

# ADVANCED STATISTICAL MODELLING

(Course notes for University of Auckland Paper STATS 330)

*By Alan Lee, Ross Ihaka and Chris Triggs*

June 2012



# Contents

<b>Preface</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Data and cases . . . . .	1
1.2 Use of computers . . . . .	3
1.3 Preparing reports . . . . .	3
<b>2 Graphical Methods</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Some examples . . . . .	5
2.3 Graphical principles . . . . .	13
2.4 Getting data into R . . . . .	15
2.4.1 Reading in a single variable . . . . .	15
2.4.2 Reading in whole data sets . . . . .	15
2.4.3 Pre-loaded data sets . . . . .	16
2.5 R code for graphics . . . . .	17
2.6 R examples . . . . .	19
2.7 Chapter summary . . . . .	21
2.7.1 Main ideas in this chapter . . . . .	21
2.7.2 R functions discussed . . . . .	21
<b>3 Regression Analysis</b>	<b>23</b>
3.1 The regression model . . . . .	23
3.1.1 Interpreting regression coefficients . . . . .	24
3.2 Fitting the model . . . . .	25
3.2.1 Fitting by least squares . . . . .	25
3.2.2 Calculating the estimates . . . . .	27
3.2.3 Interpreting the output . . . . .	30
3.2.4 Dropping and adding variables . . . . .	30
3.2.5 Fitting curves though data . . . . .	31
3.3 Prediction . . . . .	33
3.4 Diagnosing model faults . . . . .	36
3.4.1 Checking assumptions . . . . .	36
3.4.2 Outliers and high leverage points . . . . .	51
3.4.3 Summary of diagnostics . . . . .	57
3.5 Fixing up models . . . . .	58
3.5.1 Corrective action . . . . .	58
3.5.2 Summary of corrective action . . . . .	68
3.6 Case study 1: Yield of a chemical process . . . . .	68
3.7 Multicollinearity . . . . .	74
3.7.1 Mathematics of multicollinearity <sup>1</sup> . . . . .	80
3.8 Variable selection . . . . .	82
3.8.1 Model building for prediction . . . . .	83
3.8.2 Variable selection for estimation . . . . .	84

---

<sup>1</sup>This section is included for enthusiasts only – it is not examinable.

3.8.3	Selecting a subset of variables . . . . .	85
3.9	Case Study 2: Highway accident rates . . . . .	89
3.10	Smoothing . . . . .	95
3.11	Robust regression . . . . .	96
3.12	Chapter summary . . . . .	98
3.12.1	Main ideas . . . . .	98
3.12.2	R functions used . . . . .	98
<b>4</b>	<b>Models involving Factors</b>	<b>101</b>
4.1	Introduction . . . . .	101
4.2	One way Analysis of Variance . . . . .	102
4.3	Two-way analysis of variance . . . . .	104
4.4	Interaction . . . . .	105
4.5	Interactions between factors and continuous variables . . . . .	108
4.6	Chapter summary . . . . .	113
4.6.1	Main ideas . . . . .	113
4.6.2	New R functions used . . . . .	113
<b>5</b>	<b>Categorical Responses</b>	<b>115</b>
5.1	Introduction . . . . .	115
5.2	Logistic regression analysis . . . . .	115
5.2.1	Fitting the model . . . . .	116
5.2.2	Multiple logistic regression . . . . .	119
5.2.3	Analysis of grouped data using logistic regression . . . . .	122
5.2.4	Deviances . . . . .	124
5.2.5	Diagnostics for logistic regression models . . . . .	132
5.2.6	Prediction in logistic regression . . . . .	137
5.2.7	Binary anova . . . . .	139
5.3	Poisson Regression and Contingency Tables . . . . .	143
5.3.1	Poisson Regression . . . . .	143
5.3.2	Contingency tables . . . . .	147
5.3.3	Analysis of one-way tables . . . . .	148
5.3.4	Testing goodness of fit . . . . .	150
5.3.5	Analysis of two-way tables. . . . .	151
5.3.6	Analysis of three-way tables. . . . .	152
5.3.7	Odds ratios . . . . .	156
5.3.8	Odds ratios in $I \times J$ tables . . . . .	159
5.3.9	Simpson's paradox . . . . .	161
5.3.10	Association graphs . . . . .	163
5.4	Chapter summary . . . . .	164
5.4.1	Main ideas . . . . .	164
5.4.2	R functions used . . . . .	164
<b>A</b>	<b>R details</b>	<b>165</b>
A.1	R expressions . . . . .	165
A.1.1	Character expressions . . . . .	166
A.1.2	Logical expressions . . . . .	166
A.2	Data objects . . . . .	167
A.2.1	Vectors . . . . .	167
A.2.2	Matrices . . . . .	169
A.2.3	Lists . . . . .	171
A.2.4	Data frames . . . . .	172
A.3	Getting data into R . . . . .	174
A.4	Keeping track of data objects . . . . .	175
A.5	Editing data . . . . .	176
A.6	Getting online help . . . . .	176
A.7	Producing graphic images . . . . .	176

A.7.1	Drawing plots . . . . .	176
A.8	More on functions . . . . .	177
A.8.1	Arguments . . . . .	177
A.8.2	Writing functions . . . . .	178
A.9	Saving the results of the R session . . . . .	180
A.10	R packages . . . . .	180
A.11	How to get more information on R . . . . .	180
<b>B</b>	<b>R330 Documentation</b>	<b>181</b>
	acc.df . . . . .	181
	acet.df . . . . .	181
	ad.df . . . . .	182
	added.variable.plots . . . . .	183
	allpossregs . . . . .	184
	births.df . . . . .	186
	boxcoxplot . . . . .	187
	budworm.df . . . . .	188
	cancer.df . . . . .	189
	chd.df . . . . .	190
	chem.df . . . . .	190
	cherry.df . . . . .	191
	chickwts.df . . . . .	192
	coag.df . . . . .	192
	cross.val . . . . .	193
	cycles.df . . . . .	195
	diets.df . . . . .	196
	drug.df . . . . .	196
	educ.df . . . . .	197
	err.boot . . . . .	197
	ethanol.df . . . . .	199
	evap.df . . . . .	200
	fatty.df . . . . .	200
	funnel . . . . .	201
	HLstat . . . . .	202
	housing.df . . . . .	204
	influenceplots . . . . .	204
	ingots.df . . . . .	206
	kyphosis.df . . . . .	207
	lizard.df . . . . .	208
	metal.df . . . . .	208
	mines.df . . . . .	209
	onions.df . . . . .	209
	plum.df . . . . .	210
	rats.df . . . . .	211
	reg3d . . . . .	212
	ROC.curve . . . . .	212
	rubber.df . . . . .	214
	salary.df . . . . .	214
	sport.df . . . . .	215
	stamford.df . . . . .	216
	test.lc . . . . .	216
	traffic.df . . . . .	218
	vapour.df . . . . .	219
	vaso.df . . . . .	220
	WB.test . . . . .	220
	wine.df . . . . .	222



# Preface

Welcome to STATS 330 – Advanced Statistical Modelling. This course aims to give you further practical experience in the analysis of data, and builds on ideas covered in STATS 201/8.

These notes **supplement** the lecture slides, and together with the slides cover all of the material you need to master in order to pass the term tests and final exam and do the assignments.

The assignments are regarded as an important part of the course, which is why they count for 20% of the final grade. Data analysis is a practical subject, and like learning a musical instrument, cannot be mastered without a certain amount of practice. The assignments are designed to give you this practice.

Another **vital** aspect of data analysis is communication. Your best analysis is useless unless you can communicate your findings to others. Statistical work is usually done to **shed light** on important questions. If you are a statistician, the people with an interest in the answers to these questions are your customers, and your analysis and conclusions are your product. When handing in assignments, imagine that the marker is your client, who is hiring you for your professional **expertise**. Your **consulting** report (i.e. your assignment) should be **neat**, correctly spelled, well-organised, relevant and **concise** – in other words, professional.

There are weekly tutorials in this course. In the tutorials, we go over the R code that you need to do the assignments. Help in computing matters and in preparing assignments can be obtained from the lecturer and the course tutors. Some of you will be using the PC's in the Department's undergraduate computing laboratories. Students having their own computers can download R and work away from the labs as they wish.

To pass the course, you need to score at least 50% overall (i.e. for exam and coursework combined). This should not be difficult for students who complete all the assignments and master the content of these notes.





# Chapter 1

## Introduction

### 1.1 Data and cases

In this course, we will be fitting more complicated statistical models than those covered in STATS 201/8. However, we still start by collecting data on the individual objects of our study, and measuring the characteristics of interest (e.g. height, weight, length, number of defects etc) for each object.

In this course we will often call the objects *cases*. However, cases need not be human - they may be any type of object, animal, vegetable or mineral on which observations are made. The total number of cases in an investigation is called the *sample size*.

The characteristics we measure are called *variables*. The usual situation (which we confine ourselves to in these notes) is where the same variables are measured on each case. In this case the data we collect can be organised into a rectangular table or *data matrix*, with rows corresponding to cases and *columns* to variables. Sometimes, data is missing where a measurement on a particular variable is not made on a particular case.

The data in Table 1.1 illustrate this idea. Three measurements (BPD and AC measured in mm, and birth weight (BW) measured in grams) are made on each of 10 unborn infants using ultrasound techniques. Thus there are ten “cases” (infants), and each row of the data matrix consists of measurements on the same case.

As another example, a small part of the data from a dental survey is shown in Table 1.2. The variables measured are Age, the patients’ age in years, Sex, the patients’ sex (M/F) and Attitude, their attitude to dentists, as measured by the question “Do you think dentists deserve the salaries they are paid?”

Table 1.1: Ultrasound measurements on ten infants

Case	BPD	AC	Birth weight
1	77	248	1350
2	82	253	1500
3	90	265	2170
4	82	265	1780
5	86	260	1820
6	84	252	1740
7	80	230	1300
8	87	250	1730
9	79	242	1580
10	81	263	1620

Table 1.2: Data from the dental survey.

Case	Age	Sex	Attitude
1	37	M	Y
2	23	M	Y
3	45	F	Y
4	61	M	N
5	33	F	Y
6	29	F	N
7	57	M	N
8	41	F	Y

### Types of data

One obvious difference between these two examples is that while all the variables in the first data set are numeric, the second data set contains some non-numeric variables. Numerical variables can either be *continuous*, consisting of readings made along some continuous scale such as weight, length or time, or *discrete*, consisting of counts. The variables BPD, AC, BW are all measurements of length or weight and so are continuous.

In the dental example, the variable Age is continuous or possibly discrete, but the other two are *dichotomies* i.e. variables having just two categories, in this instance attitude (Yes/No) and sex (Male/Female). Using a variable such as Attitude involves careful definition of the categories, and some rule for the classification of doubtful cases, since each case must be assigned *unambiguously* to exactly one category. Variables such as Attitude and Sex that are classifications are called *categorical* variables or *factors*. Sometimes the categories are ordered, such as socioeconomic status, or age groups, in which case the factor is called *ordinal*. If there is no implied ordering in the categories (e.g. categories such as sex, race, or religious preference) then the factor is called *nominal*.

The sort of analysis *appropriate* for a given set of data (e.g. the selection of the appropriate form of model) depends on the number and type of the variables involved.

### Response and explanatory variables

In this course, there will be a particular variable that is to be studied, such as attitude or birth weight, and we want to investigate the effect that certain other variables (such as sex or socioeconomic status in the case of attitudes) have on this particular characteristic. This particular characteristic is called the *response variable*, (often abbreviated to the *response*) or sometimes the *dependent variable*. The variables whose relationship to the response is to be studied are called the *explanatory variables*, or sometimes the *independent variables*, the *predictor variables*, or the *carriers*.

### Models

To study the relationship between the response and the explanatory variables, we fit *statistical models* which describe how the distribution of the response is affected by the explanatory variables. Such models are called *regression models*. The type of model depends on the type of variables involved. Regression models where both the response and the explanatory variables are continuous are discussed in Chapter Three.

Chapter Four discusses how to incorporate factors as explanatory variables into regression models. Note that there is no full treatment of analysis of variance in this course - this is done in STATS 340.

Finally, Chapter 5 covers regression models that are suitable for categorical or count responses. We will deal with logistic regression (where the response is a binary variable), Poisson regression (where the response is a count) and log-linear models for contingency tables.

## 1.2 Use of computers

Usually the computer software necessary to fit statistical models is collected (together with facilities for the entry and manipulation of data) into a set of programs called a *statistical package*. Such a package should come complete with a consistent *interface*, a set of conventions that allows the user to communicate his or her intentions to the computer, either by typing in instructions at a keyboard, or using a pointing device such as a “mouse”, or perhaps a combination of both.

In this course, we will perform our analyses using the statistical package R, as used in STATS 201/8. We will show you how to use R to fit a variety of statistical models to data.

The program R is a open-source statistics package that is widely used around the world by statisticians in academia, business and government. It was originally created by Ross Ihaka and Robert Gentleman here in the Statistics Department at Auckland University, and is now maintained by a core group of international contributors.

R runs programs written in the statistical language S, which was originally created by researchers at Bell Laboratories in the USA. There will be quite a lot of class discussion devoted to the use of R. There are many books available on R, which fall into two categories: first, books that focus on the statistics, with an emphasis on how to make R do the necessary analyses, and second, books that focus on R as a computer language. The latter class of book is useful for coming to grips with the non-statistical aspects of R, such as data manipulation, vectorisation, subsetting and the writing of R functions. The book “R in a Nutshell” by Joseph Adler (O’Reilly, 2010) provides an easy introduction to R. Paul Murrell’s book “Introduction to Data Technologies” has a good introductory section on R. The book “An Introduction to S and S-Plus” by Phil Spector, is another good introduction. Although it describes the language S (which is almost the same as R) and is quite old book, it is relevant for R as well. Some other books are listed in the bibliography.

In addition, there is a good “on-line” help facility in R. See Section A.6 in Appendix A for more details on this.

## 1.3 Preparing reports

As we discussed in the Preface, for at least one of the assignments, you will have to write a formal report describing your analysis. To quote from the preface:

*The people with an interest in the answers are your customers, and your conclusions are your product. When handing in assignments, imagine that the marker is your client, who is hiring you for your professional expertise. Your consulting report (i.e. your assignment) should be neat, correctly spelled, well-organised, relevant and concise – in other words, professional.*

Reports must not be handwritten. They can be prepared on any word processor to which you have access.

Your reports should contain (at least) three sections.

- Section 1: Summary: A brief statement of the problem and your conclusions. You should regard this as your “executive summary” it should contain no symbols or mathematics and take up no more than one page.
- Section 2: Results: A description of your analysis. The description should include summaries of the models that you have fitted. These summaries may include output from R commands, R graphics and tables. You can include these in Word documents by directly cutting and pasting from R. Graphics and tables should be labeled “Figure 1”, “Table 2”, etc and either placed in the body, or attached to the end of the report. In the text of the report you should refer to “Figure 1”, or “Table 2”.

- Section 3: Conclusions: This section should give the conclusions from your analysis with reasons **justifying** them. You might like to bear in mind the following quote from Ronald D. Snee, who was for many years a statistical consultant with the giant U.S. chemical company DuPont:

*When I am writing a report, the single most important thing I want people to learn is always displayed in the form of a plot – preferably near the beginning. If I feel that I am obligated to include some information, but I don't want people to notice it, I display it as a table.*

There is a Powerpoint presentation about writing reports on the class web site: you may want to have a look at this.

# Chapter 2

## Graphical Methods

### 2.1 Introduction

Often a well-chosen graph gives us insight into our data, and makes clear the essential message contained in the data. We might call this the “Eureka factor”.

In STATS 201/8, you learned many graphical techniques, particularly for investigating univariate data (e.g. histograms, stem and leaf plots, quantile plots, Box-Cox plots and so on), for making comparisons (multiple boxplots, dotplots) and for investigating the relationship between variables (scatterplots). In addition, several plots that help you fit regression models were also covered (e.g. plotting residuals versus fitted values and normal plots of residuals).

Most of the graphical methods covered in 201/8 dealt with one or two variables. In this course, we will build on these plotting techniques to develop further graphical methods to deal with several (three or more) variables. A key technique will be drawing several 201/8-style plots on the same page, in order to compare plots easily. Fortunately, such graphs (called trellis plots or coplots) are easy to draw in R.

We start with several examples, which each present a data set, pose several questions about the data, and then show how the questions can be answered by a suitable plot. We also discuss the principles behind the plots. The remainder of the chapter shows you how to get the data to be plotted into R and describes the R code needed to produce different kinds of plots. We finish the chapter with a summary.

### 2.2 Some examples

#### Example 1. Rats!!

This data set consists of measurements on 16 rats, arranged in 3 groups. On each rat, measurements of weight (grams) were made on 11 dates, which are expressed in terms of elapsed days from the start of the experiment. The main purpose of the study was to investigate rat growth. We are interested in whether the growth rate is the same for each rat, or, if this is not so, at least for each group of rats. Another question is whether the growth rate is constant. In addition after day 36 (the sixth measurement) each rat had a change of diet and we want to check if this affected the growth rate.

The data have been assembled into a data frame `rats.df`. Here is a small part of this data frame:

```
> rats.df
  growth group rat change day
1    240     1   1      1    1
2    250     1   1      1    8
3    255     1   1      1   15
4    260     1   1      1   22
5    262     1   1      1   29
```

6	258	1	1	1	36
7	266	1	1	2	43
8	266	1	1	2	44
9	265	1	1	2	50
10	272	1	1	2	57
.....					
171	538	3	16	1	36
172	535	3	16	2	43
173	542	3	16	2	44
174	550	3	16	2	50
175	553	3	16	2	57
176	569	3	16	2	64

Seeing that we are interested in growth rates, a natural plot is to plot the weight of the rats versus the day the weight measurement was made. A perfectly uniform growth rate (i.e. a fixed weight gain per day) would show up as a straight line. We also need to distinguish between rats. The easiest way is to join the points corresponding to the same rat by line segments. This amounts to drawing a separate time series plot for each rat, but on the same plot. The results are shown in Figure 2.1.

This simple graph tells us most of what we need to know: the growth rates are quite uniform (the lines are quite straight). There seems to be no evidence of a major change of slope at day 36, since there is no systematic change in the slope of the lines. There seem to be two clusters of rats, with the rats in the upper (heavier) cluster having a slightly faster growth rate (the lines have a slightly greater slope.)

We might want to see if this is connected to the group the rat belongs to. To check this, we need to indicate on the graph which line corresponds to which group. There are several ways to do this. Perhaps the most obvious is to use a different way of drawing the lines for the different groups, say solid lines for group 1, dashed lines for group 2 and dashed-dotted lines for group 3. We also need to include a legend. The result is shown in Figure 2.2. It is now clear that the rats in Group 1 are lighter. They start off lighter and grow at slightly lesser rate than the other groups. Group 2 is in turn lighter than group 3, except for the trace at the top of the plot. This raises the question of whether the group of this rat has been correctly recorded. Which rat is it? The plot could be enhanced by labeling each line with the rat number: Figure 2.3 shows that rat 12 is the rat in question.

Another way of plotting these “growth curves” is to draw the points for each rat on a separate plot, but using the same axis scale for each plot. This is shown in Figure 2.4. This display shows the difference in growth rates more clearly. Rats 9,10 and 12 (i.e. rats 1,2 and 4 in group 2) have higher growth rates than the rest, as do rats 13 and 16 (rats 1 and 4 in group 3). We could also smooth out the curves using a loess smoother, and plot the data points on the same plot. This makes the differences in slope stand out. This is shown in Figure 2.5, using “Trellis Graphics”. See Section 2.5 and the document “A Tour of Trellis Graphics” on the course website for more details about trellis graphics.

### Example 2. Cycles to failure

The data shown below come from an experiment to investigate the connection between the failure of worsted yarn and several explanatory variables. The experiment was performed by subjecting a length of yarn to repeated cycles of loading, and measuring the number of cycles it took for the yarn to fail. This process was repeated under various conditions: the length of the specimens was taken to be 250, 300 or 350 mm, the cycle amplitudes were taken to be 8, 9, or 10 mm and the loads were taken to be 40, 45 or 50 grams.

There are three factors:

**yarn.length:** having 3 levels, Low, Medium and High;  
**amplitude:** having 3 levels, Low, Medium and High;  
**load:** having 3 levels, Low, Medium and High.

The data are recorded as Low, Medium or High for each of these three variables, and the response variable `cycles` records the number of load cycles it took to destroy the specimen. The data were assembled into a data frame `cycles.df`:

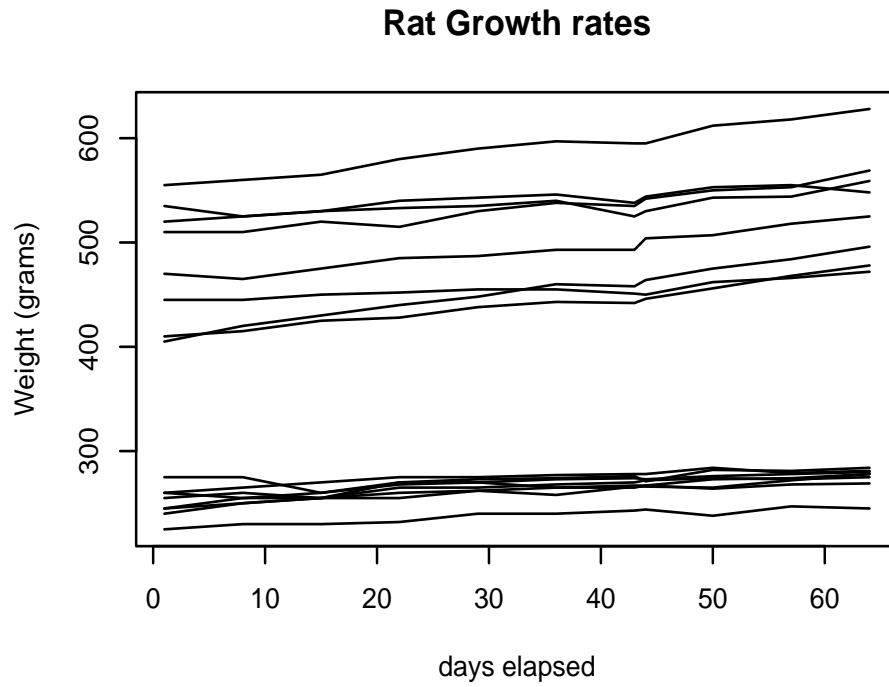


Figure 2.1: Growth rate of rats

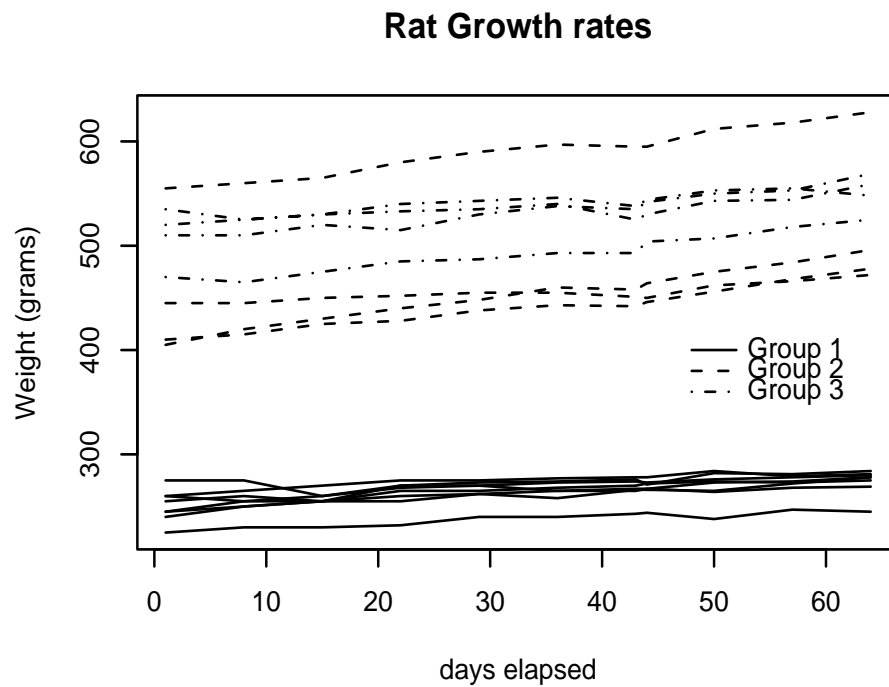


Figure 2.2: Coding groups by linetype.

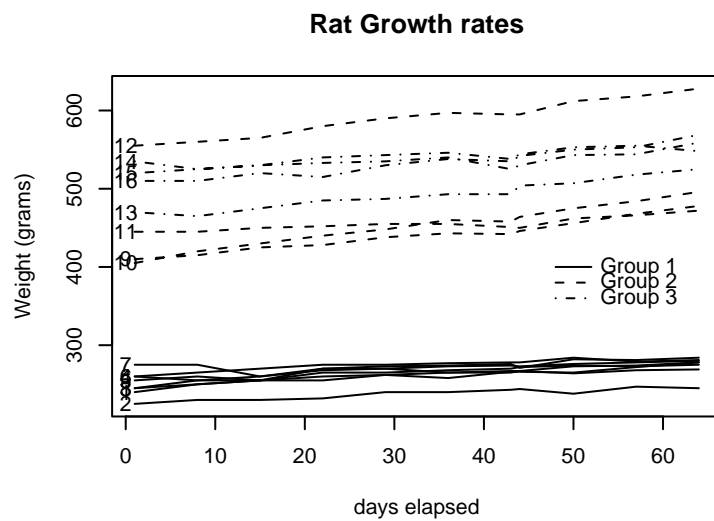


Figure 2.3: Plot with rat numbers added.

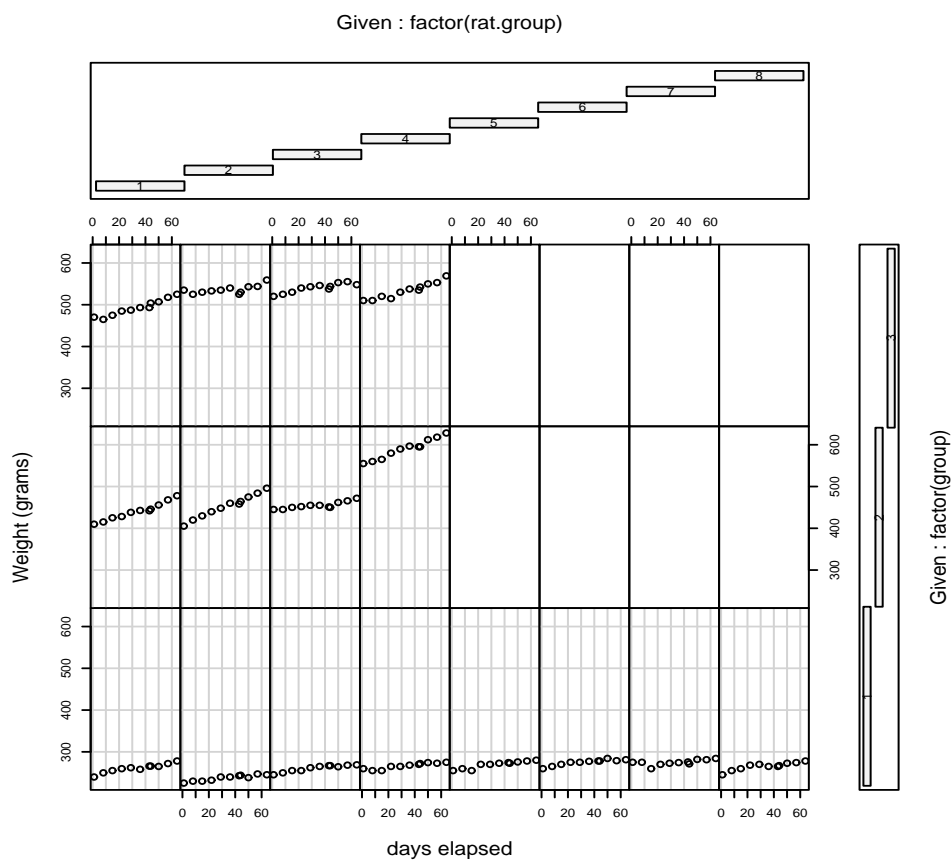


Figure 2.4: Plotting rats on separate graphs.



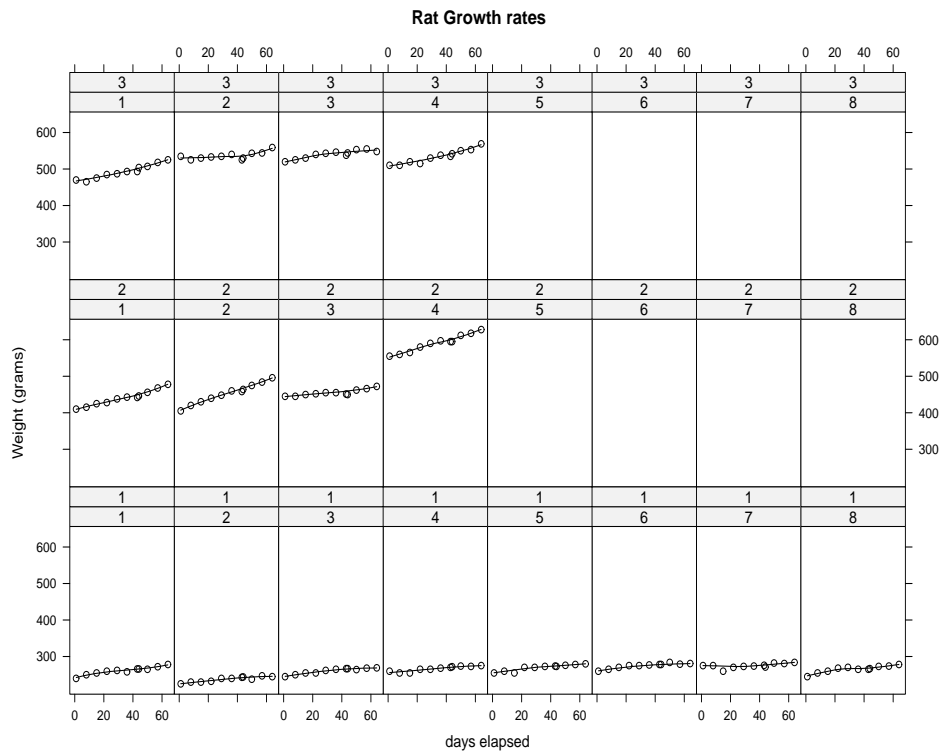


Figure 2.5: Plotting groups on separate graphs, with loess smoothing.

	yarn.length	amplitude	load	cycles
1	low	low	low	674
2	low	low	med	370
3	low	low	high	292
4	low	med	low	338
5	low	med	med	266
6	low	med	high	210
7	low	high	low	170
8	low	high	med	118
9	low	high	high	90
10	med	low	low	1414
11	med	low	med	1198
12	med	low	high	1634
13	med	med	low	1022
14	med	med	med	620
15	med	med	high	438
16	med	high	low	442
17	med	high	med	332
18	med	high	high	220
19	high	low	low	3636
20	high	low	med	3184
21	high	low	high	2000
22	high	med	low	1568
23	high	med	med	1070
24	high	med	high	566
25	high	high	low	1140
26	high	high	med	884
27	high	high	high	360

We need a way of displaying these four variables (the three factors yarn.length, amplitude and

load, and the response cycles) on a graph so that way the factors effect the response is clearly displayed.

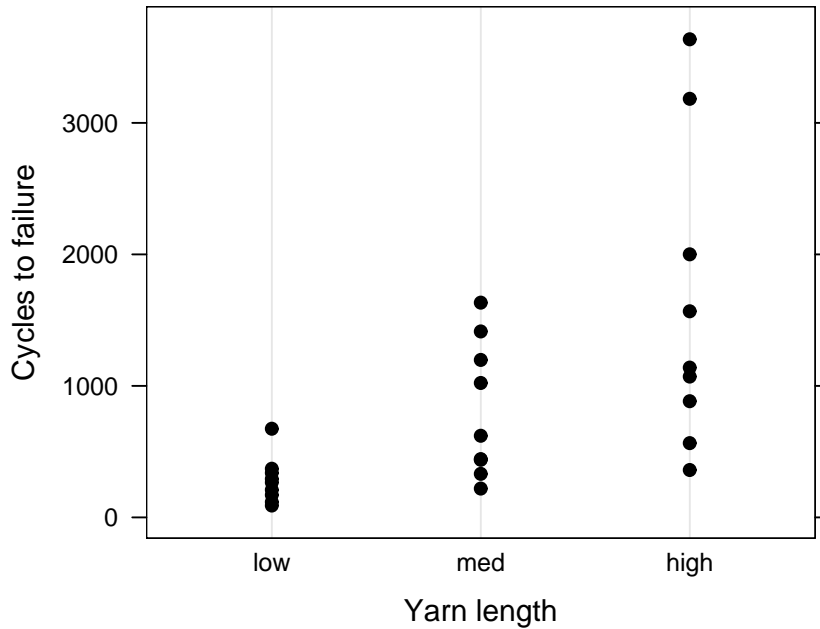


Figure 2.6: Dotplot of cycles for different levels of yarn length.

If there was just one factor, dotplots of the response separately for each value of the factor would be the standard 201/8 style method. For example, if we want to see how the “number of cycles to failure” depends on the levels of the factor “yarn.length”, we would draw a dotplot like that in Figure 2.6. We can clearly see that the longer specimens take longer (i.e. more cycles) to fail.

We can incorporate the other variables by using a separate “panel” for each combination of levels of the other two factors: the result is a “Trellis” plot shown in Figure 2.7. Now we can see that low amplitude and low load specimens take longer to fail. The “first column” and “bottom row” of dotplots are on the whole higher than the others.

### Example 3. Nitrogen

Our final example demonstrates some graphical techniques that are useful for showing relationships between three numeric variables. This example is taken from William Cleveland’s book “Visualizing Data”, page 10. The data come from an experiment to see how the amount of nitrogen oxides emitted from car engines depends on the engine parameters. Two engine parameters were studied: compression ratio (C) and the equivalence ratio (E), which is a measure of the richness of the air/fuel mixture. Eighty-eight experimental runs were made with different combinations of C and E, measuring the amount of emitted nitrogen oxides in milligrams per joule (recorded as variable NOx) for each run. The data have been assembled into a data frame `ethanol.df`, a small part of which is shown below:

	NOx	C	E
1	3.741	12.0	0.907
2	2.295	12.0	0.761
3	1.498	12.0	1.108
4	2.881	12.0	1.016
5	0.760	12.0	1.189
6	3.120	9.0	1.001
7	0.638	9.0	1.231

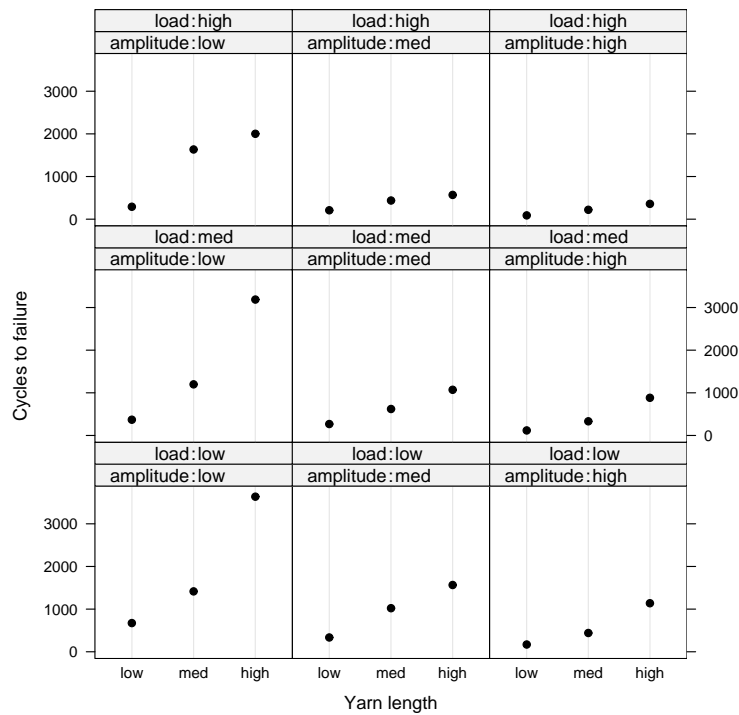


Figure 2.7: Trellis Dotplot of cycles for different levels of yarn length, separately for all other factors.

```

8 1.170 9.0 1.123
9 2.358 12.0 1.042
10 0.606 12.0 1.215
11 3.669 12.0 0.930
12 1.000 12.0 1.152
13 0.981 15.0 1.138
14 1.192 18.0 0.601
15 0.926 7.5 0.696
16 1.590 12.0 0.686
17 1.806 12.0 1.072
18 1.962 15.0 1.074
19 4.028 15.0 0.934
20 3.148 9.0 0.808
...more data...

```

There were only 5 values of  $C$  used, so we could treat  $C$  as a factor with 5 levels. However, we will treat  $C$  as a numeric variable for the purposes of this example.

If we had just two variables, we could draw a scatterplot, which would reveal the relationship between them. For example, if we ignore  $C$ , we can plot  $NOx$  versus  $E$ , as in Figure 2.8.

The plot shows an obvious pattern, with  $NOx$  peaking at a value of about 0.9 for the equivalence ratio. The question investigated in this example is how does this relationship between  $NOx$  and  $E$  depend on  $C$ ?

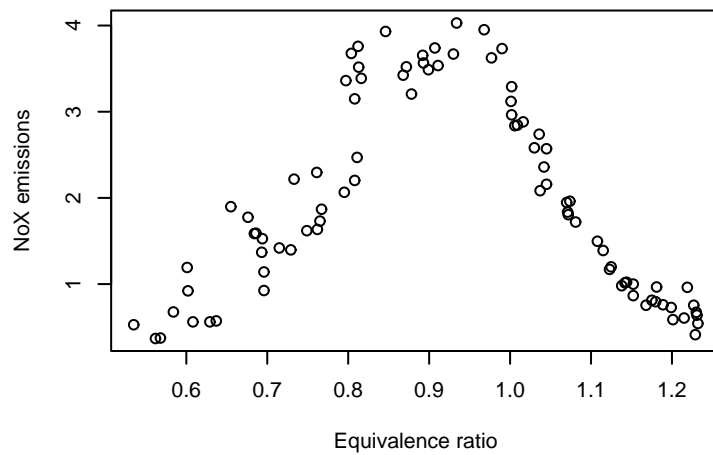


Figure 2.8: Plot of nitrogen oxides versus equivalence ratio.

To answer this, we have several tools available. We could

- Make scatterplots for all possible pairs of variables, or a 3-d scatterplot of all three.
- Draw a scatter plot of two variables, using a coded plotting symbol to show the value of the third variable.
- Draw a scatter plot for each value or range of values of the third variable.

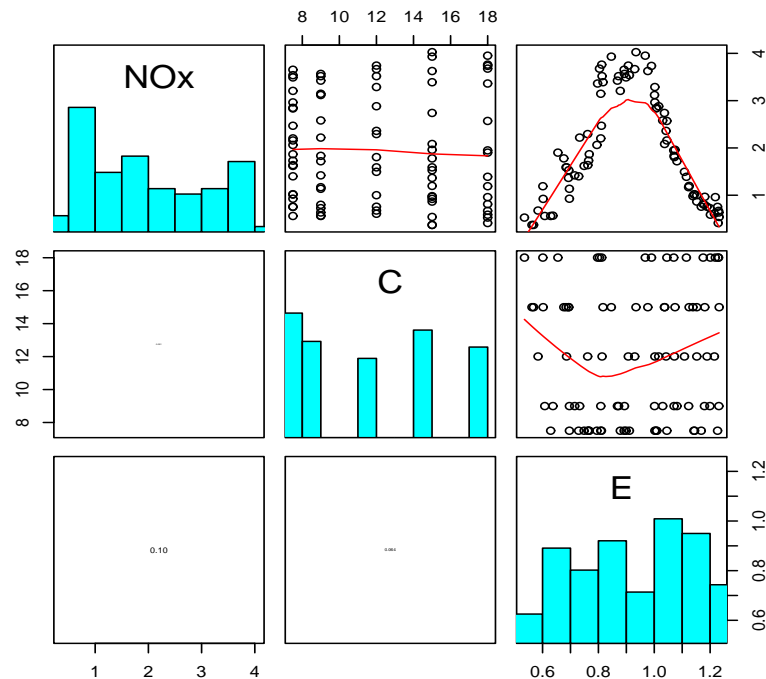


Figure 2.9: All possible scatterplots of the nitrogen data.

Here is how these options turn out for our nitrogen oxide data:

1. (All possible scatterplots: see Figure 2.9) Conclusion: We can't tell much from this: the data live in 3 dimensions and two-dimensional views are not very informative.
2. (Three-dimensional scatterplot: see Figure 2.10) Conclusion: This is not quite as informative as the previous plots, but improves if we can alter the angle at which we view the picture.
3. (Code value of third variable: see Figure 2.11) Conclusion: This picture is more informative: we see that the curves for various values of  $C$  get slightly higher as the value of  $C$  increases.
4. (Separate plot for each value of third variable: see Figure 2.12) Conclusion: This plot shows the same thing as the previous plot: as the value of  $C$  increases the  $\text{NO}_x$  peak is a little higher. Note that if the third variable has many different values, we can divide up the range of the third variable into “bins” and draw a scatterplot for the data in each bin.

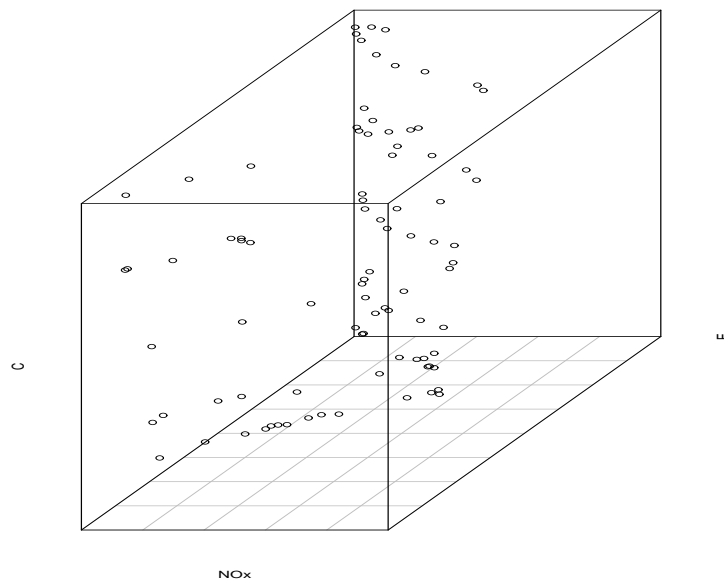


Figure 2.10: Plotting  $\text{NO}_x$ ,  $E$  and  $C$  in 3-d.

## 2.3 Graphical principles

We have looked at several different types of graph that can be used to visualise data containing several variables. We have concentrated on the case where a numerical variable (the response) is being influenced by several other variables (which may be numeric variables or factors).

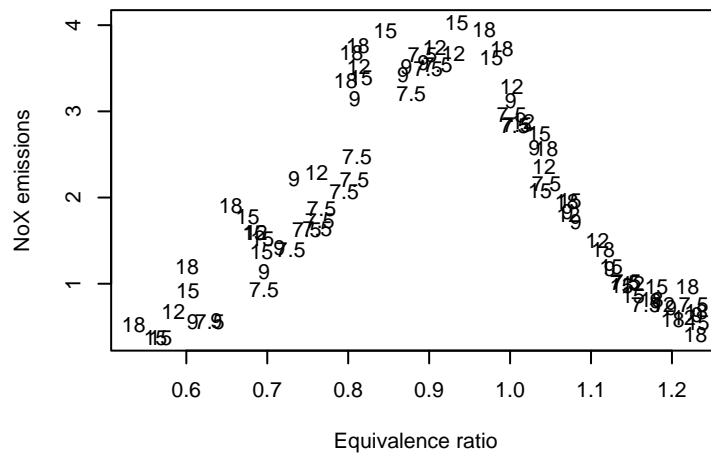


Figure 2.11: Plotting NOx versus E, coding value of C.

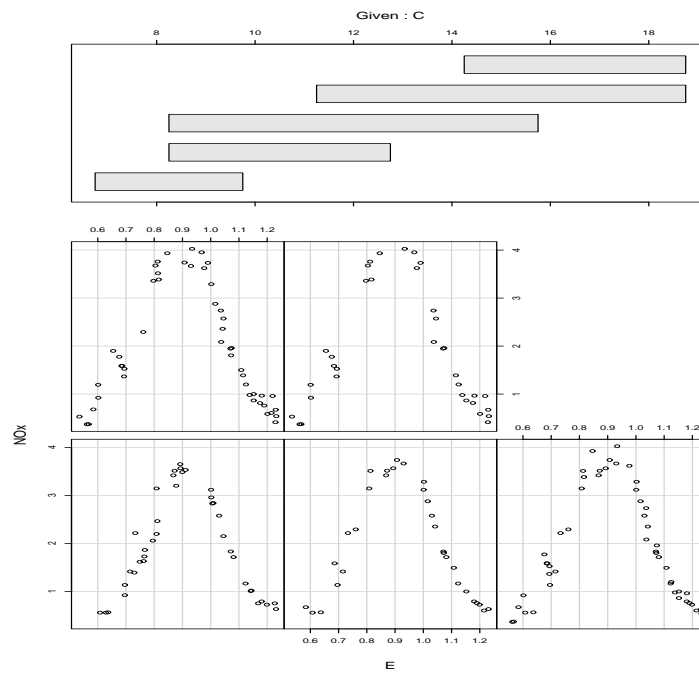


Figure 2.12: Plotting NOx versus E, separate plot for each value of C.

There are always many ways to visualise data: clever ideas help here. One way of portraying the relationships between multivariate data is to focus on the (numeric) response and one other variable, and draw the kind of plot suitable for them (scatterplots if both variables are numeric, or dotplots/boxplots if the second variable is a factor). The remaining variables can be included by either coding the plotting symbols, or by using the coplots, where we draw separate plots for each value or range of values of the extra variables. The fact that the coplots are laid out in rows and columns of plots according to the values of the extra variables, using the same scale and axes for each plot, makes comparisons easier.

## 2.4 Getting data into R

In order to draw graphs, we must first import our data into R. There are several ways to do this, depending on the format of the data. We will discuss our different situations (i) when we want to create a single variable, (ii) when we have a data set with several variables, with the data in a text file, (iii) when the data set has several variables and is in the form of an Excel spreadsheet, and (vi) where the data are part of an R package.

### 2.4.1 Reading in a single variable

If we want to create a single variable `x`, and there are only a few values, we can use the `c` (for combine) function in R:

```
> x = c(2.3, 5.6, 1.8, -0.4)
> x
[1] 2.3 5.6 1.8 -0.4
```

For more values, we can use the `scan` function:

```
> x = scan()
```

We then type in the values separated by spaces, typing multiple numbers on a line if we want. We terminate the input by a blank line. If we want, we can cut and paste numbers from another source. An alternative is to type the numbers into an external text file, called `x.txt` say. We can then type in R

```
> x = scan("x.txt")
```

to get the same result.

### 2.4.2 Reading in whole data sets

Most often, we will want to import an entire data set and create a data frame. First, let us assume that we have a data set, say the ultrasound data listed in Table 1.1, that has been typed into a text file created in Notepad or some other text editor. Suppose the first line contains the variable names, (these must begin with a letter and contain no spaces), and the remaining lines data, and we omit the case numbers, so the file looks like

```
BPD AC Birth_weight
77 248 1350
82 253 1500
90 265 2170
82 265 1780
86 260 1820
84 252 1740
80 230 1300
87 250 1730
79 242 1580
81 263 1620
```

Suppose we save the file under the name `ultra.txt`. We can read this text file into R and create a data frame `ultra.df` say using the code

```
> ultra.df = read.table(file.choose(), header=TRUE)
```

This brings up the standard Windows dialog that allows you to browse the file system in the normal way to find your file. The argument `header=TRUE` tells R that the first line of the file contains the variable names (but see below).

Alternatively, if the file is in the current (working) directory, you can type

```
> ultra.df = read.table("ultra.txt", header=TRUE)
```

We can inspect the data frame by typing its name:

```
> ultra.df
  BPD  AC Birth_weight
1   77 248         1350
2   82 253         1500
3   90 265         2170
4   82 265         1780
5   86 260         1820
6   84 252         1740
7   80 230         1300
8   87 250         1730
9   79 242         1580
10  81 263         1620
```

We have a lot of freedom in naming objects, but the names must start with a letter, use only letters and numbers (and the characters `_` and `.` and contain no spaces. Names are *case-sensitive*, so that capitals and lower case are treated as distinct symbols. This is different from SAS. However, I recommend ending data frame names with `.df`. This makes it easier to distinguish them from other R objects, such as vectors and matrices.

An alternative to using a text editor such as Notepad is to type the data into an Excel spreadsheet, using the same format as before, with the names in the first row and the data in succeeding rows, as shown in Figure 2.13. To read this into R, save it as a CSV (comma delimited) file `ultra.csv` say. Then, typing

```
> ultra.df = read.table("ultra.csv", header=TRUE, sep=",")
```

produces the same result as before.

The code above assumes that there is one column in the file for each variable. After the data has been read in, we can print it by typing the name of the data frame. Each line is labelled with a sequence number, by default the line number (1,2,... and so on). In fact, every data frame in R has such a set of sequence numbers (called *rownames*).

Sometimes input files will contain an initial column containing a sequence number or ID number that we would like to convert into *rownames*. To do this, we simply omit the argument `header=TRUE` from the argument list of `read.table`. Provided the first line of the input file contains one less variable name than the number of data columns, all will be well.

### 2.4.3 Pre-loaded data sets

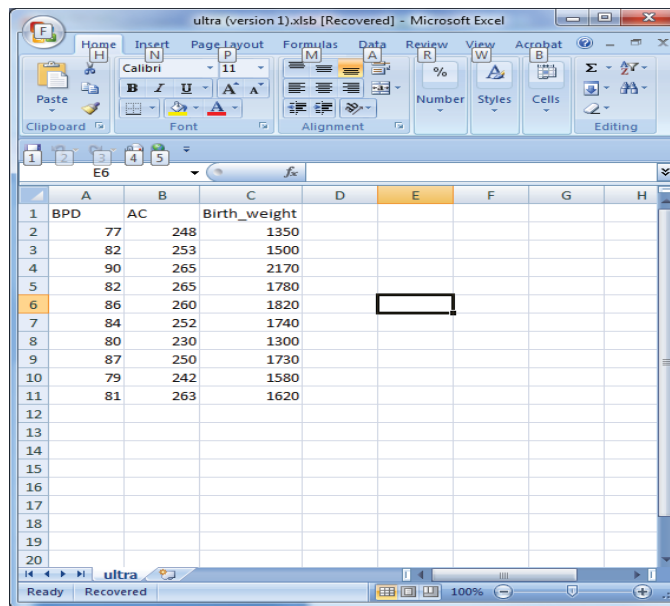
A feature of R is the use of “packages”, which are collections of functions and data that can be loaded into R. The package used in this course is called `R330`, and can be downloaded from the internet and installed in R using the method described in Appendix A.10. Assuming this has been done, we type

```
> library(R330)
```

to load the data and functions into R. One of the data sets in the package is the ethanol data, in the form of the data frame `ethanol.df`<sup>1</sup>. We make the data accessible by the command

<sup>1</sup>See Appendix A.2.4 for more information on data frames





	A	B	C	D	E	F	G	H
1	BPD	AC	Birth_weight					
2	77	248	1350					
3	82	253	1500					
4	90	265	2170					
5	82	265	1780					
6	86	260	1820					
7	84	252	1740					
8	80	230	1300					
9	87	250	1730					
10	79	242	1580					
11	81	263	1620					
12								
13								
14								
15								
16								
17								
18								
19								
20								

Figure 2.13: The ultrasound data as an Excel spreadsheet.

```
> data(ethanol.df)
```

A complete list of all the available data sets in the package can be found in Appendix B.

More details on reading in data can be found in Appendix A.3, and in the books listed in the bibliography.

## 2.5 R code for graphics

Now we describe in general terms some of the R functions used to produce the graphs in Section 2.2. For details consult the on-line help in R, as described in Section A.6 of these notes. We present the code used to draw the figures in this chapter and make some comments as we go.

The basic function in R is the function `plot`. This can have a variety of arguments, and usually produces something sensible. The most usual syntax is

```
> plot(x,y)
```

This will produce a scatterplot if the variables `x` and `y` are both numeric variables, and a set of boxplots (one for each level of `x`) if `x` is a factor. (What happens if `y` is a factor???) If `x` and `y` are not vectors, but are variables in data frame `data.df`, this will not work, as R can't see inside the data frame. You have to either attach the data frame, or, better, type

```
> plot(data.df$x,data.df$y)
```

Even better is

```
> plot(y~x,data=data.df)
```

Sometimes we want to plot coded points, or join up the plotted points with lines. For example, if the variables `x` and `y` are both numeric variables, and we want to join up the plotted points with lines, we can use the function `lines`:

```
> plot(x,y)
> lines(x,y)
```

(For this to work, the elements of the vector `x` must be sorted in ascending order. What happens if they are not??)

Sometimes we want to use a different plotting symbol. We use the function `text` to put text on the plot. Suppose we want to plot `y` versus `x` but use the corresponding level of the factor `z` as a plotting symbol. We type

```
> plot(x,y,type="n")
> text(x,y,z)
```

In this piece of code, the actual plotting is done by the function `text`, the function `plot` is just used to set up the plot, draw the box and label the axes. The extra argument `type="n"` stops anything being plotted. This is how Figure 2.11 was drawn. Finally, your graphs should always have titles or captions and axis labels.

Suppose we want to draw a separate plot of a response versus a numeric explanatory variable for each value of a third variable. For example, for the nitrogen data, we would type

```
> coplot(N0x~E|C, data=ethanol.df)
```

Note that R cannot find the variables `N0x`, `E` and `C` directly, which is why we need the argument `data=ethanol.df`. See Appendix A.2.4. Note the rather unusual argument: it is of the form

response variable  $\sim$  explanatory variable | conditioning formula

The “conditioning formula” can take several forms. For example, it can be

1. The name of a variable, say `C`. In this case the data is divided into groups according to the value of `C` and plot drawn for each group. To do this we would type

```
> coplot(N0x~E|C, data=ethanol.df)
```

The range of `C`-values for each group is shown on the plot.

2. If `Z` and `W` are factors, with say 3 and 4 levels respectively, then there are  $3 \times 4 = 12$  possible combinations of `Z` and `W` levels. If `y` and `x` are both numeric variables, and we want to plot `y` versus `x` separately for each of these 12 combinations of levels, we type

```
> coplot(y~x|Z*W)
```

This produces a  $3 \times 4$  array of scatterplots, all drawn on the same scale, with all plots in the first row of the array using the data corresponding to the first level of `Z`, all plots in the first column using the data corresponding to the first level of `W` and so on.

For adding titles and labels on the `x` and `y` axes, we use the arguments `main`, `xlab`, `ylab`.

## Trellis Graphics

A more powerful type of coplot can be drawn with the trellis graphics features available in R, in the package `lattice`. This still keeps the idea of laying out the plots in a matrix of plots, with row and columns corresponding to the conditioning variables, but allows more control over what is actually plotted. The matrix of dotplots in Figure 2.7 was drawn using this package. A document describing the use of trellis graphics is available on the course website. This document describes trellis graphics as implemented in S, but the `lattice` package in R is essentially the same.

## Getting more information

These notes are a bare minimum of information to get you started using R plots. For more information on plotting generally, I recommend the books by Paul Murrell and Joseph Adler listed in the references. For Trellis, download the manual from the course web site, or see Paul’s book. The Adler book also has information on trellis graphics.

## 2.6 R examples

In this section we discuss the R code that as used to draw the figures in Section 2.2.

**Figure 1** The code is (assuming data is in a data frame `rats.df`)

```
par(mfrow=c(1,1)) # set up one plot per page
plot(growth~day,type="n", data=rats.df,
     xlab="days elapsed",
     ylab="Weight (grams)",
     main="Rat Growth rates")

# now loop for the lines
for(i in (0:15)){
  index<-(1:11) + i*11
  lines(rats.df$day[index],rats.df$growth[index])
}
```

Note how we used the `plot` command with the `type="n"` argument to set up the plot, without drawing the lines. The actual lines are drawn with the `lines` command. We used a loop to avoid the end of one rat's line joining up with the beginning of the next. Note the use of subsetting to pick out the appropriate lines of the data frame.

**Figure 2** The code is

```
group.vec<-as.numeric(rats.df$group)
# convert group from factor to vector
plot(growth~day,type="n",
     data=rats.df,
     xlab="days elapsed",
     ylab="Weight (grams)",
     main="Rat Growth rates")
for(i in (0:15)){
  index<-(1:11) + i*11
  lines(rats.df$day[index],rats.df$growth[index],
        lty=group.vec[index[1]])
}
legend(45,400,paste("Group",1:3),lty=c(1,2,3))
```

We used the argument `lty=` to draw different line styles, and the function `legend` to put a legend on the plot. Note that the linetype (`lty`) has to be an integer. Since `group` was a factor (and hence had character values) we had to convert these into integers using the function `as.numeric`.

**Figure 3** The code is the same as the previous example, with the addition of the lines

```
ratindex<-seq(1,166,by=11)
text(0,rats.df$growth[ratindex],1:16)
```

We have used the function `text` to label each growth curve with its number.

**Figure 4** We need to first create the vector `rat.group` to define the groups for the plot. Note we have to convert `group` and `rat.group` into factors: otherwise the conditioning won't work properly. The code is

```
rat.group<-c(rep(1:8,rep(11,8)),rep(1:4,rep(11,4)),
             rep(1:4,rep(11,4)))
coplot(growth~day|factor(rat.group)*factor(rats.df$group),
       data=rats.df,
       xlab="days elapsed",
       ylab="Weight (grams)")
```

**Figure 5** Here we use Trellis graphics with a “panel function”, which allows us to control what we plot in each panel (in this case to smooth the points.) Note we have to invoke the Trellis library with the line `library(lattice)`. The code is

```
library(lattice)
xyplot(growth~day|factor(rat.group)*factor(group),
       data=rats.df,
       xlab="days elapsed",
       ylab="Weight (grams)",
       panel = function(x, y, ...){
         panel.xyplot(x,y)
         panel.loess(x, y, span = .8, ...)
       }
)
```

**Figure 6** We used the trellis function `dotplot` to draw the dotplots. Note how we specify a data frame containing the data by using the `data=` argument, if we don’t want to attach the data frame. Note also how we use the function `factor` to reorder the levels of the factors. The code is

```
cycles.df$yarn.length=factor(cycles.df$yarn.length, levels= c("low", "med", "high"))
cycles.df$amplitude=factor(cycles.df$amplitude, levels= c("low", "med", "high"))
cycles.df$load=factor(cycles.df$load, levels= c("low", "med", "high"))
dotplot(cycles~yarn.length,
       xlab="Yarn length",
       ylab="Cycles to failure",
       data=cycles.df)
```

**Figure 7** Note how we use the function `factor` to reorder the levels of the factors. The code is

```
dotplot(cycles~yarn.length|amplitude*load,
       xlab="Yarn length",
       ylab="Cycles to failure",
       data=cycles.df,
       strip=function(...)strip.default(...,strip.names=T))
```

**Figure 8** The code is

```
plot(NOx~E,
     data=ethanol.df,
     ylab="NOx emissions",
     xlab="Equivalence ratio")
```

**Figure 9** We use the `pairs20x` function to draw all possible scatter plots.

```
pairs20x(ethanol.df)
```

**Figure 10** We use the `plot` and `text` functions to set up the plot and draw the points.

```
plot(ethanol.df$E,ethanol.df$NOx,
     type="n",ylab="NoX emissions",
     xlab="Equivalence ratio")
text(ethanol.df$E,ethanol.df$NOx,ethanol.df$C)
```

**Figure 11** The code is;

```
coplot(NOx~E|C,data=ethanol.df)
```

**Figure 12** To use the `scatterplot3d` function, you have to load the `scatterplot3d` library. The code is

```
library(scatterplot3d)
scatterplot3d(x=ethanol.df$C,
y=ethanol.df$E,
z=ethanol.df$NOx,
ylab="NoX emissions",
xlab="Equivalence ratio",
zlab="NOx")
```

or, using the Trellis graphs

```
cloud(NOx~C*E, data=ethanol.df)
```

## 2.7 Chapter summary

### 2.7.1 Main ideas in this chapter

- Plotting gives insight into data.
- For a continuous response, and a continuous explanatory variable, try scatterplots.
- For a continuous response, and a explanatory variable that is a factor, try dotplots, boxplots.
- Incorporate other variables by using coplots, all possible pairs of scatterplots.
- Bin continuous variables where necessary.
- There are no rules: anything that works (i.e. displays something interesting about the data) is OK.

### 2.7.2 R functions discussed

as.numeric  
attach  
cloud  
coplot  
dotplot  
for  
legend  
lines  
pairs20x  
plot  
reg3d  
scatterplot3d  
text  
xyplot

Use the R help system to find the details for these functions. For example, to get help on the function `plot`, just type `?plot`. Alternatively, you can make a selection from the Help menu.



# Chapter 3

## Regression Analysis

### 3.1 The regression model

We use regression models when we have a continuous response variable and one or more continuous explanatory variables, and we want to explore the relationship between the response and the explanatory variables, and use the explanatory variables to predict the response. The *linear regression model* has the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + e_i \quad i = 1, 2, \dots, n \quad (3.1)$$

where we are assuming that we have  $k$  explanatory variables, observed on  $n$  cases, and that the response  $y_i$  measured on the  $i$ th case is composed of a linear combination of the explanatory variables measured on the  $i$ th case, plus a random error  $e_i$ . The random errors are supposed to be independently drawn from an  $N(0, \sigma^2)$  distribution. The linear combination describes how the explanatory variables are related to the mean value of the response:

$$E(y_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}$$

while the error variance  $\sigma^2$  describes the amount of variation about these mean values. Note that the regression coefficient  $\beta_j$  measures the average effect on the response of a unit change in  $x_j$ , given that the values of all the other variables are fixed.

In statistical terms, we are assuming that the response is normally distributed, with mean a linear function of the explanatory variables, and constant variance. In geometric terms, if we regard the data  $(x_{i1}, x_{i2}, \dots, x_{ik}, y_i)$  from the  $i$ th case as a point in a  $(k+1)$ -dimensional space, we are assuming that the data are uniformly scattered about a (hyper)plane, with the displacements from the plane in the  $y$ -direction being normally distributed.

In R, we represent the model (3.1) using the notation

$$y \sim x1 + x2 + \dots + xk$$

where  $y$  is the name of the response variable and  $x1, \dots, xk$  are the names of the explanatory variables. Note that we do not refer to the errors or the coefficients in this form: we merely specify which variable is the response and which variables are the explanatory variables. The actual model fitted will be assumed to be of the form (3.1).

The regression coefficients  $\beta_0, \dots, \beta_k$  and the error variance  $\sigma^2$  are unknown and need to be estimated from the data, which we will assume are stored in a data frame in row and column form.

The model (3.1) is often written in matrix form: if we define a matrix  $X$  and a vectors  $\beta$  by

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix} \quad \text{and} \quad \beta = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_k \end{bmatrix} \quad (3.2)$$

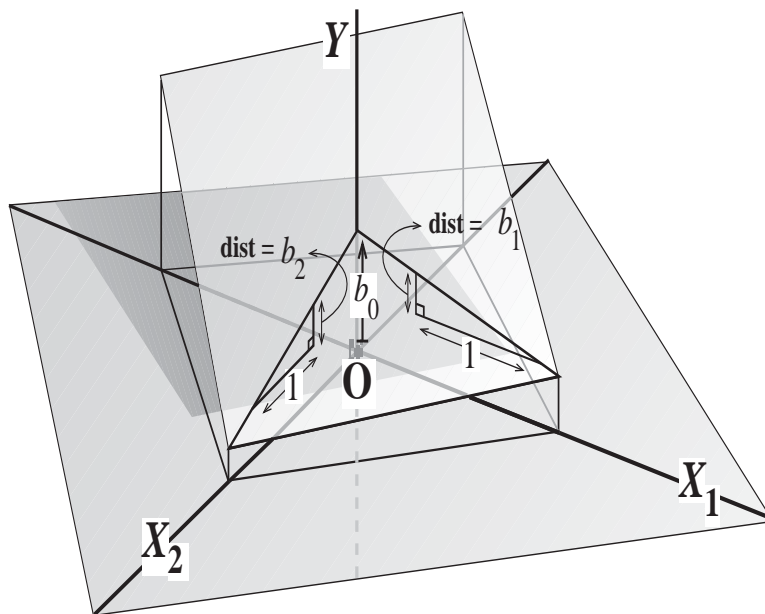


Figure 3.1: Geometric interpretation of the regression coefficients.

then the model (3.1) can be written

$$Y = X\beta + e$$

where the vectors  $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$  and  $e = \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix}$  are the vectors of responses and errors respectively.

### 3.1.1 Interpreting regression coefficients

Suppose our data follow a regression model, where the error variance is small so that the data (almost) lie on a plane. To keep things simple, suppose that there are only two independent variables  $X_1$  and  $X_2$  say, so that the data live in 3 dimensions.

The equation of the plane is

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2. \quad (3.3)$$

The parameter  $\beta_0$  is the  $y$ -intercept, and the parameter  $\beta_1$  is the slope of the line formed by intersecting the regression plane with the plane formed by the  $x_1$  and  $y$  axes. Similarly,  $\beta_2$  is the slope of the line formed by intersecting the regression plane with the plane formed by the  $x_2$  and  $y$  axes. See Figure 3.1 for the geometry of this.

The plane represents the average value of the response, given the values of the explanatory variables  $X_1$  and  $X_2$ . The height  $y$  of the plane over the point  $(x_1, x_2)$  is the expected value of an observation having  $X_1 = x_1$  and  $X_2 = x_2$ . In algebraic terms,  $y$  is given by the formula (3.3). Actual observations differ from this expected value by a random error, whose likely size is governed by the error variance  $\sigma^2$ .

Suppose we increase the value of the explanatory variable  $X_1$  by 1 to the value  $x_1 + 1$ , but keep  $X_2$  fixed at  $x_2$ . The value of the expected response is now  $\beta_0 + \beta_1(x_1 + 1) + \beta_2 x_2$ . The increase is

$$(\beta_0 + \beta_1(x_1 + 1) + \beta_2 x_2) - (\beta_0 + \beta_1 x_1 + \beta_2 x_2) = \beta_1.$$



- The regression coefficient  $\beta_1$  represents the increase in the mean response associated with a unit increase in the variable  $X_1$ , **provided**  $X_2$  is held fixed.
- The coefficient  $\beta_2$  represents the increase in the mean response associated with a unit increase in the variable  $X_2$ , provided  $X_1$  is held fixed.
- Positive coefficients suggest that an increase in the explanatory variable is associated with an **increase** in the response.
- Negative coefficients suggest that an increase in the explanatory variable is associated with a **decrease** in the response.

The regression coefficients are sometimes called partial regression coefficients to emphasise that their interpretation requires that the other variable should be held fixed. A numerical example will illustrate the point. The data frame `example.df` contains 100 observations generated from the regression model

$$y = 1 + 3x_1 - 2x_2 + e \quad (3.4)$$

where the errors  $e$  have error variance  $\sigma = 0.05$ .

If we fit the regression model (see the next two sections on how to do this in R) , we get the following estimates:

$$\begin{aligned} \beta_0: & \quad 1.006 \\ \beta_1: & \quad 2.892 \\ \beta_2: & \quad -1.889 \end{aligned}$$

Suppose we ignore the variable  $X_2$  altogether. A plot of  $Y$  versus  $X_1$  is shown in Figure 3.2. The fitted line shown on the plot has slope 0.988, which is quite different from 2.892. Why the difference? The first slope (0.988) represents the unit increase in  $Y$  for a unit increase in  $X_1$  for *all* points, irrespective of the value of  $X_2$ . The second value (2.892) represents the unit increase in  $Y$  for a unit increase in  $X_1$  for points having the *same* value of  $X_2$ . This is illustrated in Figure 3.3, where we re-plot the  $y$  versus  $x_1$ , this time showing the (coded) value of  $X_2$ . The bands of points having similar  $X_2$ -values (the bands of points in the figure plotted using the same digit) have considerably higher slope (round about 3) than the whole plot, which has slope about 1. We can see this more clearly in a coplot. Figure 3.4 shows a coplot where the separate bands of points in Figure 3.3 (corresponding to similar values of  $X_2$ ) have been plotted on separate panels. We see that the common slope of the points in these panels is about 3. Thus, the partial coefficient of  $X_1$ , representing the increase in  $y$  for a unit increase in  $X_1$  when  $x_2$  is held fixed, can be interpreted as the slope in the coplot.

## 3.2 Fitting the model

Now we describe how to fit the model to our data using least squares and R.

### 3.2.1 Fitting by least squares

The elements of  $\beta$  are estimated by least squares: we estimate them by the values that minimise

$$\sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_k x_{ik})^2. \quad (3.5)$$

Mathematically, the minimising  $\beta$ 's are found by solving the *normal equations*

$$X^T X \beta = X^T Y.$$

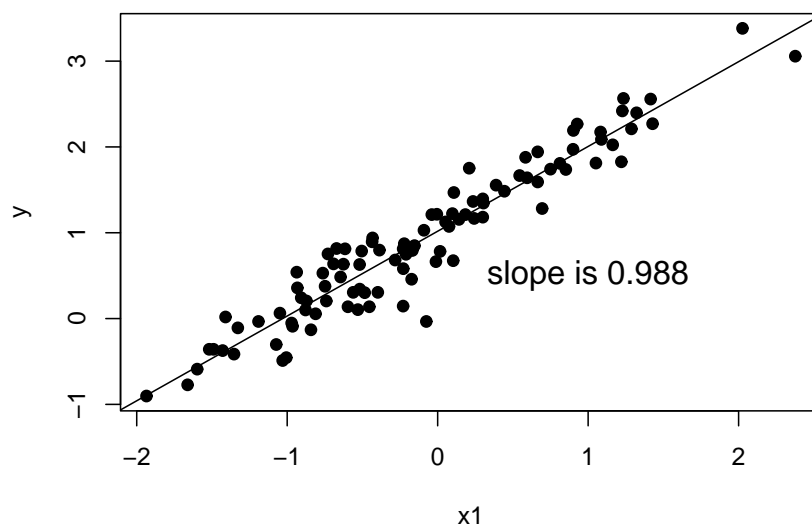


Figure 3.2: Plot of  $Y$  versus  $X_1$ , with fitted line added.

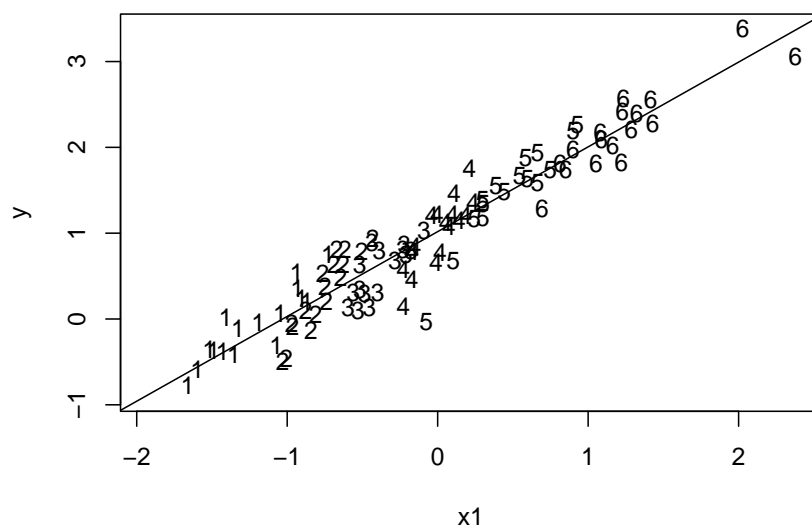


Figure 3.3: Plot of  $Y$  versus  $X_1$ , with  $X_2$  coded.

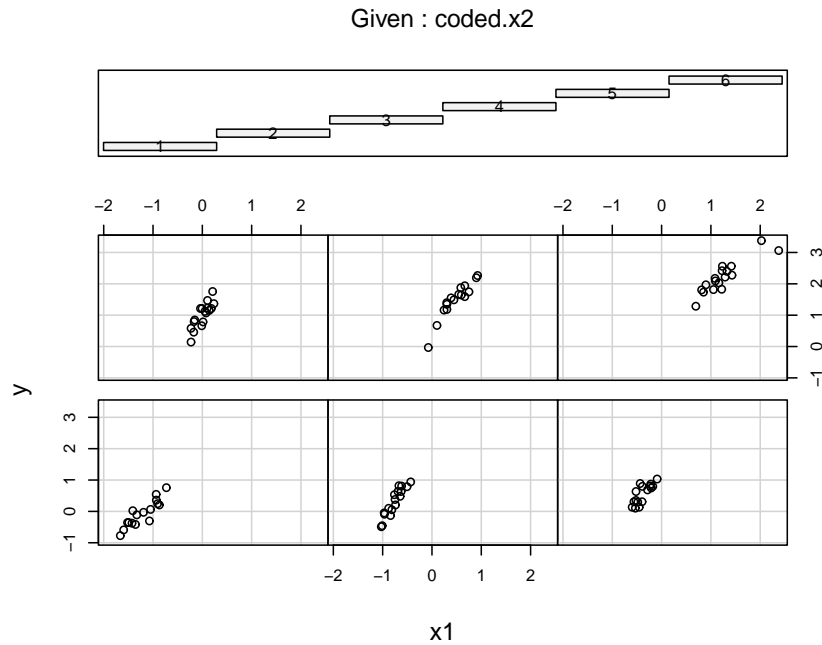


Figure 3.4: Coplot of  $Y$  versus  $X_1$ , with panels determined by the value of  $X_2$ .

The solutions to these equations, which are denoted by  $\hat{\beta}$ , are called the *least squares estimates* of the regression coefficients. The error variance  $\sigma^2$  is estimated by

$$\hat{\sigma}^2 = \frac{1}{n - k - 1} \sum_{i=1}^n \left( y_i - \hat{\beta}_0 - \cdots - \hat{\beta}_k x_{ik} \right)^2.$$

Geometrically, we are finding the best fitting hyperplane in the  $k+1$  dimensional space occupied by the data: for two variables  $X_1$  and  $X_2$  and a response  $Y$ , Figure 3.5 illustrates the situation. The fitted plane is the one minimising the sum of squared vertical distances between the responses and the plane.

*The vertical distances between the responses and the fitted plane are called the **residuals**, and are given by the formula*

$$\text{Residual for } i\text{th case} = r_i = y_i - (\hat{\beta}_0 + \cdots + \hat{\beta}_k x_{ik}).$$

*The sum of squared residuals is called the **residual sum of squares**, and we get the estimate  $\hat{\sigma}^2$  by dividing the residual sum of squares by a number called the **residual degrees of freedom**, which is the difference between the number of cases and the number of coefficients. Note that the estimate  $\hat{\sigma}^2$  is (almost) the variance of the residuals. This estimate is also called the **residual mean square**.*

### 3.2.2 Calculating the estimates

The R function `lm` is the basic tool for producing the regression estimates. We illustrate the use of this function by an example.

**Example 1.** Table 3.1 below gives data on four variables for each of twenty cases. The variables are free fatty acid level (**ffa**), age (**age**), weight (**weight**) and skin fold thickness (**skinfold**). We want to construct a model for predicting free fatty acid from the other variables.

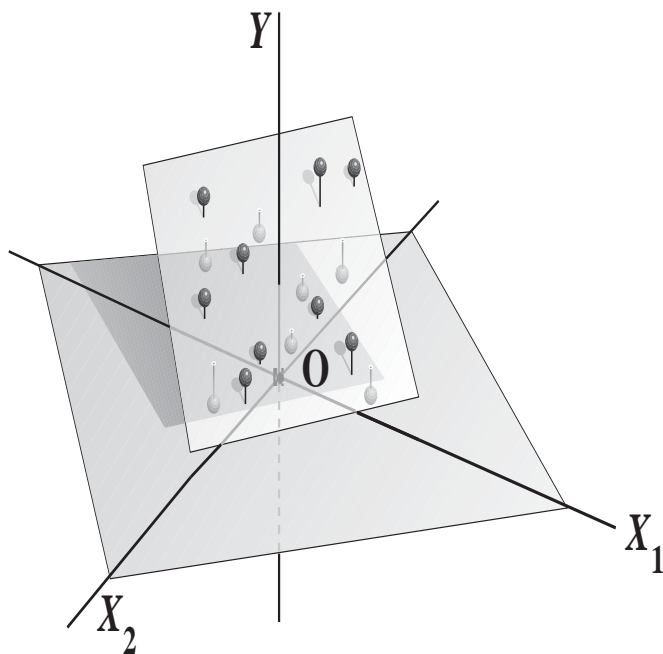


Figure 3.5: Geometric interpretation of the regression plane.

We assume the data has been read into R as described in Chapter 2, producing a data frame `fatty.txt`:

```
> fatty.df
      ffa age weight skinfold
1  0.759 105    67    0.96
2  0.274 107    70    0.52
3  0.685 100    54    0.62
4  0.526 103    60    0.76
5  0.859  97    61    1.00
6  0.652 101    62    0.74
7  0.349  99    71    0.76
8  1.120 101    48    0.62
9  1.059 107    59    0.56
10 1.035 100    51    0.44
11 0.531 100    80    0.74
12 1.333 101    57    0.58
```

Table 3.1: Fatty acid data.

FFA	Age	Weight	Skinfold	FFA	Age	Weight	Skinfold
0.759	105	67	0.96	0.274	107	70	0.52
0.685	100	54	0.62	0.526	103	60	0.76
0.859	97	61	1.00	0.652	101	62	0.74
0.349	99	71	0.76	1.120	101	48	0.62
1.059	107	59	0.56	1.035	100	51	0.44
0.531	100	80	0.74	1.333	101	57	0.58
0.674	104	58	1.10	0.686	99	58	0.72
0.789	101	54	0.72	0.641	110	66	0.54
0.641	109	59	0.68	0.355	109	64	0.44
0.256	110	76	0.52	0.627	111	50	0.60

```

13 0.674 104    58    1.10
14 0.686  99    58    0.72
15 0.789 101    54    0.72
16 0.641 110    66    0.54
17 0.641 109    59    0.68
18 0.355 109    64    0.44
19 0.256 110    76    0.52
20 0.627 111    50    0.60

```

Steps for creating a data frame:

1. Use `wordpad` or `notepad` to make a text file `xxxx.txt` to store the data
2. Type the variable names on the first line (no spaces in a name but spaces between names)
3. Type data for each case on a separate line
4. Make data frame `xxxx.df` by typing the R command
 

```
> xxxx.df<-read.table("xxxx.txt",header=TRUE)
```

The regression model is fitted by the function `lm`, which has (at least) two arguments

1. the model to be fitted, expressed in a special R form, which in the case of regression is  $y \sim x_1 + x_2 + \dots + x_k$  where  $y$  is the name of the response variable and  $x_1, \dots, x_k$  are the names of the explanatory variables,
2. the name of the data frame, expressed in the form `data=xxxx.df`.

The function name `lm` means “linear model”, i.e. a model where the mean is a linear function of the regression coefficients. The function returns a “linear model object” which contains information about the fit. For our free fatty acid data, we do the regression by typing

```
> fatty.lm<-lm(ffa~age+weight+skinfold,data=fatty.df)
```

storing our results in the linear model object `fatty.lm`. We can look at the results using the function `summary`:

```
> summary(fatty.lm)
```

Call:

```
lm(formula = ffa ~ age + weight + skinfold) [1]
```

Residuals:

```

      Min       1Q   Median       3Q      Max
-0.23163 -0.16706 -0.03901  0.11602  0.49567 [2]

```

Coefficients:

```

              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.95777    1.40138   2.824  0.01222 *
age          -0.01912    0.01275  -1.499  0.15323
weight       -0.02007    0.00613  -3.274  0.00478 **
skinfold     -0.07788    0.31377  -0.248  0.80714
--- [3]

```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.2224 on 16 degrees of freedom [4]
```

```
Multiple R-Squared: 0.4893, Adjusted R-squared: 0.3936
```

```
F-statistic: 5.11 on 3 and 16 degrees of freedom, p-value: 0.0114
```

### 3.2.3 Interpreting the output

The parts of this output are:

- [1] The model being fitted,
- [2] A 5-number summary of the distribution of the residuals,
- [3] Parameter estimates: Column 1 gives the estimates  $\hat{\beta}_0, \dots, \hat{\beta}_k$ , and column 2 gives the standard errors:

$$s.e.(\hat{\beta}_j) = (j, j \text{ element of } (X^T X)^{-1})^{\frac{1}{2}} \times \hat{\sigma}.$$

Column 3 gives the  $t$ -statistic  $t = \hat{\beta}_j / s.e.(\hat{\beta}_j)$  for testing  $\beta_j = 0$  (i.e. for testing the hypothesis that adding the  $j$ -th variable to the model is unnecessary, given all other variables are in the model). Column 4 gives the  $p$ -value of the test. The code shows the significance level of the  $p$ -value. ‘\*\*\*’ shows that the  $p$ -value is less than 0.001. ‘\*\*’ shows that the  $p$ -value is between 0.01 and 0.001, and so on. Small  $p$ -values suggest that the variable is needed in the model. Remember:

*Variables having small  $p$ -values are important in the regression, and help to explain/predict the response. Variables having a big  $p$ -value are either*

- *not related to the response, or*
- *related to the response, but **given the other variables are already in the model**, contribute nothing further to predicting the response. This is because they are related to other variables in the regression, so both are not needed.*

*Thus: we can't necessarily discard all the variables that have big  $p$ -values.*

- [4] Next comes the “Residual standard error  $\hat{\sigma}$ , together with the residual degrees of freedom  $n - k - 1$ , the multiple  $R^2$  (i.e. the ratio  $RegSS/TotalSS$ ), and the  $F$ -statistic for testing  $\beta_1 = \dots = \beta_k = 0$ : i.e. the ratio

$$F = \text{Regression mean square} / \text{Residual Mean square}$$

and together with its  $p$ -value. We see that in the present case, a null regression would give as extreme an  $F$  about 1% of the time. (This point will be expanded on in class.)

What do we conclude? Given that age and weight are in the model, skin fold thickness would appear to be unnecessary, since its  $p$ -value is quite large. There is also evidence that the variable **age** is not very important. Weight seems to be an important predictor, since its  $p$ -value is very small. The  $R^2$  is quite low at 49%, indicating that the variability of the data is quite large. The error standard deviation is estimated to be 0.2224, quite large compared to the size of the responses.

How good is the above analysis? It is only appropriate if the data really do cluster about a plane in four dimensions. How can we tell if this is true? This is not an easy question, but we discuss some methods of finding out in subsequent sections. For the moment, we shall proceed under the assumption that the model is OK.

### 3.2.4 Dropping and adding variables

The analysis above suggests that dropping skinfold thickness and possibly age from the model might be possible. We can confirm this by dropping these two variables and re-fitting the “submodel” consisting of weight alone. Dropping terms from the model always increases the residual sum of squares. However, if this increase is sufficiently small, we can drop the extra terms.

In 201/8 it was explained how to compare the fit of the *submodel*

$$\text{ffa} = \beta_0 + \beta_1 \text{weight} + e \tag{3.6}$$

with that of the *full model* which has all three explanatory variables. We get the submodel by setting the regression coefficients corresponding to age and skinfold simultaneously equal to zero. Thus we are testing the hypothesis “the coefficients of age and skinfold are simultaneously equal to 0” or in other words “the model with weight alone predicts the response as well as the model using all three variables”.

We do this by computing the residual sums of squares from the full model and the submodel (3.6) and evaluating the statistic

$$\frac{(\text{ResSS}_{(SUB)} - \text{Res SS}_{(FULL)})/d}{\text{residual mean square from full model}} \quad (3.7)$$

where  $d$  is the number of variables dropped (here  $d = 2$ .)

*Large values of the statistic (or equivalently small  $p$ -values) are evidence **against** the hypothesis of submodel adequacy. Thus a small  $p$ -value means the submodel is **not** adequate.*

We do the test in R by fitting the submodel, and then using the function `anova` to compare the two models and calculate the test:

```
> sub.lm<-lm(ffa~weight, data=fatty.df)
> anova(sub.lm,fatty.lm) # the arguments are the two models
Analysis of Variance Table

Model 1: ffa ~ weight
Model 2: ffa ~ age + weight + skinfold
  Res.Df Res.Sum Sq Df Sum Sq F value Pr(>F)
1      18   0.91007
2      16   0.79113  2  0.11895   1.2028 0.3261
```

We see that the ResSS for the full model is 0.79113, the ResSS for the submodel is 0.91007 and the value of the test statistic (3.7) is 1.2028 with a  $p$ -value of 0.3261. The large  $p$ -value indicates that the data provide no evidence against the hypothesis that the submodel is adequate.

Another way to fit the submodel is to “update” the full model: we could have typed

```
sub.lm<-update(fatty.lm,~-age-skinfold)
```

instead. This function takes two arguments, first the linear model object containing the full-model fit, and second an “updating formula” which indicates which terms are to be dropped.

Strictly speaking, we cannot rely too much on the  $p$ -value 0.3261 we calculated above. We decided to test if skinfold and age could be dropped from the model after we had fitted the full model. The test we performed assumes that we formulate the hypothesis to test *before* the data is inspected, so the fact that our choice of hypothesis was “data driven” may cause the  $p$ -value to be in error. A more accurate method of calculating the  $p$ -value is to use a technique known as the “bootstrap”, based on simulation. There is further discussion of this point in Section 3.8.

### 3.2.5 Fitting curves though data

Suppose we have a single explanatory variable  $x$ , but the relationship between the response  $y$  and  $x$  is non-linear. Under these circumstances, it may be appropriate to fit a model of the form  $y_i = f(x_i) + e_i$  where the function  $f$  is not linear. One possibility is to use a polynomial function, where  $f$  has terms in  $x$ ,  $x^2$ ,  $x^3$  and so on. For example, if the scatterplot of  $y$  versus  $x$  reveals a curve, we might use a quadratic polynomial of the form

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2.$$

We can fit such models using the `lm` function, by introducing a new variable `x2` containing the squares of the  $x$ 's, and fitting the model  $y \sim x + x2$ . If we want a cubic model, we can make an

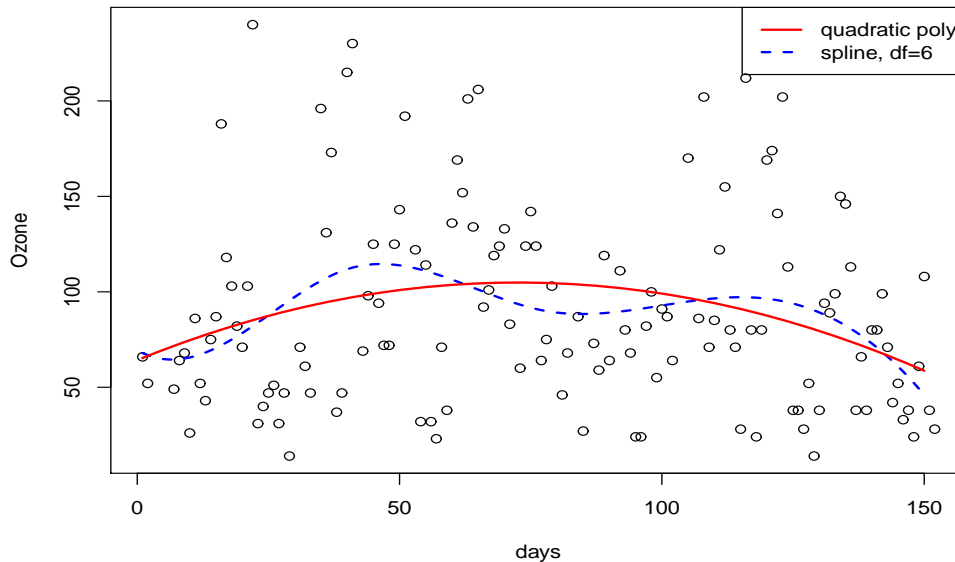


Figure 3.6: Fitting splines and polynomials.

additional variable `x3` containing the cubes of the  $x$ 's, and fit the model  $y \sim x + x^2 + x^3$ . We can fit polynomials of higher degree (i.e. with higher powers) using the `poly` function: the model  $y \sim \text{poly}(x, 2)$  fits a quadratic,  $y \sim \text{poly}(x, 3)$  fits a cubic,  $y \sim \text{poly}(x, 4)$  fits a fourth power polynomial and so on.

One problem with polynomials is that they are not very flexible: In order to fit the data well in one place, they may fit poorly in another. A solution to this problem is to divide the range of  $x$  into intervals and fit separate polynomials over each interval, making sure they join up in a smooth curve. Such piecewise polynomials are called *regression splines* and are a much more flexible version of polynomials. We fit them using the R function `bs`. We need to specify an argument `df`: this is related to the number of intervals we divide the range of  $x$  into. Some number between 5 and 10 is usually satisfactory. We illustrate the use of `poly` and `bs` in the following example.

**Example 2.** The data frame `stamford.df` contains daily ozone concentrations at Stamford, Connecticut from May 1, 1974 to Sept 30, 1974. These data has been extensively analysed by Chambers et. al. (1983). The data frame is part of the `R330` package, and contains two variables: `days`, the day the reading was taken, in days elapsed since May 1, 1974, and `ozone`, the ozone concentration in parts per billion. A plot of these data is shown in Figure 3.6. To fit a quadratic through these data, we can use the code

```
poly.fit = lm(ozone~poly(days,2), data = stamford.df)
```

The fitted quadratic is shown as a solid line in Figure 3.6. The difficulty with the quadratic fit is that the quadratic cannot capture the dip in the middle of the day range. A better solution is to use a higher degree polynomial or a spline. Using a spline, with `df` equal to 6, say we get

```
library(splines)
spline.fit = lm(ozone~bs(days,df=6), data = stamford.df)
```

This fit is also shown in Figure 3.6, and captures the dip in the middle better. Note that we have to say `library(splines)` to load the function `bs` into R.



### 3.3 Prediction

In the Introduction, we mentioned that one of the main uses of regression was prediction of the response, given the values of the explanatory variables. In this section we discuss how to do this.

Suppose we have a set of  $x$ -values  $x_1, \dots, x_k$  and we want to predict the  $y$ -value associated with these  $x$ -values. The problem is usually expressed in two ways

- (a) Predict the average value of the response for this set of  $x$ -values (i.e. estimate  $\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$ )
- (b) Predict the actual response of an individual case with this set of  $x$ -values.

In both cases the predictor is the same, but the prediction intervals are different. A natural estimate of the average value of the response is obtained by substituting the estimates  $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$  for the true  $\beta$ 's in the expression above. This results in the estimate  $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_k x_k$ . A convenient shorthand for this is to use vector notation: write  $x$  for the vector  $(1, x_1, \dots, x_k)^T$ . Then the estimate can be written  $x^T \hat{\beta}$ .

To predict a future value  $y = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k + e$ , we need to add an estimate of the error  $e$  to the estimate of  $\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$ . All that we know is that the error has mean 0. Thus we predict the error as 0, and our predicted value of  $y$  is  $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_k x_k + 0 = x^T \hat{\beta}$ , the same as before. We will use the term “predictor” to mean this common value  $x^T \hat{\beta}$ .

To get a confidence interval for the average value of the response, we need to work out the standard error of the predictor. Using techniques taught in STATS 310, this turns out to be the quadratic form

$$\text{s.e. predictor} = \hat{\sigma} \sqrt{x^T (X^T X)^{-1} x}.$$

Thus the 95% confidence interval is

$$x^T \hat{\beta} \pm \hat{\sigma} \sqrt{x^T (X^T X)^{-1} x} \times t,$$

where  $t = t_{n-k-1}(0.025)$ , the 2.5% point of the  $t$ -distribution with  $n - k - 1$  degrees of freedom.

A 95% prediction interval for  $y$  is an interval of the form  $\hat{y} - A \leq y \leq \hat{y} + A$  where  $A$  is chosen so that  $\Pr(\hat{y} - A \leq y \leq \hat{y} + A) = 0.95$ . The value of  $A$  turns out to be  $\hat{\sigma} \sqrt{1 + x^T (X^T X)^{-1} x} t_{n-k-1}$ , so the prediction interval is

$$x^T \hat{\beta} \pm \hat{\sigma} \sqrt{1 + x^T (X^T X)^{-1} x} t_{n-k-1} \left( \frac{\alpha}{2} \right).$$

We can write the formulae above in terms of the standard error of the predictor. For the confidence interval we get

$$\text{predictor} \pm \text{s.e. predictor} \times t$$

and for the prediction interval we get

$$\text{predictor} \pm \sqrt{\hat{\sigma}^2 + (\text{s.e. predictor})^2} \times t.$$

To compute the prediction intervals, we use the function `predict`.

**Example 3.** We illustrate the calculation of these intervals with some data on moisture evaporation, which are shown in Table 3.2. The data are included in the R330 package as the data frame `evap.df`. In this example we use a subset of the variables in the data frame. The variables are `avat`, the average air temp over a 24 hour period ( $\times 10$ ); `avh`, the average relative humidity over a 24 hour period ( $\times 10$ ); `wind`, the average wind speed over a 24 hour period ( $\times 100$ ) and `evap`, the amount of evaporation over a 24 hour period (units unknown).

The idea is to predict the amount of evaporation from the temperature, humidity and wind speed. Suppose our data are stored in a data frame `moisture.df`, containing the variables described above. As usual, we use `lm` to compute the basic regression quantities:

```
> moisture.lm<-lm(evap~avat+avh+wind,data=evap.df)
```

Table 3.2: Moisture evaporation data.

Obs	Temper- ature	Humid- ity	Wind Speed	Evapour- ation
1	151	398	273	30
2	159	345	140	34
3	152	388	318	33
4	158	406	282	26
5	180	379	311	41
6	147	478	446	4
7	159	462	294	5
8	159	464	313	20
9	195	430	455	31
10	206	406	604	38
11	208	393	610	43
12	211	385	520	47
13	211	405	663	45
14	201	392	467	45
15	167	448	184	11
16	162	436	177	10
17	173	392	173	30
18	177	392	76	29
19	169	398	72	23
20	170	431	183	16
21	196	379	76	37
22	199	393	230	50
23	204	394	193	36
24	201	386	400	54
25	206	389	339	44
26	208	370	172	41
27	214	396	238	45
28	210	380	118	42
29	207	365	93	50
30	202	357	269	48
31	173	418	128	17
32	168	420	423	20
33	189	399	415	15
34	210	389	300	42
35	208	384	193	44
36	215	389	195	41
37	198	380	215	49
38	196	354	185	53
39	211	364	466	53
40	198	405	399	21
41	197	447	232	1
42	205	380	275	44
43	209	379	166	44
44	208	372	189	46

```
> summary(moisture.lm)

Call:
lm(formula = evap ~ avat + avh + wind, data = evap.df)

Residuals:
    Min       1Q   Median       3Q      Max
-20.8211  -2.1270   0.1073   3.2553  14.7833

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 123.901800  24.624411   5.032 9.60e-06 ***
avat         0.222768   0.059113   3.769 0.000506 ***
avh        -0.342915   0.042776  -8.016 5.31e-10 ***
wind         0.015998   0.007197   2.223 0.031664 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.69 on 42 degrees of freedom
Multiple R-Squared:  0.805,    Adjusted R-squared:  0.7911
F-statistic:  57.8 on 3 and 42 degrees of freedom,    p-value: 5.884e-15
```

Suppose we want to make three predictions:

1. At avat=200, avh= 400, wind=300;
2. At avat=210, avh= 400, wind=350;
3. At avat=220, avh= 400, wind=400.

First we need to make a new data frame to hold these new values:

```
> new.df<-data.frame(avat=c(200,210,220),avh=c(400,400,400),
                     wind=c(300,350,400))
```

We then use `predict` to calculate the confidence intervals:

```
> predict(moisture.lm, new.df, interval="confidence")
      fit      lwr      upr
1 36.08883 33.74476 38.43290
2 39.11644 36.01729 42.21558
3 42.14405 38.07332 46.21477
```

For the prediction intervals, we type

```
>predict(moisture.lm, new.df, interval="prediction")
      fit      lwr      upr
1 36.08883 22.38520 49.79246
2 39.11644 25.26366 52.96922
3 42.14405 28.04207 56.24602
```

Note that the prediction intervals are wider.

*Use the function `predict` to calculate prediction intervals.*

## 3.4 Diagnosing model faults

Having fitted our model, we need to check that the model we have chosen is appropriate for the data. If these checks reveal any deficiencies, we need to modify the model in order to make it describe the data better. Thus we repeatedly go through the “modelling cycle” of fitting, checking and revising our models. We have already discussed fitting in Section 3.2.2. In this section we discuss model checking, and in Section 3.5 we show how to proceed if the checks indicate some of the model assumptions do not hold.

### 3.4.1 Checking assumptions

Imagine that the data cluster about a surface, the *true regression surface*. The regression model assumptions are

- The relationship between the response and the explanatory variables is linear (i.e. the true regression surface is a plane rather than a curved surface). This is the most important assumption. If we try to describe the regression surface with a flat plane when in fact it is a curved surface, we will not be getting very far.
- The errors are independent, normal, and have constant variance. These are secondary assumptions, but if they do not hold, our calculation of standard errors and  $p$ -values may be seriously in error.

We can’t observe the errors since we don’t know the true regression surface, but since

$$\begin{aligned} e_i &= y_i - \beta_0 - \dots - \beta_k x_{ik} \\ &\approx y_i - \hat{\beta}_0 - \dots - \hat{\beta}_k x_{ik} \\ &= \text{residual for the } i\text{th case} \\ &= r_i \text{ say,} \end{aligned}$$

the residuals  $r_i$  should reflect the underlying properties of the errors provided the true surface is a plane. The main tools for checking the regression assumptions are various plots of residuals. Specifically,

**Curved regression surface:** *The linearity assumption is checked by plots of residuals versus fitted values, plots of residuals versus explanatory variables, partial regression plots and gam plots.*

**Normality of errors:** *Normality of errors is checked by a normal plot of the residuals.*

**Outliers:** *The presence of outliers and high influence points is detected by normal plots, plotting residuals versus fitted values and “influence statistics”.*

**Independence:** *Lack of independence is detected by examining the residuals for evidence of serial correlation by e.g. plotting  $r_i$  vs  $r_{i-1}$ .*

**Constant variance:** *Constancy of variance is checked by examining the res/fitted plots for “funnel effects”, and by examining smoothed plots of squared residuals.*

We will illustrate these methods with examples and outline the necessary R code to produce the plots described above.

### Detecting a curved regression surface

Possible non-linearity in the relationship between  $y$  and the  $x$ ’s can be detected by

- Plotting residuals versus fitted values
- Plotting residuals versus  $x$ 's
- Partial regression plots
- Gam plots

We describe each of these in turn:

#### Plots of residuals versus fitted values:

We have defined residuals above. Also important are the *fitted values*  $\hat{y}_i$  defined by

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_k x_{ik}.$$

In matrix terms, the vector  $\hat{Y}^T = (\hat{y}_1, \dots, \hat{y}_n)$  is given by the equation

$$\hat{Y} = X(X^T X)^{-1} X^T Y.$$

We can extract these from the “linear model object” returned by `lm`, by using the functions `residuals` and `fitted.values`. To get the residuals, type

```
> res<-residuals(fatty.reg)
```

and to get the fitted values type

```
> pred<-fitted.values(fatty.reg)
```

We could produce a plot of residuals versus fitted values by means of the function `plot`, as discussed in Chapter 1. However, the `plot` function has many uses, and can take many different kinds of argument, not just a pair of vectors. The function usually produces a plot appropriate to the argument we feed it. For example, if we use a `lm` object as its argument, we get a page of plots, including a plot of residuals versus fitted values, and a normal plot of residuals.

**Example 4.** We illustrate this point using some tyre abrasion data. This is a data set with three variables, `abloss` (abrasion loss, the dependent variable), `hardness` and `tensile` (hardness and tensile strength, the explanatory variables). The idea is to model the amount of tyre lost in a standard abrasion test, as a function of the hardness and tensile strength of the rubber. The data are given in Table 3.3. The data are part of the R330 package, in the data frame `rubber.df`.

We need to make the data available, do the regression, and then plot the residuals versus the fitted values:

```
> library(R330) # do this if the R330 library is not loaded
> data(rubber.df) # make data frame available
> rubber.lm<-lm(abloss~hardness+tensile,data=rubber.df)
> par(mfrow=c(2,2)) # plot 4 plots on the same page
> plot(rubber.lm)
```

The result is shown in the top four panels of Figure 3.7. We get a plot of residuals versus fitted values and a Normal Q-Q plot. We also get a Scale-Location plot and a plot of residuals versus leverage (see Section 3.4.1).

Plotting residuals versus the explanatory variables can also be useful. If some of the plots show a curved relationship, and others do not, then we get an idea of which explanatory variables to transform. A curved plot indicates a non-linear relationship between the explanatory variables and the response, indicating that some or all of the variables should be transformed.

**Example 5.** To plot residuals against `hardness` and residuals against `tensile`, we type

```
> par(mfrow=c(1,2)) # set up 2 plots per page
> res = residuals(rubber.lm)
> plot(rubber.df$hardness,res)
> abline(0,0)
```

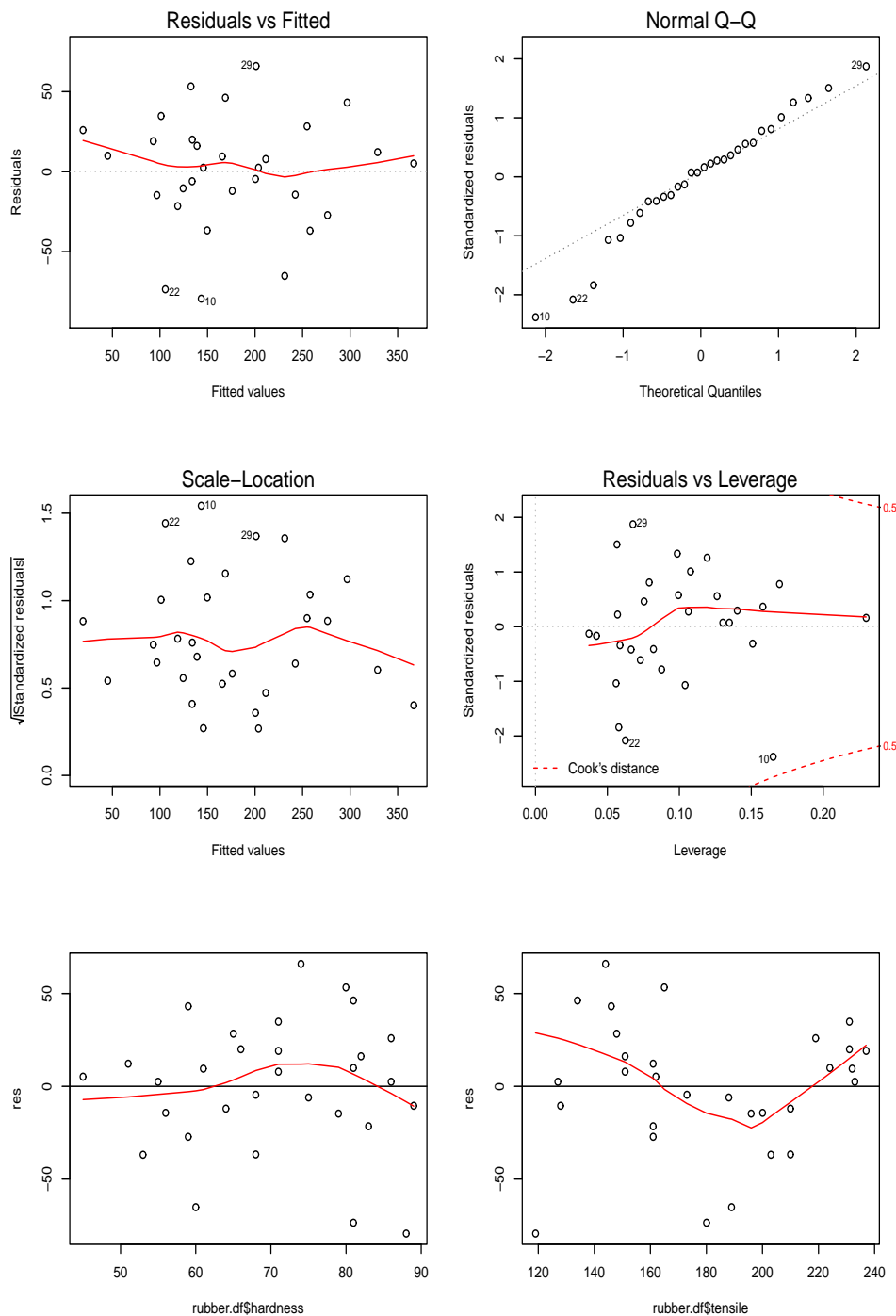


Figure 3.7: Residual plots produced by the `plot` function, plus plots of residuals versus explanatory variables.

Table 3.3: Tyre abrasion loss data.

Hardness	Tensile strength	Abrasion loss	Hardness	Tensile strength	Abrasion loss
45	162	372	55	233	206
61	232	175	66	231	154
71	231	136	71	237	112
81	224	55	86	219	45
53	203	221	60	189	166
64	210	164	68	210	113
79	196	82	81	180	32
56	200	228	68	173	196
75	188	128	83	161	97
88	119	64	59	161	249
71	151	219	80	165	186
82	151	155	89	128	114
51	161	341	59	146	340
65	148	283	74	144	267
81	134	215	86	127	148

```
> plot(rubber.df$tensile,res)
> abline(0,0)
```

The plots are shown in the bottom two panels in Figure 3.7. Since the plot for hardness is reasonably straight but that for tensile strength is curved, we would try transforming tensile strength.

One difficulty in interpreting these plots is to visually pick up the curves that indicate the need for transformation. It helps to draw in a smooth line through the points i.e. to “smooth” the plot, as in the plot of residuals versus fitted values. This is done by inserting the R commands

```
> lines(lowess(res~rubber.df$hardness))
```

and

```
> lines(lowess(res~rubber.df$tensile))
```

after the appropriate plot command. See Section 3.10 for more on smoothing plots.

#### Added variable plots:

These are plots that are useful for detecting non-linear relationships, but also have many other uses. A common problem in regression is to decide if a particular explanatory variable ( $x_k$  say) adds anything to the prediction of the response. In the last section, we discussed how to judge this by means of  $t$ -tests. Partial residual plots shed further light on this problem, and can help us decide if the variable should be dropped, or whether it needs transforming to improve the fit.

We will want to include the variable  $x_k$  if its inclusion improves the fit. This may fail to happen in two ways. First, the variable may be completely unrelated to  $y$ , so including it cannot improve the model. Second, it may be related to the response, but given that all the other explanatory variables are already in the model, its extra predictive power may be negligible. This is because  $x_k$  is related to the other explanatory variables, so any predictive ability it may have has already been taken into account by the presence of the other variables.

The part of the response not accounted for by the other variables  $x_1, \dots, x_{k-1}$  is simply the residual we get when we regress the response on these variables. We want to see if there is any relationship between these residuals and the part of  $x_k$  not related to  $x_1, \dots, x_{k-1}$ . This unrelated part of  $x_k$  is best measured by the residuals we get when we regress  $x_k$  on  $x_1, \dots, x_{k-1}$ . Thus we want to examine the relationship between

- the residuals we get by regressing  $y$  on  $x_1, \dots, x_{k-1}$ , and
- the residuals we get by regressing  $x_k$  on  $x_1, \dots, x_{k-1}$ .

We do this by plotting the first set of residuals versus the second. The resulting plot is called the *added variable plot* for variable  $x_k$ . (An alternative name is an *partial regression plot*). We draw one plot for each explanatory variable. These plots have several interesting interpretations.

**First**, since residuals always have zero mean, a line fitted through the plot by least squares will always go through the origin. Moreover, the slope of this line is  $\hat{\beta}_k$ , the estimate of the regression coefficient  $\beta_k$  in the full regression using all the explanatory variables  $x_1, \dots, x_k$ .

**Second**, the amount of scatter about the least squares line reflects how important  $x_k$  is as a predictor. This is because the correlation  $r$  between the two sets of residuals is related to the  $t$ -statistic  $t$  for testing  $\beta_k = 0$  by the equation

$$t^2 = \frac{1}{n - k - 1} \frac{r^2}{1 - r^2}.$$

Thus the closer  $r$  to zero (i.e. the larger the scatter) the smaller  $t$  is, and so the less important  $x_k$  is as a predictor, provided the other variables are included in the model. Another expression for  $r$  is

$$\hat{\beta}_j / \sqrt{\frac{ResSS}{\sum (x_{ij} - \bar{x}_j)^2 (1 - R_j^2)} + \hat{\beta}_j^2}$$

where  $R_j^2$  is the multiple correlation coefficient (i.e. the  $R^2$ ) of the regression of  $x_j$  on the other  $x$ 's. If  $x_j$  is well predicted from the other  $x$ 's we tend to get a “football shaped plot” as the correlation is then close to zero since  $R_j^2$  is close to unity and the denominator is large.

**Third**, the residuals from the fit on the partial residual plot are the residuals from the original fit, so the plot can be used to identify points with big residuals.

**Fourth**, (this is the property most important for diagnostic purposes), if the variable  $x_k$  needs transforming, the plot will be curved rather than straight. To sum up

*If the partial regression plot for a variable shows a large amount of scatter, the variable does not need to be included in the regression, provided the other variables are retained. If the partial regression plot is straight, then the variable should be included untransformed. Finally, if the plot is curved, the variable should be transformed.*

**Example 6.** We can draw added variable plots for the rubber data, one for each explanatory variable, using the function `added.variable.plots`. Note that this function is not included in the regular version of R, but is part of the R330 package.

We draw the two plots for the rubber data (one for each independent variable) by typing

```
>rubber.lm<-lm(abloss~hardness+tensile,data=rubber.df)
>added.variable.plots(rubber.lm)
```

or, more simply

```
>added.variable.plots(abloss~hardness+tensile,data=rubber.df)
```

The plots are shown in Figure 3.8. Neither plot shows excessive scatter, so both variables are required in the model. The `hardness` plot is quite straight, indicating no need to transform this variable. On the other hand, the `tensile` plot is curved, so we need to transform `tensile`. What transformation should be used? Next, we discuss some ways of finding a suitable transformations.



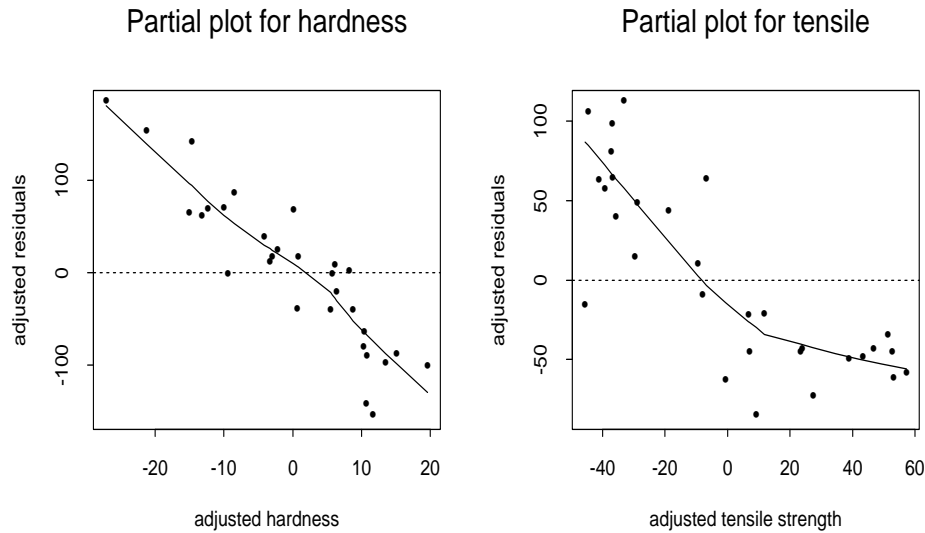


Figure 3.8: Partial regression plots for detecting non-linearity.

### Gam plots

Partial regression plots are useful, but they don't necessarily tell us the form of the transformation required. Gam plots are better for this. A generalisation of the usual regression model is to imagine that suitable transformations of the variables follow a linear model: i.e. there exists functions  $\phi_1, \dots, \phi_k$  such that

$$y = \phi_1(x_1) + \dots + \phi_k(x_k) + e_i.$$

This is equivalent to imagining that the variables when suitably transformed will follow the usual regression model (3.1). Choosing these functions by ad-hoc methods is difficult, except possibly in the case when  $k = 1$ , when we can use graphical methods and “straighten the plot” using the “ladder of powers”. In the late 1980's, a method called “generalised additive models” (hence the initials gam) was developed to handle the situation when  $k > 1$ . Roughly speaking, the idea is to choose the functions so that the  $R^2$  between the transformed variables is maximised, subject to the constraint that the functions are members of some particular class, usually that of “smooth functions”. Other possibilities are powers, logs, monotone functions etc. The actual calculations proceed iteratively, with one function being improved at each step. This is called the “backfitting algorithm”.

Thus if we have chosen functions  $\phi_2, \dots, \phi_k$  we can “improve”  $\phi_1$  by “smoothing” the plot of  $z = y - \phi_2(x_2) - \phi_3(x_3) - \dots - \phi_k(x_k)$  vs  $x_1$  as shown in Figure 3.9. The “smoothed” plot (the solid line) is then taken as  $\phi_1$ . This is now repeated, improving  $\phi_2$  in a similar manner. These iterations are continued until no further change is made. The method is useful in choosing transformations in regression, and can be carried out in R by means of the `gam` function.

The best way to use the gam method is to draw plots of the estimated functions  $\phi_1, \phi_2, \dots, \phi_k$ . We will leave the response variable untransformed at this stage. The idea is to use these optimal transformations to suggest simpler, more practical transformations such as powers, logs etc. We then use these simple transformations to transform and reanalyse the data.

The gam plots (one per explanatory variable) give us an idea which variables to transform: if the gam plot for a variable is straight, we leave that variable untransformed. If the plot for a particular variable is non-linear, the shape of the plot suggests the form of the transformation.

Thus, if the gam plot has the shape  $\curvearrowright$ , we can transform the corresponding variable with a

power less than one, and if the plot has shape  $\curvearrowleft$  we can transform with a power greater than one. If the plot seems quadratic or cubic, we can add squares or cubes in the appropriate variable, or use splines.

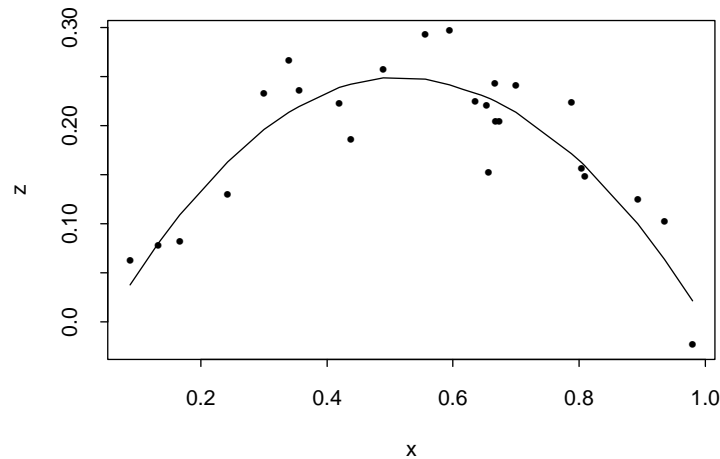


Figure 3.9: Updating functions in generalised additive models via smoothing.

**Example 7.** To illustrate the R code required to draw the plots, we use the rubber data again. To draw the gam plots, we type

```
> library(mgcv) # load the gam function
> gam.stuff<-gam(abloss~s(hardness)+s(tensile),data=rubber.df)
> par(mfrow=c(1,2)) # to plot two pictures in the window
> plot(gam.stuff)
> par(mfrow=c(1,1)) # reset plotting window to one plot
```

The plots are shown in Figure 3.10. These plots also indicate that **hardness** should remain untransformed (because the plot is relatively straight), but that **tensile** needs transforming to model the “kinks”. One possibility is to add a fourth degree polynomial in **tensile** to the regression equation, since the gam plot resembles a 4th degree polynomial. This amounts to adding terms  $\text{tensile}^2$ ,  $\text{tensile}^3$  and  $\text{tensile}^4$  to the model. We can do this in R using the `poly` function:

```
> poly.lm<-lm(abloss~hardness+poly(tensile,4),data=rubber.df)
> summary(poly.lm)
```

Call:

```
lm(formula = abloss ~ hardness + poly(tensile, 4), data = rubber.df)
```

Residuals:

Min	1Q	Median	3Q	Max
-43.784	-15.194	1.215	14.392	53.978

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	615.4012	29.2893	21.011	< 2e-16 ***
hardness	-6.2614	0.4124	-15.182	8.35e-14 ***
poly(tensile, 4)1	-264.4043	24.6171	-10.741	1.20e-10 ***
poly(tensile, 4)2	23.6269	24.8947	0.949	0.352043
poly(tensile, 4)3	119.9408	24.1991	4.956	4.64e-05 ***
poly(tensile, 4)4	-91.6965	23.2722	-3.940	0.000613 ***

---

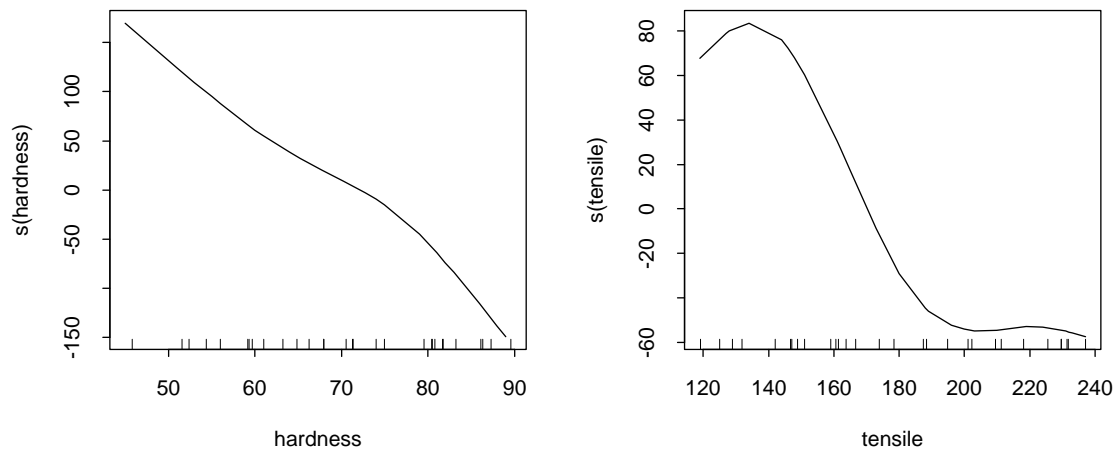


Figure 3.10: Gam plots for detecting non-linearity.

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 23.25 on 24 degrees of freedom

Multiple R-Squared: 0.9423, Adjusted R-squared: 0.9303

F-statistic: 78.46 on 5 and 24 degrees of freedom, p-value: 4.508e-14

The addition of fourth degree polynomial terms in the model has increased the  $R^2$  from 84% to 94%. We could also try splines.

### Checking the normality

We check the normality of the errors by means of a normal plot of the residuals. The plot is one of the standard diagnostic plots produced by the `plot` function as described in the previous section. We can also perform a formal test of normality by calculating the value of the Weisberg-Bingham test (this is a variant of the Shapiro-Wilk test discussed in STATS 201/8). We calculate the test statistic by computing the correlation of the values in the normal plot. For example, to check the normality of the rubber data, we can type

```
> res<-residuals(rubber.lm)
> n<-length(res)
> n
[1] 30
> normal.quantiles<-qnorm(((1:n)-0.5)/n) # ((1:n)-0.5)/n is (i-0.5)/n
> cor(normal.quantiles, sort(res))
[1] 0.985589
```

Table 3.4 contains percentage points of the Weisberg-Bingham statistic. From the table, we see that the  $p$ -value for the test of the normality hypothesis is greater than 0.10, so there is no evidence against the normality hypothesis. Alternatively, we can use a simulation approach to compute the  $p$ -value associated with the correlation above, using the function `WB.test` in the R330 package:

```
> WB.test(rubber.lm)
WB test statistic = 0.986
p = 0.5
```

This is not too surprising, given that the normal plot of the rubber data residuals in Figure 3.7 is quite straight. The normal assumption is usually not too crucial, although a “long-tailed” error distribution can result in outliers that can have an effect on the estimation of the regression plane.

Table 3.4: Percentage points of the Weisberg-Bingham statistic.

Percentage point			
$n$	0.01	0.05	0.10
30	0.902	0.931	0.942
35	0.919	0.943	0.952
40	0.922	0.945	0.956
45	0.935	0.952	0.960
50	0.935	0.953	0.963
55	0.940	0.958	0.965
60	0.946	0.963	0.968
65	0.948	0.965	0.971
70	0.952	0.967	0.972
75	0.956	0.969	0.973
80	0.958	0.970	0.975
85	0.961	0.972	0.977
90	0.962	0.973	0.978
100	0.967	0.976	0.980
150	0.977	0.982	0.985
200	0.981	0.986	0.988
250	0.984	0.989	0.990

See Section 3.4.2 for more about outliers. If the data are too far from normal, the  $p$ -values in the regression table can be in error. Moreover, prediction intervals will have the wrong width.

### Investigating the independence

We will discuss two approaches to detecting dependence in the errors: the Durbin-Watson test and graphical checks involving various plots of residuals. Both approaches assume that the dependence is caused by the observations being collected sequentially in time.

The Durbin-Watson test is designed to detect the following error structure. The errors  $e_i$  are said to be a *first-order autoregressive process* if they are of the form

$$e_{i+1} = \rho e_i + a_{i+1} \quad i = 1, 2, \dots$$

where the  $a_i$  are i.i.d. Normal  $(0, \sigma_a^2)$  and  $\rho$  is a parameter (the autocorrelation coefficient) between  $\pm 1$ . Note that  $\rho = 0$  corresponds to the errors being independent, in which case  $\sigma^2 = \sigma_a^2$ . If  $\rho \neq 0$ , then the errors are not independent.

*If the errors are not independent, least squares is not the best method of estimating the regression coefficients, and the standard errors computed by the regression program will be wrong.*

If  $\rho > 0$ , we say that the errors are positively serially correlated, while if  $\rho < 0$ , they are negatively serially correlated. Negative serial correlation is very uncommon, but positive serial correlation is very common in time series data (data that have been collected sequentially in time). Such data tend to have errors of the same sign in successive observations, and the presence of serial correlation can be detected informally by plotting each residual against the one preceding. If a linear pattern is visible, we have serial correlation, since a residual should be roughly proportional to the residual of the previous observation. See the top panels of Figure 3.11.

As well as the plot of residuals versus previous residuals, a plot of residuals versus time order can reveal positive dependence, which typically is indicated by long stretches where all residuals

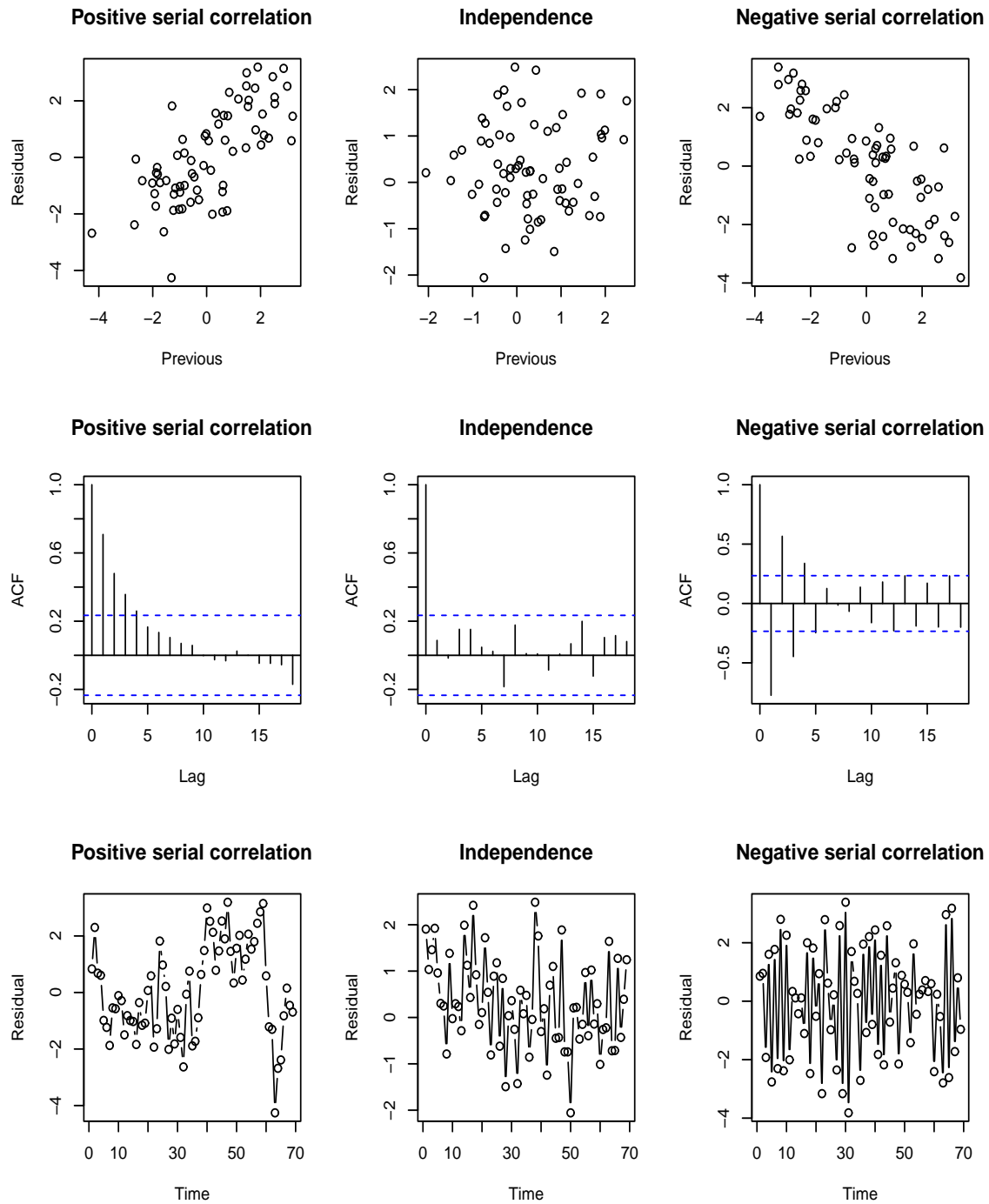


Figure 3.11: Appearance of plots for detecting serial correlation of residuals.

Table 3.5: Monthly sales and advertising for 36 months. Read downward, left to right.

Month	Sales	Advert- ising	Month	Sales	Advert- ising
1	12.0	15	19	30.5	33
2	20.5	16	20	28.0	62
3	21.0	18	21	26.0	22
4	15.5	27	22	21.5	12
5	15.3	21	23	19.7	24
6	23.5	49	24	19.0	3
7	24.5	21	25	16.0	5
8	21.3	22	26	20.7	14
9	23.5	28	27	26.5	36
10	28.0	36	28	30.6	40
11	24.0	40	29	32.3	49
12	15.5	3	30	29.5	7
13	17.3	21	31	28.3	52
14	25.3	29	32	31.3	65
15	25.0	62	33	32.2	17
16	36.5	65	34	26.4	5
17	36.5	46	35	23.4	17
18	29.6	44	36	16.4	1

are positive (or stretches of negative residuals). Negative dependence is rare, but is revealed by a plot where positive and negative residuals alternate. We can also plot the autocorrelation function of the residuals, which shows the correlation of the residuals with the residuals lagged by various amounts. If the autocorrelation function has values close to zero, there is no indication of dependence. These plots are also illustrated in Figure 3.11.

We can also perform a formal test of the independence (i.e. we can test if  $\rho = 0$ ). The parameter  $\rho$  is estimated by  $\hat{\rho} = \sum_{i=2}^n r_i r_{i-1} / \sum_{i=1}^n r_i^2$  and the test statistic for the Durbin-Watson test is

$$DW = \sum_{i=2}^n (r_i - r_{i-1})^2 / \sum_{i=2}^n r_i^2 \approx 2(1 - \hat{\rho}).$$

Thus we reject the hypothesis of independent errors if  $\hat{\rho}$  is too far from 0 (i.e. if  $DW$  is too far from 2). The exact significance points depend on  $X$ , and are not generally used. Instead, approximate significance points are used.

To test  $\rho = 0$  against an alternative of  $\rho > 0$ , the exact test would reject  $\rho = 0$  at level  $\alpha$  if  $DW < D_\alpha$ , where  $D_\alpha$  depends on  $X$ <sup>1</sup>. However, it is possible to find bounds

$$d_L(\alpha) < D_\alpha < d_U(\alpha)$$

which do not depend on  $X$ . These values are tabulated in Table 3.6. To test for independence, we

- reject  $\rho = 0$  if  $DW < d_L(\alpha)$
- accept  $\rho = 0$  if  $DW > d_U(\alpha)$ .

Note that small values of  $DW$  imply large positive values of  $\hat{\rho}$  ie high degrees of serial correlation.

**Example 8.** Consider the data in Table 3.5, which records monthly sales and advertising expenditure over 36 months . A suitable model to fit is

$$\text{Sales} = \beta_0 + \beta_1 \text{advertising} + \beta_2 \text{advertising for previous month}.$$

<sup>1</sup>The phrase “reject  $\rho = 0$  at level  $\alpha$  if  $DW < D_\alpha$  means that the value of  $DW$  that would give a  $p$ -value of  $\alpha$  is  $D_\alpha$ .

Table 3.6: Table of the Durbin-Watson percentage points

$n$	$k = 1$		$k = 2$		$k = 3$		$k = 3$		$k = 5$	
	$d_L$	$d_U$	$d_L$	$d_U$	$d_L$	$d_U$	$d_L$	$d_U$	$d_L$	$d_U$
15	1.08	1.36	0.95	1.54	0.82	1.75	0.69	1.97	0.56	2.21
16	1.10	1.37	0.98	1.54	0.86	1.73	0.74	1.93	0.62	2.15
17	1.13	1.38	1.02	1.54	0.90	1.71	0.78	1.90	0.67	2.10
18	1.16	1.39	1.05	1.53	0.93	1.69	0.82	1.87	0.71	2.06
19	1.18	1.40	1.08	1.53	0.97	1.68	0.86	1.85	0.75	2.02
20	1.20	1.41	1.10	1.54	1.00	1.68	0.90	1.83	0.79	1.99
21	1.22	1.42	1.13	1.54	1.03	1.67	0.93	1.81	0.83	1.96
22	1.24	1.43	1.15	1.54	1.05	1.66	0.96	1.80	0.86	1.94
23	1.26	1.44	1.17	1.54	1.08	1.66	0.99	1.79	0.90	1.92
24	1.27	1.45	1.19	1.55	1.10	1.66	1.01	1.78	0.93	1.90
25	1.29	1.45	1.21	1.55	1.12	1.66	1.04	1.77	0.95	1.89
26	1.30	1.46	1.22	1.55	1.14	1.65	1.06	1.76	0.98	1.88
27	1.32	1.47	1.24	1.56	1.16	1.65	1.08	1.76	1.01	1.86
28	1.33	1.48	1.26	1.56	1.18	1.65	1.10	1.75	1.03	1.85
29	1.34	1.48	1.27	1.56	1.20	1.65	1.12	1.74	1.05	1.84
30	1.35	1.49	1.28	1.57	1.21	1.65	1.14	1.74	1.07	1.83
31	1.36	1.50	1.30	1.57	1.23	1.65	1.16	1.74	1.09	1.83
32	1.37	1.50	1.31	1.57	1.24	1.65	1.18	1.73	1.11	1.82
33	1.38	1.51	1.32	1.58	1.26	1.65	1.19	1.73	1.13	1.81
34	1.39	1.51	1.33	1.58	1.27	1.65	1.21	1.73	1.15	1.81
35	1.40	1.52	1.34	1.58	1.28	1.65	1.22	1.73	1.16	1.80
36	1.41	1.52	1.35	1.59	1.29	1.65	1.24	1.73	1.18	1.80
37	1.42	1.53	1.36	1.59	1.31	1.66	1.25	1.72	1.19	1.80
38	1.43	1.54	1.37	1.59	1.32	1.66	1.26	1.72	1.21	1.79
39	1.43	1.54	1.38	1.60	1.33	1.66	1.27	1.72	1.22	1.79
40	1.44	1.54	1.39	1.60	1.34	1.66	1.29	1.72	1.23	1.79
45	1.48	1.57	1.43	1.62	1.38	1.67	1.34	1.72	1.29	1.78
50	1.50	1.59	1.46	1.63	1.42	1.67	1.38	1.72	1.34	1.77
55	1.53	1.60	1.49	1.64	1.45	1.68	1.41	1.72	1.38	1.77
60	1.55	1.62	1.51	1.65	1.48	1.69	1.44	1.73	1.41	1.77
65	1.57	1.63	1.54	1.66	1.50	1.70	1.47	1.73	1.44	1.77
70	1.58	1.64	1.55	1.67	1.52	1.70	1.49	1.74	1.46	1.77
75	1.60	1.65	1.57	1.68	1.54	1.71	1.51	1.74	1.49	1.77
80	1.61	1.66	1.59	1.69	1.56	1.72	1.53	1.74	1.51	1.77
85	1.62	1.67	1.60	1.70	1.57	1.72	1.55	1.75	1.52	1.77
90	1.63	1.68	1.61	1.70	1.59	1.73	1.57	1.75	1.54	1.78
95	1.64	1.69	1.62	1.71	1.60	1.73	1.58	1.75	1.56	1.78
100	1.65	1.69	1.63	1.72	1.61	1.74	1.59	1.76	1.57	1.78

Note that we have included the previous month's sales as an explanatory variable. This introduces some problems, as we don't know the previous month's sales for the first observation. In effect we lose one observation, since there is no data for the previous spend corresponding to the first observation. Assuming the data are in a data frame `ad.df`, with variables `sales`, `advertising spend` and previous month's advertising spend, we fit the model with the code

```
> advert.lm<-lm(sales~spend+prev.spend, data=ad.df)
> summary(advert.lm)
Call:
lm(formula = sales ~ spend + prev.spend)
Residuals:
    Min       1Q   Median       3Q      Max
-7.7903 -2.1982  0.1199  2.9635  7.2536
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 15.60361    1.34531   11.599 5.35e-13 ***
spend        0.14242    0.03518    4.049 0.000305 ***
prev.spend   0.16651    0.03606    4.617 6.03e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 3.652 on 32 degrees of freedom
Multiple R-Squared:  0.6392,    Adjusted R-squared:  0.6167
F-statistic: 28.35 on 2 and 32 DF,  p-value: 8.233e-08
```

Note also that the inclusion in the model of the previous month's advertising seems necessary, since this term has a very small  $p$ -value (0.00006).

To calculate the residuals and draw the plot, we type

```
> resids<-residuals(advert.lm)
> plot(resids[-35],resids[-1],xlab="lagged residuals",ylab="residuals")
```

The plot is shown in Figure 3.12.

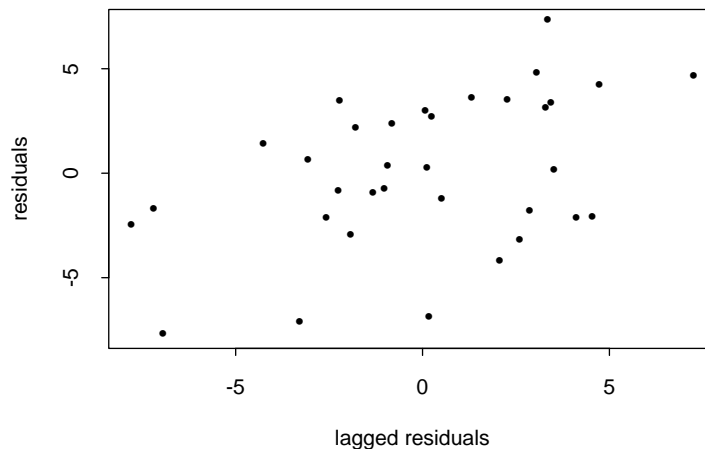


Figure 3.12: Plot of residuals versus lagged residuals.

The slight but unmistakable trend shows a definite suggestion of serial correlation. To calculate the estimate  $\hat{\rho}$  and do the formal test, we type

```
> rho<- cor(resids[-1],resids[-35])
> rho
[1] 0.4450734
```



Table 3.7: Capital value and rental value (in \$NZ) for 96 properties.

Capital	Rental	Capital	Rental	Capital	Rental	Capital	Rental
61500	6656	124000	8528	170000	8500	225000	10400
67500	6864	125000	7904	170000	12688	225000	12480
75000	4992	125000	8500	170000	8736	231000	12896
75000	7280	126000	6240	170000	9568	240000	11648
76000	6656	127000	8944	170000	11232	240000	11648
77000	4576	130000	9776	174000	10400	240000	12064
80000	4992	130000	7696	175000	9152	240000	11024
81000	6656	135000	8320	178000	11856	240000	10192
81000	4784	135000	7280	189000	12272	241000	9152
85000	7072	135000	7488	190000	8320	244500	11232
90000	6240	135000	8736	191000	12064	250000	12500
90000	6240	140000	9152	194000	11232	262000	10192
90000	4576	140000	9360	195000	11024	270000	12480
94000	8736	140000	9568	200000	10608	270000	12896
94000	5200	140000	10816	200000	10000	275000	13312
95000	7904	145000	8320	200000	12272	289000	11648
102000	9568	145000	11232	200000	9360	295000	11232
104000	7904	148000	8320	200000	8320	300000	12480
106000	6656	151000	7904	200000	13312	300000	12896
110000	7072	155000	7488	200000	10400	303000	12272
115000	5824	163000	11024	200000	12480	304000	12272
115000	7904	165000	8528	208000	10400	310000	12480
120000	11440	165000	9152	210000	11232	315000	12896
121000	12064	165000	13312	214000	8528	325000	12580

```

> DW<- 2*(1-rho)
> DW
[1] 1.109853

```

The estimate of  $\rho$  is 0.445, and the Durbin-Watson statistic is 1.11. Noting that  $n = 35$ ,  $k = 2$  we see from the table in Figure 3.6 that  $d_L(0.05) = 1.34$ . We reject the hypothesis that  $\rho = 0$  at the 5% level<sup>2</sup>, and conclude that there is evidence of positive serial correlation.

### Detecting inhomogeneity of variance

The next type of departure from our assumptions occurs when the errors do not have the same variance, and this may be detected by the “funnel effect” in the plot of residuals versus fitted values. If there are repeated  $y$ -values for each set of  $x$ -values, we can also compute standard deviations of each set of observations, and plot them against the means, or, equivalently, plot log standard deviations versus log means. (Recall from 201/8 that the slope of a line fitted to this plot gives an idea of how to transform.) However, this is often not possible, as data sets where there are a reasonable number of data points for each set of  $X$ -values are not common. An alternative is to group the data into sets of points whose  $x$ -values are similar.

A better method is to plot the squares of the residuals versus fitted values, and “smooth” the plot either visually or by using a *scatterplot smoother*<sup>3</sup>. We illustrate this idea with some data from the Auckland City Council.

**Example 9.** The data in Table 3.7 show the capital value and annual rental value of 96 domestic properties in Auckland. The data are also included in the R330 package as the data frame `acc.df`. The City Council wished to explore the relationship between these to aid the formulation of a tax policy.

<sup>2</sup>In other words, the  $p$ -value is less than 0.05.

<sup>3</sup>More details on smoothing may be found in Section 3.10.

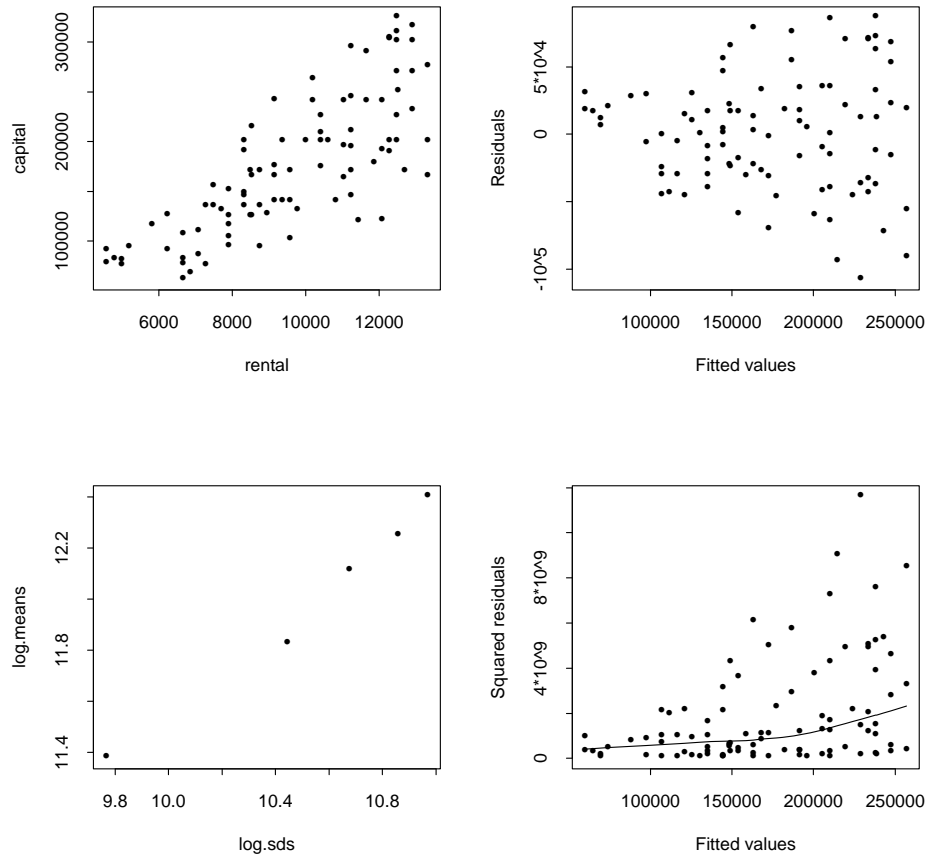


Figure 3.13: (i) Capital value versus rental value for 96 properties. (ii) Residuals versus fitted values for 96 properties. (iii) Log standard deviations versus log means for the five groups. (iv) Smoothed plot of squared residuals versus fitted values.

The ultimate aim was to predict the capital value from the rental value, so it is natural to let the capital value be the response and the rental value the explanatory variable. (In other contexts the roles of these two variables might well be reversed). A plot of capital versus rental values is shown in the top left-hand plot in Figure 3.13, and least squares residuals are plotted against fitted values in the top right-hand plot. Note how the “funnel effect” in the right plot, where the spread of the residuals increases with the fitted values, magnifies the corresponding pattern in the left hand plot. This pattern is indicative of greater variation in capital value for the higher rents than for the lower rents.

Dividing the observations up into groups according to the rental values, and plotting the log standard deviation of the capital values for each group versus the log mean values shows a clear relationship between the dispersion and mean level of the capital values. The plot is the bottom left plot in Figure 3.13. The following R program produced this plot:

```
> data(acc.df)
> groups<-cut(acc.df$rental,c(0,7000,9000,11000,12000,14000))
> log.means<-log(tapply(acc.df$capital,groups,mean))
> log.sds<-log(sqrt(tapply(acc.df$capital,groups,var)))
> plot(log.means,log.sds)
```

Here we are more concerned with smoothing the plot of squared residuals versus fitted values. This should give us an idea of how the variance of the residuals changes with the mean level of the response. In Figure 3.13 we show a plot of squared residuals from the fit plotted against the

fitted values, with a smooth curve added. We see that there is a definite upward trend in the plot, indicating that, as the fitted value (which measures the mean capital value) increases, so does the variability of the residuals, and hence the variability of the errors.

The plot was produced by the following commands assuming that the data is stored in two vectors `capital` and `rental` respectively.

```
> acc.lm<-lm(capital~rental, data=acc.df)
> sq.res<-residuals(acc.lm)^2
> pred<-fitted.values(acc.lm)
> plot(pred,sq.res,xlab="Fitted values",ylab="Squared residuals")
> lines(lowess(pred,sq.res))
```

Since we are routinely going to check for unequal variances, it is convenient to have a function `funnel` that draws all the plots we have discussed in this section. The function contained in the R330 package. We use the function as follows:

```
> funnel(acc.reg)
```

This produces the bottom two pictures in Figure 3.13.

### 3.4.2 Outliers and high leverage points

We need to distinguish between *outliers* (points that have large errors  $e_i$ ) and *high leverage points* (points that have extreme  $x$ -values.) We know from STATS 201/8 that outliers can distort the position of the least-squares plane, and that the distortion is worse if the outlier is also a high leverage point. Figure 3.14 illustrates the effect that outliers and high leverage points have on the model fit. Since a high leverage point attracts the fitted line or plane, the residual corresponding to an outlier that is also a high leverage point need not be large. On the other hand, a low leverage outlier will usually have a large residual, and can be detected by normal plots of residuals, or residuals versus fitted values plots. Since we usually want to identify the observation corresponding to a big residual, it is helpful to use the observation number as the plotting symbol.

**Example 10.** Consider the data in Table 3.8, which consists of employee salaries ( $y$ ) and various explanatory variables  $x_1, \dots, x_5$ . A regression model explaining salary in terms of the explanatory variables was fitted. Assume that the data are stored in a data frame `salary.df`. We can fit the regression, and draw a plot of residuals versus fitted values using labelled points by using the following R commands:

```
> salary.lm<-lm(Y~X1+X2+X3+X4+X5,data=salary.df)
> resids<-residuals(salary.lm)
> pred<-fitted.values(salary.lm)
> plot(pred,resids,type="n")
> ncases<-length(resids)
> text(pred,resids,1:ncases) # 1:ncases is the vector of case labels
```

This plot is shown in Figure 3.15. We see that there is one large residual, corresponding to point 6.

High leverage points do not necessarily have large residuals, so that it is occasionally difficult to recognise them from a residual plot. The usual method of detecting high leverage points is by calculating a set of numerical measures known as *hat matrix diagonals* or HMD's. Consider the formula that express the fitted value  $\hat{y}_i$  as a linear combination of the  $y_i$ 's:

$$\hat{y}_i = \sum_{j=1}^n h_{ij} y_j$$

or in matrix notation

$$\hat{Y} = HY$$

where  $H$  is the matrix  $X(X^T X)^{-1} X^T$ . The  $i$ th diagonal element of  $H$ ,  $h_{ii}$ , is called the *hat matrix diagonal* or HMD, and measures to what extent  $y_i$  determines the fitted value  $\hat{y}_i$ . Thus

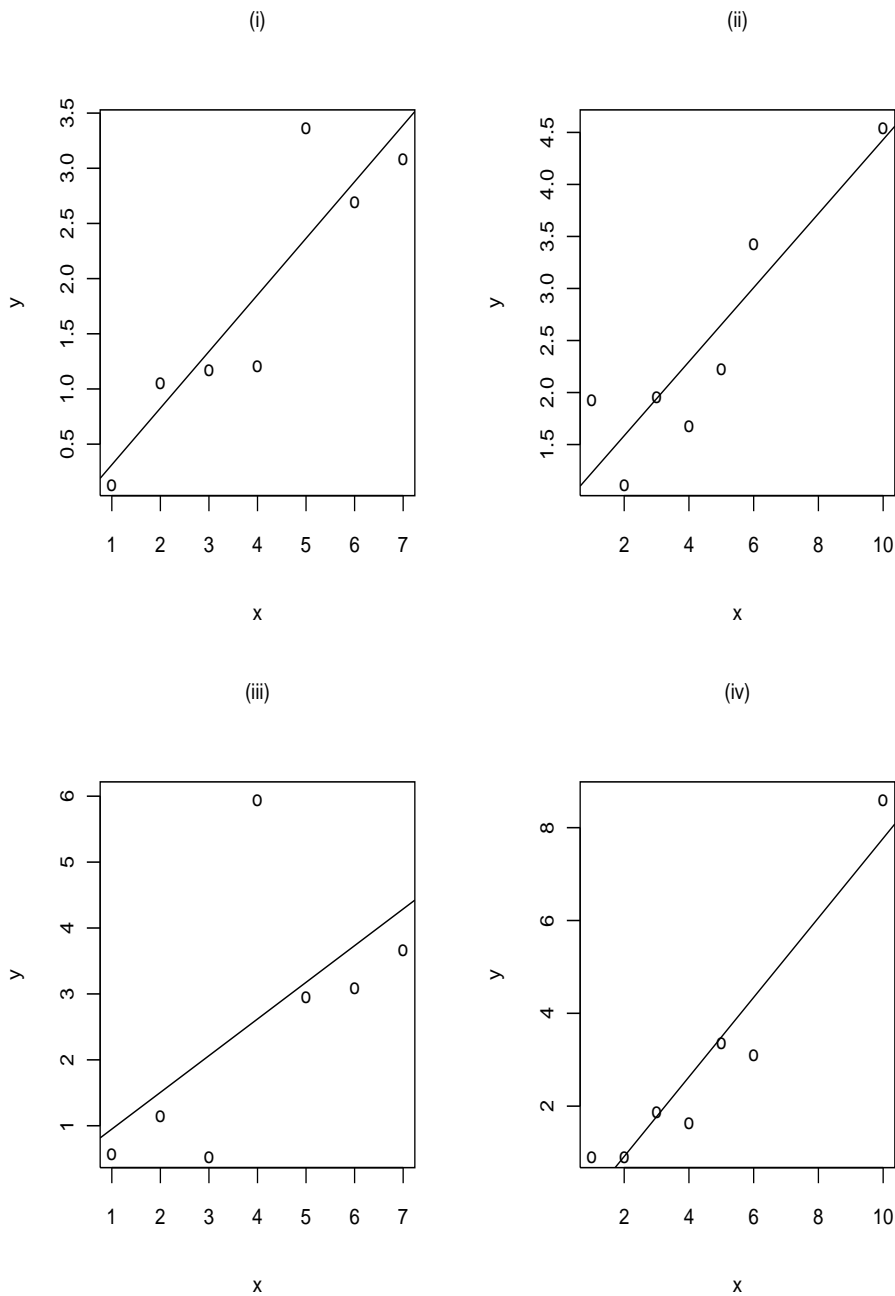


Figure 3.14: Outliers and high leverage points. (i) No outliers, no high leverage points. (ii) A high leverage point that is not an outlier. (iii) An outlier that is not a high leverage point. (iv) A high leverage outlier.

Table 3.8: Employee salary data.

X1	X2	X3	X4	X5	Y	X1	X2	X3	X4	X5	Y
350	1	2	2	5	1000	350	1	5	5	5	1400
350	0	4	4	4	1200	350	1	20	20	1	1800
425	0	10	2	3	2800	425	1	15	10	3	4000
425	0	1	1	4	2500	425	1	5	5	4	3000
600	1	10	5	2	3500	600	0	8	8	3	2800
600	0	4	3	4	2900	600	1	20	10	2	3800
600	1	7	7	5	4200	700	1	8	8	1	4600
700	0	25	15	5	5000	700	1	9	16	4	4600
700	0	20	14	5	4700	400	0	6	4	3	1800
400	1	20	8	3	3400	400	0	5	3	5	2000
500	1	22	12	3	3200	500	1	25	10	3	3200
500	0	8	3	4	2800	500	0	2	1	5	2400
800	1	10	10	3	5200	475	1	10	4	3	2400
475	0	3	3	4	2400	475	1	8	8	2	3000
475	1	6	6	4	2800	475	0	12	4	3	2500
475	0	4	2	5	2100						

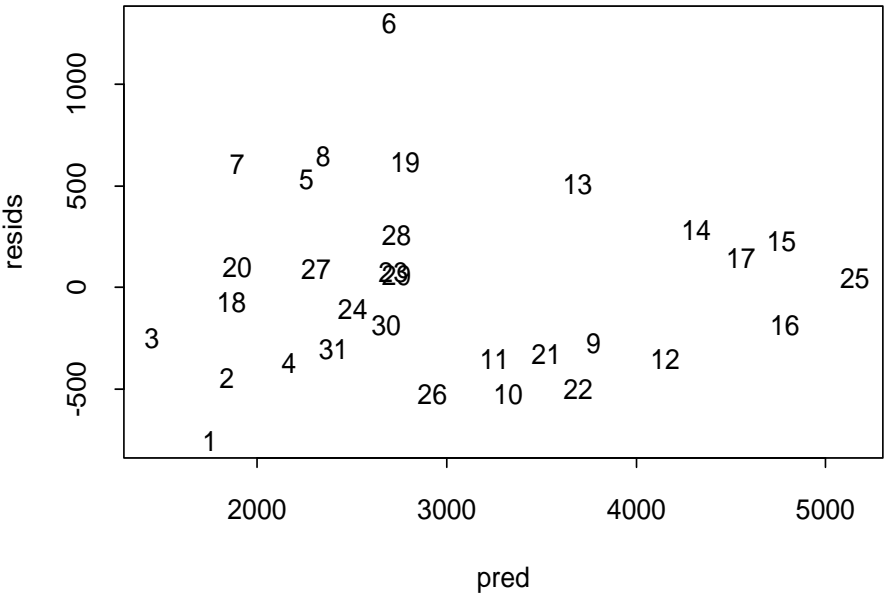


Figure 3.15: Residuals versus fitted values for the salary data.

$h_{ii}$  measures “the amount of pull” that the  $i$ th data point has on the fitted regression surface at that point. Note that the HMD’s depend only on the explanatory variables and not at all on the response.

It can be shown that  $\sum_{i=1}^n h_{ii} = k+1$  so that the “average” hat matrix diagonal is  $\frac{k+1}{n}$ . HMD’s greater than  $3(k+1)/n$  are deemed to have “excessive leverage” and points having such high HMD’s have the potential of exerting a very large influence on the position of the fitted regression plane.

**Example 11.** For the simple model having one explanatory variable and no intercept, and equation

$$y_i = \beta x_i + e_i$$

it can be shown that

$$h_{ii} = x_i^2 / \sum_{j=1}^n x_j^2$$

so that  $h_{ii}$  is large for extreme  $x$ -values.

**Example 12.** When we have a single explanatory variable, the model is

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

and

$$h_{ii} = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{j=1}^n (x_j - \bar{x})^2}$$

so that  $h_{ii}$  measures the distance of  $x_i$  from the “centre” of the  $x$ -data.

In general, when we have  $k$  explanatory variables, we can think of the of  $x$ -values of each case as points in a  $k$ -dimensional space, the “design space”. The HMD corresponding to a point measures how far from the centre of the data configuration the particular point is.

We calculate the hat matrix diagonals in R using the function `hatvalues`, which takes an “lm object” (i.e. the object produced by the function `lm`) as its argument, and returns the hat matrix diagonals.

```
> hats<-hatvalues(salary.lm)
> hats
[1] 0.23483750 0.22348752 0.14713190 0.69774292 0.18201083 0.09302908
[7] 0.11172459 0.11667179 0.18501917 0.16169049 0.12251453 0.16603209
[13] 0.20225378 0.34531946 0.37306734 0.23712410 0.29948313 0.13554444
[19] 0.21795353 0.11779806 0.13889463 0.28450981 0.08804692 0.11864810
[25] 0.29477687 0.11916143 0.09298534 0.13450846 0.10426843 0.15006812
[31] 0.10369565
> sum(hats)
[1] 6
```

Note how all the hat matrix diagonals lie between 0 and 1 and sum to  $k+1$  (6 in this case, since there are 5 explanatory variables.) To identify which points have large HMD’s, we can use the function `order`. When applied to the vector `hats`, it returns a vector whose elements are the positions in `hats` of the smallest element, the second smallest element,...., and finally the largest element. Thus if we type

```
> order(hats)
[1] 23 27 6 31 29 7 8 20 24 26 11 28 18 21 3 30 10 12 5 9 13 19 ‘
[23] 2 1 16 22 25 17 14 15 4
```

we see that the 4th and 15th observations have the most leverage, since they have the biggest HMD’s.

A convenient method to illustrate which points are high leverage and/or outlying is to plot the HMD’s versus the standardised residuals – this is the LR plot discussed in 201/8. Points that have both large residuals and large HMD’s will have a substantial impact on the fitted model. We type

```
> plot(salary.lm,which=5)
```

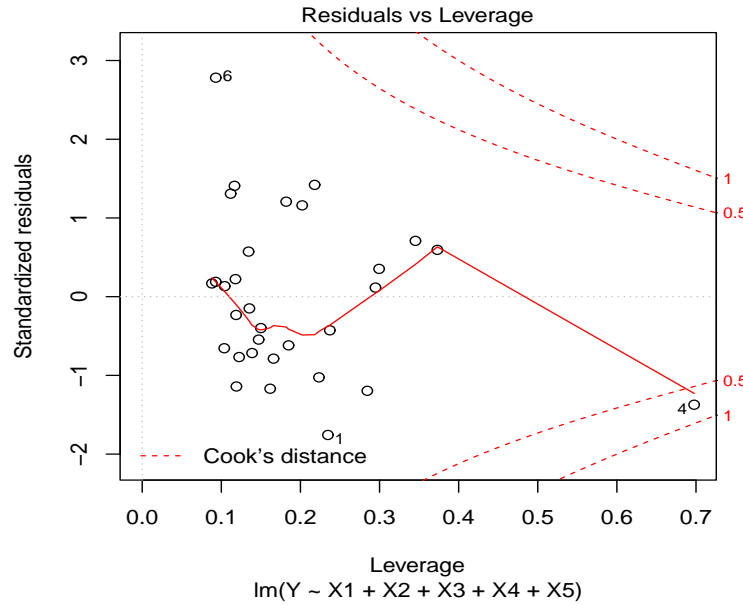


Figure 3.16: Leverage-residual plot for the salary data.

to get a single plot. This plot is one of the plots produced by the plot function as described on page 37. From the plot shown in Figure 3.16, it is plain that point 6 is an outlier and point 4 has high leverage.

*To summarise, the following plots are useful for detecting outliers and high leverage points:*

- *A plot of residuals versus fitted values, with a lowess curve added;*
- *A normal plot of residuals,*
- *A plot of residuals versus fitted values, with case numbers used as plotting symbols,*
- *A plot of residuals versus leverage.*

#### “Leave-one-out” influence measures

One way to determine the amount of influence that a point has on the fitted regression model, is to delete the point from the data set, refit the model, and see how big a change has occurred in the various regression quantities such as regression coefficients, standard errors etc. Good references for this idea are Belsey, Kuh and Welsch (1980) and Chatterjee and Hadi (1988). Some measures of this type are:

- (i) Standardised difference in coefficients, given by the formula

$$(\hat{\beta}_j - \hat{\beta}_j(-i))/s.e.(\hat{\beta}_j)$$

In this formula,  $\hat{\beta}_j(-i)$  is the estimate of  $\beta_j$  after the  $i$ th data point has been deleted from the data. Note that the standard error of estimate of  $\hat{\beta}_j$  is based on an estimate of  $\sigma$  that does not include the  $i$ th case. These measures are called DFBETAS in the references above. The DFBETAS are in units of standard deviations, so that for example a DFBETAS of 1 means dropping one data point has changed the regression coefficient by a standard deviation

– a big change. There is some disagreement in the literature about the value of a suitable cutoff value. In R, any point having a DFBETAS of more than 1 is regarded as influential. This seems to work well in practice.

(ii) Change in fitted values:

$$\frac{\hat{Y}_i - \hat{Y}_i(-i)}{s.e.(\hat{Y}_i)}$$

called DFFITS in the above references. Once again this is standardised by an estimate of the standard deviation of  $\hat{Y}_i$  that based in turn on an estimate of  $\sigma$  that does not include the  $i$ th case.<sup>4</sup> The references above suggest that any point having a DFFITS of more than 2 or 3 times  $\sqrt{\frac{k+1}{n-k-1}}$  is influential. R uses  $3\sqrt{\frac{k+1}{n-k-1}}$ .

(iii) Covariance ratio: the ratio of determinants

$$CR_i = \frac{\det(\text{Covariance matrix of estimates with } i\text{th data point removed})}{\det(\text{Covariance matrix of estimates})}$$

is called the *covariance ratio* and measures the change in the standard errors and covariances when a case is deleted. Points having values of  $CR$  with  $|CR_i - 1| > \frac{3(k+1)}{n-k-1}$  should be regarded as influential.

Another influence measure, due to Cook (Technometrics 1977), is based on the idea of a *confidence ellipsoid*, a generalization of a confidence interval. The standard theory of linear models (see STATS 310) says that a  $100(1 - \alpha)\%$  confidence ellipsoid for  $\beta$  is

$$\left\{ b : \frac{(\hat{\beta} - b)^T (X^T X)(\hat{\beta} - b)}{(k+1)\hat{\sigma}^2} \leq F_{k+1, n-k-1}(\alpha) \right\}$$

The interpretation is the same as for confidence intervals. The idea is to see if  $\hat{\beta}(-i)$  is in the ellipsoid, or equivalently, look at the quantities

$$D_i = \frac{(\hat{\beta} - \hat{\beta}(-i))^T (X^T X)(\hat{\beta} - \hat{\beta}(-i))}{(k+1)\hat{\sigma}^2}$$

and see if they are smaller than say  $F_{k+1, n-k-1}(.5)$ , the 50% point of the  $F$  distribution with  $k+1$  and  $n-k-1$  degrees of freedom. This is the same as  $\hat{\beta}(-i)$  being in a 50% confidence ellipsoid.

As an approximation, we can take  $F_{k+1, n-k-1}(.5)$  to be approximately 1. This is the explanation of the red dotted lines on the LR plot in R: points outside the red dotted line marked “1” correspond roughly to points outside the ellipsoid. (The “0.5” line corresponds roughly to a 10% confidence ellipsoid.) The measure  $D_i$  is called Cook’s  $D$ .

Computing all these measures in R is rather complicated, so there is an R function `influence.measures` to do the job. It takes as its argument the regression object produced by `lm`. Thus for the salary data, we type

```
> influence.measures(salary.lm)
Influence measures of
lm(formula = Y ~ X1 + X2 + X3 + X4 + X5, data = salary.df) :

      dfb.1_   dfb.X1   dfb.X2   dfb.X3   dfb.X4   dfb.X5   dffit cov.r   cook.d   hat inf
1  0.04899  0.32200 -0.6427  0.15125  0.06431 -0.558324 -1.0183 0.758 0.157816 0.235
2  0.02430  0.24681 -0.3003  0.12173 -0.09944 -0.338155 -0.5504 1.272 0.050386 0.223
3 -0.15037  0.13847  0.1171  0.08004 -0.08542  0.014455 -0.2236 1.394 0.008575 0.147
4 -1.16868  1.23590  0.5150  0.84691 -1.59147  0.720475 -2.1264 2.640 0.725771 0.698 *
5  0.32188 -0.08580 -0.2512  0.29675 -0.32587 -0.251359  0.5751 1.089 0.054078 0.182
6  0.35339 -0.59287  0.3335  0.10086  0.19778 -0.000170  1.0501 0.153 0.132215 0.093 *
7  0.24582 -0.07712 -0.1943 -0.14464 -0.00628 -0.074827  0.4706 0.940 0.035821 0.112
8  0.02314 -0.12674  0.3329 -0.18178  0.06168  0.172109  0.5224 0.881 0.043619 0.117
9 -0.01803 -0.14888 -0.1057 -0.04865  0.15391  0.151432 -0.2908 1.429 0.014457 0.185
10 -0.13948 -0.13212  0.3480  0.24506 -0.22458  0.229246 -0.5176 1.087 0.043962 0.162
```

<sup>4</sup>The standard deviation of the fitted value  $\hat{Y}_i$  is  $\sigma h_{ii}$ , which we estimate by  $\hat{\sigma}(-i)h_{ii}$  where  $\hat{\sigma}(-i)$  is the estimate of  $\sigma$  obtained from the data set with the  $i$ th case deleted.



11	0.01570	-0.15932	0.1076	0.06173	0.03412	0.033257	-0.2842	1.262	0.013693	0.123	
12	0.00236	-0.10755	-0.0775	-0.20208	0.17134	0.138710	-0.3483	1.318	0.020537	0.166	
13	-0.37797	0.21228	0.3602	-0.15640	0.06024	0.385661	0.5881	1.149	0.056810	0.202	
14	0.06203	0.28445	0.0509	-0.16719	-0.00193	-0.345438	0.5090	1.728	0.044076	0.345	*
15	-0.21862	0.09487	-0.1659	0.15309	0.06404	0.196085	0.4530	1.871	0.035129	0.373	*
16	0.14069	-0.07624	-0.0504	0.04792	-0.11419	-0.097227	-0.2340	1.603	0.009435	0.237	
17	-0.11152	0.06003	-0.0909	0.01753	0.07493	0.099477	0.2271	1.770	0.008907	0.299	*
18	-0.04828	0.02329	0.0377	0.01017	-0.00733	0.030344	-0.0578	1.470	0.000579	0.136	
19	0.13300	-0.27783	0.2358	0.58473	-0.39491	0.021814	0.7655	0.987	0.093536	0.218	
20	0.01695	-0.03231	-0.0265	-0.00496	0.00655	0.034197	0.0798	1.431	0.001104	0.118	
21	0.00776	0.06399	-0.0718	-0.17437	0.05069	-0.025438	-0.2846	1.310	0.013773	0.139	
22	0.06465	0.05868	-0.2002	-0.66941	0.44246	-0.053229	-0.7616	1.253	0.094933	0.285	
23	0.01062	0.00578	-0.0242	0.01656	-0.02205	-0.003401	0.0510	1.391	0.000451	0.088	
24	0.00730	-0.01580	0.0159	0.00990	0.01524	-0.029609	-0.0838	1.431	0.001218	0.119	
25	-0.03598	0.05956	0.0210	-0.02404	0.00651	-0.005161	0.0727	1.806	0.000918	0.295	*
26	-0.04537	-0.03510	-0.2513	-0.17062	0.28263	0.058508	-0.4221	1.053	0.029320	0.119	
27	0.02361	-0.00217	-0.0309	-0.02377	0.00957	-0.007541	0.0589	1.397	0.000601	0.093	
28	0.11996	-0.02965	0.0421	-0.11099	0.06855	-0.129814	0.2228	1.364	0.008503	0.135	
29	-0.00526	-0.00273	0.0293	-0.01775	0.00770	0.016370	0.0447	1.420	0.000346	0.104	
30	-0.07948	0.00858	0.0888	-0.08515	0.08002	0.073476	-0.1651	1.446	0.004705	0.150	
31	0.00608	-0.00321	0.0569	0.00577	0.03110	-0.092378	-0.2200	1.285	0.008262	0.104	

Note that points that are influential according to any of the criteria above are flagged with a \*. There is function `influenceplots` in the R330 package that can be used to draw plots of these quantities.

### More on residuals

One objection to using raw residuals in diagnostic plots is that unlike the errors  $e_i$ , they do not have equal variances even when the regression model holds exactly. In fact, it can be shown that when all the model assumptions hold that

$$\text{Var } r_i = (1 - h_{ii})\sigma^2.$$

The implication is that high influence points tend to have smaller variances. Because of this, it is common practice to standardise the residuals so that they have approximately equal variances. This can be done in several ways. *Internally studentised residuals* are defined by

$$r'_i = \frac{r_i}{\hat{\sigma}(1 - h_{ii})^{\frac{1}{2}}}.$$

These have variance approximately unity. An alternative is to replace  $\hat{\sigma}$  by  $\hat{\sigma}(-i)$  where  $\hat{\sigma}(-i)$  is the estimate of  $\sigma$  obtained by deleting the  $i$ th case. This offers some protection if the  $i$ th case is an outlier (in the sense that the variance will be closer to unity) and these residuals are said to be *externally studentised*. This terminology is well established in the literature. Confusingly, R uses the terms “standardised residual” to mean an internally studentised residual, and “studentised residual” to mean an externally studentised residual.

### 3.4.3 Summary of diagnostics

1. Checking for a curved regression surface:
  - (a) Use plots described in Chapter 2
  - (b) Plot all possible scatter plots (`pairs`)
  - (c) Plot residuals versus fitted values (`plot`)
  - (d) Plot residuals versus explanatory variables (`plot`)
  - (e) Added variable plots (`added.variable.plots`)
  - (f) Gam plots (`plot` and `gam`)
2. Checking for normality
  - (a) Normal plots (`qqnorm`, `plot`)
  - (b) Weisberg-Bingham test

3. Checking for independence
  - (a) Durbin-Watson test
  - (b) Plot residuals versus lagged residuals, time series plot, acf plot
4. Checking for outliers/influence points
  - (a) Normal plots (`qqnorm`, `plot`)
  - (b) Plot residuals versus fitted values (`plot`)
  - (c) Hat matrix diagonals
  - (d) Leave-one-out diagnostics (`influence.measures`)
  - (e) LR plot (`plot`)
5. Checking for unequal variances
  - (a) Look for funnel effect in residuals versus fitted values plot
  - (b) plot squared residuals versus fitted values and smooth (`funnel`)
  - (c) group data and plot log means versus log sds

## 3.5 Fixing up models

Now we deal with methods for revising models. If the methods we looked at on the last section indicate that the regression model does not fit well, we have to take some kind of corrective action to improve the fit. We usually have two options: either fit a more complex model, or transform the data so that the standard regression model fits the transformed data. In this section we describe how to take whatever corrective action seems indicated by the diagnostic plots and tests described in the last section.

### 3.5.1 Corrective action

#### Non-linearity

In general, the most serious problem is the possible non-linearity of the regression function. If our residual plots reveal that the regression surface is curved, we can transform the  $x$ 's or  $Y$ , perhaps by using the Box-Cox family of power transformations. For a suitable power  $p$ , we transform our data  $y$  using the formula

$$\frac{y^p - 1}{p}.$$

In R, we can transform the data by typing

```
y.trans<-(y^p-1)/p
```

We can get an idea of which variable or variables to transform, and also which power should be used by looking at the gam plots. We make the indicated transformations, refit the model, and check to see if the revised model fits well.

Sometimes we need to fit a more complicated model. For example, we can add squared or cubed terms in some or all of the variables, or perhaps cross-product terms. We can fit general quadratic or cubic models: For example, if we have two explanatory variables  $x_1$  and  $x_2$  and a response  $y$ , we can fit a general quadratic model of the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2 + e$$

which will model a variety of curved surfaces. Other approaches include smoothing, which is discussed in Section 3.10.

**Example 13.** The data in Table 3.9 are percentage conversions of n-heptane to acetylene (the variable `percent.conv`), temperature in degrees Centigrade (`temp`), ratio of  $H_2$  to n-heptane (`ratio`)

Table 3.9: Acetylene data.

% CONV	TEMP	RATIO	CONTACT
49.0	1300	7.5	0.0120
50.2	1300	9.0	0.0120
50.5	1300	11.0	0.0115
48.5	1300	13.5	0.0130
47.5	1300	17.0	0.0135
44.5	1300	23.0	0.0120
28.0	1200	5.3	0.0400
31.5	1200	7.5	0.0380
34.5	1200	11.0	0.0320
35.0	1200	13.5	0.0260
38.0	1200	17.0	0.0340
38.5	1200	23.0	0.0410
15.0	1100	5.3	0.0840
17.0	1100	7.5	0.0980
20.5	1100	11.0	0.0920
29.5	1100	17.0	0.0860

and contact time in seconds (`contact`). The data arose in a study to find the best conditions for the manufacture of acetylene, and may be found in an article by Marquart and Snee, in the *American Statistician* in 1975. We begin by fitting a regression model using the three variables `temp`, `ratio` and `contact` as predictors:

```
> summary(lm(percent.conv~temp+ratio+contact, data=acet.df))
```

Call:

```
lm(formula = percent.conv ~ temp + ratio + contact)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-6.9198 -1.8559  0.2340  2.0740  6.9476
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -121.26962   55.43571  -2.188  0.0492 *
temp          0.12685    0.04218   3.007  0.0109 *
ratio         0.34816    0.17702   1.967  0.0728 .
contact      -19.02170   107.92824  -0.176  0.8630
```

---

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 3.767 on 12 degrees of freedom

Multiple R-Squared: 0.9198, Adjusted R-squared: 0.8998

F-statistic: 45.88 on 3 and 12 degrees of freedom, p-value: 7.522e-07

It's pretty clear from the large  $p$ -value for `contact` that this variable is not doing much good, so we drop it from further models. Let's try a model with `temp` and `ratio` alone. We also plot residuals versus fitted values using the observation number as a plotting symbol:

```
> acet.lm<-lm(percent.conv~temp+ratio,data=acet.df)
```

```
> summary(acet.reg)
```

Call:

```
lm(formula = percent.conv ~ temp + ratio, data = acet.df)
```

```

Residuals:
      Min       1Q   Median       3Q      Max
-7.0337 -2.0205  0.2494  2.2035  6.8648

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -130.68986   14.14571  -9.239 4.47e-07 ***
temp          0.13396    0.01191   11.250 4.51e-08 ***
ratio         0.35106    0.16955    2.071  0.0589 .
---
Residual standard error: 3.624 on 13 degrees of freedom
Multiple R-Squared:  0.9196,    Adjusted R-squared:  0.9072
F-statistic: 74.35 on 2 and 13 degrees of freedom,    p-value: 7.654e-08
>plot(fitted.values(acet.reg),residuals(acet.reg),type="n")
>text(fitted.values(acet.reg),residuals(acet.reg),1:16)

```

The plot, shown in Figure 3.17(i), has a rather peculiar appearance, which indicates a curved relationship between the explanatory variables and the response. The three sets of points correspond to the three values of `temp`. We get a better idea of what is going on with a 3-d plot:

```
> reg3d(acet.df)
```

It is now clear from Figure 3.18 that there are three curved bands of points. To model this as a non-linear regression surface, we can try a second-degree polynomial in `temp` and `ratio`:

```

> second.lm<-lm(percent.conv ~ temp + ratio + I(temp*temp)
+                  + I(ratio*ratio) + I(temp*ratio))
> summary(second.lm)

Call:
lm(formula = percent.conv ~ temp + ratio + I(temp * temp) + I(ratio *
ratio) + I(temp * ratio))

Residuals:
      Min       1Q   Median       3Q      Max
-1.57672 -0.44853  0.02746  0.58530  1.52180

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.991e+01  8.219e+01  0.242   0.8135
temp          -2.104e-01  1.385e-01 -1.519   0.1598
ratio          9.963e+00  8.808e-01  11.310 5.09e-07 ***
I(temp * temp)  1.782e-04  5.854e-05   3.044   0.0124 *
I(ratio * ratio) -2.367e-02  1.019e-02 -2.324   0.0425 *
I(temp * ratio) -7.335e-03  7.993e-04 -9.177 3.47e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

Residual standard error: 1.066 on 10 degrees of freedom
Multiple R-Squared:  0.9946,    Adjusted R-squared:  0.992
F-statistic: 371.5 on 5 and 10 degrees of freedom,    p-value: 5.13e-11
> plot(fitted.values(second.reg),residuals(second.reg),type="n")
> text(fitted.values(second.reg),residuals(second.reg),1:16)

```

The plot (Figure 3.17(ii)) shows that the extra terms have been worthwhile: the pattern in the residuals has disappeared, the residuals are much smaller, and the  $R^2$  has increased from 92% to 99%.

## Unequal variances

The next most serious problem is when the variances of the errors are not equal, so the scatter of the points about the regression surface is different for different parts of the surface. When this

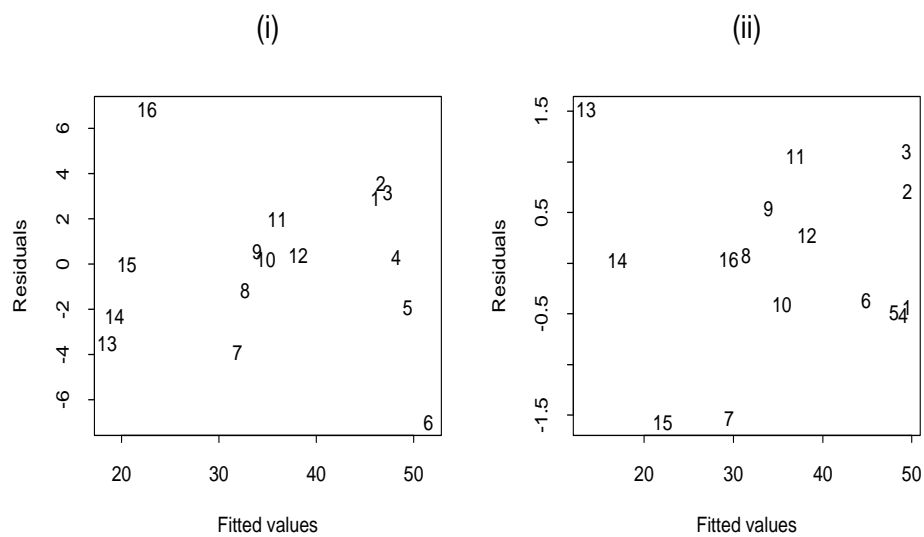


Figure 3.17: Residual plots for the acetylene data. (i) linear fit, (ii) quadratic fit.

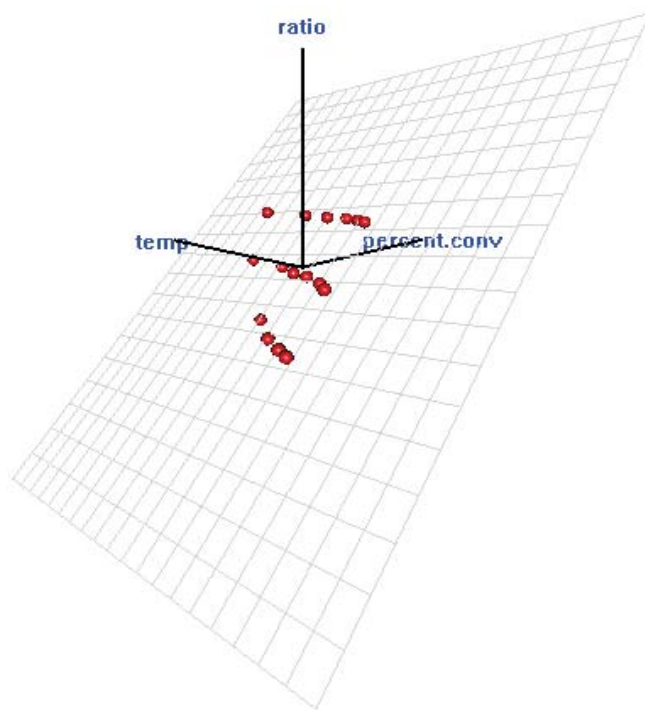


Figure 3.18: 3-d plot of the acetylene data.

happens, the standard errors calculated by `lm` will not be correct. We detect unequal scatter by the funnel effect in the residual plot. This problem can be attacked in two ways, either by transforming  $y$ , or by explicitly modelling the different variances.

**Transforming  $y$ :** To transform  $y$  we plot log standard deviations versus log means as we did in Example 8. To find the power for the transformation, which is 1 minus the slope of the fitted line, we can type

```
> funnel(acc.lm)
Slope: 0.8721885
> 1-0.8721885
[1] 0.1278115
```

This power is quite close to 0, indicating a log transformation. This transformation pretty much removes the problem.

**Modelling the different variances:** The second method for dealing with unequal variances involves modelling the data as

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \sigma_i e_i. \quad (3.8)$$

The errors are now written as  $\sigma_i e_i$ , where the  $e_i$  are independent with mean 0 and variance 1 i.e. the dispersion of the  $e_i$  does not depend on the  $x$ 's. The multipliers  $\sigma_i$  are assumed to be unknown constants (i.e. not random quantities) and model the different amounts of scatter. They must be deduced from the data. Methods for doing this are discussed below.

Assuming that this has been done, we can divide each side of (3.8) by  $\sigma_i$  to get

$$\left(\frac{y_i}{\sigma_i}\right) = \beta_0 \left(\frac{1}{\sigma_i}\right) + \cdots + \beta_k \left(\frac{x_{ik}}{\sigma_i}\right) + e_i. \quad (3.9)$$

To estimate the coefficients, we need to use the so-called method of *weighted least squares*. We need to minimise the expression

$$\sum_{i=1}^n \left( \frac{y_i}{\sigma_i} - \beta_0 \left( \frac{1}{\sigma_i} \right) - \cdots - \beta_k \left( \frac{x_{ik}}{\sigma_i} \right) \right)^2$$

which we may also write as

$$\sum_{i=1}^n \frac{1}{\sigma_i^2} (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_k x_{ik})^2$$

or more generally as

$$\sum_{i=1}^n w_i (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_k x_{ik})^2.$$

This is the same form as (3.5) except that the terms of the sum are weighted using the weights  $w_i$ . Terms with large weights dominate the sum, so the weights control how much impact each point has on the position of the fitted plane: the greater the weight the greater the impact. In the present case,  $w_i = 1/\sigma_i^2$  so that points with large values of  $\sigma_i$  will have a small influence on the fit. This procedure is called *weighted least squares* (WLS) and is done in the function `lm` by inserting an extra argument `weights`.

We need to estimate the weights  $w_i = 1/\sigma_i^2$  and then use WLS to estimate the parameters. If we use ordinary least squares (OLS) the estimates of the regression coefficients will be reasonable but the standard errors will be wrong. The  $\sigma_i$  can either be estimated by

- (i) grouping points with similar  $x$ -values together, and taking the standard deviation of the group of residuals, or
- (ii) Smoothing a plot of  $r_i^2$  versus  $\hat{y}_i$ , to get a rough idea of the form of  $\sigma_i$ .

The first method is quite common in the literature but fails badly when the groups of similar points are too small, say less than 10. See Carroll and Cline (1988) for a discussion of what can go wrong. They advocate the second method, which assumes that  $\sigma_i^2 = g(x_i^T \beta)$  for some smooth function  $g$ . Thus, since  $E(r_i^2) \approx g(x_i^T \hat{\beta})$ , plotting  $r_i^2$  vs  $x_i^T \hat{\beta} = \hat{y}_i$  should reveal something about the form of  $g$ . This method is typically more accurate.

**Example 14.** Let's carry out this method with the ACC data of Example 8. We have already shown how to smooth the plot using `funnel`. We use the smoothed function  $g(\hat{y}_i)$  as an estimate of  $\sigma_i^2$ . The vector of values  $g(\hat{y}_i)$  is returned by the function `funnel`, and we get it by typing

```
g.values<-funnel(acc.lm)
```

We then do the regression using weights the reciprocals of the elements in `g.values`:

```
> acc.lm=lm(capital~rental, weights=1/g.values,data=acc.df)
> summary(acc.lm)
```

Call:

```
lm(formula = capital ~ rental, data = acc.df, weights = 1/g.values)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.94914	-0.74150	0.06722	0.68697	2.13288

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-33117.28	12601.36	-2.628	0.0100 *
rental	21.37	1.51	14.151	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.01 on 94 degrees of freedom

Multiple R-Squared: 0.6805, Adjusted R-squared: 0.6772

F-statistic: 200.3 on 1 and 94 DF, p-value: < 2.2e-16

Compare the results with an unweighted regression:

```
> summary(lm(capital~rental,data=acc.df))
```

Call:

```
lm(formula = capital ~ rental, data = acc.df)
```

Residuals:

Min	1Q	Median	3Q	Max
-107777	-30990	1298	23588	84991

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-43372.326	17993.856	-2.41	0.0179 *
rental	22.559	1.822	12.38	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 42450 on 94 degrees of freedom

Multiple R-Squared: 0.6199, Adjusted R-squared: 0.6159

F-statistic: 153.3 on 1 and 94 DF, p-value: < 2.2e-16

The estimates of the slope are similar, but the standard errors are somewhat different. The difference in the residual standard error is because in the first output the residual standard error is estimating the variance of  $e_i$  which is 1, and in the second the error variance in the *unweighted* regression.

### Lack of normality

This is probably the least important departure and hopefully can be fixed up by transforming  $y$ . But in a toss-up between achieving linearity and normality, choose linearity.

To choose a transformation, we can use a variation of the Box-Cox technique. Suppose we transform the response with a Box-Cox transformation with power  $p$ . Let  $RSS^{(p)}$  be the residual sum of squares from the transformed regression. We calculate the quantity

$$L = \frac{n}{2} \log RSS^{(p)} + (1-p) \sum_{i=1}^n \log y_i$$

and choose the power  $p$  that minimises this expression. (Technically, this results in the maximum likelihood estimate of the power.) The easiest way to choose  $p$  is to plot the expression for several values of  $p$  and pick the minimising value. A function `boxcoxplot` to do this is included in the R package `R330`.

**Example 15.** The New York Times of March 4 1990 carried an interesting article, describing an attempt by Professor Orley Ashenfelter of Princeton University to use multiple regression to assess the quality of various Bordeaux vintages. The Bordeaux wine district of France produces some of the world's finest wines which fetch extremely high prices. The quality of the wine depends on various factors, some of which are climatic. Some years produce outstanding vintages, for example 1982, 1983 and 1985, whereas other years such as 1984 produce wine of a lesser quality.

Professor Ashenfelter's analysis relates the quality of the year's wine (as measured by an index of current wholesale prices fetched by mature wines) to three climatic variables: the rainfall at harvest time, winter rainfall (both measured in mm) and the average temperature (in degrees Centigrade) over the growing season. For a good vintage, it is desirable to have high winter rainfall to ensure a plentiful supply of water during the growing season. High temperatures during the growing season are also beneficial, as is low rainfall during harvest.

The resulting regression, using the current auction price as the response and the climatic variables and the year of vintage as the explanatory variables, explains the variations in the price reasonably well. Amusingly, the analysis has been attacked in the wine trade press by wine experts ignorant of statistics. Perhaps they resent the intrusion of alternative kinds of expertise into the rather arcane and mystical world of fine wines.

The data used by Professor Ashenfelter is shown in Table 3.10 and is included in the `R330` package as the data frame `wine.df`, with variables `price`, `w.rain`, `h.rain` `temp` and `year`. The regression of price on the other variables is

```
> wine.lm<-lm(price~temp+h.rain+w.rain+year, data=wine.df)
> summary(wine.lm)
Call:
lm(formula = price ~ temp + h.rain + w.rain + year, data = wine.df)
Residuals:
    Min       1Q   Median       3Q      Max
-14.077  -9.040  -1.018   3.172  26.991

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 1305.52761   597.31137   2.186  0.03977 *
temp         19.25337    3.92945   4.900 6.72e-05 ***
h.rain       -0.10121    0.03297  -3.070  0.00561 **
w.rain        0.05704    0.01975   2.889  0.00853 **
year        -0.82055    0.29140  -2.816  0.01007 *
---
Residual standard error: 11.69 on 22 degrees of freedom
Multiple R-Squared:  0.7369,    Adjusted R-squared:  0.6891
F-statistic: 15.41 on 4 and 22 DF,  p-value: 3.806e-06
```

First, we note that the percentage of variation explained is high at almost 74%. It is surprising that we can explain so much of the variation in something as complex as wine quality using three



Table 3.10: Data on Bordeaux vintages.

Year	Price(1961=100)	Temperature	Harvest rain	Winter rain
1952	37	17.1	160	600
1953	63	16.7	80	690
1955	45	17.1	130	502
1957	22	16.1	110	420
1958	18	16.4	187	582
1959	66	17.5	187	485
1960	14	16.4	290	763
1961	100	17.3	38	830
1962	33	16.3	52	697
1963	17	15.7	155	608
1964	31	17.3	96	402
1965	11	15.4	267	602
1966	47	16.5	86	819
1967	19	16.2	118	714
1968	11	16.2	292	610
1969	12	16.5	244	575
1970	40	16.7	89	622
1971	27	16.8	112	551
1972	10	15.0	158	536
1973	16	17.1	123	376
1974	11	16.3	184	574
1975	30	16.9	171	572
1976	25	17.6	247	418
1977	11	15.6	87	821
1978	27	15.8	51	763
1979	21	16.2	122	717
1980	14	16.0	74	578

simple climatic variables and the year. All the explanatory variables contribute significantly to the regression, as all  $p$ -values are small. As viticultural experience suggests, the coefficients of winter rain and temperature are positive (high winter rain and temperature increase quality) and the coefficient of harvest rain is negative (rain during harvest decreases quality). The negative coefficient for `year` reflects the fact that the older wines are more valuable.

The next step is to plot residuals:

```
> plot(wine.reg)
```

When we examine the residual plots (shown in Figure 3.19) there are certain aspects of the fit that are not very satisfactory. The residual versus fitted value plot shows that the large residuals are all positive. The curve in the normal plot of the residuals also shows that the residuals are right skewed. Also, apart from two largish residuals having small fitted values, there is something of a curved “funnel effect” in the residuals versus fitted values plot. Most of the large residuals correspond to large fitted values. Also, there are two points with large Cook’s distances.

This suggests that a transformation of the response `price` might be effective. To choose a suitable power, we can use the Box-Cox technique. We type

```
boxcoxplot(price~temp+h.rain+w.rain+year, data=wine.df)
```

The plot is shown in Figure 3.20. The minimum occurs at a power of about -0.3, which is reasonably close to a log. If we transform the price using logs, and refit, we get a better  $R^2$  (83%) and better residual plots, which are shown in Figure 3.21.

```
> trans.lm<-lm(log(price)~temp + h.rain + w.rain + year, data=wine.df)
> summary(trans.lm)
```

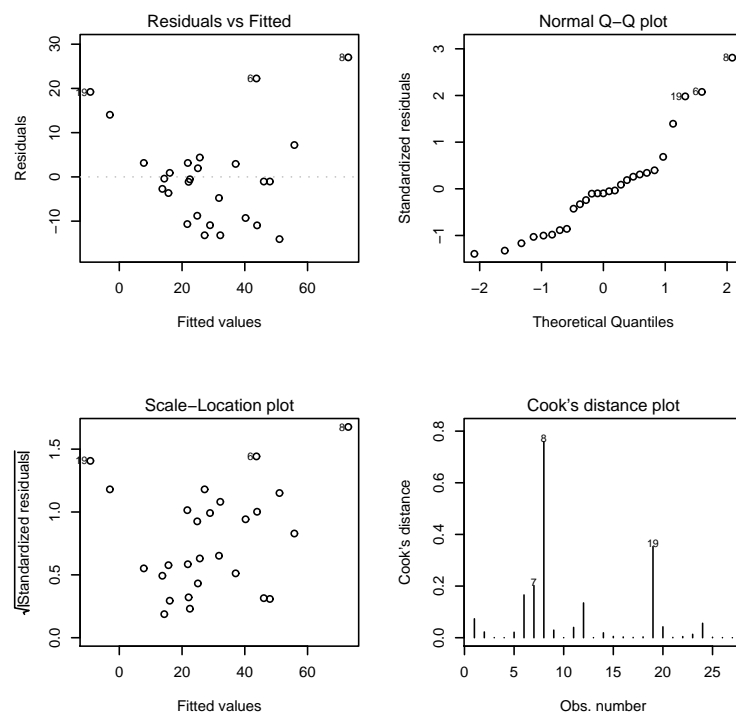


Figure 3.19: Residual plots for the wine data.

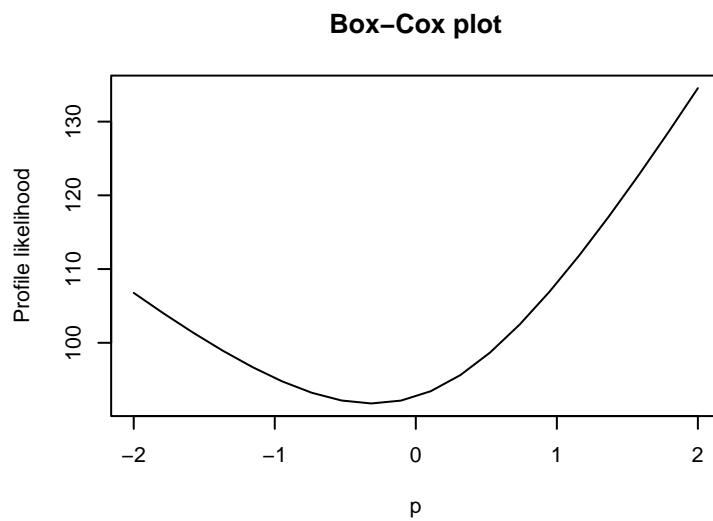


Figure 3.20: Box-Cox plot for the wine data.

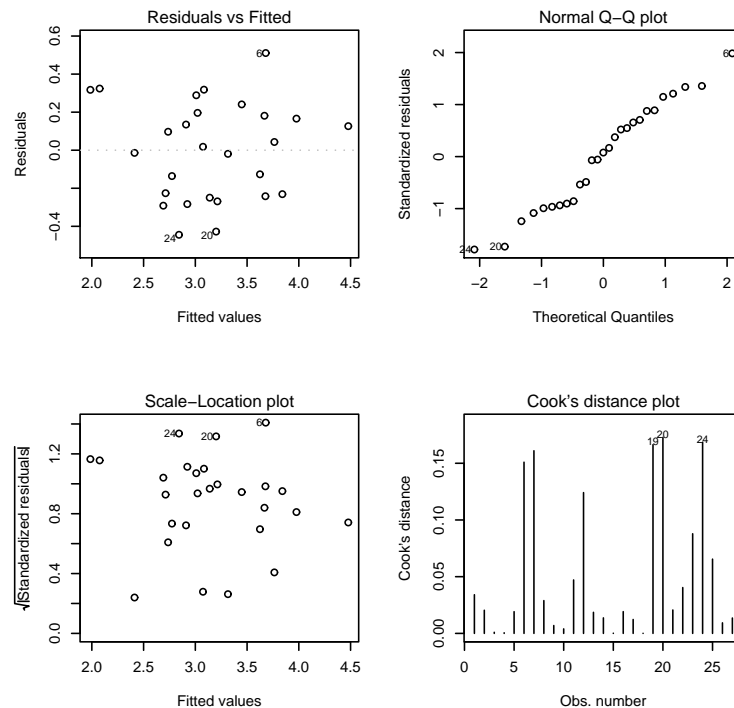


Figure 3.21: Residual plots for the transformed wine data.

```
Call:
lm(formula = log(price) ~ temp + h.rain + w.rain + year, data = wine.df)
Residuals:
    Min       1Q   Median       3Q      Max
-0.4449 -0.2362  0.0181  0.1888  0.5100
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 40.6777920  14.3374580   2.837  0.00959 **
temp         0.6187109   0.0943199   6.560 1.35e-06 ***
h.rain      -0.0037482   0.0007915  -4.736  0.00010 ***
w.rain       0.0011972   0.0004740   2.526  0.01924 *
year        -0.0243519   0.0069947  -3.481  0.00212 **

Residual standard error: 0.2805 on 22 degrees of freedom
Multiple R-Squared:  0.8308,    Adjusted R-squared:  0.8001
F-statistic: 27.01 on 4 and 22 DF,  p-value: 3.294e-08
> plot(trans.reg)
```

### Lack of independence

If the errors are not independent, we use special time series methods to estimate the parameters. These are discussed in more detail in STATS 326. Essentially, we fit a regression model where the errors are an autoregressive process, as described in the subsection “Investigating the dependence” in Section 3.4. This is done using the `arima` function. To fit the model to the sales data, we type

```
> arima(sales, order=c(1,0,0), xreg=cbind(spend, prev.spend))
Call: arima(x = sales, order = c(1, 0, 0), xreg = cbind(spend, prev.spend))
Coefficients:
      ar1  intercept    spend  prev.spend
   0.4966   16.9080   0.1218    0.1391
s.e.  0.1580    1.6716   0.0308    0.0316
```

`sigma^2` estimated as 9.476: `log likelihood = -89.16, aic = 188.32`

The estimated coefficients are in the top line of this display. The estimate of  $\rho$  (in the “ar1” column) is 0.4966. Compare these estimates and standard errors with the previous analysis. The coefficients haven’t changed much, but the standard error of the intercept is a bit higher.

### Outliers and high leverage points

In dealing with outliers, we have two choices. We can identify and delete the offending points, or we can use “robust” fitting methods that are not as affected by outliers as least squares. In the first method, we use the methods of the previous section to identify the outliers, and refit the model by least squares after deleting (or correcting, in the case of typographical errors) the offending points. However, we should resist the temptation to delete too many points, since this gives a false impression of how well the model fits.

The second method, based on fitting techniques that are not so affected by outliers, is described in Section 3.11.

### 3.5.2 Summary of corrective action

1. Dealing with a curved regression surface:
  - Transform response and/or explanatory variables
  - Use the plots in Section 3.4 to help select a power
  - Alternatively fit a polynomial surface
2. Dealing with non-normality
  - Box-Cox transformation
3. Allowing for independence
  - Fit a time series model using `arima`
4. Dealing with outliers/influence points
  - Delete and refit
  - Use robust regression (see Section 3.11)
5. Dealing with unequal variances
  - Transform response
  - Get power from plot of log sds/log means
  - model variances, use weighted least squares (`funnel`)

## 3.6 Case study 1: Yield of a chemical process

The data in Table 3.11 were gathered in the course of studying the yield of a particular chemical process. The investigators were interested in the relationships between the yield and the other variables. For example, are higher yields associated with higher or lower flows? Higher or lower conversion percentages? Also, chemical theory predicts that the variable Ratio should enter the regression as a reciprocal. (i.e. we should use the variable  $1/\text{Ratio}$  instead of Ratio). Do the data support this idea? Would some other transformation of Ratio be better? Theory also predicts that a term in  $\text{Conversion} \times \text{Flow}$  should be in the model.

The first step, is always to draw some exploratory graphics to get a rough idea of what is going on. Since all the variables in this example are continuous, we might try drawing some trellis plots and doing some spinning. We don’t show these (try them yourselves), but the general impression is that the regression surface is curved and that point 6 is an outlier in the ratio variable. Lets press on and fit a basic model:

Table 3.11: Data on a chemical process.

Observation	Yield	Conversion(%)	Flow	Ratio
1	55.45	11.79	118.9	0.155
2	54.83	11.95	105.0	0.089
3	52.21	12.14	97.0	0.094
4	50.40	12.06	101.0	0.108
5	49.32	12.04	44.0	0.100
6	41.36	12.28	30.0	0.036
7	49.28	12.36	38.0	0.113
8	47.89	12.22	32.0	0.123
9	52.40	11.90	220.0	0.135
10	52.62	11.34	350.0	0.183
11	59.34	11.20	160.0	0.166
12	45.32	12.03	328.0	0.221
13	48.97	12.04	325.0	0.192
14	49.87	12.02	330.0	0.188
15	47.06	12.02	330.0	0.201
16	40.05	12.05	322.0	0.153
17	51.29	12.05	322.0	0.194
18	43.73	12.14	326.0	0.097
19	51.44	12.05	366.0	0.136
20	49.33	12.13	350.0	0.143
21	51.85	11.94	510.0	0.116
22	52.33	12.02	513.0	0.195
23	47.72	12.02	523.0	0.160
24	46.65	11.80	430.0	0.164
25	49.83	11.40	325.0	0.197
26	51.36	11.19	380.0	0.233
27	51.19	11.19	380.0	0.211
28	48.10	11.18	383.0	0.222
29	48.28	11.21	375.0	0.223
30	51.28	11.21	375.0	0.229
31	47.81	11.88	410.0	0.170
32	46.78	12.09	485.0	0.163
33	44.24	11.97	445.0	0.153
34	53.07	11.30	232.0	0.180
35	54.08	10.90	220.0	0.126
36	54.19	10.90	223.0	0.152
37	54.26	10.90	222.0	0.184
38	55.94	10.80	240.0	0.225
39	55.85	10.90	238.0	0.169
40	56.57	10.80	258.0	0.161
41	53.85	10.75	272.0	0.197
42	54.81	10.72	280.0	0.201
43	54.14	11.10	426.0	0.221
44	53.74	11.12	404.0	0.215

```
> chem.reg<-lm(yield~.,data=chem.df)
# NB . means include all the variables in the model
> summary(chem.reg)
Coefficients:
              Value Std. Error  t value Pr(>|t|)
(Intercept)  113.0113   14.6319    7.7236  0.0000
conversion    -5.1894    1.1497   -4.5139  0.0001
flow         -0.0070    0.0045   -1.5633  0.1259
ratio        -0.0558   15.7558   -0.0035  0.9972
Residual standard error: 3.118 on 40 degrees of freedom
Multiple R-Squared:  0.46
F-statistic: 11.36 on 3 and 40 degrees of freedom,
the p-value is 1.591e-05
```

On the face of it, only conversion appears to contribute to the regression. Low yield would seem to be associated with a high conversion percentage, since the coefficient of “conversion” is negative. Flow and ratio do not seem to have much effect, since the  $p$ -values are small.

Do the data support the idea that yield depend on the reciprocal (ie 1/over) of ratio? Would some other transformation of ratio be better? Theory also predicts that a term in Conversion  $\times$  Flow should be in the model. Is this correct? First fit a model with 1/ratio instead of ratio:

```
> recip.reg<-lm(yield~conversion+flow+I(1/ratio),data=chem.df) #note use of I()
> summary(recip.reg)
Coefficients:
              Value Std. Error  t value Pr(>|t|)
(Intercept)  105.9357   10.6944    9.9057  0.0000
conversion    -4.2631    0.9618   -4.4322  0.0001
flow         -0.0115    0.0039   -2.9439  0.0054
I(1/ratio)    -0.3455    0.1560   -2.2148  0.0325
Residual standard error: 2.943 on 40 degrees of freedom
Multiple R-Squared:  0.519
F-statistic: 14.39 on 3 and 40 degrees of freedom, the p-value is 1.663e-06
```

According to the  $p$ -value and the  $R^2$  using 1/ratio looks pretty good! What about other transformations? Let's try some gam plots

```
> layout(2,2)
> plot(gam(yield~s(conversion)+s(flow)+s(ratio),data=chem.df))
```

It looks like conversion doesn't need transforming, flow could be modelled as a cubic, and ratio as a quadratic (or possibly a quartic). Lets fit some models and see:

```
> mod1.reg<-lm(yield~conversion+poly(flow,3)
               +poly(ratio,2),data=chem.df)
> summary(mod1.reg)
Coefficients:
              Value Std. Error  t value Pr(>|t|)
(Intercept)   99.1211   12.7559    7.7706  0.0000
conversion    -4.1699    1.0976   -3.7990  0.0005
poly(flow, 3)1 -10.5183    3.4596   -3.0403  0.0043
poly(flow, 3)2  3.6636    3.3372    1.0978  0.2794
poly(flow, 3)3  10.0990    3.1038    3.2537  0.0024
poly(ratio, 2)1  6.9578    4.2540    1.6356  0.1104
poly(ratio, 2)2 -4.5273    3.0141   -1.5020  0.1416
Residual standard error: 2.557 on 37 degrees of freedom
Multiple R-Squared:  0.6641
F-statistic: 12.19 on 6 and 37 degrees of freedom,
the p-value is 1.592e-07
```

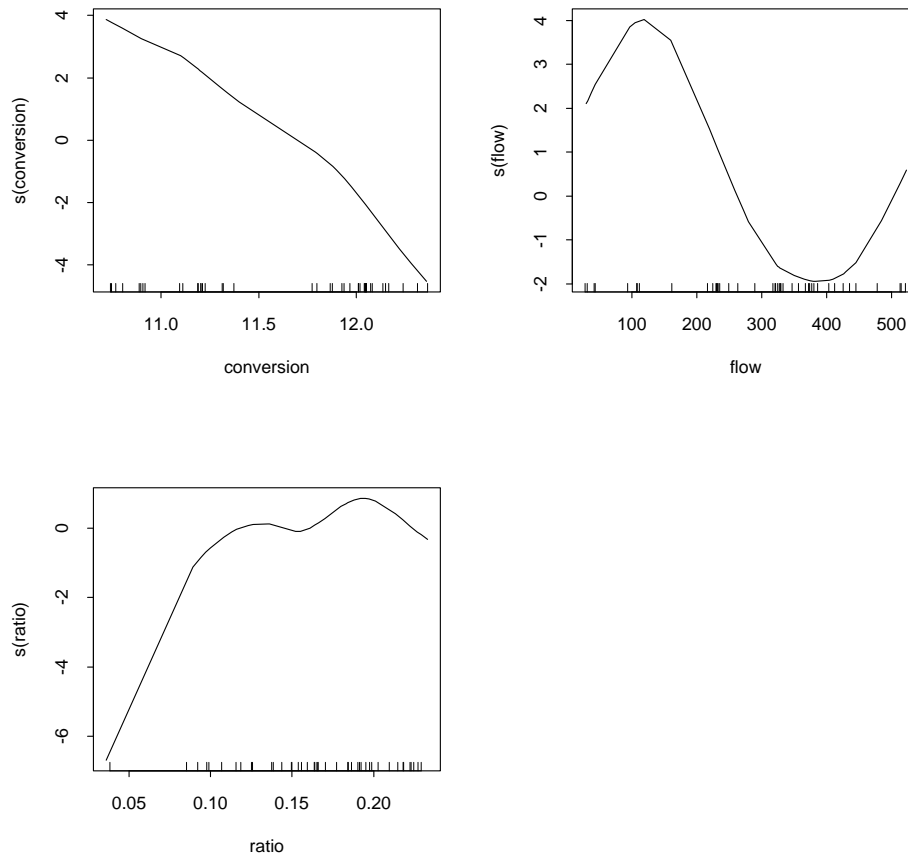


Figure 3.22: Gam plots.

The  $R^2$  has improved a lot to 66%, and flow now is significant, indicating a cubic in flow is a useful addition to the model. It is a bit of a worry that ratio seems to be not significant, although a quadratic was strongly indicated in the gam plots. Lets do a model/submodel test to see if the ratio terms can be dropped:

```
> mod2.reg<-lm(yield~conversion+poly(flow,3),data=chem.df)
> anova(sub.reg,full.reg)
Analysis of Variance Table
```

Response: yield

	Terms	Resid. Df	RSS
1	conversion +poly(flow,3)	39	272.7760
2	conversion+poly(flow,3)+poly(ratio,2)	37	241.9044

	Test Df	Sum of Sq	F Value	Pr(F)
1				
2 +poly(ratio, 2)	2	30.87156	2.360948	0.108392

```
>
> summary(mod2.reg)
```

Coefficients:

	Value	Std. Error	t value	Pr(> t )
(Intercept)	104.4106	11.1228	9.3871	0.0000
conversion	-4.6253	0.9569	-4.8335	0.0000

```
poly(flow, 3)1  -6.0147    2.6610   -2.2603    0.0295
poly(flow, 3)2   0.8144    3.1734    0.2566    0.7988
poly(flow, 3)3  10.9140    2.7354    3.9899    0.0003
```

Residual standard error: 2.645 on 39 degrees of freedom

Multiple R-Squared: 0.6213

F-statistic: 15.99 on 4 and 39 degrees of freedom,  
the p-value is 7.858e-08

From the F-test ( $p$ -value is 0.108) it seems that the submodel is adequate, and the terms in ratio can be dropped. Thus we have a paradox: if we put in  $1/\text{ratio}$  it is significant, but if we put in a quadratic in ratio it is not. Arrrgg###!!\$????? But remember that point 6 is an outlier....

Let's press on anyway and add a term in  $\text{conversion} \times \text{flow}$  to the two contending models (i.e. with and without  $1/\text{ratio}$ , and a cubic in flow):

```
> summary(lm(yield~conversion+poly(flow,3)+I(1/ratio)
+I(flow*conversion),data=chem.df))
```

Coefficients:

	Value	Std. Error	t value	Pr(> t )
(Intercept)	99.4292	10.6763	9.3131	0.0000
conversion	-0.1866	4.1155	-0.0453	0.9641
poly(flow, 3)1	127.6243	139.0612	0.9178	0.3647
poly(flow, 3)2	1.7394	3.7595	0.4627	0.6463
poly(flow, 3)3	13.3002	4.1724	3.1876	0.0029
I(1/ratio)	-0.3540	0.1409	-2.5128	0.0165
I(flow * conversion)	-0.0130	0.0131	-0.9929	0.3272

Residual standard error: 2.47 on 37 degrees of freedom

Multiple R-Squared: 0.6866

F-statistic: 13.51 on 6 and 37 degrees of freedom,  
the p-value is 4.7e-08

```
> summary(lm(yield~conversion+poly(flow,3)+I(flow*conversion)
,data=chem.df))
```

Coefficients:

	Value	Std. Error	t value	Pr(> t )
(Intercept)	101.6953	11.3577	8.9539	0.0000
conversion	0.1263	4.3918	0.0288	0.9772
poly(flow, 3)1	157.9020	147.9085	1.0676	0.2925
poly(flow, 3)2	-1.4682	3.7753	-0.3889	0.6995
poly(flow, 3)3	14.7571	4.4114	3.3452	0.0019
I(flow * conversion)	-0.0155	0.0140	-1.1084	0.2747

Residual standard error: 2.637 on 38 degrees of freedom

Multiple R-Squared: 0.6331

F-statistic: 13.12 on 5 and 38 degrees of freedom,  
the  $p$ -value is 1.917e-07

It seems that the term in  $\text{conversion} \times \text{flow}$  seems unnecessary in either model, since the  $p$ -value in either model is large.

Lets provisionally entertain the model that includes  $1/\text{ratio}$ . We will examine this model for influence.

```
> recip.reg<-lm(yield~conversion+poly(flow,3)+I(1/ratio),data=chem.df)
> diag.plots(recip.reg)
> influence.measures(recip.reg)
```

	.Intercept.	conversion	poly.flow..3.1	poly.flow..3.2	poly.flow..3.3
6	-0.184	0.272	-0.264	0.043	0.219
7	-0.010	0.017	-0.091	0.067	-0.050



8	-0.029	-0.004	0.400	-0.336	0.267	
16	0.833	-0.836	-0.233	0.830	-0.014	
	I.1.ratio.	dffits	cov.ratio	cooks.d	hats	inf
6	-1.075	-1.285	8.077	0.281	0.861	*
7	-0.065	0.125	1.630	0.003	0.285	*
8	0.305	-0.551	1.667	0.051	0.356	*
16	-0.080	-1.174	0.193	0.172	0.091	*

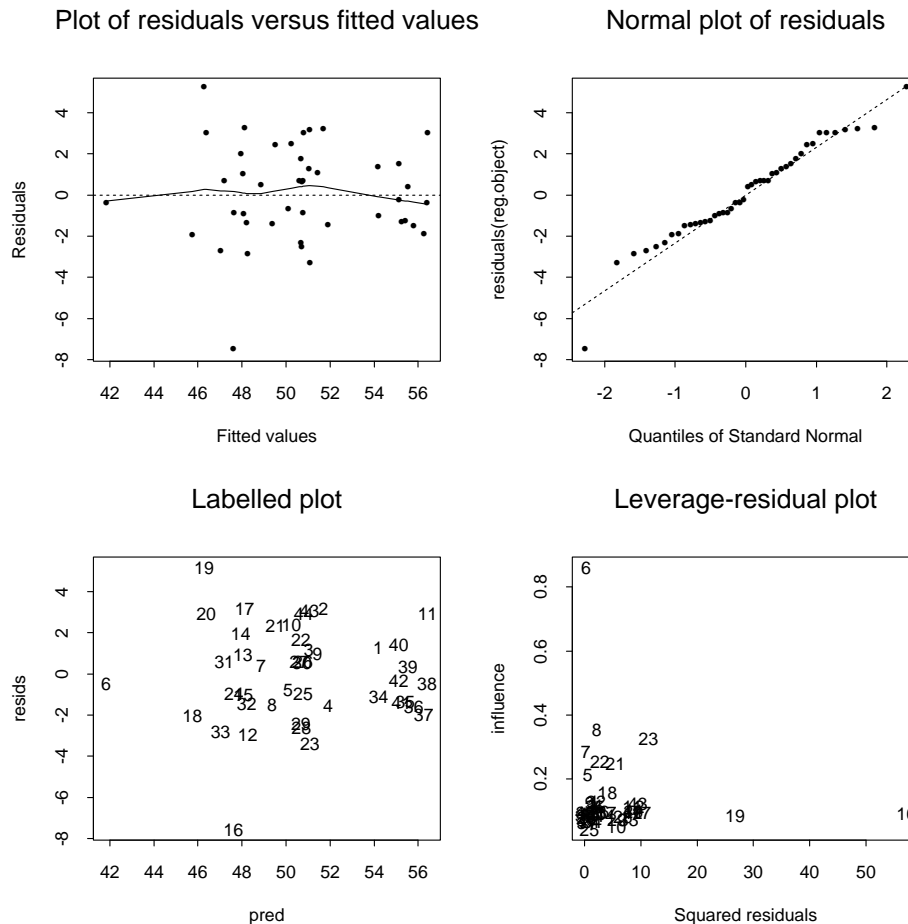


Figure 3.23: Residual plots for detecting non-linearity and non-normality.

(We have deleted all the output for unstarred points.) We see that point 6 is very influential. If we examine the original data, we see that the value for ratio for case 6 is 0.036. When we take reciprocals, we get  $1/0.036=27.77$ . The next biggest reciprocal is  $1/0.089=11.23$ . It is clear that point 6 is very influential (as we can see from the LR plot.) The DFFBETAS for the variable  $1/\text{ratio}$  is more than 1 in magnitude. Also the hat matrix diagonal is very large.  $(3(k+1)/n = 18/44 = 0.409)$  as is the covariance ratio. We also note the big outlier at point 16. Let's delete these two points and refit:

```
> summary(lm(yield~conversion+poly(flow,3)+I(1/ratio),
              data=chem.df[-c(6,16),]))
```

Coefficients:

	Value	Std. Error	t value	Pr(> t )
(Intercept)	96.0543	9.8763	9.7257	0.0000
conversion	-3.7628	0.9155	-4.1102	0.0002
poly(flow, 3)1	-9.2321	2.7876	-3.3119	0.0021
poly(flow, 3)2	2.0092	2.6767	0.7506	0.4577

```
poly(flow, 3)3    9.0940    2.3355    3.8939    0.0004
I(1/ratio)   -0.2025    0.2828   -0.7159    0.4787
```

Residual standard error: 2.155 on 36 degrees of freedom

Multiple R-Squared: 0.6727

F-statistic: 14.8 on 5 and 36 degrees of freedom,  
the p-value is 6.765e-08

Now we can see what is going on. The significance of (1/ratio), which seemed puzzling before, is entirely due to the influential point 6. Deleting this point leads us to the conclusion that ratio (transformed or not) is not required in the model, since the  $p$ -value is large.

The final step is to fit the model using conversion and the polynomial in flow:

```
> summary(lm(yield~conversion+poly(flow,3),data=chem.df[-c(6,16),]))

Call: lm(formula = yield ~ conversion + poly(flow, 3), data = chem.df[ - c(6, 16),
])

Residuals:
    Min       1Q   Median       3Q      Max
-4.18  -1.444  0.2131  1.717  4.039
```

Coefficients:

	Value	Std. Error	t value	Pr(> t )
(Intercept)	98.4299	9.2406	10.6518	0.0000
conversion	-4.0787	0.7968	-5.1188	0.0000
poly(flow, 3)1	-7.9682	2.1429	-3.7183	0.0007
poly(flow, 3)2	1.5336	2.5758	0.5954	0.5552
poly(flow, 3)3	8.5939	2.2138	3.8820	0.0004

Residual standard error: 2.141 on 37 degrees of freedom

Multiple R-Squared: 0.668

F-statistic: 18.61 on 4 and 37 degrees of freedom, the p-value is 1.844e-08

Correlation of Coefficients:

	(Intercept)	conversion	poly(flow, 3)1	poly(flow, 3)2
conversion	-0.9994			
poly(flow, 3)1	-0.0464	0.0464		
poly(flow, 3)2	0.5558	-0.5562	-0.0258	
poly(flow, 3)3	-0.2548	0.2550	0.0118	-0.1418

This is a reasonable final model, explaining 67% of the variation.

### 3.7 Multicollinearity

Assuming all the model assumptions hold, how well can we estimate the regression coefficients? When we only have one explanatory variable, the accuracy of estimating the slope of the true regression line is summed up by its standard error

$$\frac{\sigma}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}}.$$

This formula suggests that the accuracy with which we can estimate the coefficients is partly determined by the extent of the scatter about the true regression line (as measured by  $\sigma$ ) but also by the configuration of the  $x$ 's: the more spread out the  $x$ 's, the larger is  $\sum_{i=1}^n (x_i - \bar{x})^2$ , hence the smaller the standard error and the more accurate the estimate. Intuitively, when the  $x$ 's are spread out, the regression line is "well supported" and the least squares line estimates the true line very well.

On the other hand, if the  $x$ 's are bunched together, then the least squares line is not well supported. It is rather like a see-saw balancing on a single point, and a very small change in the data will make a big change in the slope, just as a small movement on a see-saw can produce a

big change! To illustrate the difference between these two situations, we generated some data from the model

$$y_i = \beta_0 + \beta_1 x_i + e_i, \quad i = 1, \dots, 10.$$

We set  $\beta_0 = 1$ ,  $\beta_1 = 2$  and  $\sigma = 2$ . We looked at two cases: first where the ten  $x$ 's were equally spaced from 1 to 10, so that  $\sum_{i=1}^n (x_i - \bar{x})^2 = 82.5$ , and second when the  $x$ 's were equally spaced from 4.6 to 5.5, so that  $\sum_{i=1}^n (x_i - \bar{x})^2 = 0.825$ . We generated 20 sets of data for each of these two cases. For each set, we calculated the regression line and drew in on a graph. The graphs for the two cases are shown in Figure 3.24. We see in the left-hand plot that the twenty regression lines are all quite similar - there is not much variation from sample to sample. The situation is quite different in the right-hand plot, when the  $x$ 's are tightly clustered. Now the lines are extremely variable: some even have a negative slope!

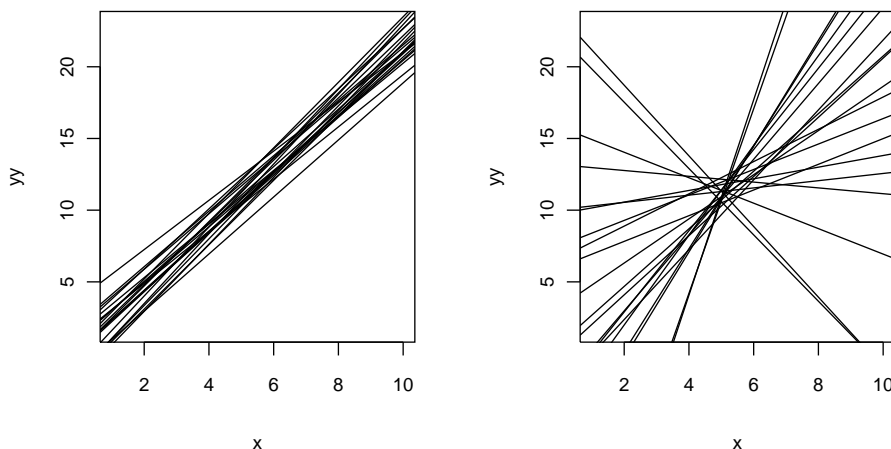


Figure 3.24: Instability of least squares lines. Left panel: stable case. Right panel: unstable case

What happens with two explanatory variables? The regression coefficients tend to be unstable when the explanatory variables are highly correlated and there is a strong linear relationship between them. In this case, we should be cautious about interpreting the coefficients.

When the explanatory variables are highly correlated, the data tends to form a “knife edge” in three dimensional space, as illustrated in Figure 3.25. When we try to fit the least squares plane, the plane is balancing on the knife edge and is unstable, so that a small change in the data produces a large change in the fitted plane. On the other hand, if the explanatory variables are uncorrelated (or *orthogonal*), then they will be well spread out and support the fitted plane well. In this case small changes in the data will produce only small changes in the fitted plane.

We can do another simulation to illustrate this. Suppose we have a model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + e_i, \quad i = 1, \dots, 10.$$

We set  $\beta_0 = \beta_1 = \beta_2 = 1$  and  $\sigma = 0.2$ . We again look at two cases: first where the ten pairs of  $x$ 's were almost uncorrelated, with  $r = -0.0519$ , and the second where the correlation was high at 0.9944. We illustrate the two configurations in Figure 3.26. Also, in Figure 3.27, we show the true regression plane, and in Figure 3.28 we show the result of fitting the model to 18 sets of data simulated from the model above. The first nine sets have the low-correlation configuration of  $x$ 's, and the fitted plane is stable, accurately approximating the true plane. In the bottom nine plots, the data sets are generated using the correlated  $x$ 's, and the fitted planes are very unstable.

This phenomenon is called *multicollinearity*. Whenever the explanatory variables cluster about a point (in the case of a single explanatory variable) or about a line (in the case of two) or about a hyperplane of dimension less than  $k$  in the case of  $k$ , then the fitted line or plane is not well

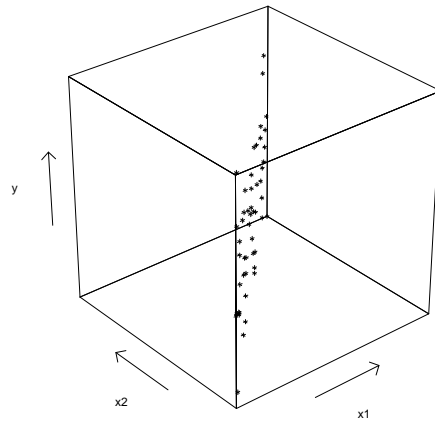


Figure 3.25: Three-dimensional scatterplot of data having high correlations between the explanatory variables

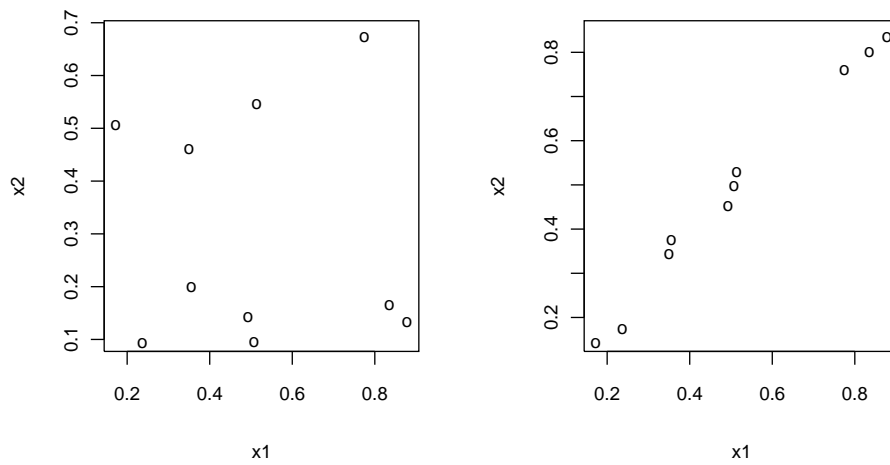


Figure 3.26: Configurations of  $x_1$  and  $x_2$  for the simulation. Left panel: correlation -0.0519. Right panel: correlation 0.9944.

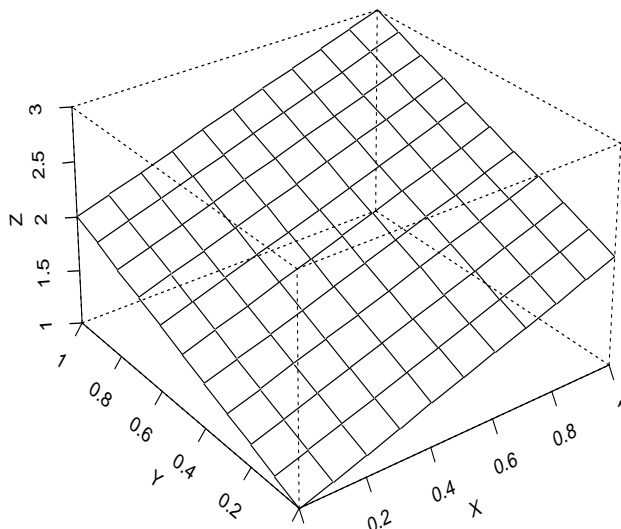


Figure 3.27: The true regression plane.

supported, and the standard errors of the regression coefficients are inflated, with small changes in the data producing large changes in the fitted line or plane. We should be wary about interpreting the estimated coefficients in this case. For example, when there are two explanatory variables, it can be shown that the standard errors of the coefficients are proportional to  $(1 - r^2)^{-\frac{1}{2}}$ , where  $r$  is the correlation between the explanatory variables. A correlation close to  $\pm 1$  will result in a large inflation of the standard errors of the coefficients.

In terms of the matrix representation of the regression model, multicollinearity occurs when the columns of the matrix

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix},$$

are almost linearly related. In this case the elements of the inverse matrix  $(X^T X)^{-1}$  will be large. Since the standard errors of the regression coefficients are proportional to the square roots of the diagonal elements of this matrix, this will inflate the standard errors and result in unstable fits.

Often multicollinearity occurs when one of the explanatory variables has very little variation, and is thus almost constant. If a constant term is included in the model, we are effectively trying to fit two constants, and variance inflation will result.

In addition, the magnitudes (and hence the variances) of the regression coefficients depend on the units of measurement of the corresponding explanatory variable as well as the possible relationships between the columns of  $X$ . We can think of multicollinearity caused by these two factors as inessential, since we can remove it by *standardising* the data: that is transforming each variable so that it has zero mean and either unit standard deviation (s.d. scaling) or zero mean and unit sum of squares (unit norm scaling). Thus in the first case we are making the transformation

$$x'_{ij} = (x_{ij} - \bar{x}_j) / \text{s.d.}(x_j)$$

where  $\bar{x}_j$  and  $\text{s.d.}(x_j)$  are the mean and standard deviation of the  $j$ th variable, and in the second case

$$x'_{ij} = (x_{ij} - \bar{x}_j) / \left\{ \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2 \right\}^{\frac{1}{2}}.$$

“Essential” multicollinearity is what remains after standardisation, and is caused by approximate linear relationships between the explanatory variables. We can recognise which if any of the

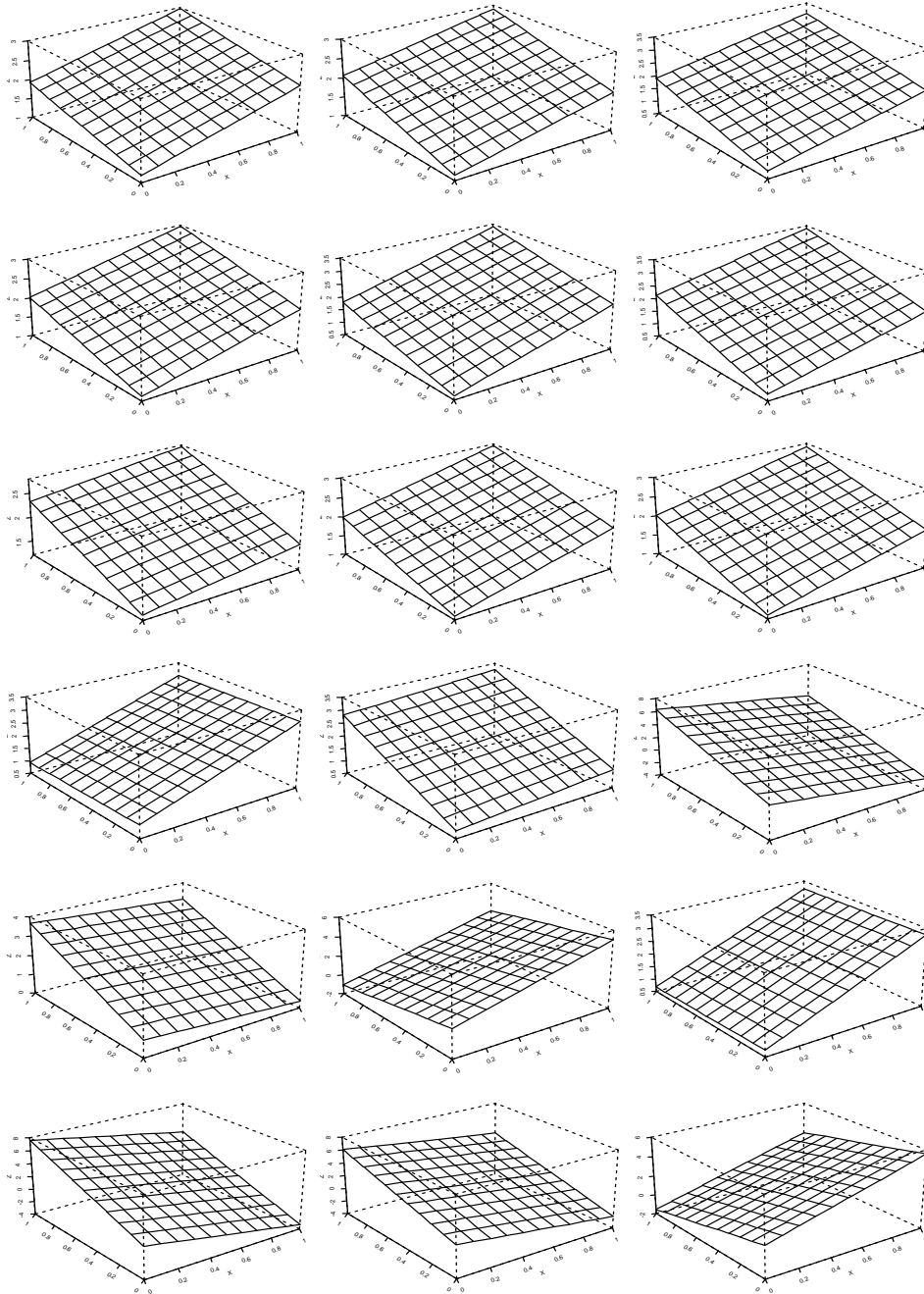


Figure 3.28: Fitted planes.

variables are affected by calculating *variance inflation factors* or VIF's, which are just the diagonal elements of  $(X^T X)^{-1}$  after we have centred and scaled the explanatory variables.

Given that we have standardised our explanatory variables, the smallest possible variances will occur when the explanatory variables are uncorrelated. In this case the VIF is 1 for each variable. If we calculate the VIF's for our regression, and find that some or all of the VIF's are much larger than one, then we know that we have a multicollinearity problem: our regression coefficients will be too inaccurate to be of much use.

What can we do about multicollinearity? It occurs when there are approximate linear relationships between the explanatory variables. Under these circumstances, if one variable is related to some of the others, it will make little or no extra contribution to the model, so it can be dropped. This will destroy the linear relationship between the explanatory variables and solve the multicollinearity problem.

Which linear relationships are causing the problem? If there is a linear relationship between just two of the explanatory variables, then we can tell which two variables are involved by examining the correlations between the explanatory variables: the offending pair of variables will have a correlation close to  $\pm 1$ . If there is a relationship between three or more variables, we need to use some matrix theory to identify the relationship. This (optional) theory is presented below. First, we look at an example.

**Example 16.** In Example 2, we looked at a regression model for soil evaporation, using the explanatory variables `avat` (average air temperature, `avh` (average humidity) and `wind`. These variables were contained in the data frame `evap.df` in the R330 package. In actual fact the data frame contains more variables: minimum and maximum soil temperature, and minimum and maximum air temperature. We might suspect that there is substantial correlation between the six temperature variables, since they are all measuring approximately the same thing.

We can fit the model using all eight explanatory variables (six temperature variables, plus humidity and wind).

```
> evap.lm<-lm(evap~maxst+minst+avst+maxat+minat+avat
+              +avh+wind,data=evap.df)

> summary(evap.lm)

Call:
lm(formula = evap ~ maxst + minst + avst + maxat + minat + avat +
    avh + wind, data = evap.df)

Residuals:
    Min       1Q   Median       3Q      Max
-17.1816  -1.9813   0.7584   3.1655  13.0982

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  57.612008   64.006779   0.900   0.3739
maxst         1.108602    0.706811   1.568   0.1253
minst        -0.111546    1.091838  -0.102   0.9192
avst         -0.494191    0.313478  -1.576   0.1234
maxat         0.358649    0.564430   0.635   0.5291
minat         0.110547    0.719427   0.154   0.8787
avat         0.311029    0.159195   1.954   0.0583 .
avh          -0.325183    0.066200  -4.912 1.85e-05 ***
wind          0.011589    0.009006   1.287   0.2061
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.547 on 37 degrees of freedom
Multiple R-Squared:  0.8355,    Adjusted R-squared:  0.7999
F-statistic: 23.49 on 8 and 37 degrees of freedom,    p-value: 2.719e-12
```

We note that the regression is very significant (the  $p$ -value is very small at 2.719e-12) but most of the temperature variables are not significant. This indicates that each may be dropped, provided the others are retained. In our earlier regression, the single temperature variable was strongly significant, so it is clear that temperature helps prediction. However, we don't seem to need all the temperature variables.

Let's calculate the variance inflation factors. There are various ways to do this in R, but the easiest is to invert the matrix of correlation coefficients: assuming we have centred and scaled the variables, the elements of the matrix  $X^T X$  corresponding to  $\beta_1, \dots, \beta_k$  are the correlations between the explanatory variables. If we type

```
> X<-evap.df[, -9]
```

we form a matrix of data consisting of the explanatory variables alone. We can get the matrix  $X^T X$  using the function `cor`, invert it using `solve`, and extract the diagonal elements using `diag`:

```
> VIF<-diag(solve(cor(X)))
[1] 19.243449 13.594342 41.571247 8.582273 7.301924 11.697691 3.997904
[8] 1.892209
```

All the temperature variables have high variance inflation factors (several more than 10), indicating that some should be dropped. Note that, after centring and scaling, the estimate of  $\beta_0$  is  $\bar{y}$ , and its inflation factor is 1.

Which variables, if any, should be dropped? We discuss this question in Section 3.8. Which linear relationships are causing the problems? We can look at the correlations:

```
> round(cor(X),3)
      maxst minst avst maxat minat avat avh wind
maxst 1.000 0.851 0.953 0.906 0.466 0.825 -0.757 -0.092
minst 0.851 1.000 0.933 0.840 0.681 0.818 -0.482 0.030
avst 0.953 0.933 1.000 0.914 0.594 0.870 -0.679 -0.093
maxat 0.906 0.840 0.914 1.000 0.570 0.874 -0.655 -0.090
minat 0.466 0.681 0.594 0.570 1.000 0.783 -0.065 0.411
avat 0.825 0.818 0.870 0.874 0.783 1.000 -0.537 0.128
avh -0.757 -0.482 -0.679 -0.655 -0.065 -0.537 1.000 0.223
wind -0.092 0.030 -0.093 -0.090 0.411 0.128 0.223 1.000
```

(Note we have rounded to three decimal places to make reading easier.) Note the high correlations among the temperature variables, indicating in particular a strong relationship between the three soil temperature readings. However, for a more detailed look at these relationships, we need to study the matrix in more depth.

### 3.7.1 Mathematics of multicollinearity<sup>5</sup>

Suppose we have centred and standardised our data. Then for  $j = 1, 2, \dots, k$  the standard error of  $\hat{\beta}_j$  is the square root of the  $j - j$ th element of the matrix

$$\sigma^2(X^T X)^{-1},$$

where  $X$  is the matrix of standardised explanatory variables. How do linear relationships between the columns of  $X$  affect the size of  $\hat{\beta}_j$ ? Equivalently, how do they affect the  $j$ th diagonal element of  $(X^T X)^{-1}$ ? To answer this, we need to look at the eigenvalues and eigenvectors of  $X^T X$ .

An *eigenvalue* of  $X^T X$  is a number  $\lambda$  such that

$$X^T X \mathbf{u} = \lambda \mathbf{u} \tag{3.10}$$

for some non-zero vector  $\mathbf{u}$ . We can assume that  $\mathbf{u}$  has *unit length* i.e.  $|\mathbf{u}| = (\mathbf{u}^T \mathbf{u})^{\frac{1}{2}} = 1$ .  $\mathbf{u}$  is called the *eigenvector* corresponding to  $\lambda$ . The matrix  $X^T X$  is  $k \times k$  and almost always in

<sup>5</sup>This section is included for enthusiasts only – it is not examinable.



regression will have  $k$  positive eigenvalues  $\lambda_1 > \lambda_2 > \dots > \lambda_k > 0$ , with corresponding eigenvectors  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k$  say.

Let  $U = [\mathbf{u}_1, \dots, \mathbf{u}_k]$  be the matrix whose columns are the eigenvectors. Then the equations

$$X^T X \mathbf{u}_j = \lambda_j \mathbf{u}_j \quad j = 1, 2, \dots, k$$

can be written in matrix form as

$$X^T X U = U \Lambda \quad (3.11)$$

where  $\Lambda$  is a diagonal matrix with diagonal elements  $\lambda_1, \dots, \lambda_k$ .

Now it is a property of the eigenvectors  $\mathbf{u}_1, \dots, \mathbf{u}_k$  that they are orthogonal i.e.  $\mathbf{u}_j^T \mathbf{u}_{j'} = 0$  when  $j \neq j'$ . Thus  $U^T U = I$ , a  $k \times k$  identity, so that  $U^{-1} = U^T$ . Post multiplying (3.11) by  $U^T$  gives

$$X^T = U \Lambda U^T$$

so

$$\begin{aligned} (X^T X)^{-1} &= (U \Lambda U^T)^{-1} \\ &= (U^T)^{-1} \Lambda^{-1} U^{-1} \\ &= U \Lambda^{-1} U^T \end{aligned}$$

where  $\Lambda^{-1}$  is a diagonal matrix with entries  $\lambda_1^{-1}, \lambda_2^{-1}, \dots, \lambda_k^{-1}$ .

Now let  $U = (u_{j\ell})$ . Then the  $j, j$  element of  $(X^T X)^{-1}$  is the  $j, j$ th element of  $U \Lambda^{-1} U^T$  i.e. is equal to  $\sum_{\ell=1}^k u_{j\ell}^2 / \lambda_\ell$ , so that

$$\text{s.e.}(\hat{\beta}_j) = \sigma \sqrt{\sum_{\ell=1}^k u_{j\ell}^2 / \lambda_\ell}.$$

Since  $\sum_{j=1}^k u_{j\ell}^2 = 1$  (the columns of  $U$  have unit length) none of the  $u_{j\ell}$ 's are very big. However, it may be that some of the  $\lambda_\ell$ 's are quite small, so that  $u_{j\ell}^2 / \lambda_\ell$  can be quite large. Thus small eigenvalues  $\lambda_\ell$  inflate the standard error of  $\hat{\beta}_j$ .

When do we get small eigenvalues? For any eigenvalue  $\lambda$  and corresponding eigenvector  $\mathbf{u}$  we get

$$\mathbf{u}^T X^T X \mathbf{u} = \lambda \mathbf{u}^T \mathbf{u}$$

by pre-multiplying (3.10) by  $\mathbf{u}^T$ , so that

$$\begin{aligned} \|X \mathbf{u}\|^2 &= (X \mathbf{u})^T (X \mathbf{u}) \\ &= \mathbf{u}^T X^T X \mathbf{u} \\ &= \lambda \mathbf{u}^T \mathbf{u} \\ &= \lambda \end{aligned}$$

since  $\mathbf{u}$  has unit length. Hence if  $\lambda$  is very small, then  $\|X \mathbf{u}\|^2 \approx 0$  and so  $X \mathbf{u} \approx 0$ . The equation  $X \mathbf{u} \approx 0$  means that a linear combination of the columns of  $X$  is approximately zero, the elements of  $\mathbf{u}$  being the coefficients of the linear combination: i.e. if  $\mathbf{x}_1, \dots, \mathbf{x}_k$  are the columns of  $X$ ,  $u_{1\ell}, \dots, u_{k,\ell}$  are the elements of  $\mathbf{u}_\ell$  and  $\lambda_\ell$  is small, then

$$0 \approx X \mathbf{u}_\ell = u_{1\ell} \mathbf{x}_1 + \dots + u_{k,\ell} \mathbf{x}_k$$

so that  $u_{1\ell}, \dots, u_{k,\ell}$  are the coefficients of the linear combination that is approximately zero.

### Finding the relationships causing variance inflation

From the last section we see that

*Every small eigenvalue corresponds to an approximate linear relationship between the columns of  $X$ . The coefficients of the linear relationship are the elements of the eigenvector corresponding to the small eigenvalue. It is these linear relationships that make estimation of the regression coefficients inaccurate.*

We can find out what linear relationships are causing the variance inflation by using R to calculate the eigenvalues and vectors and picking out the elements of the eigenvectors corresponding to any small eigenvalues. The next example shows how.

**Example 17.** To find which linear relationships between the explanatory variables in the soil data are causing the variance inflation, we need to calculate the eigenvalues and eigenvectors of the matrix  $X$  of the last example. We do this using the function `eigen`:

```
> eigen.stuff<-eigen(cor(X))
> eigen.stuff$values
[1] 5.40308506 1.51962009 0.57587334 0.23580044 0.13139355 0.06970962 0.04726019
[8] 0.01725771

> round(eigen.stuff$vectors,2)
      wind  avh  avat minat maxat avst minst maxst
maxst 0.41 -0.15 0.14 0.24 -0.19 0.51 0.53 0.39
minst 0.40 0.08 -0.21 0.60 0.29 -0.24 -0.41 0.35
avst 0.42 -0.08 -0.06 0.24 0.15 0.16 0.08 -0.84
maxat 0.41 -0.08 -0.05 -0.11 -0.70 -0.57 0.06 -0.04
minat 0.29 0.53 -0.31 -0.38 0.39 -0.20 0.44 0.09
avat 0.40 0.16 -0.05 -0.48 -0.14 0.47 -0.58 0.06
avh -0.29 0.42 -0.62 0.30 -0.43 0.27 0.03 -0.08
wind 0.00 0.69 0.66 0.22 -0.14 -0.02 -0.04 -0.08
```

The eigenvector corresponding to the smallest eigenvalue 0.01725771 is the eighth column of the matrix above, since 0.01725771 is the eighth element of the vector of eigenvalues. If we examine the eigenvector, the first three elements are large, and the rest are negligible. (All elements have to be less than one because the length of the eigenvectors is always 1, so the sum of the squares of the elements is 1.) This indicates a relationship between the first three variables (i.e. the soil temperature variables). The third is roughly the average of the first two. This is because the average soil temperature variable is about halfway between the maximum and the minimum temperatures. The next smallest eigenvector shows a relationship between the first two variables and the 5th and 6th i.e. between max and min soil and max and min air temperatures.

- *Multicollinearity is caused by relationships between explanatory variables*
- *Detected by big VIFs (say more than 10)*
- *Check eigenvalues and vectors of `Cor(X)`*

### 3.8 Variable selection

Suppose we have a set of potential explanatory variables that we could use to build a regression model. How do we decide which variables should enter the model? What can go wrong?

If we include too few variables, we are *underfitting*, and if we include too many variables we are *overfitting*. In both cases certain unpleasant consequences may occur. The exact consequences depend on why we are fitting the regression. We can identify two reasons for building a regression model:

- We want to build a model for prediction, without needing to interpret the coefficients;

- We are building the model to help understand how the explanatory variables affect the response. In particular we may want to see if the data support the existence of a causal relationship between the response and a particular explanatory variable.

### 3.8.1 Model building for prediction

Here the idea is to build a model with the smallest possible prediction error. We have to find a way to measure the prediction error, and then compare various models with respect to this criterion. Suppose we have a model under consideration, based on a particular set of variables, and use it to construct a prediction rule, as we did in Section 3.3. The prediction rule will depend (through the estimated regression coefficients) on both the model under consideration and also the data used to fit the model. We want to use the prediction rule to predict the  $y$ -value associated with a new  $x$ -observation. The prediction  $\hat{y}$  say, depends on the new data  $x$  and the prediction rule constructed from the old data. If  $y$  is the actual response corresponding to the new  $x$ , the squared prediction error is  $(y - \hat{y})^2$ . Averaged over *all future data*, the mean squared error of prediction is

$$E[(y - \hat{y})^2].$$

The symbol  $E$  denotes averaging over all future data. The idea is to choose the model so that the prediction rule based on that model minimises the mean squared error of prediction.

Let  $\mu = E(y)$  denote the expected value of the new data  $y$ , and  $E(\hat{y})$  the expected value of the prediction. (Again, the symbol  $E$  denotes averaging over all future data.) Note that  $\mu$  and  $E(\hat{y})$  may not be the same: if they are not, the prediction rule is biased.

We can split the mean squared error of prediction into two parts:

$$\begin{aligned} E[(y - \hat{y})^2] &= E[(y - \mu)^2] + E[(\hat{y} - \mu)^2] \\ \text{EMSE} &= \text{VARIANCE} + \text{SQUARED BIAS} \end{aligned}$$

The effect of overfitting is to increase the variance, whereas the effect of underfitting is to increase the bias.

How do we measure the mean squared error of prediction? There are several possibilities:

- **Data splitting.** If we have a lot of data, we can divide the data into two parts, the *training set* and the *test set*. We use the training set to build the prediction rule as in Section 3.3. We then predict the  $y$ -value for each  $x$ -value in the test set. Since we also know the true  $y$ 's in this case, we can calculate the average squared difference between  $y$  and  $\hat{y}$  over the test set, to get an estimate of the mean squared error of prediction. The bigger the test set, the better this estimate will be. This method is a good one but only gives good results if we have plenty of data, since, if we do not, there will be insufficient data to both estimate the model from the training set and estimate the error from the test set. If there is not sufficient data to provide estimates of both these, what can be done?
- **Use the residual sum of squares.** We could use all the available data to both build the prediction rule and estimate the prediction error. This amounts to using the residual sum of squares (divided by the sample size) as the estimated mean squared error of prediction. This is usually too optimistic and underestimates the prediction error.
- **Cross-validation.** Here we randomly split the data into say 10 parts. One part acts as a test set, the rest as the training set. We compute the prediction error from the test set as before. We then repeat the procedure another 9 times, using a different 10th as the test set each time. We average the estimates to get a better estimate of prediction error. We can repeat for different “random splits”, averaging the results over the random splits. This is “10-fold cross-validation”. We can do 5-fold, or  $n$ -fold (where the test set has a single observation), but 10-fold seems to be best. See Efron and Tibshirani (1993) p. 247.
- **The bootstrap.** Here we “resample” the data to create a training set. That is, if the original data set has  $n$  rows, we select  $n$  rows at random with replacement from the original data set to construct a training set and then fit the model to the training set. We then use

the original data for the test set and calculate the estimated prediction error as before. This repeated say 200 times and the results averaged. (It may seem a bit odd to use the resampled data as the training set but this seems to give better results than using it for the test set.) The method described here is sometimes called the “simple bootstrap”. There are variations on this idea that are a bit more accurate - see Efron and Tibshirani (1993) p. 247 for more details. The R330 function `err.boot` implements one of these variations.

In general, data splitting is less accurate than bootstrapping and cross-validation, but better than using the residual sum of squares which is not recommended. When data splitting, the result depends on the split chosen, and in any event does not make best use of all the data. Cross-validation is less biased than the bootstrap but is more variable, but in general performs about as well. The residual sum of squares method is severely biased downwards - it gives estimates of error that are too small.

### 3.8.2 Variable selection for estimation

Consider the case where we want to estimate the effect of an exposure  $E$  on a disease  $D$ , for example the effect of coffee drinking on heart disease. Of course, there are other causes of heart disease, such as smoking ( $S$ ). It may be that smoking and coffee drinking are associated (heavy coffee drinkers are more likely to be smokers), so that if we observe an association between coffee drinking and heart disease, there are two possibilities:

- Coffee drinking causes heart disease.
- Coffee drinking does not cause heart disease, but heavy coffee drinkers tend to be heavy smokers, and smoking does cause heart disease.

To distinguish between these, we need to examine the association between coffee and disease, *with smoking held constant*. This amounts to fitting the model

$$D = \beta_0 + \beta_1 E + \beta_2 S + e$$

and examining the coefficient of  $E$ . If this is much different from 0, we conclude that there may be a causal link between coffee and disease. If the coefficient of  $E$  is not significantly different from 0, there is no evidence of a causal link.

If we fit the model without  $S$ ,

$$D = \beta'_0 + \beta'_1 E + e,$$

we get different estimates in general. The association between coffee and smoking may make the estimated coefficient of  $E$  in this model significantly different from 0, falsely encouraging the conclusion that coffee causes disease. The parameters  $\beta_1$  and  $\beta'_1$  are different things, as we saw in section 3.1.1: the former refers to the conditional distribution of  $D$  given  $E$  and  $S$ , while the latter refers to the conditional distribution of  $D$  given  $E$ . As we saw, the former distribution is the correct one to make judgments about causality.

If smoking and exposure are associated, then there can be big differences between  $\beta_1$  and  $\beta'_1$ . We refer to the difference as the *bias*. Variables that are associated with the exposure and the response are called *confounders*. Leaving confounders out of the model can cause bias.

The problem is that we do not ever know if all relevant confounders have been included in the model, as we may not have ever measured them. We might try and put every possible variable in the model. The downside of this is that adding variables not associated with the response will degrade the estimation of the exposure coefficient: they do not introduce bias, but they do increase the variance (or standard error) of the exposure coefficient.

Thus, in both the prediction and estimation contexts we must trade off the bias against the variance. As a very rough generalisation, for prediction overfitting is worse than underfitting, whereas for estimation the reverse is true.

We also note in passing the general principle of *Occam's razor*: this states that simple explanations are to be preferred to complex ones, or, in other words, that models having small numbers of variables are to be preferred (as long as they fit well). The name derives from William of Occam, a 14th century English philosopher.

### 3.8.3 Selecting a subset of variables

We have already mentioned some ways of selecting a model for prediction. These involved the method of “all possible regressions”. For each possible submodel, i.e. for each subset of variables, we defined some measures of “goodness”, by assessing the submodel’s ability as a predictor. We then evaluate this measure for all possible subsets and pick the best model. Another name for this is “exhaustive search.” We can also use this method for model selection in the estimation context.

Another type of variable selection method is one that moves from one model to another by adding or deleting variables, gradually arriving at a good model. These are usually called *stepwise* methods.

Finally, a very simple method is to base our selection of variables on the  $p$ -values in the original regression fit. Suppose we want to assess the effect of an exposure on disease, and we want to make a selection of confounders from a given set. We fit the model using all the variables, and retain all variables whose  $p$ -value exceeds some threshold, typically greater than 0.05.

#### All possible regressions

For these methods, we need a criterion for “model goodness”. Possible criteria are

- (i) Cross-validation/bootstrap (for prediction)
- (ii) AIC/BIC,
- (iii) Adjusted  $R^2$ , denoted by  $\bar{R}_p^2$ ,
- (iv) size of  $\hat{\sigma}^2$ ,
- (vi) Mallows’s  $C_p$ .

We have already discussed the bootstrap and cross-validation. For the other criteria:

**AIC:** This is the Akaike Information Criterion, and is an estimate of the difference between the fitted model and the actual true model. It is defined by the equation

$$AIC = \frac{RSS_p}{s^2} + 2(p + 1)$$

where  $RSS_p$  is the residual sum of squares of the model in question, and  $\hat{\sigma}^2$  is the estimate of the residual variance based on the full model. Thus, the RSS is penalised for big models, with  $p$  large. Small values of AIC indicate a good model.

**BIC:** This is the Bayesian Information Criterion, and is an estimate of the posterior probability that the fitted model is the correct one. (Posterior means after looking at the data.) The BIC is defined by the equation

$$BIC = \frac{RSS_p}{s^2} + \log(n)(p + 1).$$

Again, this penalises the RSS for big models, more severely than AIC. Small values of BIC indicate a good model.

**adjusted  $R^2$ :** Increasing the number of variables in a subset always increases the  $R^2$ , even if the added variables are random numbers. To compensate for this, the measure (i) adjusts  $R^2$  to account for the number of variables in the submodel. It is defined by

$$\bar{R}^2 = 1 - (1 - R^2) \frac{n - 1}{(n - p - 1)}$$

where  $p$  is the number of variables in the submodel. Big values of the adjusted  $R^2$  indicate a good model. We can view this as a crude way of compensating for the tendency of the residual sum of squares to underestimate the prediction error.

**size of  $\hat{\sigma}^2$ :** Since  $\hat{\sigma}^2 = \text{ResSS}/(n - p - 1)$ ; this estimate also adjusts for the increasing number of predictors. We pick the model for which  $\hat{\sigma}^2$  is a minimum. This measure is equivalent to the adjusted  $R^2$ .

**Mallow's  $C_p$ :** Before the invention of cross-validation and the bootstrap, Mallow's  $C_p$  was used as an approximation to the prediction error. We will not discuss it further.

We compute regressions for all  $2^k - 1$  possible models (i.e. with all possible subsets of variables), and pick out the best subset, on the basis of the criteria above. The R330 R function `allpossregs` does the calculations. The function returns information about the best subsets of each size, according to the residual sum of squares. (For a fixed subset size, the subset having the smallest RSS will also have the smallest value of each of our measures, except possibly CV and BOOT.)

**Example 18.** To print out the value of these measures for the fatty acid data, we type

```
> allpossregs(ffa~weight+age+skinfold, data=fatty.df)
      rssp sigma2 adjRsqr Cp      AIC      BIC      CV weight age skinfold
1 0.910  0.051  0.380 2.406 22.406 24.397 0.114      1  0      0
2 0.794  0.047  0.427 2.062 22.062 25.049 0.103      1  1      0
3 0.791  0.049  0.394 4.000 24.000 27.983 0.111      1  1      1
```

The best models of sizes 1, 2, 3 are listed in order. Thus, we see that on the adjusted  $R^2$ ,  $\hat{\sigma}^2$  and AIC criteria, the model with `weight` and `skinfold` is best. The model with `weight` alone is best on the other criteria, with  $C_p$  giving no strong indication between the two. Which of these criteria we use depends on the purpose of the model fitting. If we are fitting the model for the purposes of prediction, we would use the prediction error criteria CV and BOOT. If we are estimating, then AIC or BIC is more appropriate.

For this example, there are only 7 possible models, assuming all have a constant term. In general, there are  $2^k - 1$ , and this is a very large number if  $k$  is large. To reduce the output to manageable size, we may want to only print out the best few models for each subset size. This is done by means of the argument `best` in the `allpossregs` function. The default value, which we used above, is 1.

**Example 19.** Let's try the method on the big version of the soil data, with 10 explanatory variables.

```
> allpossregs(evap~maxst+minst+avst+maxat+minat+avat+maxh+minh+avh+wind, data=evap.df)
      rssp sigma2 adjRsqr Cp      AIC      BIC      CV maxst minst avst maxat
1 3071.255 69.801  0.674 30.519 76.519 80.177 286.520      0      0      0      0
2 2101.113 48.863  0.772  9.612 55.612 61.098 205.238      0      0      0      0
3 1879.949 44.761  0.791  6.390 52.390 59.705 189.606      0      0      0      0
4 1696.789 41.385  0.807  4.065 50.065 59.208 188.959      1      0      1      0
5 1599.138 39.978  0.813  3.759 49.759 60.731 189.714      1      0      1      0
6 1552.033 39.796  0.814  4.647 50.647 63.448 194.380      1      0      1      0
7 1521.227 40.032  0.813  5.920 51.920 66.549 214.563      1      0      1      1
8 1490.602 40.287  0.812  7.197 53.197 69.654 231.768      1      0      1      1
9 1483.733 41.215  0.808  9.034 55.034 73.321 249.134      1      0      1      1
10 1482.277 42.351  0.802 11.000 57.000 77.115 261.261      1      1      1      1
      minat avat maxh minh avh wind
1      0      0      0      0      1      0
2      0      1      0      0      1      0
3      0      1      0      0      1      1
4      0      1      0      0      1      0
5      0      1      0      1      1      0
6      0      1      0      1      1      1
7      0      0      1      1      1      1
8      0      1      1      1      1      1
9      1      1      1      1      1      1
10     1      1      1      1      1      1
```

The best model with 4 predictors seems to be a good model, with the best CV. The models with 5 and 6 predictors are also good. Any of these models would be satisfactory, depending on the purpose of the fitting, although we often prefer a simple model over a more complicated one.

## Stepwise regression

In this technique, we start with an initial model, and then seek to improve it by dropping or adding variables sequentially. The decision about which variable to add or delete at each stage can be made on the basis of an  $F$ -test or  $t$ -test or by comparing AIC's. We describe the latter, as this is the way it is done in R. There are three variations of stepwise regression:

**Forward selection:** We start with the null model with no variables, just a constant term. We then examine all one variable models, and select the one with the smallest AIC. If the null model is the best, we select this. We then consider all two-variable models that include the variable added at the first stage. If none of these has a lower AIC, we choose the best one variable model. Otherwise we continue. Thus, at each stage, we add the variable to the current model that has the best decrease in AIC. If no added variable decreases the AIC, we are done.

**Backward elimination:** Here we start with the full model with all the variables. We then delete the variable from the full model that results in the biggest decrease in AIC. If no variable decreases the AIC, we stop. Otherwise, we repeat the process and delete from the new model the variable that gives the biggest decrease in AIC. Thus, at each stage, we delete the variable from the current model that gives the biggest decrease in AIC. If no deleted variable decreases the AIC, we stop.

**Stepwise regression:** Here we start with the null model. We repeatedly perform a cycle of forward selection, then a cycle of backward elimination, by either adding a variable not currently in the model, or deleting a variable currently in the model, to achieve the best decrease in AIC. If none of these actions results in an improvement, we stop.

The calculations are done by the R function `step`. It takes several arguments. The first is an `lm` object representing the starting model, and the second is a formula representing the full model. Another argument is `direction`, which is one of `forward` (for forward selection), `backwards` (for backward elimination) or `both` (for stepwise regression). The default is `both`.

**Example 20.** Thus, to perform stepwise regression on the moisture data, we type

```
null.model=lm(evap~1, data=evap.df)
full.model.formula = evap~maxst+minst+avst+maxat+minat+avat+maxh+minh+avh+wind
step(null.model, full.model.formula, trace=0) # trace=0 supresses some output
Call:
lm(formula = evap ~ avh + avat + wind + avst + maxst, data = evap.df)
```

Coefficients:

(Intercept)	avh	avat	wind	avst	maxst
70.53895	-0.32310	0.36375	0.01089	-0.48809	1.19629

We can get a complete record of what happens at each stage by setting `trace=1` (the default). Note that this is not the best 5 variable model as chosen by `allpossregs`: there is no guarantee that the stepwise methods will result in the “best model”. To find a model by forward selection, we type

```
> step(null.model, full.model.formula, direction="forward", trace=0)
Call:
lm(formula = evap ~ avh + avat + wind + avst + maxst, data = evap.df)
```

Coefficients:

(Intercept)	avh	avat	wind	avst	maxst
70.53895	-0.32310	0.36375	0.01089	-0.48809	1.19629

and for backward elimination

```
> # backward elimination
> full.model = lm(evap~maxst+minst+avst+maxat+minat+avat+maxh+minh+avh+wind,
  data=evap.df)
> selected.model = step(full.model, formula(full.model), direction="backward",
  trace=0)
Call:
lm(formula = evap ~ maxst + avst + maxat + maxh + minh + avh + wind,
  data = evap.df)
```

Coefficients:					
(Intercept)	maxst	avst	maxat	maxh	minh
-59.9896	2.3800	-0.6657	0.7206	1.3146	1.0000
avh	wind				
-0.6050	0.0126				

Note that backward elimination results in a bigger model than the other two methods.

### Which method should we use?

Generally, for selecting models in the prediction context, the “all possible regressions method” is to be preferred, using either CV or the bootstrap as our estimate of prediction error. In the estimation context, backward elimination generally works well, as does “all possible regressions” using the AIC criterion.

One problem with stepwise methods is that a small change in the data can result in choosing a very different model. One way to deal with this is to bootstrap the selection process, so that we can see how the selected model changes with different bootstrap samples. We can include in the final model those variables that occur in say 50% of the bootstrap samples.

**Example 21.** Consider the evaporation data. In Example 20, we selected the model `evap~maxst + avst + maxat + maxh + minh + avh + wind` using backward elimination. Let’s imagine this is in fact the true model. We can generate some data from this model using the R code

```
n = dim(evap.df)[1]
y = fitted(selected.model) + rnorm(n, sd = summary(selected.model)$sigma)
new.data.df = data.frame(evap.df[,-11], y)
```

Now we know the “true model”, we can apply the bootstrap selection procedure above and compare the results with the true state of affairs. Using the new data in the data frame `new.data.df`, we can repeatedly draw 1000 bootstrap samples and record the coefficients of the selected model in an array called `coef.mat`. The following code does just this:

```
B=1000
# create an array to hold the results
# each row is a bootstrap replication, and each column
# corresponds to a variable
coef.mat = matrix(0,B, 10) # create an array to hold the results
for(b in 1:B){
  use = sample(n,n,replace=TRUE)
  use.df = new.data.df[use,]
  full.model.boot = lm(y~., data=use.df) # fit model to bootstrap sample
# do subset selection
  selected.mod.boot = step(full.model.boot, y~1, direction="backward", trace=0)
#record coefficients
  selected.vars = names(coef(selected.mod.boot))[-1]
  index = match(selected.vars, names(new.data.df)[-11])
  coef.mat[b,index] = coef(selected.mod.boot)[-1]
}
```

Note that if a variable is not selected, the corresponding coefficient is zero. We can identify the variables that appear in more than 50% of the replications:

```
> proportions = apply(coef.mat!=0,2,mean)
> proportions
[1] 0.808 0.500 0.576 0.269 0.394 0.288 0.569 0.803 0.999 0.817
> names(new.data.df)[-11][proportions>0.5]
[1] "avst" "minst" "maxst" "avh" "minh" "maxh" "wind"
```

Compare this with the true model, which had variables `avst`, `maxst`, `avh`, `maxat`, `minh`, `maxh`, and `wind`. Refitting the model using the variables selected by the 50% rule and comparing coefficients we get



	Bootstrap selection	True model
avst	2.8303	2.3800
minst	0.3625	0.0000
maxst	-0.6944	-0.6657
avat	0.0000	0.7206
avh	1.6017	1.3146
minh	1.1236	1.0000
maxh	-0.6494	-0.6050
wind	0.0109	0.0126

so the procedure has done a reasonable job of selecting the correct variables.

### Testing after model selection

Note that it is not valid to fit a model selected using the variable selection methods we have described, and then delete further variables based on their  $p$ -values. The  $p$ -value is only valid for a model selected *before* inspecting the data, not for models selected by a data-driven variable selection procedure. One possibility for dealing with this situation is to use the bootstrap: we repeatedly resample the data, use our favourite variable selection method each time, and inspect the resulting distributions of the regression coefficients. We can estimate the coefficient by the mean of the bootstrapped replications. This can be done using the matrix `coef.mat` computed above. We can draw histograms for each variable (i.e. for the data in each column of the matrix. These are shown in Figure 3.29. The results show that, for this example at least, the mean of the bootstrapped replications does slightly worse than fitting the model using the variables selected by the 50% method.

- *Underfitting biases predictions and estimation.*
- *Overfitting increases the variances of predictions and estimated coefficients.*
- *Use cross-validation and the bootstrap to estimate prediction error.*
- *Use `allpossregs` to fit all possible regressions.*
- *Use `step` to do stepwise regression.*

## 3.9 Case Study 2: Highway accident rates

The data in Table 3.12 were gathered in the course of studying the relationship between accident rates on US highways and various characteristics of the highway. Each “case” is a particular stretch of highway. The variables are:

**rate:** The accident rate per million vehicle miles (the response),

**len:** The length of the segment of highway in miles,

**adt:** Average daily traffic count (‘000),

**trks:** The percentage of the traffic that are trucks,

**slim:** The speed limit in mph,

**lwid:** The lane width in feet,

**shld:** The width of the shoulder in feet,

**itg:** Number of freeway interchanges per mile,

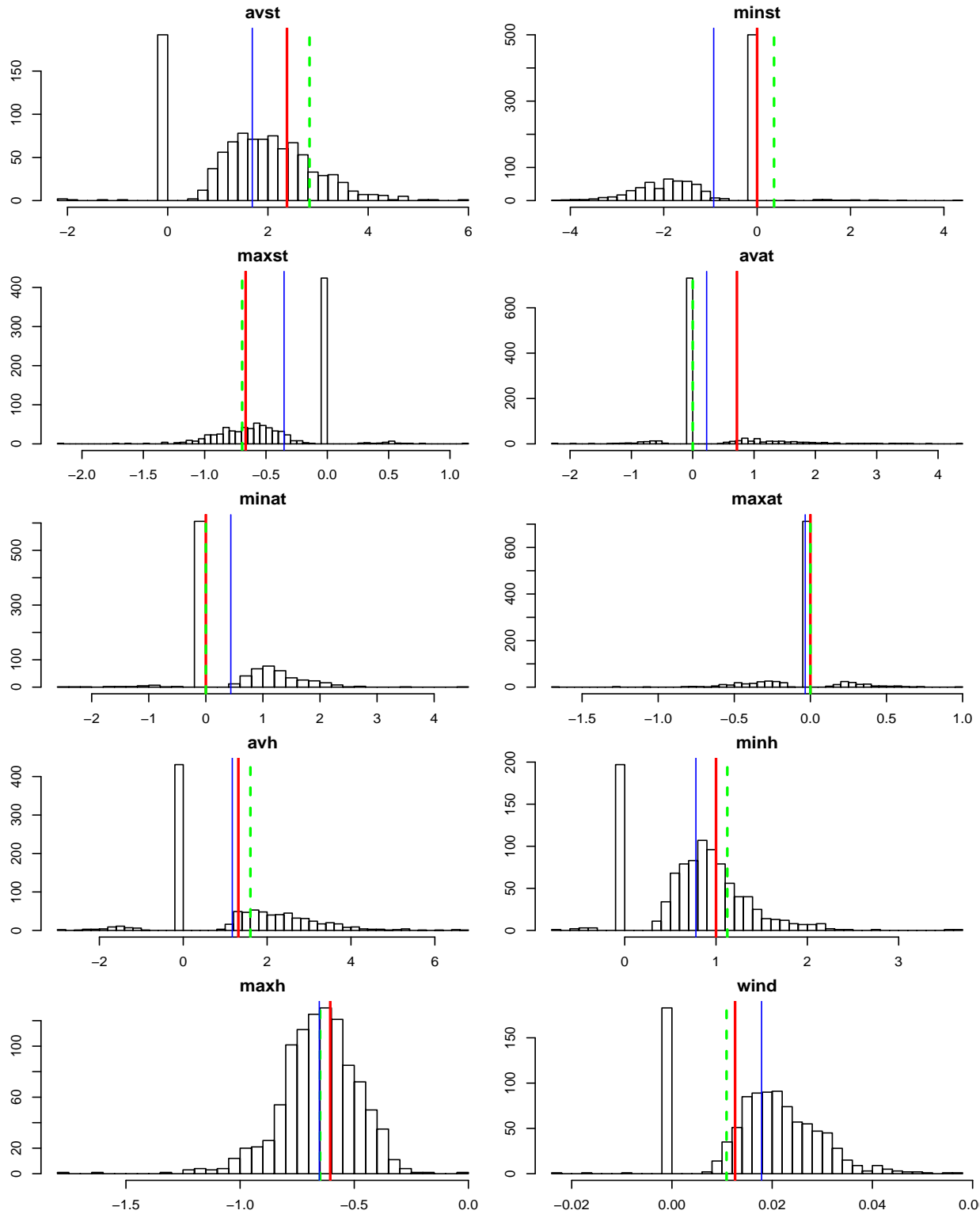


Figure 3.29: Histograms of bootstrapped coefficients. The thick vertical line is the true value of the coefficient and the thin line the mean of the bootstrapped coefficients. The dashed line represents the coefficients of the model selected by the 50% method.

**sigs:** Number of entrances controlled by lights per mile,

**acpt:** Number of access points per mile,

**lane:** Number of lanes in each direction,

**fai:** Dummy variable, equal to 1 if an interstate highway, zero otherwise,

**pa:** equal to 1 if a principal highway, 0 otherwise,

**ma:** equal to 1 if a major highway, 0 otherwise.

There are four types of highway in the data set, interstate, principal, major and “major collector” for which all the dummy variables are zero. The data were collected to explore the likely causes of accidents, and to identify variables associated with the accident rate. They are included in the R330 package under the name `traffic.df`.

Table 3.12: Data on highway accidents.

obs	rate	len	adt	trks	slim	lwd	shld	itg	sigs	acpt	lane	fai	pa	ma
1	4.58	4.99	69	8	55	12	10	1.20	0	4.60	8	1	0	0
2	2.86	16.11	73	8	60	12	10	1.43	0	4.40	4	1	0	0
3	3.02	9.75	49	10	60	12	10	1.54	0	4.70	4	1	0	0
4	2.29	10.65	61	13	65	12	10	0.94	0	3.80	6	1	0	0
5	1.61	20.01	28	12	70	12	10	0.65	0	2.20	4	1	0	0
6	6.87	5.97	30	6	55	12	10	0.34	1.84	24.80	4	0	1	0
7	3.85	8.57	46	8	55	12	8	0.4	0.70	11.00	4	0	1	0
8	6.12	5.24	25	9	55	12	10	0.38	0.38	18.50	4	0	1	0
9	3.29	15.79	43	12	50	12	4	0.95	1.39	7.50	4	0	1	0
10	5.88	8.26	23	7	50	12	5	0.12	1.21	8.20	4	0	1	0
11	4.20	7.03	23	6	60	12	10	0.29	1.85	5.40	4	0	1	0
12	4.61	13.28	20	9	50	12	2	0.15	1.21	11.20	4	0	1	0
13	4.80	5.40	18	14	50	12	8	0	0.56	15.20	2	0	1	0
14	3.85	2.96	21	8	60	12	10	0.34	0	5.40	4	0	1	0
15	2.69	11.75	27	7	55	12	10	0.26	0.60	7.90	4	0	1	0
16	1.99	8.86	22	9	60	12	10	0.68	0	3.20	4	0	1	0
17	2.01	9.78	19	9	60	12	10	0.20	0.10	11.00	4	0	1	0
18	4.22	5.49	9	11	50	12	6	0.18	0.18	8.90	2	0	1	0
19	2.76	8.63	12	8	55	13	6	0.14	0	12.40	2	0	1	0
20	2.55	20.31	12	7	60	12	10	0.05	0.99	7.80	4	0	1	0
21	1.89	40.09	15	13	55	12	8	0.05	0.12	9.60	4	0	1	0
22	2.34	11.81	8	8	60	12	10	0	0	4.30	2	0	1	0
23	2.83	11.39	5	9	50	12	8	0	0.09	11.10	2	0	1	0
24	1.81	22.00	5	15	60	12	7	0	0	6.80	2	0	1	0
25	9.23	3.58	23	6	40	12	2	0.56	2.51	53.00	4	0	0	1
26	8.60	3.23	13	6	45	12	2	0.31	0.93	17.30	2	0	0	1
27	8.21	7.73	7	8	55	12	8	0.13	0.52	27.30	2	0	0	1
28	2.93	14.41	10	10	55	12	6	0	0.07	18.00	2	0	0	1
29	7.48	11.54	12	7	45	12	3	0.09	0.09	30.20	2	0	0	1
30	2.57	11.10	9	8	60	12	7	0	0	10.30	2	0	0	1
31	5.77	22.09	4	8	45	11	3	0	0.14	18.20	2	0	0	1
32	2.90	9.39	5	10	55	13	1	0	0	12.30	2	0	0	1
33	2.97	19.49	4	13	55	12	4	0	0	7.10	2	0	0	1
34	1.84	21.01	5	12	55	10	8	0	0.10	14.00	2	0	0	1
35	3.78	27.16	2	10	55	12	3	0.04	0.04	11.30	2	0	0	1
36	2.76	14.03	3	8	50	12	4	0.07	0	16.30	2	0	0	1
37	4.27	20.63	1	11	55	11	4	0	0	9.60	2	0	0	1
38	3.05	20.06	3	11	60	12	8	0	0	9.00	2	0	0	0
39	4.12	12.91	1	10	55	12	3	0	0	10.40	2	0	0	0

A `pairs` plot doesn’t tell us very much, but reveals no obvious outliers. One problem is the sheer number of variables: we need to cut down the number to make things simpler. Initially, we will ignore the dummy variables and just analyse the continuous ones. A preliminary fit yields

```
> traffic.lm = lm(rate~len+adt+trks+slim+lwd+shld+sigs+acpt+lane, data=traffic.df)
> summary(traffic.lm)
```

Call:

```
lm(formula = rate ~ len + adt + trks + slim + lwd + shld + sigs +
    acpt + lane, data = traffic.df)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-1.8815 -0.8435  0.1379  0.6169  2.6327
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 13.004085   5.891543   2.207  0.03536 *
len          -0.064301   0.032170  -1.999  0.05508 .
adt           0.007711   0.018727   0.412  0.68355
trks         -0.117811   0.101533  -1.160  0.25538
slim         -0.059506   0.060706  -0.980  0.33508
lwd          -0.395028   0.466381  -0.847  0.40393
shld         -0.093958   0.104479  -0.899  0.37590
sigs          0.331827   0.399142   0.831  0.41257
acpt          0.090974   0.030341   2.998  0.00552 **
lane          0.033733   0.266787   0.126  0.90026
```

---

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 1.16 on 29 degrees of freedom

Multiple R-squared: 0.7396, Adjusted R-squared: 0.6587

F-statistic: 9.15 on 9 and 29 DF, p-value: 2.09e-06

The  $R^2$  is quite good, but most of the variables can be dropped from the model, provided the others are retained. Note that, using a  $p$ -value threshold of 0.1, only the variables `len` and `acpt` would be retained.

Lets have a look at the correlations:

```
> round(cor(traffic.df[,2:12]), 1)
```

```
      rate len  adt trks slim  lwd shld  itg sigs acpt lane
rate  1.0 -0.5  0.0 -0.5 -0.7  0.0 -0.4  0.0  0.6  0.8  0.0
len   -0.5  1.0 -0.3  0.5  0.2 -0.3 -0.1 -0.2 -0.3 -0.2 -0.2
adt    0.0 -0.3  1.0 -0.1  0.2  0.1  0.5  0.9  0.1 -0.2  0.8
trks  -0.5  0.5 -0.1  1.0  0.3 -0.2  0.0 -0.1 -0.5 -0.4 -0.2
slim  -0.7  0.2  0.2  0.3  1.0  0.1  0.7  0.2 -0.4 -0.7  0.3
lwd    0.0 -0.3  0.1 -0.2  0.1  1.0  0.0  0.1  0.0  0.0  0.1
shld  -0.4 -0.1  0.5  0.0  0.7  0.0  1.0  0.4 -0.1 -0.4  0.5
itg    0.0 -0.2  0.9 -0.1  0.2  0.1  0.4  1.0  0.1 -0.2  0.7
sigs   0.6 -0.3  0.1 -0.5 -0.4  0.0 -0.1  0.1  1.0  0.5  0.2
acpt   0.8 -0.2 -0.2 -0.4 -0.7  0.0 -0.4 -0.2  0.5  1.0 -0.2
lane   0.0 -0.2  0.8 -0.2  0.3  0.1  0.5  0.7  0.2 -0.2  1.0
```

Seems as though at least `len`, `trks`, `slim`, `shld`, `sigs` and `acpt` have some relationship with the response. The non-significance of these variables in the regression table could be due to multicollinearity. There are a few largish associations between the explanatories: between `adt` and `itg`, `slim` and `shld`, `slim` and `acpt`, `lane` and `adt`, `lane` and `itg`. These could be masking the significance. Lets investigate the multicollinearity, using the continuous variable only:

```
> X<-traffic.df[,3:12]
> eigen.stuff<-eigen(cor(X))
> eigen.stuff$values
[1] 3.3921725 2.6765015 1.1432671 0.8732114 0.6018518 0.4606080 0.3714477
[8] 0.2473021 0.1665959 0.0670421

> diag(solve(cor(X)))
```

```

      len      adt      trks      slim      lwd      shld      itg      sigs
1.701583 9.358752 1.614839 3.678741 1.289684 2.944450 5.705195 1.830873
      acpt      lane
2.282032 3.769093

```

Thus, there are moderate but not severe problems - the maximum VIF is a bit under 10. Lets find a suitable subset of variables:

```

> allpossregs(traffic.lm)
      rssp sigma2 adjRsq      Cp      AIC      BIC      CV
1 65.119  1.760  0.554 13.376 52.376 55.703 5.770
2 52.139  1.448  0.633  5.733 44.733 49.724 5.098
3 44.847  1.281  0.675  2.316 41.316 47.970 4.604
4 42.333  1.245  0.684  2.449 41.449 49.767 5.069
5 40.928  1.240  0.686  3.405 42.405 52.386 4.898
6 40.456  1.264  0.679  5.054 44.054 55.699 5.879
7 39.857  1.286  0.674  6.609 45.609 58.917 6.280
8 39.058  1.302  0.670  8.016 47.016 61.988 6.582
9 39.037  1.346  0.659 10.000 49.000 65.636 6.992
      len adt trks slim lwd shld sigs acpt lane
1   0   0   0   0   0   0   0   1   0
2   1   0   0   0   0   0   0   0   0
3   1   0   0   1   0   0   0   0   0
4   1   0   0   1   0   0   1   1   0
5   1   0   1   1   0   0   1   1   0
6   1   0   1   1   1   0   1   1   0
7   1   0   1   1   1   1   1   1   0
8   1   1   1   1   1   1   1   1   0
9   1   1   1   1   1   1   1   1   1

> step(traffic.lm, scope=formula(traffic.lm), direction="back", trace=0)

```

Call:

```
lm(formula = rate ~ len + slim + sigs + acpt, data = traffic.df)
```

Coefficients:

```

(Intercept)      len      slim      sigs      acpt
  8.81443    -0.06856   -0.09599    0.48538    0.08940

```

Model 3 in the `allpossregs` output looks good: it has the smallest AIC, and the smallest CV (although we are not so interested in prediction in this example.) Model 4 has the next best AIC. It is also the model chosen by the stepwise regression algorithm.

If we plot this subset using `pairs`, we note some possible outliers: an influence analysis (not shown) indicates that point 25 is influential. Is it necessary to transform the response to improve the fit? (We don't explore the possibility of transforming the explanatory variables to save space, but in a real analysis this should be done, using the techniques illustrated in Case Study 1.)

After deleting point 25, a Box-Cox plot for Model 4 indicates a log transformation might be helpful. Refitting the model using a logged response does not improve the fit - in fact the  $R^2$  goes down slightly, from 69.0% to 68.6%. But the residual plots look a bit better, so we decided to go with the logged response and point 25 deleted. The regression summary for this model is

Call:

```
lm(formula = log(rate) ~ len + slim + sigs + acpt, data = traffic.df,
    subset = -25)
```

Residuals:

```

      Min      1Q    Median      3Q      Max
-0.51486 -0.20918 -0.01086  0.21646  0.44126

```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	2.715810	0.631857	4.298	0.000143	***
len	-0.018520	0.005915	-3.131	0.003639	**
slim	-0.028161	0.010136	-2.778	0.008945	**
sigs	0.179730	0.084828	2.119	0.041725	*
acpt	0.023487	0.008241	2.850	0.007476	**

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2614 on 33 degrees of freedom

Multiple R-squared: 0.6866, Adjusted R-squared: 0.6486

F-statistic: 18.08 on 4 and 33 DF, p-value: 5.968e-08

Next, we included the dummy variables in the analysis. The  $R^2$  goes up to 75.6%, indicating that the dummy variables could be quite important. After some experimentation, including a backward elimination, we decided to retain the variable `pa`, giving a final model with an  $R^2$  of 73.8%. The regression summary for this model is

Call:

```
lm(formula = log(rate) ~ len + slim + sigs + acpt + pa, data = traffic.df,
    subset = -25)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.57786	-0.13654	0.00801	0.15279	0.38266

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	2.999528	0.597448	5.021	1.87e-05	***
len	-0.020113	0.005529	-3.638	0.000957	***
slim	-0.030266	0.009448	-3.203	0.003069	**
sigs	0.284420	0.089144	3.191	0.003174	**
acpt	0.017633	0.007999	2.204	0.034815	*
pa	-0.234881	0.093694	-2.507	0.017453	*

---

Residual standard error: 0.2427 on 32 degrees of freedom

Multiple R-squared: 0.7381, Adjusted R-squared: 0.6971

F-statistic: 18.03 on 5 and 32 DF, p-value: 1.73e-08

It is well known that fitting models after using a variable selection procedure tends to lead to biased estimates of the coefficients, so we should be cautious about interpreting coefficients unless the p-values are very small. Thus in the present example, we might be cautious about interpreting `acpt`, and possibly `pa`. It seems that there is a strong connection between accident rate and length, speed limit and the number of entrances controlled by lights. The more entrances, the higher the rate - this is probably saying that intersections are dangerous. Note also that the higher the speed limit, the lower the accident rate. This seems paradoxical, but it probably means that on intrinsically dangerous parts of the highway, the speed limit is low anyway. Also, the longer the length, the less the accident rate. This may be because longer highways (motorways) have less traffic density, and hence a lower accident rate. We can't be sure unless we know more about the data.

The coefficient of `pa` is negative, indicating a lower accident rate on "principal highways". Note also that `acpt` and `slim` are negatively correlated, so that speed limits are lower on roads having many access points. To some extent, these variables are acting as proxies for one another.

## 3.10 Smoothing

The previous sections discussed the fitting of relationships to either transformed or untransformed variables. In most cases the form of the relationship was linear, but we also considered polynomials and splines. With the exception of the spline fits, which are quite flexible, we thus put considerable constraints on the form of the fitted relationship.

In this section we describe how to fit non-linear relationships between a response  $y$  and a single explanatory variable  $x$  using curves that are not subject to such strong restrictions. Rather, they are data-driven in the sense that the data determines the form of the relationship to an extent that is under the control of the experimenter. These methods come under the general heading of *scatterplot smoothers*. The idea is to produce a function  $\hat{y}(x)$  that smooths the scatterplot and estimates the relationship between  $y$  and  $x$ .

We will discuss only one type of smoother, the loess smoother, although there are many different types. Loess stands for “local regression”.

**Example 22.** We illustrate the use of the loess smoother with the ozone data that was used in Example 2. A plot of ozone concentration against time ( $x_i = 1, 2, \dots, 153$ ) is shown in Figure 3.30. The nature of the relationship is obscured by excessive variability. The figure also shows the scatterplot smoothed using loess, using various amounts of smoothing.

We see that the resulting “smooth” i.e. the resulting estimated relationship is highly dependent on the amount of smoothing done. The loess method is based on “local regression”. Each data

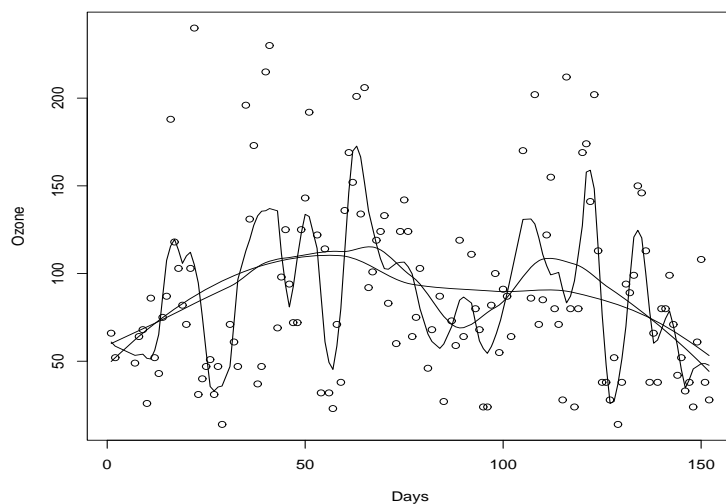


Figure 3.30: Smoothing the ozone data.

point  $(x_i, y_i)$  is smoothed by replacing it with a point  $(x_i, \hat{y}_i)$  where  $\hat{y}_i$  is a fitted value from a weighted regression of  $y$  on  $x$ . The weights for case  $j$  depend on the distance between  $x_i$  and  $x_j$ .

More precisely, each point  $(x_j, y_j)$  is given a weight  $c_{ij}$  depending on the distance of  $x_j$  from  $x_i$ . The weights are given by

$$c_{ij} = h(x_i - x_j)$$

where  $h$  is some function satisfying

- (i)  $h(k) = h(-k)$ ,
- (ii) If  $|k_1| < |k_2|$ , then  $h(k_1) \geq h(k_2)$ ,
- (iii)  $h(k) = 0$  for  $|k| > d$ .

The function used in loess is the tricube function

$$h(k, d) = \begin{cases} \left[1 - \left(\frac{|k|}{d}\right)^3\right]^3, & \text{if } -d \leq k \leq d, \\ 0, & \text{otherwise.} \end{cases}$$

The value of  $d$  depends on  $x_i$ , and is the distance from  $x_i$  one must go to include a fraction  $\alpha$  of the data, called the *span*. The fraction  $\alpha$  thus controls the amount of smoothing: the bigger  $\alpha$ , the more smoothing. A line is now fitted through the points by weighted least squares, resulting in a fitted value  $\hat{y}_i$ . We can use either linear or quadratic regression.

The function in R for fitting loess regressions is `loess`. It is easy to use for drawing smooth curves on scatterplots.

**Example 23.** Assuming the data are in a data frame `stamford.df`, to draw the plot and add the smooth curve using `loess`, we type

```
plot(stamford.df$days, stamford.df$ozone, xlab="Days", ylab="Ozone")
loess.stuff=loess(ozone~days, data=stamford.df, span=0.75)
lines(loess.stuff$x, loess.stuff$fitted)
```

For more smoothing, we can set the value of the parameter `span`. The default is 0.75, but we can smooth to a greater or lesser extent by varying this parameter. The greater the value of `span`, the more smoothing. The three smooths in Figure 3.30 are for values of `span` equal to 0.1, 0.4 and the default value 0.75 (this last smooth is the one produced by the code above).

### 3.11 Robust regression

In Sections 3.4 and 3.5 we saw one common approach to the problem of outliers: remove the offending points, set them aside for separate discussion, then fit the line to the remaining bulk of the data using least squares. This type of procedure is usually quite successful, but requires close inspection of the results and the diagnostic information relating to a regression. An alternative is an automatic procedure which can routinely allow for the presence of such outlying observations.

One possible approach is to change the least squares minimisation criterion: what if, instead of minimising a sum of squares, we use some other function of the residuals? The problem with least squares is that points far from a possible regression line will have a very large squared residual so the best way to minimise the sum of squared differences is to make the largest squared differences smaller - i.e. least squares tries to make these residuals small, and the line is attracted to the point. To avoid this, we might try a function that does not magnify the contribution of large residuals to the minimisation criterion. One possibility is the absolute value function: we could choose  $\beta_0, \dots, \beta_k$  to minimise

$$\sum_{i=1}^n |r_i| = \sum_{i=1}^n |y_i - \beta_0 - \beta_1 x_{i1} - \dots - \beta_k x_{ik}|.$$

This usually turns out to be quite successful in reducing the effect of outlying points. Computational procedures for producing these estimates are not as straightforward as those for least squares, but are reasonably commonly available. These estimates are called LAD, or *least absolute deviation* estimates.

A more general approach is to choose  $\beta_0, \dots, \beta_k$  to minimise

$$\sum_{i=1}^m \rho(r_i) = \sum_{i=1}^m \rho(y_i - \beta_0 - \beta_1 x_{i1} - \dots - \beta_k x_{ik})$$

where  $\rho$  is some function having the properties

- (i)  $\rho(x) \geq 0$  for all  $x$ , and has a minimum at 0;
- (ii)  $\rho(x) = \rho(-x)$  for all  $x$ ;



- (iii)  $\rho(x)$  increases as  $x$  increases from 0, but does not get too large as  $x$  increases.

Estimates of  $\beta_0, \dots, \beta_k$  using this idea are called  $M$ -estimates; for a detailed discussion see Huber (1981). LAD regression corresponds to choosing

$$\rho(x) = |x|$$

and least squares to

$$\rho(x) = x^2.$$

Another possibility is the biweight function

$$\rho(x) = \begin{cases} \left(1 - \left(\frac{x}{k}\right)^2\right)^2, & \text{if } |x| < k, \\ 0, & \text{otherwise.} \end{cases}$$

The constant  $k$  is chosen to be proportional to the spread of the residuals. Property (iii) above guarantees that a line estimated with such a  $\rho$  will be resistant to the effect of outlying observations. Accordingly, such techniques they are called *robust* methods.

An alternative approach is to replace the sum in the least squares criterion by something more robust, such as quantiles or trimmed sums. Rousseeuw (1984) considers a least quantile of squares line (LQS line) which minimises the  $(n + p + 2)/2$  quantile of the squared residuals. Another possibility is a trimmed version of least squares, which minimises, say, the sum of the smallest half of the squared residuals. For descriptions of still further approaches, Rousseeuw and Leroy (1987), Chapter 2. We compare these procedures using an example.

**Example 24.** Suppose we have the following set of ultrasound data:

```
> BPD
[1] 77 82 90 82 86 84 80 87 79 81
> BW
[1] 1350 1500 2170 525 1820 1740 1300 1730 1580 1620
```

If we plot the data, we see that point 4 is an outlier, but it is not influential. In Figure 3.31(i), we plot the data, and also fit lines by four methods: least squares, LQS, Least trimmed squares, and finally  $M$ -estimate regression. We see that all three robust methods have ignored the outlier, but the least squares line has been pulled down. We fit robust regressions in the usual way, using a model formula to define the model. We use the functions `lqs` for least quantile of squares and least trimmed squares, and `rlm` for  $M$ -estimation.

Robust methods deal well with outliers, but are not so good at dealing with high influence points. Suppose that the data had been

```
> BPD
[1] 77 82 90 82 86 55 80 87 79 81
> BW
[1] 1350 1500 1817 1780 1820 525 1300 1730 1580 1620
```

Now the data contains a high influence point. If we fit lines using the same four methods, we see in Figure 3.31(ii) that only the LQS method provides much protection against high influence points.

Another way to compare these methods is by comparing their *breakdown points*, i.e. the percentage of the data that can be outlying before the fitted line is attracted to the outlying points. Least squares has a breakdown point of 0%, since one serious outlier can spoil the fit. The  $M$ -estimate methods also has breakdown points of 0%, but can be modified to achieve values in the region of 20–30%. The LMS method has a breakdown point of 50%. See Rousseeuw and Leroy (1987) pp65–70 for an excellent discussion of breakdown points, and an experiment to contrast the breakdown points of the various methods.

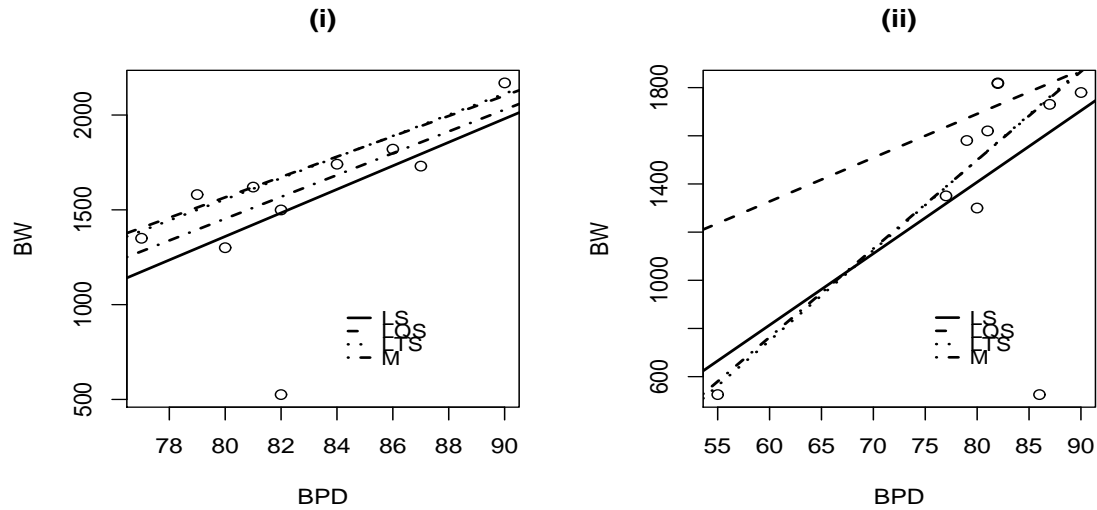


Figure 3.31: Fitting using various methods. (i) Outliers, (ii) High influence points.

## 3.12 Chapter summary

### 3.12.1 Main ideas

- The regression model
- Interpreting regression coefficients
- Fitting the model
- Interpreting the output
- Dropping and adding variables
- Prediction
- Diagnosing model faults (see section summary for details)
- Fixing up models (see section summary for details)
- Multicollinearity
- Subset selection: Consequences of incorrect choice, all possible regressions, stepwise methods
- Smoothing
- Robust regression

### 3.12.2 R functions used

Note: \* denotes a function in the R330 package

`allpossregs*`  
`anova`

arima  
boxcoxplot\*  
cbind  
coefficients  
cor  
cut  
data.frame  
diag  
diag.plots  
eigen  
fitted  
funnel\*  
gam  
hatvalues  
influence.measures  
influenceplots  
length  
lines  
lm  
lm.influence  
loess  
order  
par  
partial.plots  
poly  
predict  
qnrm  
read.table  
residuals  
round  
seq  
solve  
step  
summary  
tapply  
text  
update

Use the R help system to find the details for these functions.



## Chapter 4

# Models involving Factors

### 4.1 Introduction

In the previous chapter we were concerned with fitting models relating a continuous response variable to other continuous explanatory variables. For example, if we had a data frame where the observations were different makes of cars and we wished to relate the fuel consumption, `Mileage`, to the weights of the car and the size of its engine, `Weight` and `Disp.` respectively, then we would specify our model as

```
Mileage ~ Weight + Disp.
```

But a variable might not be intrinsically continuous, such as the weight of a car. It might take values from a specified, but finite, set of possible levels. We call such a variable a *factor*. For example a factor `Sex` would represent one of two values “Male” or “Female”. We can see that the levels of our new factor are in no sense numeric, indeed are not even ordered.

Factors enter model formulae in the same way as numeric variables, but the interpretation of the output from fitting the model needs a little care. In fact when fitting a model we obtain a set of coefficients corresponding to a factor. Thus unlike a continuous variable, where model fitting produces a single coefficient or parameter, a factor gives rise to  $L - 1$  coefficients, where  $L$  is the number of distinct levels of the factor. Suppose that we had fitted the model

```
Salary ~ Age + Sex
```

where `Age` is a continuous variable and `Sex` is a factor having two levels, Female and Male. Our model says that the expected value of the salary of the  $i$ 'th person is given by

$$E[\text{Salary}_i] = \mu + \text{Age}_i\beta + \alpha_m \quad \text{if the } i\text{'th person is male, and}$$

$$E[\text{Salary}_i] = \mu + \text{Age}_i\beta + \alpha_f \quad \text{if the } i\text{'th person is female,}$$

where  $\alpha_m$  and  $\alpha_f$  are two parameters representing the two levels of `Sex`. One way that this model might be fitted is to construct two numeric variables, `Xm`, `Xf`. The  $i$ 'th value of `Xm` would be 1 if the  $i$ 'th person were male, 0 if female, and of course `Xf` would take the opposite values. We see immediately that `Xm + Xf = 1`. We could specify our model not referring to factors at all:

```
Salary ~ Age + Xm + Xf
```

But what will happen to the estimates of the parameters,  $\mu$ ,  $\alpha_m$  and  $\alpha_f$ ? The fitted values, estimating  $E[\text{Salary}_i]$  will be entirely unaffected if some constant,  $\delta$  say, is added to  $\alpha_m$  and  $\alpha_f$  and subtracted from  $\mu$ . This is why we can only determine  $L - 1$  coefficients for a factor with  $L$  levels. In the case of `Sex`,  $L = 2$ , so we cannot separately determine  $\alpha_m$  and  $\alpha_f$ , only one of them. R adopts the convention that the coefficient corresponding to the first level of a factor is set equal to zero. The coefficients corresponding to levels  $2 \dots L$ , measure the *difference* between that level and level 1. Thus for our model relating `Salary` to `Age` and `Sex` we have

$$E[\text{Salary}_i] = \mu + \text{Age}_i\beta \quad \text{if the } i\text{'th person is male, and}$$

Table 4.1: Coagulation times for four diets.

A	B	C	D
62	63	68	56
60	67	66	62
63	71	71	60
59	64	67	61
	65	68	63
	66	68	64
			63
			59

$$E[\text{Salary}_i] = \mu + \text{Age}_i\beta + \alpha_f \quad \text{if the } i\text{'th person is female.}$$

We will find that  $\alpha_m$  does not appear in the output. The estimate of  $\mu$  depends upon the mean salary of the males in the sample. As a consequence the estimate of  $\alpha_f$  depends on the difference between the means of the salaries of females and males.

R does not distinguish between factors and numeric variables in the specification of models. We can fit models containing only continuous variables, stored as vector objects, as in

```
Fuel ~ Weight + Disp
```

above. We can fit models containing a mixture of vectors and factors, as in

```
Salary ~ Age + Sex
```

and we can fit models containing only factors. How does R tell if a variable is a factor or a numerical vector? If the entries in a vector are characters (i.e. enclosed in quotes), there is no problem. All character vectors are taken to be factors. However, sometimes numbers are used as factor levels e.g. we might have treatment 1, treatment 2 etc. We want the labels 1 and 2 to be regarded as levels of a factor, not values of a numerical variable. In this case we declare the variable to be a factor using the `factor` function. See the next section for examples.

In earlier courses, models only involving only factors and no continuous explanatory variables have been referred to as “Analysis of Variance” or ANOVA models. Before the development of computers and statistical packages ANOVA models were treated differently from “regression” models, such as

```
Fuel ~ Weight + Disp.
```

which involve only continuous variables. This was because the arithmetic of fitting ANOVA models could often be arranged in such a way to not require inverting matrices, indeed not to require the use of matrices at all. Sometimes this led to considerable economies of calculation. Unfortunately it also has led to a false dichotomy in statistical practice - the belief that ANOVA and regression are different and that continuous variables and factors should not be freely mixed in a statistical investigation.

We will show the models of one- and two-way analysis of variance from 201/8 can be directly fitted in R using the function `lm` we have already used to do regression. We revisit the concept of interaction, and then return to models mixing factors and continuous variates.

## 4.2 One way Analysis of Variance

The simplest kind of experiments are those in which a single continuous response variable is measured a number of times for each of several levels of experimental factor. For example consider the data below which consist of blood coagulation times for each four diets.

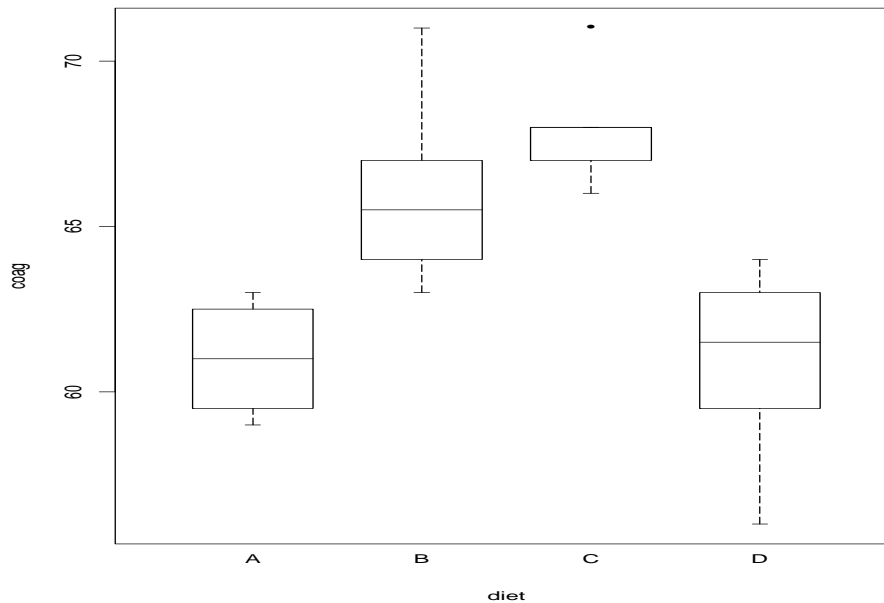


Figure 4.1: Boxplots of data for one way analysis of variance

Blood coagulation time, `coag`, is the response variable, and the factor, `diet`, has four levels: A, B, C, and D. The question of interest is whether the different levels of the factor, `diet`, have any effect on the means of the response.

Let us suppose that we have already read in the data above into the vector `coag`:

```
> coag
[1] 62 60 63 59 63 67 71 64 65 66 68 66 71 67 68 68 56 62 60 61 63 64
[23] 63 59
```

The factor `diet` can be formed using the statement

```
> diet <- factor(rep(LETTERS[1:4], c(4,6,6,8)))
```

and the two variables `coag` and `diet` combined to make a data frame.

```
> coag.df <- data.frame(diet,coag)
```

We carry out a preliminary graphical analysis of the data using the function `plot`.

When one of the arguments is a factor, the function produces a boxplot of the response data for each level of the factor:

```
> plot(diet,coag)
```

The plot produced by this function, shown in Figure 4.1, indicates that the spreads of the individual samples are comparable and so it is reasonable to proceed with fitting the model. We also see that diets B and C have a higher response than diets A and D.

We fit the model and print the analysis of variance table by typing

```
> coag.lm <- lm(coag ~ diet,data = coag.df)
> anova(coag.lm)
Analysis of Variance Table

Response: coag
Df Sum Sq Mean Sq F value    Pr(>F)
```

```
diet      3  228.0    76.0  13.571 4.658e-05 ***
Residuals 20  112.0     5.6
---
Signif. codes:  0  '***'  0.001  '**'   0.01  '*'   0.05  '.'   0.1  ' '   1
```

The  $p$ -value is calculated as 0.000047, which is certainly  $<0.001$  and so there is a significant difference between the means of the different diets. These probabilities should not be interpreted too rigidly, as the assumptions under which they are calculated are rarely more than approximately satisfied. Thus rather than quote a very small probability to many decimal places it is preferable merely to state that it is  $<0.001$ .

```
> summary(coag.lm)

Call:
lm(formula = coag ~ diet, data = coag.df)

Residuals:
    Min       1Q   Median       3Q      Max
-5.000e+00 -1.250e+00 -2.498e-16  1.250e+00  5.000e+00

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.100e+01  1.183e+00   51.554 < 2e-16 ***
dietB        5.000e+00  1.528e+00    3.273 0.003803 **
dietC        7.000e+00  1.528e+00    4.583 0.000181 ***
dietD        2.991e-15  1.449e+00  2.06e-15 1.000000
---
Signif. codes:  0  '***'  0.001  '**'   0.01  '*'   0.05  '.'   0.1  ' '   1

Residual standard error: 2.366 on 20 degrees of freedom
Multiple R-Squared:  0.6706,    Adjusted R-squared:  0.6212
F-statistic: 13.57 on 3 and 20 degrees of freedom,    p-value: 4.658e-05
```

We can confirm where the differences are from the summary output. The mean of diet A is 61 with standard error 1.2. The difference between the means of diet B and diet A is estimated to be 5 and the standard error of the difference is 1.5. The  $t$ -statistic of 3.27 is found by dividing the estimate of the difference by its standard error. The residual sum of squares, and hence the estimates of the standard errors have 20 degrees of freedom. Referring the  $t$ -statistic, 3.27, to a  $t$ -distribution with 20 degrees of freedom we find that the  $p$ -value is 0.004 confirming that diet B has a higher mean than diet A. Similarly the mean of diet C is 7 units greater than that of diet A with  $SED = 1.5$  which has a  $t$ -statistic with a  $p$ -value  $<0.001$ . As the plots indicated the sample means of diets A and D are identical.

### 4.3 Two-way analysis of variance

Penicillin was the first antibiotic and was isolated from a living fungus, *Penicillium*. Its commercial development required the determination of exact and optimal conditions for fungal growth to maximise the production of the drug. The treatments A, B, C and D represent different choices of growth conditions. The second factor, blend, which has five levels, represents 5 different batches of fungus. The main question of interest is whether the treatment factor affects the yield of penicillin. The factor blend is only of secondary interest, introduced to increase the precision of the comparisons between levels of the treatment factor.

The factors and the vector `yield` containing the responses are constructed and placed into a data frame:

```
> blend <- factor(rep(paste("Blend ",1:5),4))
> treatment <- factor(rep(LETTERS[1:4],rep(5,4)))
> yield <- c(89,84,81,87,79,88,77,87,92,81,97,92,87,89,80,94,79,85,84,88)
```



```
> pen.df <- data.frame(blend,treatment,yield)
```

The boxplots shown in Figure 4.2, suggest that the differences between treatment means are smaller than the differences between the blends. The boxplots for both blends and treatments indicate essentially constant variability, symmetric distributions and show no evidence of outliers.

```
> pen.lm <- lm(yield ~ blend + treatment,data = pen.df)
> anova(pen.lm)
```

Analysis of Variance Table

Response: yield

Terms added sequentially (first to last)

	Df	Sum of Sq	Mean Sq	F Value	Pr(F)
blend	4	264	66.00000	3.504425	0.0407462
treatment	3	70	23.33333	1.238938	0.3386581
Residuals	12	226	18.83333		

After fitting the two-way analysis of variance model,  $\text{yield} \sim \text{blend} + \text{treatment}$ , we see from the analysis of variance table that the observed differences between the treatment means are small compared to the variation within the data, the  $p$ -value is 0.33. The  $p$ -value calculated from the differences between the blend means is 0.04, suggesting that there is some evidence for blend differences.

There is only one observation on each treatment for each blend. We cannot see directly whether the differences between the treatments were similar in blends 2 and 5 with low yields to differences between the treatments in the high yielding blend 1. To do this we need to take more than one observation for each treatment blend combination.

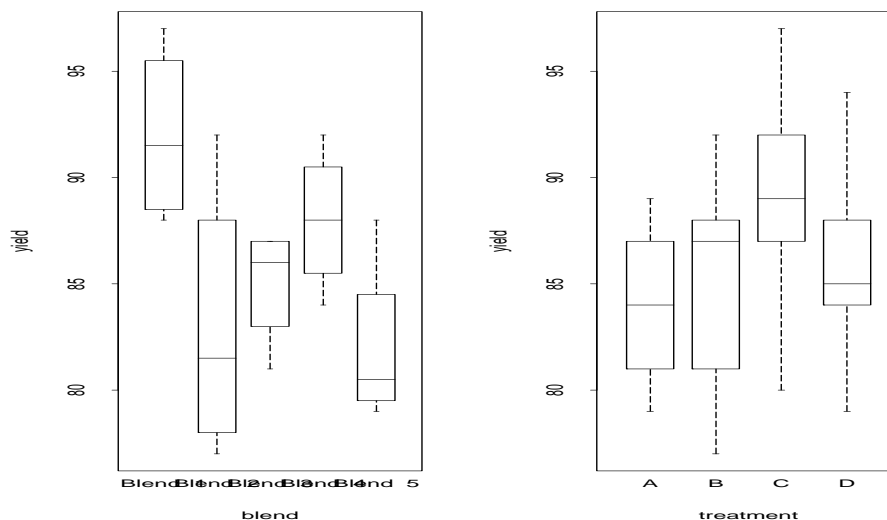


Figure 4.2: Boxplots of data for two way analysis of variance

## 4.4 Interaction

If the differences in mean responses between the levels of one factor are the same for all levels of the other factor, then we say that the two factors do not interact. If, in contrast, the first factor behaves differently at different levels of the second factor then the simple additive two-way analysis of variance model will not fit the data well, and so will not provide an adequate summary. The model is extended by fitting the interaction between the two factors.

	source=Beef	source=Cereal	Source=Pork
amount=High	$\mu_{11}$	$\mu_{12}$	$\mu_{13}$
amount=Low	$\mu_{21}$	$\mu_{22}$	$\mu_{23}$

Figure 4.3: Table of means

The first textbook on Applied Statistics, Snedecor's *Statistical Methods*, was published in 1936. It is still in print, in its 8th edition, Snedecor, G.W., and Cochran, W.G., (1989), *Statistical Methods*. An example from the first edition describes an experiment on the weight gains of rats. There were two treatment factors, source, the source of protein (beef, pork or cereal), and its amount (high or low). The 60 rats that were the experimental units were allocated at random into six groups each of ten rats, one for each treatment combination.

```
> gain <- c(,73,98,94,90,107,49,102,74,79,76,95,82,118,56,96,90,97,73,
104,111,98,64,80,86,81,95,102,86,98,81,107,88,102,51,74,97,
100,82,108,72,74,106,87,77,91,90,67,70,117,86,120,95,89,61,
111,92,105,78,58,82)
> source <- factor(rep(c("Beef", "Cereal", "Pork"), 20))
> level <- factor(rep(rep(c("High", "Low"), rep(3, 2)), 10))
> diets.df <- data.frame(gain, source, level)
```

We now fit the model `gains ~ source + amount + source:amount`. This model fits a separate mean to each combination of source and level - six means in all. We can lay out the means in a table as shown in Figure 4.3. We call the first level of each factor the “baseline level”. The table is made up of cells, corresponding to the intersection of a row and column. The cell in row  $i$  and column  $j$  is the “ $i, j$ ” cell. The mean in that cell is written  $\mu_{ij}$  and represents the population mean of all observations taken when “amount” is at level  $i$  and “source” at level  $j$ .

### Baseline cell

The cell in row 1 column 1 is the “baseline cell”, corresponding to “amount” = High and “source” = Beef. The mean in this cell is the baseline mean, in symbols  $\mu_{11}$ .

### Main effects

We define a “level main effect” for each row of the table in Figure 4.3 (i.e. each level), except for the baseline row. This has main effect zero by definition. The main effect for row  $i$  is defined as the difference between the mean in row  $i$ , column 1, and the baseline mean, in symbols  $\mu_{i1} - \mu_{11}$ . This is the difference marked A in the figure above. It measures the effect on the mean from changing “amount” from the baseline to some other row  $i$  (assuming the other factor “source” is at its baseline.) Similarly, there is a main effect for each column, apart from the first. The main effect for column  $j$  is defined as the difference between the mean in row 1, column  $j$ , and the baseline mean, in symbols  $\mu_{1j} - \mu_{11}$ . This is the difference B in the figure above, and is the effect on the mean of changing “source” from the baseline level to some other level  $j$ , assuming “amount” is at its baseline.

Table 4.2: Means for the rat example.

	source=Beef	source=Cereal	source=Pork
level=High	100	85.9	99.5
level=Low	79.2	83.9	78.7

### Interactions

By convention, we measure the main effects of a factor when the other factor is at its baseline level. If we set the other factor to have some other level, the result would in general be different. This difference marked A in Figure 4.3, and would in general be different from the difference  $\mu_{i3} - \mu_{13}$  marked C. The interaction is defined as the difference of these differences: each cell (except those in the baseline row and the baseline column) has an interaction. The interaction for the cell in row  $i$ , column  $j$  is the difference between the main effect of “amount” for row  $i$  (assuming “source” is at its level  $j$ ) and the main effect of “amount” for row  $i$  (assuming “source” is at its baseline). Thus the interaction for the “ $i, j$ ” cell is  $(\mu_{ij} - \mu_{1j}) - (\mu_{i1} - \mu_{11})$  which can be written  $\mu_{ij} - \mu_{1j} - \mu_{i1} + \mu_{11}$ . It follows that we can express the means  $\mu_{ij}$  in terms of the baseline level, the main effects and the interactions:

$$\begin{aligned}\mu_{ij} &= \mu_{11} + \{\mu_{i1} - \mu_{11}\} + \{\mu_{1j} - \mu_{11}\} + \{\mu_{ij} - \mu_{i1} - \mu_{1j} + \mu_{11}\} \\ &= \text{overall level} + \text{amount main effect} + \text{source main effect} + \text{interaction}\end{aligned}$$

If all the interactions are zero, the differences between the means in row  $i$  and row 1 are the same for every column, so the effect of changing the factor “amount” does not depend on which level “source” is at. This simplifies the interpretation of the main effects. We can also define the interactions as the differences in source main effects when “amount” is at level  $i$  and level 1. This gives exactly the same definition.

### Numerical example

The table of sample means is shown in Table 4.2. The main effect for level = Low (when source is at its baseline) is  $79.2 - 100 = -20.8$ . The effect of level (when source = Pork) is  $78.7 - 99.5 = -20.8$ . The interaction for the 2,3 cell (i.e. for level = Low, source = Pork) is  $-20.8 - (-20.8) = 0$ . On the other hand, the difference for source = Pork (when level = Low) is  $78.7 - 79.2 = -0.5$ . The main effect for Source = pork (when level is at its baseline) is  $99.5 - 100 = -0.5$ . The interaction for the 2,3 cell (i.e. for level = Low, source = Pork) is  $-0.5 - (-0.5) = 0$ . Thus we get the same result.

The model formula `gain ~ source + level + source:level` indicates to R that we want to split up the means as described above. The term `source:level` in the model is the interaction between the factors source and level. It allows us to assess whether the differences between diets containing high and low amounts of protein is the same irrespective of the protein’s source.

```
> diets.lm <- lm(gain ~ source + level + source:level, data=diets.df)
> anova(diets.lm)
Analysis of Variance Table

Response: gain
          Df Sum Sq Mean Sq F value    Pr(>F)
source      2   266.5    133.3   0.6211 0.5411319
level       1  3168.3   3168.3  14.7666 0.0003224 ***
source:level 2  1178.1    589.1   2.7455 0.0731879 .
Residuals   54 11586.0    214.6
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that the  $p$ -value for the interaction term, is 0.07 - suggesting that the differences between high and low protein diets may not be the same for each source of protein. This can be checked using

the function `interaction.plot`. This function takes three arguments: The first two arguments are the two factors and the third is a vector containing the responses. An interaction plot shows the mean gain for each level of the second factor, `amount`, plotted at each level of the first factor, `source`. To produce the plot, type

```
> interaction.plot(source,amount,gains)
```

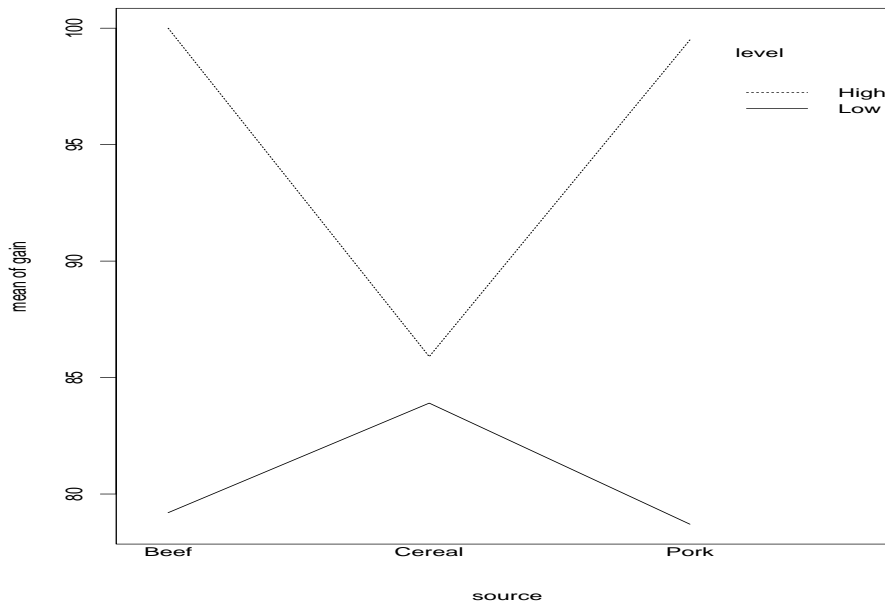


Figure 4.4: Interaction plot for the rat data

This plot, shown in Figure 4.4, shows very dramatically that the difference in weight gain for the two diets with cereal as a protein source are very close, unlike the differences for the two meat based diets. Since the response of the factor `amount` is not the same at each level of `source` we say that the factors interact. An interaction plot of two factors that do not interact will have profiles or traces that are approximately parallel.

If there are an equal number of observations for each possible combination of levels of all the factors then we say that the design is *balanced*. For example, each combination of protein source and protein level was tested on 10 rats, so the data set in the dataframe `rats.df` is balanced. For balanced data we can carry out a more informative analysis, with better summary functions, using the command `aov`. This topic is studied in the course STATS 340.

## 4.5 Interactions between factors and continuous variables

The data frame `car.test.frame` contains data on 60 different American cars, and is found in the R package `rpart`. The variables are `Price`, `Country`, `Reliability`, `Mileage`, `Type`, `Weight`, `Disp.` and `HP`. All but `Country` and `Type` can be treated as continuous variables, while `Country` is a factor giving the country of origin of the vehicle and `Type` describes the type of vehicle. Suppose that we wish to build a model which relates a car's fuel economy `Mileage` in miles per gallon (m.p.g. - this is American data) to the size of its engine `Disp.` in cubic inches and to what type of vehicle it is. We could start by checking that there is at least a gross relationship between the size of a car's engine and its fuel economy.

```
> library(rpart)
> data(car.test.frame)
> car.m1 <- lm(Mileage ~ Disp.,data=car.test.frame)
```

Table 4.3: Sample of data from data frame `car.test.frame`

	Mileage	Disp.	Type
Eagle Summit	33	97	Small
Subaru Justy	34	73	Small
Honda Accord	26	132	Compact
Acura Legend	20	163	Medium
Buick Le Sabre	23	231	Large
Mitsubishi Wagon	20	143	Van

```
> anova(car.m1)
Analysis of Variance Table

Response: Mileage
      Df Sum Sq Mean Sq F value    Pr(>F)    
Disp.    1  650.90   650.90   53.649 8.348e-10 ***
Residuals 58  703.68    12.13                 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(car.m1)

Call:
lm(formula = Mileage ~ Disp., data = car.test.frame)

Residuals:
      Min       1Q   Median       3Q      Max
-6.6477 -2.2328 -0.8693  2.9120  8.0595

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 33.907958   1.350146  25.114 < 2e-16 ***
Disp.       -0.061326   0.008373  -7.325 8.35e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.483 on 58 degrees of freedom
Multiple R-Squared:  0.4805,    Adjusted R-squared:  0.4716 
F-statistic: 53.65 on 1 and 58 degrees of freedom,    p-value: 8.348e-10
```

Not surprisingly, as the size of a car's engine increases by one cubic inch, its fuel economy decreases by 0.061 mpg, and the slope of this regression line is different from zero with a  $p$ -value  $< 0.001$ . Thus we have fitted a single straight line to the data, with an intercept of 34 (standard error 1.4) and slope of -0.061 (s.e. 0.0084).

We could also treat the data as though a one way analysis of variance model were appropriate, using `Type` as an explanatory variable. `Type` is a factor having 6 levels, which we can inspect by attaching the data frame and using the function `levels`:

```
> levels(car.test.frame$Type)
[1] "Compact" "Large"   "Medium"  "Small"   "Sporty"  "Van"
```

We can fit the model us usual using `lm`:

```
> car.m1a <- lm(Mileage ~ Type, data=car.test.frame)
> anova(car.m1a)
```

## Analysis of Variance Table

Response: Mileage

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Type	5	943.02	188.60	24.746	7.213e-13 ***
Residuals	54	411.56	7.62		

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

&gt; summary(car.m1a)

Call:

lm(formula = Mileage ~ Type, data = car.test.frame)

Residuals:

	Min	1Q	Median	3Q	Max
	-7.0000	-1.1333	0.1868	1.2308	7.0000

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	24.1333	0.7128	33.856	< 2e-16 ***
TypeLarge	-3.8000	1.7460	-2.176	0.033921 *
TypeMedium	-2.3641	1.0461	-2.260	0.027886 *
TypeSmall	6.8667	1.0461	6.564	2.1e-08 ***
TypeSporty	1.8667	1.1640	1.604	0.114628
TypeVan	-5.2762	1.2637	-4.175	0.000109 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.761 on 54 degrees of freedom

Multiple R-Squared: 0.6962, Adjusted R-squared: 0.668

F-statistic: 24.75 on 5 and 54 degrees of freedom, p-value: 7.213e-13

This model has a smaller residual mean square (7.6), than the simple regression on `Disp.` (12.1). There are very significant differences between the `Types` of cars. As expected, Vans have worse fuel economy than Compact cars.

These analyses suggest using a model that includes both the `Type` of car and the size of its engine, `Disp.`. We have several choices at this point. Let us suppose that  $y_i$  is the `Mileage` of the  $i$ 'th car, that it has engine size, `Disp.`, of  $x_i$  and that it is of `Type`  $j$ . One possible model is

Mileage ~ Type + Disp.

or, in algebraic notation

$$E[y_i] = \mu + \alpha_j + x_i\beta. \quad (4.1)$$

The model (4.1) postulates that the change in fuel economy, for a unit change in engine size is the same,  $\beta$ , for each type of car, but that intercept,  $\mu + \alpha_j$ , differs from type to type of car. Thus the model corresponds to fitting parallel lines (hence with common slope) to the data. Each level of the factor `Type` has its own line, with its own intercept. The differences between the intercepts,  $\mu + \alpha_j$ , give estimates of the differences in fuel economy between cars of different types, but with the same size of engine.

The second model we might fit is

Mileage ~ Type:Disp.

which in algebraic notation is

$$E[y_i] = \mu + x_i\beta_j. \quad (4.2)$$

This model fits a common intercept to the whole data set, but postulates that each type of car has its own slope,  $\beta_j$ . It postulates that the change in `Mileage` due to a given increase in engine size is the same for all cars of a given type, but may vary between the different levels of `Type`.

These two models have the same number of parameters, and one is not a submodel of the other so they cannot formally be compared. Informally their goodness of fit to the data can be compared graphically using residual plots, and by comparing their residual sums of squares.

Fitting the first model gives

```
> car.m2 <- lm(Mileage ~ Type+Disp.,data=car.test.frame)
> anova(car.m2)
Analysis of Variance Table

Response: Mileage
      Df Sum Sq Mean Sq F value    Pr(>F)
Type    5  943.02   188.60   37.757 2.220e-16 ***
Disp.    1  146.82   146.82   29.392 1.481e-06 ***
Residuals 53  264.75     5.00
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

and fitting the second gives

```
> car.m3 <- lm(Mileage ~ Type:Disp.,data=car.test.frame)
> anova(car.m3)
Analysis of Variance Table

Response: Mileage
      Df Sum Sq Mean Sq F value    Pr(>F)
Type:Disp. 6 1049.77   174.96   30.422 1.665e-15 ***
Residuals 53  304.81     5.75
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The model (4.1) with a common slope and varying intercepts has a residual mean square of 5.00, which is a little smaller than that of 5.75 arising from the model (4.2) with a common intercept and varying slopes.

Both of these models are submodels of

```
Mileage ~ Type*Disp.
```

The operator “\*” is the “Crossing” operator. Formally it expands to

```
Mileage ~ Type + Disp. + Type:Disp.
```

What does this model mean in terms of slopes and intercepts? The term `Type + Disp.` gives us a single slope and a separate intercept for each level of `Type`, whereas `Type:Disp.` gives a model with separate slope for each level of `Type`. Thus

```
Mileage ~ Type*Disp.
```

is a model with a separate line, i.e. a separate slope and intercept, for each type of car.

```
> car.m4 <- lm(Mileage ~ Type*Disp.,data=car.test.frame)
> anova(car.m4)
Analysis of Variance Table

Response: Mileage
      Df Sum Sq Mean Sq F value    Pr(>F)
Type    5  943.02   188.60   42.355 2.220e-16 ***
Disp.    1  146.82   146.82   32.972 6.195e-07 ***
```

```

Type:Disp.  5  51.01   10.20   2.291   0.06027 .
Residuals  48 213.74    4.45
---
Signif. codes:  0  '***'  0.001  '**'  0.01  '*'  0.05  '.'  0.1  ' '  1

```

We see that this last model, which in algebraic notation is written

$$E[y_i] = \mu + \alpha_j + x_i\beta_j \quad (4.3)$$

has (of course) a smaller residual mean square, 4.45, than either of its sub-models, (4.1) and (4.2). We can picture all the models as a lattice, where two models are joined by a line if one is a submodel of the other. In Figure 4.5 each model is given together with its residual mean sum of squares and degrees of freedom. Note that the “null model” (fitting an intercept only) is also included in the Figure. This amounts to treating the data as a single batch: the residual mean square is the variance of the responses.

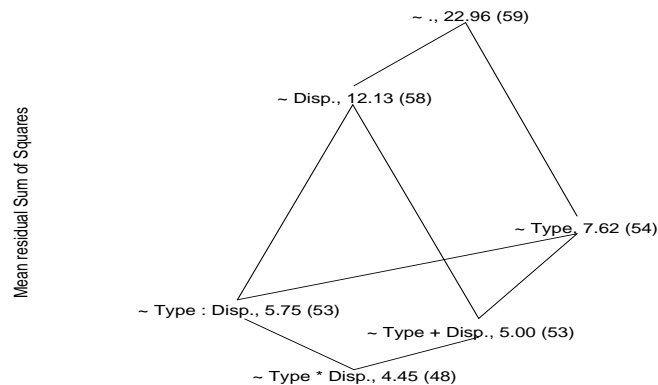


Figure 4.5: Lattice of models involving a factor and a continuous variable, with Residual Mean Sum of Squares (and degrees of freedom)

We can test whether it is necessary to consider a more complex model, or if one of its sub-models above it in the lattice provides an adequate description of the data by constructing F-statistics. For example, to test whether the model with one slope for **Disp.** is adequate or whether we need separate slopes and intercepts for every level of the factor **Type** we compare the models

```
Mileage ~ Type + Disp.
```

and

```
Mileage ~ Type * Disp.
```

Our F-statistic is

$$F = \frac{\frac{264.7454 - 213.7384}{53 - 48}}{4.4529} = 2.29097$$



which can be compared to an F-distribution with  $5 = 53 - 48$  and 48 degrees of freedom to give a  $p$ -value of 0.06027 suggesting that the simple model, with one slope

```
Mileage ~ Type + Disp.
```

is (just) adequate. Testing this model with several parallel lines against its two sub-models of just a single line, or the one way ANOVA model with no dependence on `Disp.` gives F-statistics of 15.1 and 27.8 respectively. Unsurprisingly both of these F-statistics give  $p$ -values  $< 0.001$ . The model with varying regression coefficient or slope for `Disp.` also has the single line and the one way ANOVA model as sub-models and it fits the data better than either of the sub-models.

The choice between these two models can either be made in terms of their subject matter interpretability, or if the diagnostic plots for each are comparable in terms of which fits the data better.

## 4.6 Chapter summary

### 4.6.1 Main ideas

- factors
- Models and factors
- Interpreting factor coefficients
- Anova as regression with factors
- Interaction
- Interaction between factors and continuous variables
- Parallel and non-parallel lines
- Model lattices

### 4.6.2 New R functions used

`factor`

`interaction.plot`

`levels`

`rep`

Use the R help system to find the details for these functions.



# Chapter 5

## Categorical Responses

### 5.1 Introduction

In this chapter we deal with regression models that are appropriate for categorical responses. First, we consider logistic regression models where the response is a binary variable that indicates whether or not a particular event has occurred. Then we consider the analysis of contingency tables where the response can be considered to be a count. To accomplish this, we need to make some adaptations to the techniques that were covered in earlier chapters.

### 5.2 Logistic regression analysis

**Example 1.** The data in Table 5.2 are taken from a book by Hosmer and Lemeshow (*Applied Logistic Regression, 2nd Ed.*, Wiley (2000)) and were gathered by examining 100 randomly selected patients and recording the presence or absence of coronary heart disease (**chd**, absent = 0 present = 1) and the patient's age (**age**). Our aim is to use the patient's age to predict the probability that the patient has coronary heart disease. It may be tempting to fit an ordinary regression model using **chd** as the response and **age** as an explanatory variable but a moment's thought should convince us this is not a good idea. First, a key assumption of ordinary regression is that the response has a Normal distribution. Clearly since **chd** can only take values 0 or 1 its distribution is not even close to Normal – a binomial distribution would be a much more suitable model. Second

Table 5.1: CHD and AGE for 100 patients

age	chd	age	chd	age	chd	age	chd	age	chd	age	chd
20	0	23	0	24	0	25	0	25	1	26	0
26	0	28	0	28	0	29	0	30	0	30	0
30	0	30	0	30	0	31	1	32	0	32	0
33	0	33	0	34	0	34	0	34	1	34	0
34	0	35	0	35	0	36	1	36	0	36	0
37	0	37	1	37	0	38	0	38	0	39	0
39	1	40	0	40	1	41	0	41	0	42	0
42	0	42	0	41	1	43	0	43	0	43	1
44	0	44	0	44	1	44	1	45	0	45	1
46	0	46	1	47	0	47	0	47	1	48	0
48	1	48	1	49	0	49	0	49	1	50	0
50	0	50	1	52	0	52	1	53	1	53	1
54	1	55	0	55	1	55	1	56	1	56	1
56	1	57	0	57	0	57	1	57	1	57	1
57	1	57	1	58	1	58	1	59	1	59	1
60	0	60	0	61	1	62	1	62	1	63	1
64	0	64	1	65	1	69	1				

as we want to predict a probability we would like a model that produces predictions that are between 0 and 1. There is no guarantee that this will be the case if we use ordinary regression.

We will use a technique known as logistic regression to create a suitable model. Let  $\pi$  represent the probability a patient will have CHD. The logistic model that relates  $\pi$  to the age of the patient has the form:

$$\pi = \frac{\exp(\beta_0 + \beta_1 \text{age})}{1 + \exp(\beta_0 + \beta_1 \text{age})}. \quad (5.1)$$

Notice that this model ensures that  $0 < \pi < 1$ , since (i)  $\exp(\beta_0 + \beta_1 \text{age}) > 0$  and (ii)  $1 + \exp(\beta_0 + \beta_1 \text{age}) > \exp(\beta_0 + \beta_1 \text{age})$ .

In general, a logistic model that uses a single numeric explanatory variable has a sigmoidal (S or reverse S) shape. Examples are shown in Figure 5.1. The values of  $\beta_0$  and  $\beta_1$  determine the characteristics of the logistic model:

1. The sign of  $\beta_1$  determines whether  $\pi$  increases ( $\beta_1 > 0$ ) or decreases ( $\beta_1 < 0$ ) as the value of the explanatory variable increases.
2. The absolute value of  $\beta_1$  determines how sensitive  $\pi$  is to changes in the explanatory variable – the larger the value of  $|\beta_1|$  the more sensitive  $\pi$  is to changes in the value of the explanatory variable. The curves in Figure 5.1 would become steeper if  $|\beta_1|$  was increased and more gradual if  $|\beta_1|$  was decreased.
3. The value of  $\beta_0$  determines the value of  $\pi$  when the explanatory variable is 0. Thus the value of  $\beta_0$  controls the location of the logistic curve on the  $x$ -axis. If we were to change  $\beta_0$  (but hold  $\beta_1$  constant) then the curves in Figure 5.1 would stay the same shape but would be shifted left or right along the  $x$ -axis.

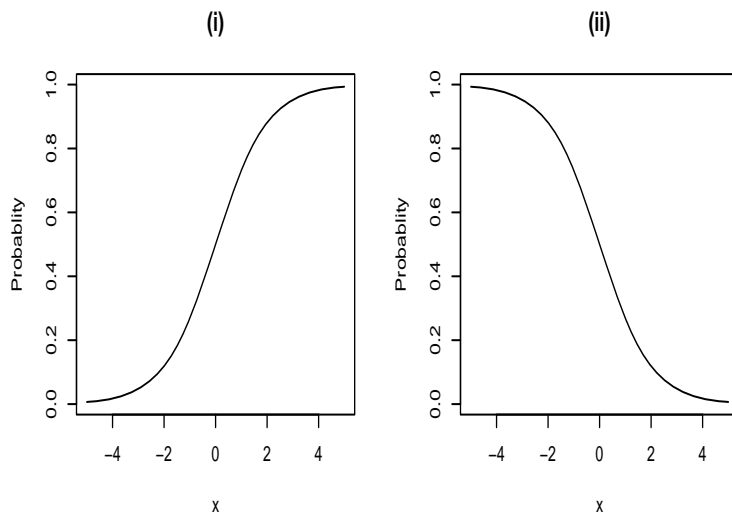


Figure 5.1: Shape of the logistic curve  $\pi(x) = \exp(\beta_0 + \beta_1 x) / (1 + \exp(\beta_0 + \beta_1 x))$ . (i)  $\beta_1 > 0$ , (ii)  $\beta_1 < 0$ .

### 5.2.1 Fitting the model

For ordinary regression models, least squares is the standard method used to estimate the model parameters. Least squares works well when the response is Normal and has constant variance but it is not so good for fitting logistic regression models. Instead, we will use a method known as *maximum likelihood*. The idea behind maximum likelihood estimation is to select the values for the parameters that are most compatible with the observed data. Put another way, the fitted parameters are those values that make the observed data most likely to occur. Note that if the

response is Normal, then the maximum likelihood method gives the same estimated coefficients as least squares.

Consider the CHD data. We assume that for each patient our response variable `chd` has a binary distribution (binomial with  $n = 1$ ). Further we assume that  $\pi_i$ ,  $\Pr(\text{chd} = 1)$  for patient  $i$ , is related to the age of patient  $i$  in the manner described by equation 5.1. Thus we have the following probability function for `chdi`:

$$\begin{aligned}\Pr(\text{chd}_i = 1) &= \frac{\exp(\beta_0 + \beta_1 \text{age}_i)}{1 + \exp(\beta_0 + \beta_1 \text{age}_i)}, \\ \Pr(\text{chd}_i = 0) &= 1 - \frac{\exp(\beta_0 + \beta_1 \text{age}_i)}{1 + \exp(\beta_0 + \beta_1 \text{age}_i)} = \frac{1}{1 + \exp(\beta_0 + \beta_1 \text{age}_i)}.\end{aligned}\tag{5.2}$$

Provided that we assume that the observations are independent, we can calculate the probability of getting the observed set of values for the `chdi`'s by multiplying the individual probabilities:

$$\Pr(\text{chd}_1 = 0, \text{chd}_2 = 0, \dots, \text{chd}_{100} = 1) = \Pr(\text{chd}_1 = 0) \times \Pr(\text{chd}_2 = 0) \times \dots \times \Pr(\text{chd}_{100} = 1).$$

Thus by plugging in the appropriate expression from (5.2) for each term we can get an expression for the probability of getting the observed data that is a function of  $\beta_0$  and  $\beta_1$ . This expression is called the *likelihood function* and is denoted by  $L(\beta_0, \beta_1)$ . The *maximum likelihood estimates* of  $\beta_0$  and  $\beta_1$  are the values of  $\beta_0$  and  $\beta_1$  that maximise  $L(\beta_0, \beta_1)$ . For our current example  $L(\beta_0, \beta_1)$  is quite a complicated expression and it would be very tedious to find the maximum likelihood estimates (MLE's) by hand. Of course, computer programmes exist that use numerical algorithms to find the MLE's and we will use one of these.

**Example 2.** To fit the logistic model to our CHD data, we can use the `glm` function in R. GLM stands for “generalised linear model” and designates a class of statistical models that includes both the logistic model and the ordinary (Normal) regression model.

The `glm` function works in much the same manner as the `lm` function. Suppose we have entered the data from Table 5.1 into a data frame in R named `chd.df`. Then we can produce a “glm object” that contains the fitted logistic regression model by typing

```
> chd.glm<-glm(chd~age,family=binomial,data=chd.df)
```

Notice that the `family=binomial` option is included in this command to indicate that the response variable is assumed to have a binomial distribution, and that the logistic curve is to be used. We can examine the fitted model using the `summary` function just as we did for regression objects produced using `lm`.

```
> summary(chd.glm)

Call:
glm(formula = chd ~ age, family = binomial, data = chd.df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9686  -0.8480  -0.4607   0.8262   2.2794

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.2784     1.1296  -4.673 2.97e-06 ***
age           0.1103     0.0240   4.596 4.30e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 136.66  on 99  degrees of freedom
Residual deviance: 107.68  on 98  degrees of freedom
AIC: 111.68

Number of Fisher Scoring iterations: 3
```

This summary is similar to that produced for a `lm` object but there are several differences. For now, just note that the MLE's for the two parameters are in the `Estimate` column in the section headed `Coefficients` (the other parts of this output will be discussed later).

The fitted logistic regression model can be written in a number of different forms. We will consider three: the “logistic form”, the “odds form”, and the “logit form”. Keep in mind that these three forms are simply different ways of expressing the same model.

The logistic form of the fitted model is obtained by putting the estimates  $\hat{\beta}_0 = -5.2784$  and  $\hat{\beta}_1 = 0.1103$  into (5.1):

$$\hat{\pi} = \frac{\exp(-5.2784 + 0.1103 \text{ age})}{1 + \exp(-5.2784 + 0.1103 \text{ age})}.$$

This model relates the probability a subject has CHD,  $\pi$ , to the subject's age. The fitted model is superimposed on a plot of `chd` versus `age` in Figure 5.2. Note that as age increases the proportion of subjects with `chd` (`chd` = 1) increases and that this trend is captured by the fitted logistic model. To predict  $\pi$  for a specific value of age we can simply plug the value of age into this formula or we can use the `predict` function in R. For example to get the estimated probability of CHD for `age` = 45:

```
> predict(chd.glm,data.frame(age=45),type="response")
[1] 0.4221367
```

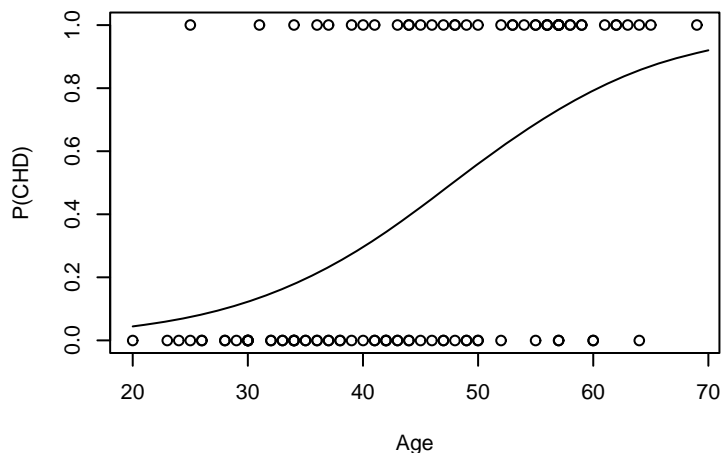


Figure 5.2: Fitted logistic regression model for the CHD example.

A second way to express the logistic model is in terms of odds. Recall that the odds an event  $E$  occurs is the probability  $E$  occurs divided by the probability  $E$  doesn't occur:  $\text{odds}(E) = \Pr(E)/(1 - \Pr(E))$ . Thus if  $\text{odds}(E) = 2$  this means that the probability  $E$  occurs is twice the probability that  $E$  does not occur which implies  $\Pr(E) = 2/3$ . The odds of CHD can be estimated by  $\widehat{\text{odds}}(\text{chd}) = \hat{\pi}/(1 - \hat{\pi})$ . Using the expression for  $\hat{\pi}$  from our fitted model and a bit of algebraic manipulation gives:

$$\widehat{\text{odds}}(\text{chd}) = \exp(-5.2784 + 0.1103 \text{ age}).$$

One advantage of this form of the logistic regression model is that it allows the coefficient of `age` to be interpreted in a convenient manner. Suppose we fix `age` at a specified value  $\text{age}_o$  and consider the effect on  $\widehat{\text{odds}}(\text{chd})$  of increasing `age` by 1 year. For  $\text{age} = \text{age}_o$  we can write:

$$\begin{aligned} \widehat{\text{odds}}(\text{chd} | \text{age} = \text{age}_o) &= \exp(-5.2784 + 0.1103 \text{ age}_o) \\ &= \exp(-5.2784) \times \exp(0.1103 \text{ age}_o). \end{aligned}$$

For  $\mathbf{age} = \mathbf{age}_o + 1$  we have:

$$\begin{aligned}\widehat{\text{odds}}(\text{chd}|\mathbf{age} = \mathbf{age}_o + 1) &= \exp(-5.2784 + 0.1103(1 + \mathbf{age}_o)) \\ &= \exp(-5.2784) \times \exp(0.1103) \times \exp(0.1103 \mathbf{age}_o). \\ &= \exp(0.1103) \times \widehat{\text{odds}}(\text{chd}|\mathbf{age} = \mathbf{age}_o).\end{aligned}$$

Thus for each increase of 1 year in age, the estimated odds of CHD is multiplied by  $\exp(0.1103) = 1.117$ .

The third way we can write the logistic regression function is in the *logit* form. The logit function is simply the log-odds:

$$\text{logit}(\mathbf{E}) = \log \frac{\Pr(\mathbf{E})}{1 - \Pr(\mathbf{E})}.$$

Thus by taking the log of each side of the odds form of the fitted model we get the logit form:

$$\widehat{\text{logit}}(\text{chd}) = -5.2784 + 0.1103\mathbf{age}.$$

Note that now the right hand side of the model is simply a linear function of  $\mathbf{age}$ . Thus for each increase in  $\mathbf{age}$  of one unit  $\widehat{\text{logit}}(\text{chd})$  increases by 0.1103.

We now consider statistical inference for the regression coefficients. The estimates of  $\hat{\beta}_0$  and  $\hat{\beta}_1$  have sampling distributions that are asymptotically Normal and thus we can use the Normal distribution to produce (approximate) confidence intervals and perform hypothesis tests. The section headed **Coefficients** from the output generated by the **summary** function provides us with the estimated values of the parameters, the standard errors of the estimates, a test statistic for testing  $\beta_j = 0$ , and the  $p$ -value for this test. Thus the line for **age** indicates that the maximum likelihood estimate of  $\beta_1$  is  $\hat{\beta}_1 = 0.1103$  and that the standard error of this estimate is  $s.e.(\hat{\beta}_1) = 0.0240$ . Further it indicates that we can test  $H_0: \beta_1 = 0$  using the statistic  $z = \hat{\beta}_1 / s.e.(\hat{\beta}_1) = 4.596$  which produces a  $p$ -value of  $4.30\text{e-}06$  (very strong evidence against  $H_0$ ). The given  $p$ -value represents  $2 \times \Pr(Z \geq 4.596)$  where  $Z \sim N(0, 1)$ . A 95% confidence interval for  $\beta_1$  can be calculated as

$$\begin{aligned}\hat{\beta}_1 &\pm s.e.(\hat{\beta}_1) \times 1.960 \\ 0.1103 &\pm 0.0240 \times 1.960 \\ 0.1103 &\pm 0.0470.\end{aligned}$$

since  $\Pr(-1.96 \leq Z \leq 1.96) = .95$ .

### 5.2.2 Multiple logistic regression

The logistic regression model can easily be extended to situations where there is more than one explanatory variable. As before, we wish to model the probability that an event  $E$  occurs which we will denote by  $\pi$ . However now  $\pi$  will depend on the values of several explanatory variables  $X_1, \dots, X_k$ . The multiple logistic regression model is:

$$\pi = \frac{\exp(\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k)}{1 + \exp(\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k)}.$$

This model can also be written in the odds form,

$$\frac{\pi}{1 - \pi} = \exp(\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k),$$

or in the logit form,

$$\log \frac{\pi}{1 - \pi} = \beta_0 + \beta_1 X_1 + \dots + \beta_k X_k.$$

The parameters are estimated using the maximum likelihood method. As before, the likelihood function  $L$  uses the assumed model to express the probability of getting the observed data as

Table 5.2: Data for Example 5.

Obs	Kyphosis	Age	Number	Start	Obs	Kyphosis	Age	Number	Start
1	absent	71	3	5	41	present	73	5	1
2	absent	158	3	14	42	absent	35	3	13
3	present	128	4	5	43	absent	143	9	3
4	absent	2	5	1	44	absent	61	4	1
5	absent	1	4	15	45	absent	97	3	16
6	absent	1	2	16	46	present	139	3	10
7	absent	61	2	17	47	absent	136	4	15
8	absent	37	3	16	48	absent	131	5	13
9	absent	113	2	16	49	present	121	3	3
10	present	59	6	12	50	absent	177	2	14
11	present	82	5	14	51	absent	68	5	10
12	absent	148	3	16	52	absent	9	2	17
13	absent	18	5	2	53	present	139	10	6
14	absent	1	4	12	54	absent	2	2	17
15	absent	168	3	18	55	absent	140	4	15
16	absent	1	3	16	56	absent	72	5	15
17	absent	78	6	15	57	absent	2	3	13
18	absent	175	5	13	58	present	120	5	8
19	absent	80	5	16	59	absent	51	7	9
20	absent	27	4	9	60	absent	102	3	13
21	absent	22	2	16	61	present	130	4	1
22	present	105	6	5	62	present	114	7	8
23	present	96	3	12	63	absent	81	4	1
24	absent	131	2	3	64	absent	118	3	16
25	present	15	7	2	65	absent	118	4	16
26	absent	9	5	13	66	absent	17	4	10
27	absent	8	3	6	67	absent	195	2	17
28	absent	100	3	14	68	absent	159	4	13
29	absent	4	3	16	69	absent	18	4	11
30	absent	151	2	16	70	absent	15	5	16
31	absent	31	3	16	71	absent	158	5	14
32	absent	125	2	11	72	absent	127	4	12
33	absent	130	5	13	73	absent	87	4	16
34	absent	112	3	16	74	absent	206	4	10
35	absent	140	5	11	75	absent	11	3	15
36	absent	93	3	16	76	absent	178	4	15
37	absent	1	3	9	77	present	157	3	13
38	present	52	5	6	78	absent	26	7	13
39	absent	20	6	9	79	absent	120	2	13
40	present	91	5	12	80	present	42	7	6
					81	absent	36	4	13



function of the unknown parameters  $\beta_0, \beta_1, \dots, \beta_k$ . The values of the parameters that maximize  $L$  (the MLE's) can be found using the `glm` function in R.

**Example 3.** The data in Table 5.1 were collected on 83 children undergoing corrective spinal surgery. The objective was to identify risk factors for kyphosis (flexing of the spine) following surgery. The variable **Kyphosis** is a binary response with “present” indicating the presence of kyphosis and “absent” the absence of kyphosis.

The risk factors studied were age in months (**Age**), the starting vertebrae level of the surgery (**Start**), and the number of vertebrae involved (**Number**). The data are in a data frame called `kyphosis.df`.

The first step is to plot the variables:

```
> pairs20x(kyphosis.df)
```

The plot of the response versus age, shown in Figure 5.3, indicates that as age increases the risk

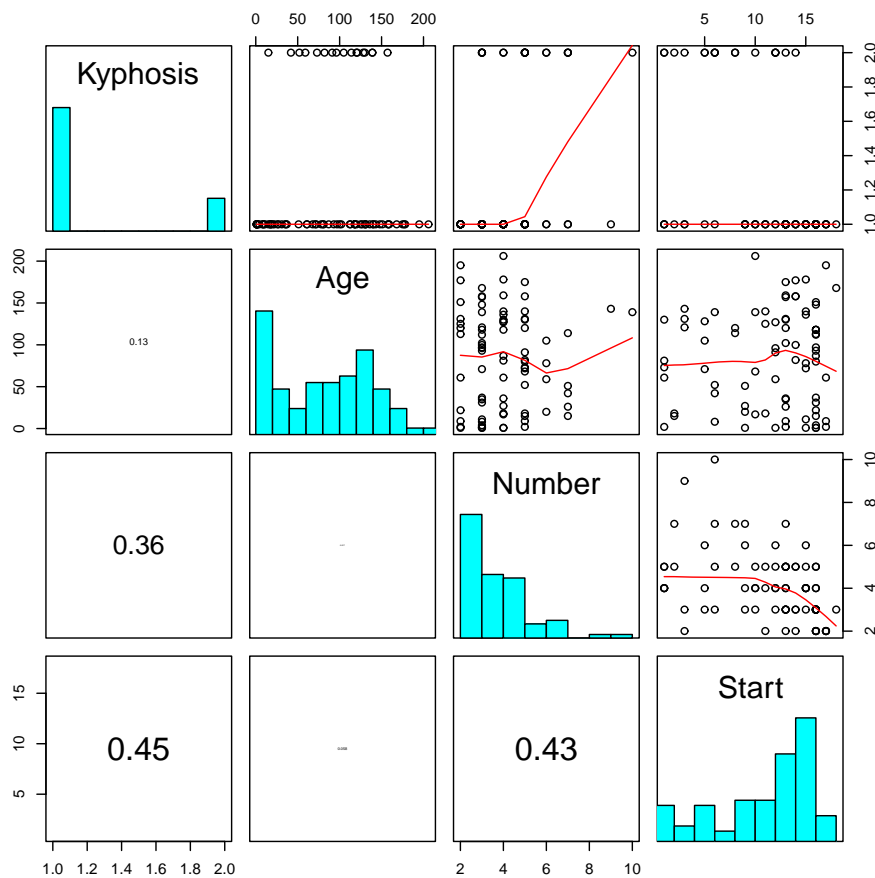


Figure 5.3: Scatterplots for the kyphosis data.

first increases and then decreases. This suggests that a quadratic term in age should be used to model this aspect of the data. Note that R is not fazed by the fact that the response is a factor: it treats “absent” as 1 and “present” as 2.

When the response is a factor, as is the case here, R interprets the first level as 0 and the second (“present”) as 1, so we are modelling the probability that kyphosis is present. To deal with the quadratic term, we fit a model of the form

$$\text{logit}(\text{Kyphosis present}) = \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Age}^2 + \beta_3 \text{Number} + \beta_4 \text{Start}$$

to the data, using the command:

```
> kyphosis.glm<-glm(Kyphosis~Age+I(Age^2)+Number+Start, family=binomial,
                    data=kyphosis.df)
> summary(kyphosis.glm)
```

Call:

```
glm(formula = Kyphosis ~ Age + I(Age^2) + Number + Start,
     family = binomial, data = kyphosis.df)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.23572	-0.51241	-0.24509	-0.06109	2.35494

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-4.3834531	2.0478366	-2.141	0.0323 *
Age	0.0816390	0.0343840	2.374	0.0176 *
I(Age^2)	-0.0003965	0.0001897	-2.090	0.0366 *
Number	0.4268603	0.2361167	1.808	0.0706 .
Start	-0.2038411	0.0706232	-2.886	0.0039 **

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 83.234 on 80 degrees of freedom

Residual deviance: 54.428 on 76 degrees of freedom

AIC: 64.428

Number of Fisher Scoring iterations: 5

There is evidence that  $\text{Age}^2$  is required in the model, since the  $p$ -value is small (0.0366). The coefficient of  $\text{Age}^2$  is negative, indicating that the chance of kyphosis first increases with age and then decreases. The variable **Start** seems important ( $p$ -value = 0.0039), but the contribution of the variable **Number** is less certain ( $p$ -value = 0.0706). The coefficient of **Start** is negative which indicates that the higher the value of **Start** the smaller the probability of kyphosis. We should always be cautious when interpreting the  $p$ -values produced by **summary**. The Normal approximation used is not very reliable and thus these  $p$ -values should be viewed as rough indications of the strength of evidence against the coefficient being 0. In Section 5.2.4 we will introduce a more reliable method of judging the significance of regressors.

### 5.2.3 Analysis of grouped data using logistic regression

Suppose that we have a data set containing  $n$  cases and  $k$  explanatory variables and that some of the cases have identical sets of values for the explanatory variables. We will say that such cases have the same “covariate vector” - for each case the covariate vector is simply a vector that contains the values of the explanatory variables for that case. If there is a good deal of duplication of covariate vectors among the cases then it is often more convenient to record and analyse the data by grouping cases that have identical covariate vectors together.

Suppose that there are  $m$  distinct covariate vectors in the data set and label these as  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$ . For each  $\mathbf{x}_i$ , let  $n_i$  represent the number of cases and suppose that  $s_i$  of these that are successes ( $Y = 1$ ) and that  $n_i - s_i$  are failures ( $Y = 0$ ). For each  $\mathbf{x}_i$ , we can regard the  $n_i$  cases having this covariate vector as a sequence of Bernoulli trials, of which  $s_i$  are successes. Under these conditions  $s_i$  has a  $\text{Bin}(n_i, \pi_i)$  distribution, where  $\pi_i = \Pr[Y = 1 | \mathbf{x}_i]$  i.e. the probability that an individual case having covariate vector  $\mathbf{x}_i$  will be a “success” and have  $Y = 1$ . As usual, the logistic model assumes that  $\text{logit}(\pi) = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$ .

For each of  $m$  possible covariate vectors we have two observations, the number of cases  $n_i$  that have the covariate vector  $\mathbf{x}_i$ , and the number  $s_i$  out of  $n_i$  that are successes. We fit the model using the *proportion* of successes as the response, and the *number* of trials as a “weight” that

is specified as in weighted least squares. We illustrate the procedure using some data from an industrial experiment.

**Example 4.** The data in Table 5.2 come from Cox, “The Analysis of Binary Data”, p. 11, and consist of observations on the “readiness for rolling” of metal ingots prepared with different soaking times and different heating times. For each combination of heating and soaking times (except one) the number “not ready for rolling” ( $s_i$ ) and the total number of ingots examined ( $n_i$ ) are given. The first number in each pair is  $s_i$ , the second  $n_i$ . Thus 0,10 signifies 0 not ready out of 10.

Table 5.3: Data for Example 6

Soaking time	Heating time			
	7	14	27	51
1.0	0,10	0,31	1,56	3,13
1.7	0,17	0,43	4,44	0,1
2.2	0,7	2,33	0,21	0,1
2.8	0,12	0,31	1,22	0,0
4.0	0,9	0,19	1,16	0,1

The data are first entered into a data frame `ingots.df`, with variables `heat`, `soak`, `notready` and `total`:

```
> ingots
  heat soak notready total
1    7  1.0        0    10
2   14  1.0        0    31
3   27  1.0        1    56
4   51  1.0        3    13
5    7  1.7        0    17
6   14  1.7        0    43
7   27  1.7        4    44
8   51  1.7        0     1
9    7  2.2        0     7
10  14  2.2        2    33
11  27  2.2        0    21
12  51  2.2        0     1
13   7  2.8        0    12
14  14  2.8        0    31
15  27  2.8        1    22
16  51  2.8        0     0
17   7  4.0        0     9
18  14  4.0        0    19
19  27  4.0        1    16
20  51  4.0        0     1
```

Note that no observations were made for soaking time 2.8 and heating time 51. We would get the same output from *R* if this line were deleted. We designate  $\pi = \text{Pr}(\text{notready})$  and fit the model

$$\text{logit}(\text{notready}) = \beta_0 + \beta_1 \text{heat} + \beta_2 \text{soak}$$

by typing

```
> ingots.glm<-glm(notready/total~heat+soak,
                  family=binomial,data=ingots.df, weight=total)
```

An alternative is to type

```
> ingots.glm<-glm(cbind(notready, total-notready)~heat+soak,
                  family=binomial,data=ingots.df)
```

To get a summary, type

```

> summary(ingots.glm)

Call:
glm(formula = notready/total ~ heat + soak, family = binomial,
    data = ingots.df, weights = total)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.28311 -0.78183 -0.50514 -0.09701  1.71923

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.55917     1.11969  -4.965 6.87e-07 ***
heat         0.08203     0.02373   3.456 0.000548 ***
soak         0.05677     0.33121   0.171 0.863906
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 25.395  on 18  degrees of freedom
Residual deviance: 13.753  on 16  degrees of freedom
(1 observation deleted due to missingness)
AIC: 34.08

Number of Fisher Scoring iterations: 5

```

There is no evidence that soaking time affects the probability of being not ready, since the  $p$ -value for the hypothesis  $\beta_2 = 0$  is 0.8639. However, there is very strong evidence that heating time affects this probability since the  $p$ -value for  $\beta_1 = 0$  is 0.0005. As the coefficient for **heat** is positive, increasing the heating time increases the probability of being not ready.

### 5.2.4 Deviances

At the bottom of the output produced when the `summary` command is applied to `glm` objects are two lines that report the “Null Deviance” and the Residual Deviance”. A deviance is a measure of how well an estimated model fits the data – the Null Deviance indicates how well a model that just contains an intercept term fits the data and the Residual Deviance indicates how well the specified model fits the data. Note that deviances are always  $\geq 0$  and that the larger the deviance the worse the fit (a deviance of 0 indicates a perfect fit). For GLM’s the deviance fills the role that the residual sums of squares plays for ordinary regression models. The way that the deviance of a model is calculated will be illustrated using the following example.

**Example 5.** The larvae of the tobacco budworm, *Heliothis virescens*, are responsible for considerable damage to cotton and tobacco crops in the United States, and Central and Southern America. As a result of intensive cropping practices and the misuse of pesticides, particularly synthetic pyrethroids, the insect has become an important crop pest. Many studies on the resistance of the larvae to pyrethroids have been conducted, but the object of the experiment by Holloway (1989) was to examine levels of resistance in the adult moth to the pyrethroid trans-cypermethrin.

In the experiment batches of pyrethroid resistant moths of each sex were exposed to a range of doses of cypermethrin two days after emergence from pupation. The number of moths which were either knocked down (movement of the moth uncoordinated) or dead (the moth is unable to move and does not respond when poked with a blunt instrument) were recorded 72 hours after treatment. Reference: Holloway, J.W. (1989), A comparison of the toxicity of the pyrethroid trans-cypermethrin, with and without the synergist piperonyl butoxide, to adult moths from two strains of *Heliothis virescens*. University of Reading, Ph.D. thesis, Department of Pure and Applied Zoology.

The problem is to assess the effect of increasing dose of cypermethrin on toxicity. The data are shown in Table 5.4.

Table 5.4: Data from the tobacco budworm toxicity experiment.

Sex of moth	Dose (mg) of cypermethrin	Number affected out of 20
Male	1.0	1
	2.0	4
	4.0	9
	8.0	13
	16.0	18
	32.0	20
Female	1.0	0
	2.0	2
	4.0	6
	8.0	10
	16.0	12
	32.0	16

Clearly we are dealing with grouped data. There are  $m = 12$  distinct covariate vectors: (Male, 1.0), (Male, 2.0), (Male, 4.0), (Male, 8.0), (Male, 16.0), (Male, 32.0), (Female, 1.0), (Female, 2.0), (Female, 4.0), (Female, 8.0), (Female, 16.0) and (Female, 32.0). We will label these covariate vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{12}$ . There are 20 cases (moths) observed for each covariate pattern, ( $n_i = 20$  for  $i = 1, 2, \dots, 12$ ) and the total number of cases is  $n = 20 \times 12 = 240$ .

Now assume that the responses are independent, and that the probability that a moth will be “knocked down” ( $Y = 1$ ) depends only on the covariates. Then  $s_i$  is an observation from a binomial distribution,  $\text{Bin}(n_i, \pi_i)$ , where  $\pi_i$  is the probability of  $Y = 1$  for a case having covariate vector  $\mathbf{x}_i$ . Thus the probability of observing  $Y = 1$   $s_i$  times out of  $n_i$  cases is

$$\binom{n_i}{s_i} \pi_i^{s_i} (1 - \pi_i)^{n_i - s_i}.$$

The likelihood function is produced by multiplying the probabilities for the  $m$  covariate patterns together:

$$L = \prod_{i=1}^m \binom{n_i}{s_i} \pi_i^{s_i} (1 - \pi_i)^{n_i - s_i}. \quad (5.3)$$

Before we can define the deviance for a model, we must first introduce the concept of a *maximal model*. The maximal model is defined as the model that gives the best possible fit to the data. In other words, it is the model that gives the highest possible value of the likelihood function. For binary response data, specifying the maximal model is equivalent to specifying the set of  $\pi_i$ 's that will maximize equation (5.3) when no restrictions are placed on the  $\pi_i$ 's. It is not difficult to show (using calculus) that this is accomplished by setting  $\hat{\pi}_i = s_i/n_i$ . Plugging these values into equation 5.3 gives the highest possible value for the likelihood function which we will denote as  $L_{\max}$ . This value serves as a benchmark to which we will compare other models.

The maximal model represents the most complicated model we can fit for the given set of covariate vectors. Usually, it is desirable to find a simpler model that can describe the relationship between the  $\mathbf{x}_i$ 's and the  $\pi_i$ 's. The simplest model that we might consider is the *null model* which has  $\hat{\pi}_i = \text{constant}$  for all  $i$ . Note that this model severely restricts our choices for the  $\hat{\pi}_i$ 's (they must all be the same). Under this restriction, the maximum value of the likelihood function is obtained by setting  $\hat{\pi}_i = s/n$  for all  $i$  where  $s = \sum s_i$  and  $n = \sum n_i$ . We will call this value  $L_{\text{null}}$ . The deviance for the null model is calculated as

$$\text{null model deviance} = 2 \log L_{\max} - 2 \log L_{\text{null}}.$$

The model that we are really interested in is the logistic model. For this model the  $\hat{\pi}_i$ 's must satisfy:

$$\hat{\pi}_i = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 \text{sex}_i + \hat{\beta}_2 \text{dose}_i)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 \text{sex}_i + \hat{\beta}_2 \text{dose}_i)}.$$

The fitted coefficients (the  $\hat{\beta}$ 's) for this model are selected so that the resulting  $\hat{\pi}_i$ 's make the value of  $L$  as large as possible (under this model). These are exactly the maximum likelihood estimates. Let  $L_{\text{mod}}$  represent the maximum value of  $L$ . Then the deviance for the logistic model is

$$\text{logistic model deviance} = 2 \log L_{\text{max}} - 2 \log L_{\text{mod}}.$$

Table 5.5: Data from the tobacco budworm toxicity experiment.

$i$	sex	dose	$s_i$	$n_i$	$\hat{\pi}_i$ 's		
					maximal	logistic	null
1	0	1	1	20	0.05	0.27	0.46
2	0	2	4	20	0.20	0.30	0.46
3	0	4	9	20	0.45	0.37	0.46
4	0	8	13	20	0.65	0.53	0.46
5	0	16	18	20	0.90	0.80	0.46
6	0	32	20	20	1.00	0.98	0.46
7	1	1	0	20	0.00	0.12	0.46
8	1	2	2	20	0.10	0.14	0.46
9	1	4	6	20	0.30	0.18	0.46
10	1	8	10	20	0.50	0.30	0.46
11	1	16	12	20	0.60	0.60	0.46
12	1	32	16	20	0.80	0.95	0.46

The fitted probabilities for each covariate pattern in the budworm data set are given in Table 5.5 for the maximal, the logistic, and the null models. Plugging these values into (5.3) gives

$$\begin{aligned} 2 \log L_{\text{max}} &= -30.110, \\ 2 \log L_{\text{mod}} &= -58.078, \\ 2 \log L_{\text{null}} &= -154.986. \end{aligned}$$

Now the deviances for the logistic model and the null model can be calculated:

$$\begin{aligned} \text{logistic model deviance} &= -30.1104 - (-58.0784) = 27.968, \\ \text{null model deviance} &= -30.1104 - (-154.986) = 124.876. \end{aligned}$$

Of course, in practice, we never go to the bother of doing all these calculations ourselves since they are easily obtained using the `summary` function in R.

```
> bugs.glm<-glm(s/n~sex+dose,family=binomial,weight=n,data=budworm.df)
> summary(bugs.glm)

Call:
glm(formula = s/n ~ sex + dose, family = binomial, data = budworm.df,
     weights = n)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5567  -1.3326   0.3384   1.1254   1.8838

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.1661      0.2615  -4.459 8.24e-06 ***
sex          -0.9686      0.3295  -2.939 0.00329 **
dose           0.1600      0.0234   6.835 8.19e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 124.876 on 11 degrees of freedom
Residual deviance: 27.968 on 9 degrees of freedom
AIC: 64.078
```

Clearly the deviance for the logistic model (Residual deviance) is much smaller than that for the null model (Null deviance). This indicates that the logistic model gives a much better fit to the data than does the null model. But how do we tell when the deviance for the logistic model is small enough to indicate that it provides a reasonable fit to the data? Suppose the logistic model is the true model. Then, provided  $m$  is not too big and the  $n_i$ 's are large, the asymptotic distribution of the deviance is  $\chi^2_{m-k-1}$  – although this approximation is often not very accurate. We can perform a test of the hypothesis “the logistic model is reasonable” by comparing the model deviance to a  $\chi^2_{m-k-1}$  distribution. Since large values of the deviance are evidence *against* the logistic model being correct, a suitable  $p$ -value is calculated by finding the area under the  $\chi^2_{m-k-1}$  curve to the *right* of the value of the deviance. Given the questionable nature of the approximation, we should treat the  $p$ -value as a rough indication of how well the the model fits the data.

For our example the residual deviance is 27.968 and  $m - k - 1 = 12 - 2 - 1 = 9$  (note this is the degrees of freedom for the residual deviance given on the output). In R the  $p$ -value can be found using

```
> 1-pchisq(27.968,9)
[1] 0.0009656815
```

This test gives very strong evidence against the hypothesis that the fitted logistic model is adequate and thus we should explore alternative models. It is known from previous experience that a logistic model using log dose rather than dose often fits mortality data from pesticide experiments well. So we will try fitting the model

$$\text{logit } \pi = \beta_0 + \beta_1 \text{sex} + \beta_2 \log(\text{dose}).$$

The output for this model is:

```
> logbugs.glm<-glm(s/n ~ sex + log(dose), family = binomial,
                    weight=n, data=budworm.df)
> summary(logbugs.glm)
```

Call:

```
glm(formula = s/n ~ sex + log(dose), family = binomial, data = budworm.df,
     weights = n)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.10540	-0.65343	-0.02225	0.48471	1.42945

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-2.3724	0.3854	-6.156	7.46e-10 ***
sex	-1.1007	0.3557	-3.094	0.00197 **
log(dose)	1.5353	0.1890	8.123	4.54e-16 ***

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 124.876 on 11 degrees of freedom
Residual deviance: 6.757 on 9 degrees of freedom
AIC: 42.867
```

Number of Fisher Scoring iterations: 3

The residual deviance is now much smaller (6.757) indicating a better fit. If we test the hypothesis that this model is adequate, we get a large  $p$ -value :

```
> 1-pchisq(6.757,9)
[1] 0.6624024
```

Thus there is no evidence against the hypothesis and we conclude that the model using  $\log(\text{dose})$  is adequate. Note that there may be other models that are adequate and that may provide an even better fit to the data.

We can assess the fit graphically by plotting the logits of the observed proportions against the log dose, with the fitted lines for males and females added. Since  $\text{sex} = 0$  for males and  $\text{sex} = 1$  for females, the fitted lines are:

$$\begin{aligned}\text{males: } \text{logit}(y) &= -2.37 + 1.54 \log(\text{dose}) \\ \text{females: } \text{logit}(y) &= -3.47 + 1.54 \log(\text{dose})\end{aligned}$$

To draw the plot, using  $M$ 's for males and  $F$ 's for females, and draw in the fitted lines in  $R$ :

```
> fitted.logits = log((budworm.df$s+0.5)/(budworm.df$n-budworm.df$s+0.5))
> my.colours = rep(c("red","blue"), c(6,6))
> plot(log(budworm.df$dose),fitted.logits,type="n")
> text(log(budworm.df$dose), fitted.logits, ifelse(budworm.df$sex==0,"M","F"),
       col=my.colours)
> abline(-2.372412,1.535336,lty=1)
> abline(-2.372412-1.100743,1.535336,lty=2)
> legend(0,3,c("M","F"),lty=c(1,2), col=c("red","blue"))
```

Note that when calculating the logits of the observed proportions, we have added a “fudge factor” of 0.5 to avoid trying to take the log of zero. The result, shown in Figure 5.4, indicates that the model fits quite well.

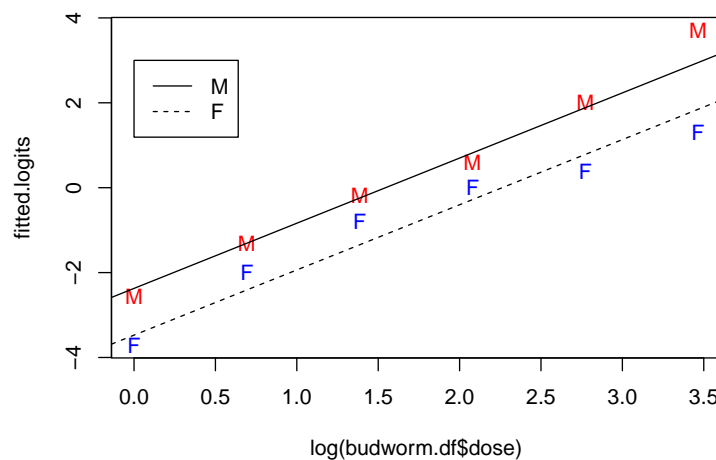


Figure 5.4: Fitted lines for the budworm data.

**Example 6.** For the ingot example (pp. 13-15), we can test the adequacy of the logistic model by following the procedure above. The residual deviance for the fitted model was 13.753 on 16 degrees of freedom. Thus our test statistic is  $\chi^2_o = 13.753$  which we compare to a  $\chi^2_{16}$  distribution:

```
> 1-pchisq(13.753,16)
```



[1] 0.617109

The  $p$ -value 0.6171 is quite large, so there is no evidence that the fitted logistic model is inadequate.

### The “sparse” case and the Hosmer-Lemeshow statistic

What happens when  $n_i = 1$  for most or all of the distinct covariate vectors? Then the conditions for the asymptotics do not hold, and so the assumption that the deviance has approximately a  $\chi^2$  distribution is not valid. In fact, the deviance can be written as a function of the fitted values alone, so can't be interpreted as a goodness-of-fit measure. In these circumstances, we can test the goodness-of-fit by means of the Hosmer-Lemeshow test.

Even if the data are ungrouped, we can form an approximate grouping by grouping together observations with similar fitted probabilities. An obvious way to do this is to divide the data into say ten groups, with all observations having fitted probabilities between 0 and 0.1 in group 1, all those between 0.1 and 0.2 in group 2 and so on. We then form the  $2 \times 10$  table, recording the numbers in each group that have  $y = 1$  and  $y = 0$ . The expected number in each cell of this table group can be estimated by averaging the fitted probabilities in a group and multiplying by the number in the group. We can then compare the observed and expected numbers using a  $\chi^2$  test. The bigger the value of  $\chi^2$ , the worse the fit, so small  $p$ -values indicate a poor fit to the logistic model.

The function `HLtest` in the R330 package performs the test. It calculates the  $\chi^2$  test statistic and an approximate  $p$ -value. We illustrate its use using the kyphosis data.

**Example 7.** To test the goodness of fit to the kyphosis data of the model including a quadratic term in age, we can type

```
> kypho.glm =glm(Kyphosis~Age+I(Age^2) + Number + Start, family=binomial,
  data=kyphosis.df)
> HLstat( kypho.glm)
Value of HL statistic = 6.498
P-value = 0.592
```

or, even more compactly

```
> HLstat(Kyphosis~Age+I(Age^2) + Number + Start, family=binomial, data=kyphosis.df)
Value of HL statistic = 6.498
P-value = 0.592
```

The large  $p$ -value indicates that there is no evidence of a poor fit.

### Comparing models

Suppose that we have a logistic model, and we want to see if we can drop a subset of variables from the model. In other words, we want to see if a submodel of the original logistic model is adequate. Let  $L_{\text{sub}}$  and  $L_{\text{full}}$  represent the maximum values of the likelihood function under the submodel and under the full model respectively. Standard likelihood theory indicates that if the submodel is adequate, the difference  $2 \log L_{\text{full}} - 2 \log L_{\text{sub}}$  will have a distribution that is approximately  $\chi_d^2$  where  $d$  is the number of variables dropped. This difference can be expressed as a difference of deviances:

$$\begin{aligned} 2 \log L_{\text{full}} - 2 \log L_{\text{sub}} &= (2 \log L_{\text{max}} - 2 \log L_{\text{sub}}) - (2 \log L_{\text{max}} - 2 \log L_{\text{full}}) \\ &= \text{deviance of submodel} - \text{deviance of full model.} \end{aligned}$$

This difference represents the increase in the deviance when we drop the  $d$  terms from the model. While this difference will always be positive, if the increase is small then dropping the extra terms will not increase the deviance by very much and so the variables can be dropped in the interests of getting a simpler model. The difference in the deviance has approximately a  $\chi_d^2$  distribution if the dropped variables are not needed in the model. Thus we can calculate a  $p$ -value to test the

hypothesis that the dropped variables are not needed by comparing this difference in deviances to a  $\chi_d^2$  distribution. A small  $p$ -value provides evidence *against* the submodel - i.e. a small  $p$ -value indicates that we should not drop all  $d$  of the variables from the model.

Unlike the approximation to the deviance itself, the  $\chi^2$  approximation to this *difference* between deviances is usually quite accurate, even when the data are not grouped. In particular it will be good provided that  $m$  (the number of distinct covariate vectors) is large. Note that we don't need the  $n_i$ 's to be large.

**Example 8.** For the kyphosis example, suppose we want to test whether we should drop both of the variables `start` and `number`. First we fit the submodel which only uses `age` and `age`<sup>2</sup> as regressors:

```
> sub.glm<-glm(Kyphosis~ Age+I(Age^2),family=binomial,data=kyphosis.df)
```

We can use the `deviance` command in R to extract the residual deviance from our `glm` object:

```
> deviance(sub.glm)
[1] 72.73858
```

Thus the submodel deviance is 72.73858, so the difference between the submodel and full model deviances is  $72.739 - 54.428 = 18.311$ . To get the  $p$ -value we use a  $\chi^2$  distribution with  $d = 2$  degrees of freedom:

```
> 1-pchisq(18.311,2)
[1] 0.0001056372
```

Thus the test is highly significant which indicates that variables `Start` and `Number` should **not** be dropped from the model.

The `anova` command in R can be used to perform the above test for us as follows:

```
> anova(sub.glm,kyphosis.glm,test="Chisq")
Analysis of Deviance Table

Model 1: Kyphosis ~ Age + I(Age^2)
Model 2: Kyphosis ~ Age + I(Age^2) + Number + Start
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1      78      72.739
2      76      54.428  2    18.311 0.0001056
```

Note that we simply enter the objects containing the submodel and the full model and use the `test="Chisq"` option to indicate we want to use a  $\chi^2$  reference distribution.

### Dropping regressors sequentially

The above procedure can be used to investigate dropping variables one at a time if the submodel is created by dropping one variable from the full model. This approach provides a more reliable method of assessing the importance of regressors than the  $p$ -values provided by the `summary` command. The `anova` function provides a convenient way to examine the changes in the deviance as terms are added to the model one at a time. For the kyphosis example, the following table can be produced using `anova`:

```
> anova(kyphosis.glm,test="Chisq")
Analysis of Deviance Table
Model: binomial, link: logit
Response: Kyphosis
Terms added sequentially (first to last)
  Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                                80    83.234
Age      1     1.302      79    81.932    0.254
I(Age^2) 1     9.194      78    72.739    0.002
Number   1     8.876      77    63.863    0.003
```

Start	1	9.435	76	54.428	0.002
-------	---	-------	----	--------	-------

Each line in this table represents an application of the submodel testing procedure and considers adding the variable listed for that line to the model that contains all variables listed above that line in the table. Thus this table summarises the following series of tests:

line	submodel	full model
Age	null	Age
I(Age^2)	Age	Age+I(Age^2)
Number	Age+I(Age^2)	Age+I(Age^2)+Number
Start	Age+I(Age^2)+Number	Age+I(Age^2)+Number+Start

In interpreting this table we should start at the bottom and work our way up. The line for **Start** indicates there is strong evidence that **Start** should be retained in the model. Given this result the line for **Number** isn't of much relevance since it considers removing **Number** from the model that does not contain **Start** (this line would be relevant had we decided to remove **Start**).

The order that terms are added in the output from `anova` is determined by the order used when specifying the model in `glm`. Thus we can investigate adding the terms in a different order as follows:

```
> kyphosis.glm2<-glm(Kyphosis~ Age+I(Age^2)+Start+Number,
                     family=binomial,data=kyphosis.df)
> anova(kyphosis.glm2,test="Chisq")
Analysis of Deviance Table
Model: binomial, link: logit
Response: Kyphosis
Terms added sequentially (first to last)
```

	Df	Deviance	Resid. Df	Resid. Dev	P(> Chi )
NULL			80	83.234	
Age	1	1.302	79	81.932	0.254
I(Age^2)	1	9.194	78	72.739	0.002
Start	1	14.324	77	58.414	0.0001539
Number	1	3.986	76	54.428	0.046

The last line for this table tests whether **Number** should be dropped from the model that contains all the variables and gives a  $p$ -value of 0.046. Compare this to the line for **Number** from the output from **Summary** on page 13 – this line is testing the same hypothesis but gives a  $p$ -value of 0.0706. The reason for this discrepancy is that the two tests are based on different approximations: the test used in **summary** assumes that the estimated coefficient has a Normal distribution whereas the test used in **anova** assumes that the change in deviance has a  $\chi^2$  distribution. Usually the two tests will agree fairly well but sometimes there is a discrepancy. In such cases, the  $\chi^2$  test is more reliable.

### Testing the significance of the regression

A special case of testing a submodel is the problem of testing the overall significance of the regression – are any of the explanatory variables useful in explaining the response? For this test, the null model is used as the submodel and the fitted model is used as the full model. Thus the test statistic is the difference in deviance between the null model and the fitted model. The reference distribution is  $\chi_k^2$  where  $k$  is the number of explanatory variables in the fitted model.

**Example 9.** For the kyphosis example, consider the model that contains **Age**, **Age<sup>2</sup>**, **Start**, and **Number** as explanatory variables and suppose we want to test the hypothesis that the coefficients for all four regressors are zero. In this case, the deviance of the fitted model is 54.428, and that of the null model is 83.234. The difference is  $83.234 - 54.428 = 28.806$ . This value is compared to a  $\chi^2$  distribution with 4 degrees of freedom – note that this value is the difference between the degrees of freedom for the null deviance and the residual deviance.

```
> 1-pchisq(28.806,4)
[1] 8.559766e-06
```

The  $p$ -value is very small indicating strong evidence against the hypothesis that none of the regressors are needed in the model.

### 5.2.5 Diagnostics for logistic regression models

For logistic regression models we require diagnostic procedures to check the validity of our fitted model. These procedures are often analogous to the procedures we used for ordinary regression models. We start by introducing two types of residuals which are required for a number of our diagnostics.

1. Pearson residuals. These are defined by

$$r_i = \frac{s_i - n_i \hat{\pi}_i}{\sqrt{n_i \hat{\pi}_i (1 - \hat{\pi}_i)}}.$$

The Pearson  $r_i$  represents the difference between the observed number of successes and the predicted number of successes for the  $i$ th covariate pattern divided by the standard error of the predicted number of successes. Thus Pearson residuals are similar to standardised residuals for the ordinary linear regression model.

2. Deviance residuals. Let  $\hat{\pi}_i$  and  $\tilde{\pi}_i$  be the estimated probabilities of success for the  $i$ th covariate pattern using the logistic model and the maximal model respectively. Then it can be shown that the deviance of the logistic model can be written as  $\sum_{i=1}^m d_i^2$  where

$$d_i = \pm \left\{ -2 \left( y_i \log \left( \frac{\hat{\pi}_i}{\tilde{\pi}_i} \right) + (n_i - s_i) \log \left( \frac{1 - \hat{\pi}_i}{1 - \tilde{\pi}_i} \right) \right) \right\}^{\frac{1}{2}}.$$

The sign of  $d_i$  is selected to be the same as the sign of  $r_i$ . The  $d_i$ 's are called *deviance residuals*.

Note that these residuals are most useful for grouped data. For ungrouped data, the a residual is large if the observation is a success, but the model gives it a low probability of success. (or conversely, if the observation is a failure, but the model gives it a high probability of success.)

### Outlier detection

Large values of  $|r_i|$  or  $|d_i|$  denote covariate patterns that are poorly fitted by the model. More precisely a large  $|r_i|$  indicates a covariate patterns where there is a large discrepancy between the observed and the predicted number of successes and a large value of  $|d_i|$  indicates an observation that makes an usually large contribution to the residual deviance of the model.

Both types of residual are calculated as part of the “glm object”, and are extracted with the `residuals` function. For the ingots data, we can extract the residuals as follows:

```
> d.resids<-residuals(ingots.glm,type="deviance")
> p.resids<-residuals(ingots.glm,type="pearson")
```

Simple index plots of the residuals are useful for outlier detection. We can use the following commands in R to get an index plots of each type of residual.

```
> plot(p.resids,type="n",main="Pearson residuals")
> text(p.resids)
> abline(h=0,lty=3)
> plot(d.resids,type="n",main="Deviance residuals")
> text(d.resids)
> abline(h=0,lty=3)
```

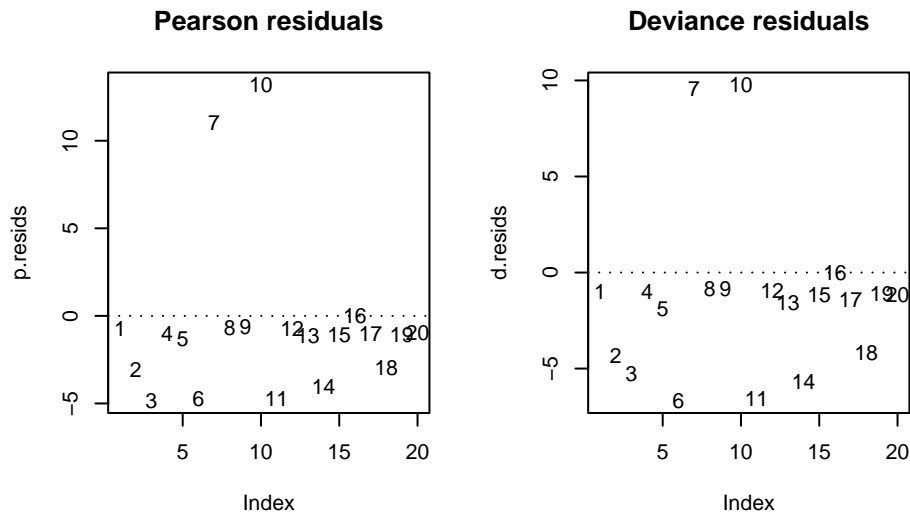


Figure 5.5: Plots of residuals for the ingot data.

These plots indicate that the 7th and 10th covariate patterns have large Pearson residuals and large deviance residuals. We can identify the cases by printing out the corresponding rows of the data frame:

```
ingots.df[c(7,10),]
  heat soak ready total
7    27  1.7     4    44
10   14  2.2     2    33
```

In both cases  $r_i > 0$  which indicates the observed number of successes are larger than what could reasonably be expected under the fitted model.

### Non-linear regression surfaces

Recall that the logistic regression model assumes that  $\text{logitPr}(Y = 1)$  is a linear function of the regressors:

$$\text{logitPr}(Y = 1) = \beta_0 + \beta_1 X_1 + \dots + \beta_k X_k.$$

We can check this assumption by (i) plotting the Pearson residuals versus the linear predictors ( $\hat{\beta}_0 + \hat{\beta}_1 X_1 + \dots + \hat{\beta}_p X_p$ ) and (ii) plotting the Pearson residuals versus each of the numerical explanatory variable. In each case, we want to observe a patternless horizontal band of points. Clear evidence of curvature indicates that the linearity assumption is not valid.

To illustrate these plots consider the first model we tried for the budworm data (page 18):  $\text{logit}(Y) = \beta_0 + \beta_1 \text{sex} + \beta_2 \text{dose}$ . Recall that we found evidence that this model was not adequate. Lets see if we can find evidence of non-linearity using the above plots. The following commands can be used to produce these plots in *R*:

```
> l.pred<-predict(bugs.glm)
> p.res<-residuals(bugs.glm,type="pearson")
> plot(l.pred,p.res,xlab="linear predictors", ylab="Pearson residuals",
+      main="Pearson residuals vs linear predictors")
> lines(lowess(l.pred,p.res),lty=3)
> plot(budworm.df$dose,p.res,xlab="dose",ylab="Pearson residuals",
+      main="Pearson residuals vs dose")
> lines(lowess(budworm.df$dose,p.res),lty=3)
```

The plots produced in this manner are given in Figure 5.6. These plots both give clear evidence that the linearity assumption is not valid for the fitted model.

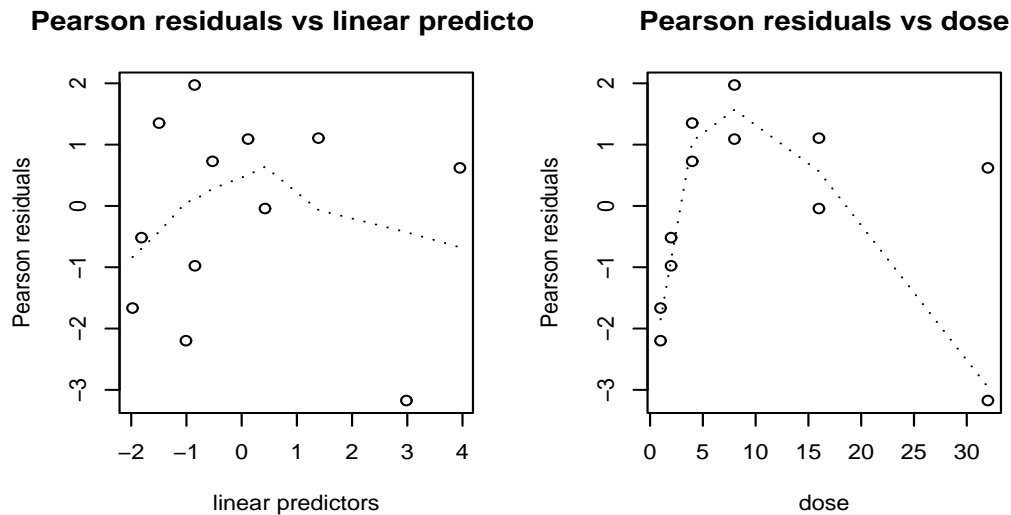


Figure 5.6: Plots of Pearson residuals for the budworm data.

### Detecting high leverage and influential points

The diagnostics used to detect high leverage and influential points for logistic regression models are, for the most part, similar to those used for ordinary regression models. To illustrate these diagnostics, we will consider the fitted model we used for the budworm data from page 19 – note that this is the model that used  $\log(\text{dose})$  as a predictor. To evaluate the leverage of points we use the diagonal elements of the “hat” matrix as was done for ordinary regression models. These can be obtained as follows:

```
> hats = hatvalues(logbugs.glm)
> round(hats,4)
      1      2      3      4      5      6      7      8      9     10     11
0.2319 0.2907 0.2921 0.2809 0.2459 0.1732 0.1281 0.1993 0.2565 0.2898 0.3201
     12
0.2915
```

To evaluate the overall influence of observations on the fitted model we will consider two diagnostics: (i) Cook’s distance and (ii) deviance changes. The definition of Cook’s distance for logistic regression models is analogous to the definition used for ordinary regression models and it is interpreted in the same manner – an unusually large value indicates an influential point. For our current example the Cook’s distances are calculated as follows:

```
> cooks = cooks.distance(logbugs.glm)
> round(cooks,4)
      1      2      3      4      5      6      7      8      9     10     11
0.0419 0.0038 0.0018 0.0335 0.0256 0.0885 0.0348 0.0084 0.1641 0.0758 0.1604
     12
0.1344
```

The second diagnostic we used to evaluate the impact of observations on the response involves “deviance changes”. This is a “leave-one-out” diagnostic that indicates how much the residual deviance would change if that observation was deleted. Unusually large values of deviance changes indicate an influential point. The deviance changes are computationally expensive to calculate exactly and so are usually approximated by:

$$\Delta D_i = d_i^2 + \frac{r_i^2 h_{ii}}{1 - h_{ii}}$$

where  $d_i$  is the  $i$ th deviance residual. In *R* they can be calculated as follows:

```
> d.res<-residuals(logbugs.glm,type="deviance")
> p.res<-residuals(logbugs.glm,type="pearson")
> hats = hatvalues(logbugs.glm)
> dcs<-d.res^2 + p.res^2 *(hats/(1-hats))
> round(dcs,4)
      1      2      3      4      5      6      7      8      9     10     11
0.4671 0.0278 0.0131 0.2524 0.2493 2.2629 1.3131 0.0964 1.3320 0.5537 0.9931
     12
0.9056
```

Values greater than 4 indicate problems.

Index plots of leverage, Cook's distance and deviance changes for the budworm model are given in Figure 5.7. In these plots we are simply looking for observations that stand out as being substantially larger than the others. For the current example, none of the observations have large enough values of any of these diagnostics to be of real concern.

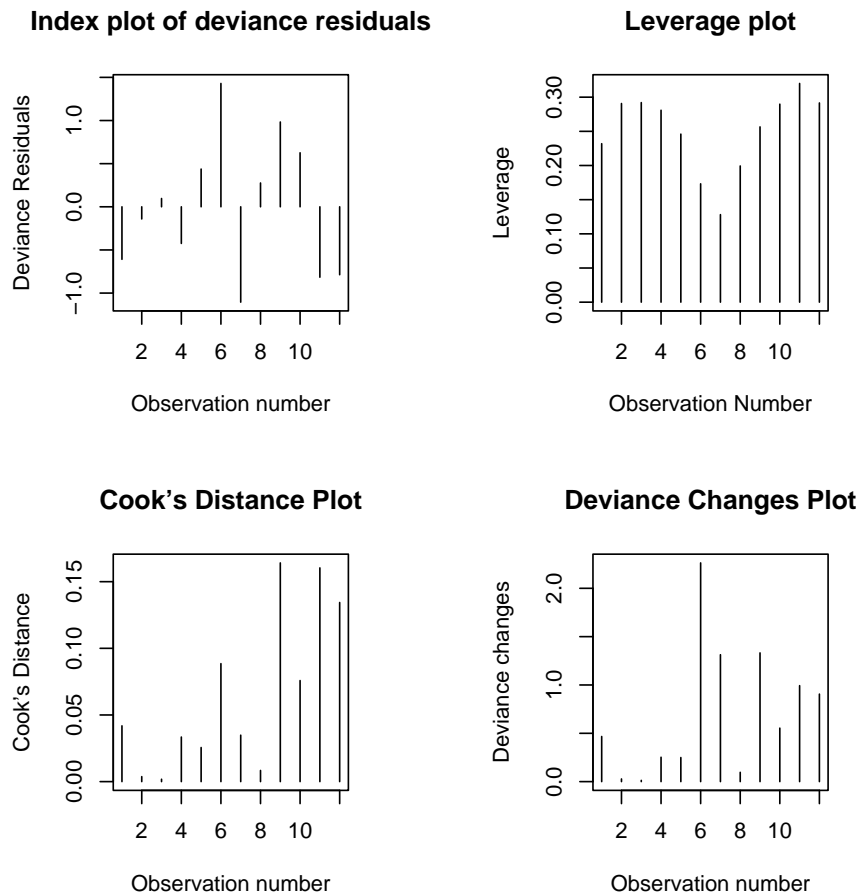


Figure 5.7: Leverage, Cook's distance and deviance changes for the budworm model.

Changes in the coefficients are measured by quantities similar to those in ordinary linear regression, and in fact the function `influence.measures` described in Chapter 3 for the calculation of influence statistics computes these when given a “glm object” as its argument.

**Example 10.** The data in Table 5.3 were obtained in a study of the effect of the rate and volume of air inspired on a transient vaso-constriction in the skin (Finney, *Biometrika*, 1947, p. 320). The nature of the measurement process was such that only the occurrence or non-occurrence of

vaso-constriction could be reliably measured. Three subjects were involved in the study: the first contributed 9 responses, the second contributed 8 responses, and the third contributed 22 responses. It was decided to use  $\log(\text{Volume})$  and  $\log(\text{Rate})$  as the regressors for the logistic model.

```
> vaso.glm<-glm(Response~log(Volume)+log(Rate),
                 data=vaso.df,family="binomial")
> summary(vaso.glm)
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.875      1.306   -2.200  0.02777 *
log(Volume)    5.179      1.843    2.810  0.00496 **
log(Rate)      4.561      1.818    2.509  0.01211 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Null deviance: 54.040  on 38  degrees of freedom
Residual deviance: 29.227  on 36  degrees of freedom
```

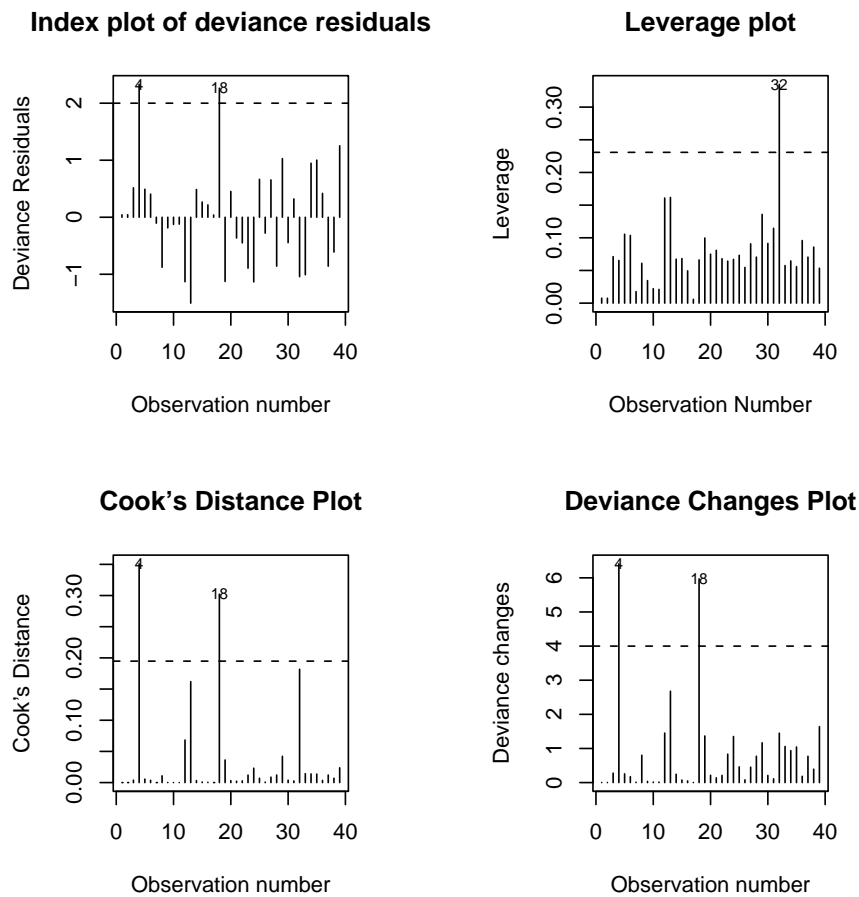


Figure 5.8: Influence plots for the vaso-constriction data.

Index plots of leverage, Cook's distance, and deviance changes are shown in Figure 5.8. These plots show that two observations, the 4th and 18th, have a considerable influence on the fit. The effect of deleting these points can be determined by refitting: (note the use of the `subset` argument)

```
> new.glm = glm(Response ~ log(Volume)+ log(Rate),
                 subset = -c(4,18), data=vaso.df)
```



```

> summary(new.glm)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -24.58      13.80  -1.781   0.0750 .
log(Volume)   39.54      22.89   1.728   0.0840 .
log(Rate)     31.93      17.49   1.826   0.0678 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
Null deviance: 51.266 on 36 degrees of freedom
Residual deviance: 7.361 on 34 degrees of freedom
AIC: 13.361

```

Note the large changes in the coefficients. Although the  $p$ -values generated by `summary` are borderline for both variables, the  $\chi^2$  tests indicate both variables are required:

```

> anova(new.glm,test="Chisq")
Analysis of Deviance Table

            Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                                36      51.266
log(Volume)  1      8.764          35      42.502    0.003
log(Rate)    1     35.141          34       7.361 3.067e-09

> new2.glm<-glm(Response~log(Rate)+log(Volume),
                data=vaso[-c(4,18),],family="binomial")
> anova(new2.glm,test="Chisq")
Analysis of Deviance Table

            Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                                36      51.266
log(Rate)    1      5.204          35      46.062    0.023
log(Volume)  1     38.701          34       7.361 4.939e-10

```

It is clear that these two points make a big difference to the regression. However, it is not clear that they are not genuine points (after all, events of low probability do sometimes occur). In logistic regression, we need to be very careful about deleting points, unless they are found to be incorrectly recorded. We will leave points 4 and 18 in the regression as a consequence.

### 5.2.6 Prediction in logistic regression

In the logistic regression context, we want to predict if an event (such as vasoconstriction) will occur. This is done by calculating the predicted probability, given the individual's covariates. If the probability is more than some threshold (usually 0.5), we predict that the event will occur. If the predicted probability is less than 0.5, we predict the event will not occur.

The prediction error is defined as

$$PE = \begin{cases} 0, & \text{if prediction is correct,} \\ 1, & \text{if prediction is incorrect.} \end{cases}$$

If we take the expected prediction error over future data, we get

$$\begin{aligned}
 E(PE) &= Pr[\text{Prediction incorrect}] \\
 &= Pr[Y = 0, \text{prediction} = 1] + Pr[Y = 1, \text{prediction} = 0] \\
 &= Pr[\text{prediction} = 1|Y = 0]Pr[Y = 0] + Pr[\text{prediction} = 0|Y = 1]Pr[Y = 1]
 \end{aligned}$$

using standard probability formulas. The conditional probabilities are related to the *sensitivity* and *specificity* of the prediction, which are defined as

$$\begin{aligned}
 \text{sensitivity} &= Pr[\text{prediction} = 1|Y = 1] \\
 &= \text{Probability of a true positive} \\
 &= 1 - Pr[\text{prediction} = 0|Y = 1]
 \end{aligned}$$

and

$$\begin{aligned}
 \text{specificity} &= Pr[\text{prediction} = 0|Y = 0] \\
 &= 1 - Pr[\text{prediction} = 1|Y = 0] \\
 &= 1 - \text{Probability of a false positive.}
 \end{aligned}$$

The sensitivity is often called the True Positive Rate (TPR) and one minus the sensitivity the False Positive Rate (FPR). Both sensitivity and specificity should be close to one for good predictions. However, this is often not possible. For example, for rare events the probability will be low and we will almost always predict that the event will not occur (i.e. predict  $Y = 0$ .) Consequently the sensitivity will be low, but the specificity will be very good.

To estimate the sensitivity and specificity, we have the same options as we did in the normal regression case. If we have plenty of data, we can use the training set/test set approach. We use the training set to estimate the regression coefficients (i.e. to construct the prediction rule) and evaluate the predicted probabilities using the test set. We estimate the sensitivity by the proportion of cases having  $Y = 1$  that have a prediction of 1, and the specificity by the proportion of cases having  $Y = 0$  that have a prediction of 0. If we don't have enough data, we can use cross-validation or the bootstrap as we did in Section 3.3. As before, using the training set to estimate the sensitivity and specificity will usually result in overly optimistic estimates.

### ROC curves

We mentioned above that using a predictor with a threshold of 0.5 when dealing with rare events will result in poor sensitivity but good specificity. It is clear that both sensitivity and specificity depend on the prediction rule, and in particular on the threshold used. We may want to adjust the threshold in such a way to make the sensitivity and specificity more equal. If we adjust the threshold down, we are making the prediction of a “1” more likely, so we are increasing the sensitivity. But at the same time, we are making the prediction of an “0” less likely, so we are decreasing the specificity. Thus, by adjusting the threshold, we can control the tradeoff between sensitivity and specificity.

This tradeoff can be made explicit by means of an ROC (Receiver-Operator Characteristic) curve. In this curve we plot the sensitivity versus one minus the specificity (or, equivalently, the TPR versus the FPR), for different threshold values. As the threshold varies, the points trace out a curve as shown in Figure 5.9. If we set the threshold at 0, then the sensitivity is 1 and the FPR is also 1. Conversely, if we set the threshold at 1, then the sensitivity is 0 and the FPR is also 0. For other values, the curve traces out a path between (0,0) and (1,1). The closer the curve goes to the top left corner of the plot, the better the prediction rule, as the ideal is a TPR of 1 and a FPR of 0. The area under the curve in this case approaches 1. The area under the curve gives a good measure of the quality of the prediction rule.

**Example 11.** We use the vaso-constriction data of Example 10. In this example, there is quite a rapid transition between high and low fitted probabilities, so that prediction should work well. To estimate the sensitivity and specificity, we can use both the bootstrap and cross-validation. First, we compute predictions from the training set and cross-classify them with the actual responses. The following code follows on from Example 9:

```

> prediction = predict(vaso.glm, type="response")>0.5
> table(vaso.df$Response, prediction)
  prediction
  FALSE TRUE
0      14    5
1       2   18

```

The proportion incorrectly classified is  $7/39 = 0.179$  so this is the “training set” estimate of the prediction error. The sensitivity is  $18/20 = 0.90$  and the specificity  $14/19 = 0.737$ . We expect these estimates to be too optimistic. The cross-validation and bootstrap estimates are

```

> cross.val(vaso.glm)

```

```

Mean Specificity = 0.7225549
Mean Sensitivity = 0.874521
Mean Correctly classified = 0.785
> err.boot(vaso.glm)
$err
[1] 0.1794872

$Err
[1] 0.1881026

```

We see that the cross-validation estimates of the sensitivity and specificity are a bit lower than the training set estimates, as expected. The cross-validated estimate of prediction error is  $1 - 0.785 = 0.215$ , a bit more than the training set estimate. The bootstrap estimate of the prediction error is 0.188 (the one labeled `$ Err` in the output—the one labelled `$ err` is the training set estimate).

To draw the ROC curve we type

```
ROC.curve(vaso.glm)
```

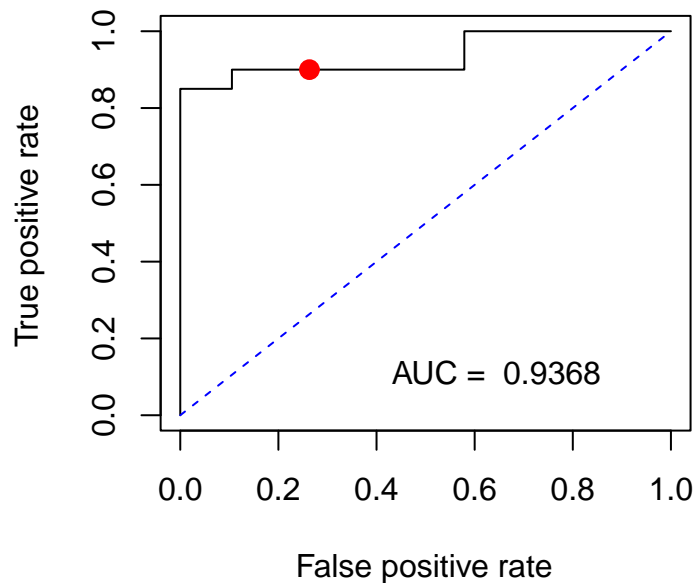


Figure 5.9: ROC curve for the vaso-constriction data. The dot indicates a predictor with threshold 0.5.

### 5.2.7 Binary anova

The logistic regression method can be easily extended to situations where all or some of the explanatory variables are factors. The terms “binary ANOVA” (all regressors are factors) and “binary ANCOVA” (some of the regressors are factors) are sometimes used to denote these situations. Note that we have already considered a case of binary ANCOVA (Example 5). We now consider two examples of binary ANOVA.

**Example 12.** A study was conducted on the reproduction of plum trees by taking cuttings from older trees. Half the cuttings were planted immediately while the other half were bedded in sand

until spring when they were planted. Two lengths of cuttings were used: long (12 cm) and short (6cm). A total of 240 cuttings were taken for each of the 4 combinations of planting time and cutting length and the number of cuttings that survived in each situation was recorded:

Length of cutting	Time of planting	Number surviving (out of 240)
short	at once	107
	in spring	31
long	at once	156
	in spring	84

This is a simple example of a two-way binary ANOVA. There are two explanatory variables each of which is a factor. Possible models for this data are:

1. The null model
2. Planting time used as a factor
3. Cutting length used as a factor
4. Both planting time and cutting length used as factors (no interaction)
5. Both planting time and cutting length used as factors and the interaction is included

We want to find the simplest model that can adequately explain the data. To do this we start with the simplest model (1) and investigate adding terms sequentially, in the order specified in the model.

```
> len<-factor(c("short","short","long","long"))
> time<-factor(c("at once","spring","at once","spring"))
> survive<-c(107,31,156,84)
> plum.glm<-glm(cbind(survive,240-survive)~time*len, family=binomial)
> anova(plum.glm,test="Chisq")
Analysis of Deviance Table
Model: binomial, link: logit
Response: cbind(survive, 240 - survive)
Terms added sequentially (first to last)
```

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			3	151.019	
time	1	97.579	2	53.440	< 2.2e-16 ***
len	1	51.147	1	2.294	8.572e-13 ***
time:len	1	2.294	0	0.000	0.1299

This table suggests (with a  $p$ -value of 0.130) that the interaction term is not needed in the model. Given that we drop the `time:len` interaction then we can use this table to consider dropping `len` as well. The small  $p$ -value for the `len` line indicates very strong evidence that `len` should not be dropped from the model. Note that since `len` is retained the line for `time` from this table is not relevant. To test whether `time` should be kept in the model, given that `len` is included, we need to consider the table that adds the variables in the other order:

```
> plum2.glm<-glm(cbind(survive,240-survive)~len+time,family=binomial)
> anova(plum2.glm,test="Chisq")
Analysis of Deviance Table
Model: binomial, link: logit
Response: cbind(survive,240-survive)
Terms added sequentially (first to last)
```

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			3	151.019	
len	1	45.837	2	105.182	1.285e-11 ***
time	1	102.889	1	2.294	< 2.2e-16 ***

Thus we conclude that both `len` and `time` are needed in the model but not the `time:len` interaction. We can get the fitted probabilities for the four combinations of `len` and `time` by using the `predict` command:

```
> predict(plum2.glm,type="response")
      1      2      3      4
0.4245994 0.1504006 0.6712339 0.3287661
```

Thus our model predicts that the following probabilities of survival:

length	time	$\hat{\pi}$
short	at once	0.42
	in spring	0.15
long	at once	0.67
	in spring	0.33

We conclude that survival probabilities are higher for: (i) cuttings planted at once compared to those planted in the spring and (ii) long cuttings compared to short cuttings.

**Example 13.** Table 5.6 contains the results of a series of experiments to compare the effects of x-rays and beta-rays on the mitotic rates in grasshopper neuroblasts. For each experiment, embryos from the same egg were divided into three groups, one serving as a control, and the other two being exposed to physically equivalent doses of x-rays and beta-rays. After irradiation, approximately equal numbers of cells in each of the 12 experiment  $\times$  treatment combinations were examined and the number of cells passing through mid-mitosis was noted. We thus have a binary ANOVA, with the cells being the experimental units (cases), the binary response being 1 if the cell has passed mid-mitosis and zero otherwise. There are two factors, the experiment with 4 levels (1 through 4) and the treatment with three levels (Control, X-ray, Beta-ray).

For our analysis we will read the data into *R*, define the factors, create a data frame, and fit models using `glm`. We type

```
> # Enter data
> yes<-c(12,3,4,14,5,6,9,5,2,17,5,7)
> no<-c(10,15,12,3,10,9,11,17,17,2,14,13)
> total<-yes+no

> # Create factors
> experiment<-factor(rep(1:4,c(3,3,3,3)))
> treat<-rep(c("Control","X-ray","Beta-ray"),4)
> # We want the levels in the order they occur
> treat<-factor(treat,levels=unique(treat))

> # Make a data frame
> grass.df<-data.frame(experiment,treat,yes,total)
> grass.df
  experiment    treat yes total
1           1 Control  12    22
2           1   X-ray   3    18
3           1 Beta-ray   4    16
4           2 Control  14    17
5           2   X-ray   5    15
6           2 Beta-ray   6    15
7           3 Control   9    20
8           3   X-ray   5    22
9           3 Beta-ray   2    19
10          4 Control  17    19
11          4   X-ray   5    19
12          4 Beta-ray   7    20
```

Table 5.6: Grasshopper data.

Experiment	Mid Mitosis	Control	X-ray	Beta-ray
1	Yes	12	3	4
	No	10	15	12
2	Yes	14	5	6
	No	3	10	9
3	Yes	9	5	2
	No	11	17	17
4	Yes	17	5	7
	No	2	14	13

```
# Fit the model using both factors plus their interaction
> grass.glm<-glm(cbind(yes,total-yes) ~ experiment*treat, family="binomial",
  data=grass.df)
> anova(grass.glm,test="Chisq")
Analysis of Deviance Table
Model: binomial, link: logit
Response: cbind(yes,total-yes)
Terms added sequentially (first to last)
```

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			11	54.838	
experiment	3	11.662	8	43.175	0.008634 **
treat	2	38.212	6	4.963	5.039e-09 ***
experiment:treat	6	4.963	0	0.000	0.548549

There is no evidence that the interaction is needed in the model. So we drop the interaction and consider the line for `treat`. This test gives extremely strong evidence that `treat` is needed in the model. For this example, the `experiment` factor should be considered as a blocking variable. That is we are not really interested in the effect of `experiment` but we want to take this effect into account when we compare levels of `treat`.

As with ordinary ANOVA, an interaction plot is a useful graphic display to complement the test for no interaction. The only difference is that we want to plot the logit values for each cell rather than the cell means. The *R* function `interaction.plot` can be used, but we need to supply an extra argument to plot the logits of the cell proportions rather than the proportions. (For binary data, proportions and means are the same.)

```
> attach(grass.df)
> logit<-function(x){log(x/(1-x))}
> interaction.plot(experiment,treatment,yes/total,fun=logit)
```

The plot is shown in Figure 5.10. As in ordinary interaction plots, parallel lines (approximately) indicate no interaction. Apart from a rather high value for the logit of the X-ray proportion in Experiment 3, the profiles are quite parallel, indicating the absence of interaction. Thus the effect of changing from one treatment to another is the same for all four experiments, although there are significant differences between experiments.

The output from the function `dummy.coef` can be useful for comparing levels for the different factors.

```
grass2.glm<-glm(cbind(yes,total-yes) ~ experiment + treat,
  family = "binomial", data=grass.df)
> dummy.coef(grass2.glm)
Full coefficients are

(Intercept):      0.366342
experiment:      1      2      3      4
0.0000000  1.0393635 -0.2951794  0.9551598
```

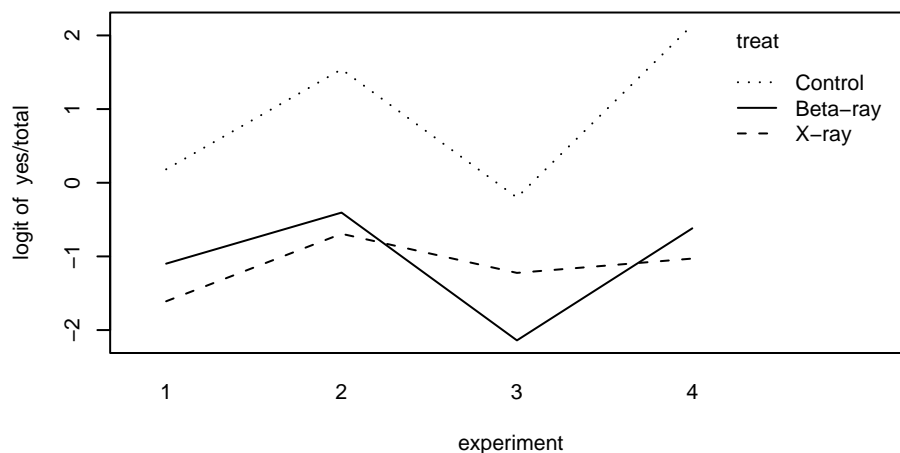


Figure 5.10: Interaction plot for the grasshopper data.

```
treat:          Control      X-ray  Beta-ray
          0.000000   -1.957590  -1.848509
```

For this example, we will concentrate on `treat` since `experiment` can be considered as a blocking factor. That is we are not really interested in the effect of `experiment` but we want to take this effect into account when we compare levels of `treat`. In comparing the levels of `treat`, it is most convenient to consider the impact on the odds of mitosis. The differences between the entries for Control, Beta-ray, and X-ray from `dummy.coef` represent the differences in the estimated logits (log odds). These differences can be converted to multiplicative factors for comparing the odds for different levels by applying the exponential function. For example, according to the difference in log odds of mitosis between the control group and the X-ray group is  $0 - (-1.96) = 1.96$ . Thus the odds of mitosis for the control group are estimated to be  $\exp(1.96) = 7.10$  times the odds of mitosis for the X-ray group. In a similar manner, our model estimates that the odds for the control group are  $\exp(1.85) = 6.36$  times the odds for the Beta-ray group, and that the odds for the Beta-ray group are  $\exp(.109) = 1.12$  times the odds for the X-ray group.

## 5.3 Poisson Regression and Contingency Tables

### 5.3.1 Poisson Regression

So far, we have considered two kinds of regression:

- Regressions where the responses are continuous. We assumed that the responses were normally distributed, with the mean depending on the covariates.
- Regressions where the responses are binary, or the number of “sucesses” out of a fixed number of “trials”. We assumed that the responses were binomially distributed, with the success probability depending on the covariates.

Now we examine a third type of regression, *Poisson regression*, where the responses are counts. We will assume that an individual response  $Y$  is a random count, (which must be a non-negative integer) whose mean depends on the covariates. The count follows a Poisson distribution, with

$$Pr[Y = y] = \frac{e^{-\mu} \mu^y}{y!}.$$

Table 5.7: Number of accidents in a 3-month period in West Virginia.

COUNT	INB	EXTRP	AHS	AGE
2	50	70	52	1.0
1	230	65	42	6.0
0	125	70	45	1.0
4	75	65	68	0.5
1	70	65	53	0.5
2	65	70	46	3.0
0	65	60	62	1.0
0	350	60	54	0.5
4	350	90	54	0.5
4	160	80	38	0.0
. . . 44 lines in all				

The mean count  $\mu$  is a positive number, and is related to the covariates  $x_1, \dots, x_k$  by

$$\mu = \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k).$$

The coefficients are interpreted in the same way as in the “odds” form of the logistic model: a unit increase in  $x_k$  is associated with an increase in the mean response by a factor of  $\exp(\beta_k)$ . This is because

$$\begin{aligned}
 \text{new mean} &= \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_k (x_k + 1)) \\
 &= \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k + \beta_k) \\
 &= \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k) \exp(\beta_k) \\
 &= \text{old mean} \times \exp(\beta_k).
 \end{aligned}$$

The Poisson regression model is fitted in almost the same way as the logistic model: the only change is that we set `family=poisson` rather than `family=binomial`. We illustrate with an example.

**Example 14.** This example features the number of accidents per mine in a 3 month period in 44 coal mines in West Virginia. The variables are

**COUNT** : the number of accidents (response)

**INB** : inner burden thickness

**EXTRP** : percentage of coal already extracted from the mine

**AHS** : the average height of the coal seam in the mine

**AGE** : the age of the mine

The data are shown in Table 5.7. We want to model the mean number of accidents as

$$\mu = \exp(\beta_0 + \beta_1 \text{INB} + \beta_2 \text{EXTRP} + \beta_3 \text{AHS} + \beta_4 \text{AGE}).$$

Assuming the data are in a data frame `mines.df` we fit the model in R using the code

```
> mines.glm<-glm(COUNT ~ INB + EXTRP + AHS + AGE, family=poisson, data=mines.df)
> summary(mines.glm)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-3.6097078	1.0284740	-3.510	0.000448 ***
INB	-0.0014441	0.0008415	-1.716	0.086145 .
EXTRP	0.0622011	0.0122872	5.062	4.14e-07 ***
AHS	-0.0017578	0.0050737	-0.346	0.729003
AGE	-0.0296244	0.0163143	-1.816	0.069394 .



Table 5.8: Deaths from childhood cancers 1951-1960 in Northumberland and Durham, classified by Cytology (Lymphoblastic / Myeloblastic), Residence (Rural/Urban), Age (0-5, 6-14).

Cytology	Residence	Age	Deaths	Population
L	R	0-5	38	103,857
M	R	0-5	5	103,857
L	R	6-14	13	155,786
M	R	6-14	8	155,786
L	U	0-5	51	135,943
M	U	0-5	13	135,943
L	U	6-14	37	203,914
M	U	6-14	20	203,914

```

Null deviance: 74.984 on 43 degrees of freedom
Residual deviance: 37.717 on 39 degrees of freedom
AIC: 143.99
> 1-pchisq(37.717,39)
[1] 0.5283455

```

The large  $p$ -value associated with the residual deviance indicates there is no evidence of lack of fit. (We interpret the  $p$ -value as for grouped data in logistic regression, provided the responses are not too small, say more than 2 or 3). The coefficients are interpreted as follows: Since the coefficient of INB (-0.0014441) is negative, as the inner burden thickness increases, the number of accidents goes down (but this coefficient is only weakly significant, so we should interpret this coefficient with caution.) The coefficient of EXTRP (0.0622011) is positive, so as the extraction percentage goes up the mean accidents go up. This is a very significant coefficient, so this interpretation can be made with a bit more confidence. As the age of the mine increases, the number of accidents goes down (but again this is only weakly significant.) Note that all these interpretations assume the other variables are held fixed.

### Offsets

Sometimes we want to model various kinds of rates, such as death rates or failure rates, and explain these in terms of covariates such as age or exposure to hazardous environments. A rate is usually expressed as so many events occurring per unit of time or, in the case of death rates, so many deaths per 1000 population. (Sometimes a bigger base is used, e.g. for a rare disease, we might speak of deaths per 100,000 population.) Thus, if there were 10 deaths in a year from a type of cancer, in a population of 500,000, the death rate would be  $10 \times 500,000/100,000$  or 50 per 100,000. Suppose we want to model the mean rate  $\mu$  as a function of a covariate  $x$ , using a model of the form  $\mu = \exp(\beta_0 + \beta_1 x)$ . The mean *number* of deaths in a particular population will be

$$\begin{aligned}
 \text{mean number of deaths} &= \text{death rate per } 100,000 \times \text{population size}/100,000 \\
 &= \exp(\beta_0 + \beta_1 x) \exp(\log(\text{population size}/100,000)) \\
 &= \exp(\beta_0 + \beta_1 x + \log(\text{population size}/100,000)).
 \end{aligned}$$

Thus, to model the rate  $\mu$ , we can use the counts as a response, provided we include the special variable  $\log(\text{population size}/100,000)$  in the model. It is special because its regression coefficient is fixed to be 1. Such special variables are called **offsets**. Our next example illustrates their use.

**Example 15.** In a study to investigate death rates from child cancer in the UK, the data in Table 5.8 were obtained. We want to examine the effect of the covariates age (0-4, 5-14) and type of residence (rural/urban) on the death rates. Two different sets of rates were required, those where the child's cytology was lymphoblastic, and those where the cytology was myeloblastic. The child population of Northumberland and Durham was classified according to the two factors of age

and type of residence, resulting in 4 sub-populations. We want to estimate a rate for each sub-population/cytology combination. The data were assembled into an R data frame `cancer.df` with variables `Cytology`, `Residence`, `Age`, `Deaths` and `Pop`. The following R-code does the analysis.

```
cancer.glm<-glm(Deaths ~ Cytology*Residence*Age, family=poisson,
offset=log(Pop/100000), data=cancer.df)
> anova(cancer.glm, test="Chisq")
Analysis of Deviance Table
Model: poisson, link: log
Response: Deaths
Terms added sequentially (first to last)
```

	Df	Deviance	Resid.	Df	Resid. Dev	P(> Chi )
NULL				7	92.452	
Cytology	1	48.952		6	43.500	2.624e-12
Residence	1	5.848		5	37.652	0.016
Age	1	23.875		4	13.777	1.028e-06
Cytology:Residence	1	1.110		3	12.667	0.292
Cytology:Age	1	8.717		2	3.950	0.003
Residence:Age	1	2.895		1	1.054	0.089
Cytology:Residence:Age	1	1.054		0	5.107e-15	0.304

Since there are no significant interactions between Residence and the other factors, this analysis suggests that the model `Deaths~Cytology*Age + Residence` is appropriate, i.e. the difference (on the log scale) between Rural and Urban is the same for all age/cytology combinations. Fitting this model confirms this:

```
submodel.glm<-glm(Deaths ~ Cytology*Age +Residence, family=poisson,
offset=log(Pop/100000), data=cancer.df)
summary(submodel.glm)
Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      3.3893     0.1465  23.139 < 2e-16 ***
CytologyM        -1.5983     0.2584  -6.184 6.24e-10 ***
Age6-14          -0.9821     0.1767  -5.557 2.75e-08 ***
ResidenceU         0.3677     0.1546   2.379 0.01736 *
CytologyM:Age6-14  1.0184     0.3500   2.910 0.00362 **
---
Null deviance: 92.4517  on 7  degrees of freedom
Residual deviance:  5.0598  on 3  degrees of freedom
AIC: 52.858
```

Note the residual deviance of 5.0598 on 3 degrees of freedom - this indicates the model fits well. (Deviances for Poisson regression models are interpreted as for grouped logistic regression). Using this model, we can calculate the predicted death rates per 100,000 for the different combinations

```
> rates = exp(predict(submodel.glm)-log(cancer.df$Pop/100000))
> data.frame(Age=cancer.df$Age, Residence=cancer.df$Residence,
+ Cytology=cancer.df$Cytology, Rate=round(rates,2))
  Age Residence Cytology  Rate
1  0-5         R        L 29.65
2  0-5         R        M  6.00
3 6-14         R        L 11.10
4 6-14         R        M  6.22
5  0-5         U        L 42.82
6  0-5         U        M  8.66
7 6-14         U        L 16.04
8 6-14         U        M  8.98
```

This indicates that for younger children, the rates are much higher for lymphoblastic cancers than for myeloblastic. Rates are higher across the board for urban children, by a factor of 44% (since  $\exp(0.3677) = 1.4444$ ). Note that we had to adjust the prediction: using predict alone gives the result with the offset included (i.e. predicts the mean counts, not the mean rates).

### 5.3.2 Contingency tables

A contingency table is convenient way of displaying data that results from the classification of a number of subjects or cases into different categories. We will illustrate the analysis of contingency table data using three examples.

**Example 16.** In our first example, Table 5.9 contains data on U.S. deaths by falling (1970 data). Each death is classified by a single factor, the month of occurrence. The table records the number of deaths that occurred during each month in 1970.

Table 5.9: U.S. deaths by falling, 1970.

Month	Number of falls	Month	Number of falls
Jan	1688	July	1406
Feb	1407	Aug	1446
Mar	1370	Sept	1322
Apr	1309	Oct	1363
May	1341	Nov	1410
June	1388	Dec	1526

**Example 17.** For our second example, consider the data in Table 5.10, where 655 students chosen at random from the student body at Auckland University are cross-classified according to their degree course and socio-economic status (SES). This is a two-way table since students are classified by two criteria (factors). Each combination of SES and degree creates a cell in the table. As there are 7 categories of degree and 6 categories of SES the total number of cells is  $7 \times 6 = 42$ . The table records the number of students that fall into each cell.

Table 5.10: University of Auckland students classified by degree and SES.

SES	Degree enrolled for						
	Arts	Science	Law	Engineering	Commerce	Medicine	Other
1	76	28	38	28	17	23	27
2	44	31	25	17	24	9	14
3	37	14	8	20	16	4	12
4	38	12	9	19	15	2	110
5	4	55	0	3	1	1	1
6	9	4	3	4	2	1	0

**Example 18.** For our third example of a contingency table, consider the data in Table 5.11, where 123 patients suffering from diabetes are classified on the basis of three criteria. Each criterion divides the patients into 2 categories. Thus we have a three-way table with a total of  $2 \times 2 \times 2 = 8$  cells.

In general a  $k$ -way contingency table consists of cases or subjects that have been cross-classified using  $k$  different criteria. The total number of cells in the table, denoted by  $m$ , is simply the product of the numbers of categories for each of the criteria. For contingency table data our analysis involves comparing the probabilities that an observation occurs in the different cells. For  $k$ -way tables where  $k > 1$ , we are typically interested in investigating the relationships between

Table 5.11: Diabetes patients classified by three criteria.

Family history of diabetes		Yes		No	
Dependent on insulin injections		Yes	No	Yes	No
Age at onset	< 45	6	1	16	2
	$\geq 45$	6	36	8	48

the  $k$  criteria. The models we use differ from regression models in that none of the  $k$  factors is singled out as a response – in effect the cell counts are the response.

### 5.3.3 Analysis of one-way tables

Consider the death by falling data (Example 16) and suppose we wish to answer the question: Is the frequency of death by falling related to month? Our strategy will be to identify a suitable model for the data and then use this model to answer the question. We will consider two different models that could be used for the data in Table 5.9: (i) a multinomial sampling model, and (ii) a Poisson sampling model.

#### Multinomial sampling model

Suppose that the probability that a death chosen at random falls in the  $i$ th month is  $\pi_i$ , for  $i = 1, \dots, m$ , where  $m = 12$ . We assume that the data are such that each death is classified into exactly one month. This implies that  $\pi_1 + \dots + \pi_m = 1$ . Let  $Y_1, Y_2, \dots, Y_m$  be random variables denoting the number of deaths that are classified into each of the months. Provided that the sample represents a small fraction of the total population, the observed set of counts  $(y_1, y_2, \dots, y_m)$  can be considered as an observation from a multinomial distribution. The multinomial distribution extends the binomial distribution to situations where there are more than 2 possible outcomes. The multinomial probability function is:

$$\Pr(Y_1 = y_1, Y_2 = y_2, \dots, Y_m = y_m) = \frac{n!}{y_1! y_2! \dots y_m!} \pi_1^{y_1} \pi_2^{y_2} \dots \pi_m^{y_m}. \quad (5.4)$$

We want to test the hypothesis that  $\pi_1 = \pi_2 = \dots = \pi_{12} = \frac{1}{12}$  i.e. someone who has died from falling is equally likely to have done so in any given month. To do this, we will consider whether a multinomial model with  $\pi_i = 1/12$  for  $i = 1, \dots, 12$  provides reasonable fit to our data.

The test we will use is based on the deviance for this model. The likelihood function for the multinomial model is the probability function from equation 5.4, regarded as a function of the  $\pi_i$ 's. To find the deviance we need to find the maximum value of the likelihood function both under the maximal model and under the hypothesised model:

$$\text{deviance} = 2 \log L_{\max} - 2 \log L_{\text{mod}}.$$

Let  $\tilde{\pi}_i$ ,  $i = 1, \dots, m$ , be the values of the probabilities for the maximal model (these are the values that maximise equation 5.4 with the only restriction being  $\sum_i \tilde{\pi}_i = 1$ ) and let  $\hat{\pi}_i$ 's be the probabilities that maximise equation 5.4 under the constraints imposed by the hypothesised model (in this case that each  $\pi_i$  equals  $1/12$ ). Then the deviance can (with a bit of algebraic manipulation) be written as:

$$\text{deviance} = 2 \sum_{i=1}^m y_i \log \tilde{\pi}_i - 2 \sum_{i=1}^m y_i \log \hat{\pi}_i.$$

For the multinomial distribution, the values of the  $\tilde{\pi}_i$ 's are always given by the observed proportions:  $\tilde{\pi}_i = y_i/n$ . Thus to get  $L_{\max}$  we substitute these into equation 5.4. For this example, the hypothesised model completely specifies the  $\hat{\pi}_i$ 's and so to get  $L_{\text{mod}}$  we substitute  $\pi_i = 1/12$  for all  $i$  into equation 5.4. Thus the expression for the deviance becomes:

$$\text{deviance} = 2 \sum_{i=1}^{12} y_i \log \left( \frac{y_i}{n} \right) - 2 \sum_{i=1}^{12} y_i \log \left( \frac{1}{12} \right).$$

This can be evaluated using R as follows:

```
> y<-c(1688,1407,1370,1309,1341,1388,1406,1446,1322,1363,1410,1526)
> n<-sum(y)
> maximal<-2*sum(y*log(y/n))
> model<-2*sum(y*log(1/12))
> maximal-model
[1] 81.09515
```

Under the hypothesised model, the deviance has a  $\chi^2$  distribution with  $m - 1 - c$  degrees of freedom, where  $c$  is the number of parameters that must be estimated under the hypothesised model. In our case  $c = 0$  since the hypothesised model completely specifies the  $\hat{\pi}_i$ 's. Thus the degrees of freedom for the  $\chi^2$  reference distribution is  $12 - 1 - 0 = 11$  and the  $p$ -value is calculated by  $\Pr(\chi_{11}^2 \geq 81.0951)$ :

```
> 1-pchisq(81.09515,11)
[1] 9.05831e-13
```

The  $p$ -value is very small indicating very strong evidence against the hypothesised model. That is we have very strong evidence against  $\pi_i = 1/12$  for all  $i$ .

### Poisson sampling model

A second way we could look at the death by falling data is to use a Poisson sampling model. In this case, we consider the number of deaths by falling for month  $i$  is an observation from a Poisson distribution with mean  $\lambda_i$ . To answer our question, "Is the frequency of deaths by falling related to month?", we test the hypothesis that  $\lambda_1 = \lambda_2 = \dots = \lambda_{12}$ . The Poisson probability function is:

$$\Pr(Y = y) = \frac{e^{-\lambda} \lambda^y}{y!}.$$

Thus if we assume that the observations are independent, then the likelihood function is:

$$\Pr(Y_1 = y_1, Y_2 = y_2, \dots, Y_m = y_m) = \prod_{i=1}^m \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}. \quad (5.5)$$

Again we will use a test that is based on the deviance of the hypothesised model. To calculate this deviance we will need to determine the values of  $L_{\max}$  and  $L_{\text{mod}}$  for the Poisson distribution. The maximal model corresponds to the set of values for the  $\lambda_i$ 's that maximise equation 5.5 when no restrictions are placed on the  $\lambda_i$ 's. We will denote these values as  $\tilde{\lambda}_i$  for  $i = 1, \dots, m$ . The maximal model corresponds to  $\tilde{\lambda}_i = y_i$  and  $L_{\max}$  is obtained by substituting these into equation (5.5). To find  $L_{\text{mod}}$  we need to maximise equation 5.5 under the restriction that  $\lambda_1 = \lambda_2 = \dots = \lambda_m$ . This is achieved by selecting  $\hat{\lambda}_i = (\sum y_i)/n = \bar{y}$  for all  $i$ . Using these values for the  $\tilde{\lambda}_i$ 's and the  $\hat{\lambda}_i$ 's and with a bit of algebraic manipulation, the deviance for the Poisson sampling model can be written as:

$$\begin{aligned} \text{deviance} &= 2 \log L_{\max} - 2 \log L_{\text{mod}} \\ &= 2 \sum_{i=1}^m (\tilde{\lambda}_i + y_i \log \tilde{\lambda}_i) - 2 \sum_{i=1}^m (\hat{\lambda}_i + y_i \log \hat{\lambda}_i) \\ &= 2 \sum_{i=1}^m (y_i + y_i \log y_i) - 2 \sum_{i=1}^m (\bar{y} + y_i \log \bar{y}). \end{aligned}$$

Using R to calculate the deviance gives:

```
> ybar<-mean(y)
> 2*(sum(-y+y*log(y))) -2*(sum(-ybar+y*log(ybar)))
[1] 81.09515
```

Notice we get exactly the same deviance as we did using the multinomial model. This is no accident. The two procedures will always give us the same deviance and thus the same  $p$ -value

when testing if an hypothesised model is adequate. A third, more convenient, method of getting the same result is to use the `glm` function of *R* to create a Poisson regression model. To do this we create a factor whose levels represent the different cells in our contingency table. Then the cell counts are modeled as function of this factor using the `family=poisson` option of `glm`:

```
y<-c(1688,1407,1370,1309,1341,1388,1406,1446,1322,1363,1410,1526)
> month<-c("jan","feb","mar","apr", "may", "jun",
            "jul","aug","sep", "oct","nov","dec")
> month<-factor(month)
> falls.glm<-glm(y~month,family=poisson)
> anova(falls.glm,test="Chi")
Analysis of Deviance Table
Model: poisson, link: log
Response: y
Terms added sequentially (first to last)
      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                                11      81.095
month 11      81.095                0  5.151e-14 9.059e-13
```

### 5.3.4 Testing goodness of fit

The contingency table techniques we have been studying can also be used to determine if a given set of data follows a particular distribution. We saw above how to test if a given model fits a set of frequencies. We can use this idea to see if the data come from a Poisson or binomial distribution (or any other mathematically described distribution for that matter.) We illustrate with an example.

**Example 19.** The data in Table 5.12 relate to 6115 families in Saxony, Germany, each of which had 12 children. The families are classified according to the number of boys in each family. Are the sexes of successive children independent? Is a boy more likely to be followed by a girl or another boy?

Table 5.12: Numbers of boys in 6115 families of size 12.

Number of Boys	Frequency	Relative frequency	Binomial probability
0	3	0.0005	0.0002
1	24	0.0039	0.0020
2	104	0.0170	0.0117
3	286	0.0468	0.0423
4	670	0.1096	0.1027
5	1033	0.1689	0.1775
6	1343	0.2196	0.2236
7	1112	0.1818	0.2070
8	829	0.1356	0.1397
9	478	0.0782	0.0671
10	181	0.0296	0.0217
11	45	0.0074	0.0043
12	7	0.0011	0.0004

If the sexes of the children in a family are independent of one another, then the births are Bernoulli trials, like coin tosses, and the number of boys should have a Binomial distribution with parameters  $n = 12$  and  $p$ , where  $p$  is the probability a child is a boy. (We would expect that this is a bit over 50%.) In the right-most two columns of Table 5.12, we calculate the relative frequencies for the Saxon data (i.e. the proportions of the 6115 families that had 0,1,2,...,12 boys) and the probabilities  $Pr[X = x], x = 0, 1, \dots, 12$ , where  $X$  has a binomial distribution with  $n = 12$ . Here, we have estimated the probability  $p$  by 0.519215. This the number of boys ( $0 \times 3 + 1 \times 24 + 2 \times 104 + \dots$ ) divided by the number of children ( $12 \times 6115$ ). Is the match between the relative frequencies and the theoretical binomial probabilities close enough for us to believe

that the data have a binomial distribution (which would imply the births are independent?) To answer this question, we calculate the residual deviance and  $p$ -value for the binomial model. We use the code

```
> y=c(3,24,104,286,670,1033,1343,1112,829,478,181,45,7)
> n=sum(y)
> maximal = sum(y*log(y/n))
> phat=sum(y*(0:12))/(12*n)
> bin.probs = dbinom(0:12, 12, phat)
> model = sum(y*log(bin.probs))
> deviance = 2*maximal - 2*model
> deviance
[1] 97.0065
> 1-pchisq(deviance, 11)
[1] 0
```

The  $p$ -value is 0, indicating that the differences between the relative frequencies and the binomial probabilities are too large to be due to chance - the hypothesis that the data have a binomial distribution is rejected. It seems that the births may not be independent (or perhaps the probability of a boy is not constant?) Note that when calculating the degrees of freedom, we subtracted 2 from the number of classes (13) rather than 1. This is because we estimated one parameter ( $p$ ) in the fitted distribution.

### 5.3.5 Analysis of two-way tables.

Consider Table 5.10 which classifies University of Auckland students according to degree programme and socio-economic status. Often with 2-way tables we are interested in how the two factors are related. For the current example, suppose we wish to investigate how the degree a student enrolls in is related to socio-economic status. The first question we should ask is “Are the two factors related?”. To do this we test the hypothesis that the factors are independent.

For two-way tables it is convenient to use a slightly different system of indexing the cells in the table: let cell  $i, j$  be the cell corresponding to row  $i$  and column  $j$ . Thus an observation that is put into cell  $i, j$  is in category  $i$  of the row factor and in category  $j$  of the column factor. Let  $Y_{ij}$  be the random variable representing the count for cell  $i, j$  and  $y_{ij}$  be the observed number of counts. Further let  $I$  and  $J$  represent the number of categories for the row factor and for the column factor respectively and thus the total number of cells is  $m = IJ$ .

Suppose we adopt a multinomial sampling model. Let  $\pi_{ij}$  represent the probability that a randomly selected individual is placed in the  $i, j$  cell. If the row factor and the column factor are independent, then this probability will be equal to the probability that the individual occurs in row  $i$  (call this  $\pi_{i+}$ ) multiplied by the probability that the individual occurs in row  $j$  (call this  $\pi_{+j}$ ):

$$\pi_{ij} = \pi_{i+} \times \pi_{+j} \quad \text{if the factors are independent.}$$

Note that  $\pi_{i+}$  is the sum of the  $\pi_{ij}$ ’s for the cells in row  $i$ ,  $\pi_{i+} = \sum_j \pi_{ij}$ , and that  $\pi_{+j}$  is the sum of the  $\pi_{ij}$ ’s for the cells in column  $j$ ,  $\pi_{+j} = \sum_i \pi_{ij}$ .

To test the hypothesis that the row factor and the column factor are independent we need to calculate the deviance for the “independence model”. Thus we need to find  $L_{\max}$  and  $L_{\text{mod}}$ . If we adjust the probability function for the multinomial distribution (equation 5.4) to our new indexing system we get

$$\Pr(Y_{11} = y_{11}, Y_{21} = y_{21}, \dots, Y_{IJ} = y_{IJ}) = \frac{n!}{y_{11}! y_{21}! \dots y_{IJ}!} \pi_{11}^{y_{11}} \pi_{21}^{y_{21}} \dots \pi_{IJ}^{y_{IJ}}. \quad (5.6)$$

For the maximal model, the  $\hat{\pi}_{ij}$ ’s are selected to maximise this expression with the only constraint being that  $\sum_i \sum_j \hat{\pi}_{ij} = 1$ . This is accomplished by setting  $\hat{\pi}_{ij} = y_{ij}/n$ . For the independence model, the  $\hat{\pi}_{ij}$ ’s are selected to maximise equation 5.6 under the restrictions:

1.  $\hat{\pi}_{ij} = \hat{\pi}_{i+} \times \hat{\pi}_{+j}$  for all  $ij$ . Note that  $\hat{\pi}_{i+}$ ’s and the  $\hat{\pi}_{+j}$  can be interpreted as the estimated row probabilities and the estimated column probabilities respectively.

2.  $\sum_i \hat{\pi}_{i+} = 1$  and  $\sum_j \hat{\pi}_{+j} = 1$ .

For the independence model we get  $\hat{\pi}_{i+} = y_{i+}/n$  for  $i = 1$  to  $I$  and  $\hat{\pi}_{+j} = y_{+j}/n$  for  $j = 1$  to  $J$  where  $y_{i+}$  is the total counts in row  $i$  and  $y_{+j}$  is the total counts in column  $j$  of the contingency table. As a result we get  $\hat{\pi}_{ij} = y_{ij}/n$ .

Using these results gives us the following expression for the deviance:

$$\begin{aligned} \text{deviance} &= 2 \log L_{\max} - 2 \log L_{\text{mod}} \\ &= 2 \sum_{i=1}^I \sum_{j=1}^J y_{ij} \log \frac{y_{ij}}{n} - 2 \sum_{i=1}^I \sum_{j=1}^J y_{ij} \log \left( \frac{y_{i+} y_{+j}}{n^2} \right). \end{aligned}$$

If we evaluate this expression for our current example we get a deviance of 355.612. Note that there is a small complication that arises in this calculation since two of the cell counts are 0 which give us terms of  $0 \times \log 0$  – to get the correct deviance these terms should be taken as being equal to 0. To find the  $p$ -value for our hypothesis test we need to use a  $\chi^2$  distribution with  $m - 1 - c$  degrees of freedom. For a two-way table  $m = IJ$ . Recall that  $c$  represents the number of parameters that must be estimated under the hypothesised model. For an independence model we must estimate  $\hat{\pi}_{i+}$  for  $i = 1$  to  $I$  and  $\hat{\pi}_{+j}$  for  $j = 1$  to  $J$ . However, since  $\sum_i \hat{\pi}_{i+} = 1$  and  $\sum_j \hat{\pi}_{+j} = 1$  we only actually estimate  $I - 1$  of the  $\hat{\pi}_{i+}$ 's and  $J - 1$  of the  $\hat{\pi}_{+j}$ 's. Thus  $c = I + J - 2$  and our expression becomes  $m - 1 - c = IJ - 1 - (I + J - 2) = (I - 1) \times (J - 1)$ . For our example  $I = 6$  and  $J = 7$  so we use a  $\chi^2$  distribution with  $5 \times 6 = 30$  degrees of freedom.

```
> 1-pchisq(355.612,30)
[1] 0
```

Thus the  $p$ -value is extremely small which represents extremely strong evidence against the hypothesis of independence.

A easier method of obtaining this result would be to use the `glm` function to fit a Poisson regression model. To do this we define variables to represent the row and the column factors and use these as regressors in a Poisson regression model of the cell counts. In this framework, we test to see whether the two factors are related by testing whether the interaction between the two factors is needed in the model.

```
> degree<-rep(c("arts","sci","law","eng","com","med","oth"),6)
> ses<-rep(1:6,rep(7,6))
> nums<-c(76,28,38,28,17,23,27,44,31,25,17,24,9,14,37,14,8,20,
+ 16,4,12,38,12,9,19,15,2,110,4,55,0,3,1,1,1,9,4,3,4,2,1,0)
> degree<-factor(degree)
> ses<-factor(ses)
> enrol.glm<-glm(nums~degree*ses,family=poisson,maxit=25)
> anova(enrol.glm,test="Chi")
Analysis of Deviance Table
Model: poisson, link: log
Response: nums
Terms added sequentially (first to last)
      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                                41      829.93
degree          6      182.38          35      647.56 1.062e-36
ses             5      291.95          30      355.61 5.394e-61
degree:ses     30      355.61           0      2.458e-06 2.367e-57
```

Note that the line for `degree:ses` gives us the same deviance and degrees a freedom as we got for testing the independence hypothesis using the multinomial sampling model.

### 5.3.6 Analysis of three-way tables.

Suppose we now have classification factors  $A$ ,  $B$ , and  $C$  with  $I$ ,  $J$ , and  $K$  levels respectively. The three-dimensional table is illustrated in Figure 5.11. Factor  $A$  is the “row” factor, factor  $B$  the “column” factor and factor  $C$  the “slice” factor. We extend the terminology we used for two-way



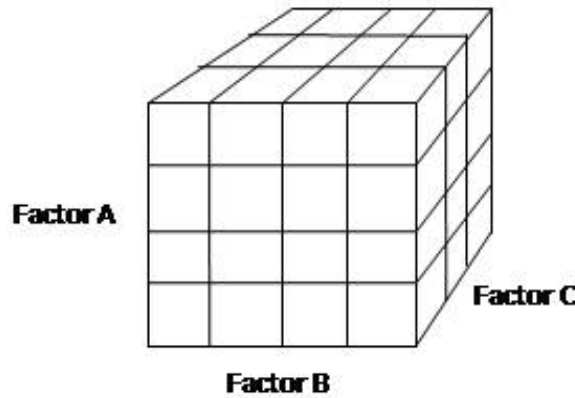


Figure 5.11: A three-dimensional contingency table.

tables in the obvious manner. Let  $\pi_{ijk}$  be the probability that a case is classified into the  $ijk$ th cell, let  $Y_{ijk}$  be the random variable of counts for this cell and let  $y_{ijk}$  be the observed counts.

Our aim is to investigate how the classification factors are related to each other. The first job will be to determine which pair of factors are related to each and which are not (i.e. are independent). There are a number of possibilities for the types of independence that can occur among the factors.

### Mutually independent factors

From the multinomial sampling perspective, the mutual independence of  $A$ ,  $B$  and  $C$  is equivalent to

$$\pi_{ijk} = \pi_{i++}\pi_{+j+}\pi_{++k} \quad (5.7)$$

for all  $ijk$ , where  $\pi_{i++} = \sum_j \sum_k \pi_{ijk}$ ,  $\pi_{+j+} = \sum_i \sum_k \pi_{ijk}$  and  $\pi_{++k} = \sum_i \sum_j \pi_{ijk}$ . We could proceed by developing the expression for the deviance under multinomial sampling for this situation and using this value to test whether this mutual independence model is compatible with the observed. Instead we will take the easier option of performing an equivalent test using Poisson regression. The mutual independence model is equivalent to a Poisson regression model that does not contain any interactions whereas the maximal model is equivalent to a Poisson regression model that contains all possible interactions between the three factors. Thus to test for mutual independence we compare the no interaction model (submodel) to the model that includes all possible interactions (full model).

Suppose we wish to test the hypothesis that the three classification factors for the diabetes data (Table 5.11) are mutually independent. This can be done in *R* as follows:

```
> onset<-rep(c("over","under"),c(4,4))
> history<-c("y","y","n","n","y","y","n","n")
> depend<-c("y","n","y","n","y","n","y","n")
> counts<-c(6,1,16,2,6,36,8,48)
> diabetes.df<-data.frame(onset,history,depend,counts)
> full.glm<-glm(counts~onset*history*depend,family=poisson,data=diabetes.df)
> sub.glm<-glm(counts~onset+history+depend,family=poisson,data=diabetes.df)
> anova(sub.glm,full.glm,test="Chi")
```

Analysis of Deviance Table

Model 1: counts ~ onset + history + depend

```

Model 2: counts ~ onset * history * depend
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         4      51.933
2         0  4.441e-16  4   51.933 1.424e-10

```

This test produces a very small  $p$ -value and thus we have very strong evidence against the hypothesis that the three factors are mutually independent.

### One factor is independent of the other two

The above test rules out the possibility that the three factors are mutually independent. However, it still may be the case that one of the factors is independent of the other two. In terms of multinomial sampling, if  $A$  is independent of  $B$  and  $C$ , then

$$\pi_{ijk} = \pi_{i++}\pi_{+jk} \quad (5.8)$$

for all  $ijk$ , where  $\pi_{i++} = \sum_j \sum_k \pi_{ijk}$  and  $\pi_{+jk} = \sum_i \pi_{ijk}$ . Again we will test the hypothesis that this is an adequate model by performing an equivalent test using Poisson regression. To test that one factor is independent of the other two using Poisson regression, we define the submodel to be the model that does not contain any interactions involving that factor. For the diabetes example suppose we wish to test that **onset** is independent of the other two factors. The full model is the maximal model (as before) but now the submodel contains the three main effects and the **history:depend** interaction. The test using  $R$  is:

```

> sub2.glm<-glm(counts~onset+history*depend,family=poisson,data=diabetes.df)
> anova(sub2.glm,full.glm,test="Chi")
Analysis of Deviance Table
Model 1: counts ~ onset+history*depend
Model 2: counts ~ onset*history*depend
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         3      51.023
2         0  4.441e-16  3   51.023 4.838e-11

```

The small  $p$ -value provides strong evidence against the hypothesis that **onset** is independent of the other two factors.

Of course, we should also test that (i) **history** is independent of **onset** and **depend** and (ii) **depend** is independent of **onset** and **history**:

```

> sub3.glm<-glm(counts~history+onset*depend,family=poisson,data=diabetes.df)
> anova(sub3.glm,full.glm,test="Chi")
Analysis of Deviance Table
Model 1: counts ~ history+onset*depend
Model 2: counts ~ history*onset*depend
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         3      1.94634
2         0  4.441e-16  3   1.94634  0.58362

> sub4.glm<-glm(counts~depend+onset*history,family=poisson,data=diabetes.df)
> anova(sub4.glm,full.glm,test="Chi")
Analysis of Deviance Table
Model 1: counts ~ depend+onset*history
Model 2: counts ~ history*onset*depend
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         3      50.034
2         0  4.441e-16  3   50.034 7.859e-11

```

The first test gives no evidence against the hypothesis that **history** is independent of **onset** and **depend** whereas the second test gives strong evidence against the hypothesis that **depend** is independent of **onset** and **history**. If we consider the results from all the hypothesis tests, we

conclude that they are consistent with `onset` and `depend` being related each other but both being independent of `history`. This conclusion is supported by looking at the output from `anova` for the full model:

```
> anova(full.glm, test="Chisq")
Analysis of Deviance Table
Model: poisson, link: log
Response: counts
Terms added sequentially (first to last)
      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                                7      125.163
onset                1      46.314             6      78.848 1.007e-11
history              1       5.117             5      73.731   0.024
depend               1      21.798             4      51.933 3.029e-06
onset:history        1       1.900             3      50.034   0.168
onset:depend         1      49.987             2       0.047 1.548e-12
history:depend       1       0.007             1       0.039   0.932
onset:history:depend 1       0.039             0      4.441e-16   0.843
```

Clearly, no interactions between history and the other two factors are required - all the  $p$ -values are large.

### Two factors are conditionally independent given the third

In terms of multinomial sampling, if  $A$  and  $B$  are conditionally independent given  $C$ , then

$$\pi_{ijk} = \frac{\pi_{i+k}\pi_{+jk}}{\pi_{+++}} \quad (5.9)$$

for all  $ijk$ , where  $\pi_{i++} = \sum_j \sum_k \pi_{ijk}$ ,  $\pi_{i+k} = \sum_j \pi_{ijk}$  and  $\pi_{+jk} = \sum_i \pi_{ijk}$ . This is equivalent to  $A$  and  $B$  being independent in all the  $K$  separate  $I \times J$  tables corresponding to the fixed levels of  $C$ . In terms of Poisson regression, this is equivalent to the  $AB$  and  $ABC$  interactions being zero, so that the model is `counts ~ A * C + B * C`. We can test the hypothesis that this is an adequate model by performing an equivalent test using Poisson regression.

**Example 20.** In this example, Table 5.13, the 674 “cases” are convicted defendants in homicide indictments in 20 Florida counties 1976-1987. The factors are Defendant’s Race, Victim’s Race, and Death Penalty. To identify how the three factors are related to each other, we want to identify

Table 5.13: The Florida murder data.

Defendant’s Race	Victim’s Race			
	Black		White	
	Death Penalty		Death Penalty	
	Yes	No	Yes	No
Black	13	195	23	105
White	1	19	39	265

which interactions are needed in a Poisson regression model that uses the cell counts as the response. Rather than sort through different possible models ourselves as we did in the previous example, we will let  $R$  do this for us. The `step` function in  $R$  can be used to do stepwise model selection and thus eliminate much of the hassle in identifying which of the interactions are needed in the model. For this example, first we create a data frame:

```
> defendant<-rep(c("b","w"),c(4,4))
> victim<-c("b","b","w","w","b","b","w","w")
> dp<-rep(c("y","n"),4)
> counts=c(13,195,23,105,1,19,39,265)
> dp.df<-data.frame(defendant,victim,dp,counts)
> dp.df
```

	defendant	victim	dp	counts
1	b	b	y	13
2	b	b	n	195
3	b	w	y	23
4	b	w	n	105
5	w	b	y	1
6	w	b	n	19
7	w	w	y	39
8	w	w	n	265

For the `step` command, we need to specify a starting model and a model that contains all the terms that we want to be considered. For our purposes, the starting model will just contain the main effects (i.e. the independence model) and the largest possible model will contain the main effects plus all the interactions. The largest possible model is specified using the second argument in `step`:

```
> dp.glm=glm(counts~., data=dp.df, family=poisson)
> # ~. is shorthand for the additive (independence model)
> full.model.formula = ~.^3 # this is shorthand for full model
> step(dp.glm, full.model.formula, trace=0)
```

```
Call: glm(formula = counts ~ defendant + victim + dp + defendant:victim +
          victim:dp, family = poisson, data = dp.df)
```

Coefficients:

(Intercept)	defendantw	victimw	dpy
5.2742	-2.3418	-0.5771	-2.7269
defendantw:victimw	victimw:dpy		
3.2068	0.9406		

Degrees of Freedom: 7 Total (i.e. Null); 2 Residual

Null Deviance: 774.7

Residual Deviance: 1.922 AIC: 56.64

The selected model contains only the `defendant:victim` and the `victim:dp` interactions. This is the model where defendant and death penalty are independent, given the victim's race. What this means is that if we fix the level of a third factor (victim's race in this example) they become independent. So if we were to only consider the data where victim's race = black then death penalty and defendant's race are independent. Similarly, if we only consider the data where victim's race = white then death penalty and defendant's race are independent. **But** if we combine the data for victim's race = black and victim's race = white then death penalty and defendant's race appear to be related. We will discuss this phenomenon further in Section 5.3.9.

### 5.3.7 Odds ratios

The odds ratio is a common method of exploring relationships between classification factors. Consider a two way table with two factors each at two levels as illustrated in Table 5.14.

Table 5.14: The  $2 \times 2$  table.

		Columns		
		1	2	Total
Rows	1	$\pi_{11}$	$\pi_{12}$	$\pi_{1+}$
	2	$\pi_{21}$	$\pi_{22}$	$\pi_{2+}$
Total		$\pi_{+1}$	$\pi_{+2}$	1

Suppose we want to investigate the relationship between the row factor and the column factor. The odds ratio does this by considering the odds that a case occurs in a given row conditional on the column that it occurs in. Let  $\text{odds}(r = i | c = j)$  represent the odds that a case occurs in row  $i$

given that it occurs in column  $j$  and recall that the definition of odds is  $\text{odds}(r = i|c = j) = \Pr(r = i|c = j) / \Pr(r \neq i|c = j)$ . For our table we can investigate the relationship between the row factor and the column factor by considering  $\rho = \text{odds}(r = 2|c = 2) / \text{odds}(r = 2|c = 1)$  which for obvious reasons is called an odds ratio. We can write the odds ratio in terms of the cell probabilities:

$$\begin{aligned} \rho &= \frac{\text{odds}(r = 2|c = 2)}{\text{odds}(r = 2|c = 1)} \\ &= \frac{(\pi_{22}/\pi_{+2})/(\pi_{12}/\pi_{+2})}{(\pi_{21}/\pi_{+1})/(\pi_{11}/\pi_{+1})} \\ &= \frac{\pi_{11} \pi_{22}}{\pi_{21} \pi_{12}} \end{aligned} \quad (5.10)$$

The odds ratio has the following properties:

1. If we reverse the roles of rows and columns, we get exactly the same expression (equation 5.10) in terms of the cell probabilities. If we swap the factor levels of the rows, we get the reciprocal of the odds ratio. Similarly, if we swap the factor levels of the columns, we get the reciprocal of the odds ratio.
2. If the row factor and the column factor are independent then the odds ratio will be 1. To see this recall that independence of two factors is equivalent to  $\pi_{ij} = \pi_{i+} \pi_{+j}$  for all  $ij$  and apply this relationship to each term in equation 5.10.

Note that we get the same value if we use the definition

$$\rho = \frac{\text{odds}(r = 1|c = 1)}{\text{odds}(r = 1|c = 2)}.$$

This is sometimes more convenient when describing the meaning of the odds ratio, if we want to make a reference to level 1 instead of level 2. See Example 21.

### The estimated odds ratio

In practice, we have to estimate the odds ratio. Suppose we take a sample of  $n$  cases and classify them into a  $2 \times 2$  table where  $y_{11}, y_{21}, y_{12}, y_{22}$  represent the observed counts. An obvious estimate of the odds ratio is obtained by using  $\hat{\pi}_{ij} = y_{ij}/n$  and substituting the estimated probabilities into equation 5.10:

$$\hat{\rho} = \frac{\hat{\pi}_{11} \hat{\pi}_{22}}{\hat{\pi}_{12} \hat{\pi}_{21}} = \frac{y_{11} y_{22}}{y_{12} y_{21}}.$$

The sampling distribution of  $\hat{\rho}$  is highly skewed but for large  $n$  the distribution of  $\log \hat{\rho}$  is approximately Normal with mean  $\log \rho$  and standard error:

$$\text{s.e.}(\log \hat{\rho}) = \sqrt{\frac{1}{y_{11}} + \frac{1}{y_{12}} + \frac{1}{y_{21}} + \frac{1}{y_{22}}}.$$

An approximate test of  $\log \rho = 0$  (i.e. of  $\rho = 1$  or independence) can be conducted by calculating the test statistic:

$$z_o = \log \hat{\rho} / \text{s.e.}(\log \hat{\rho}).$$

The  $p$ -value is given by  $2 \times \Pr(Z \geq |z_o|)$  where  $Z \sim N(0, 1)$ .

Confidence intervals for  $\log \rho$  can be created based on this Normal approximation and then transformed into intervals for  $\rho$  using the exponential the exponential function. For example, a 95% confidence interval for  $\log \rho$  would be given by:

$$\log \hat{\rho} \pm 1.96 \sqrt{\frac{1}{y_{11}} + \frac{1}{y_{12}} + \frac{1}{y_{21}} + \frac{1}{y_{22}}}.$$

Applying the exponential function to the endpoints will give an interval for  $\rho$ .

**Example 21.** The data in Table 5.15 was extracted from a larger study. It cross classifies 2121 people according to personality type (A: show signs of stress, uneasiness, and hyperactivity or B: relaxed, easygoing and exercise regularly) and cholesterol level (N = normal, H = High).

Table 5.15: Subjects classified by cholesterol level and personality type

Personality	Cholesterol		Total
	Normal	High	
A	795	232	1027
B	886	208	1094
total	1681	440	2121

First we should test the hypothesis that personality type and cholesterol are independent:

```
> counts<-c(795,232,886,208)
> personality<-c("A","A","B","B")
> chol<-c("N","H","N","H")
> personality<-factor(personality)
> chol<-factor(chol, levels=c("N","H"))
> fit<-glm(counts~chol*personality,family=poisson)
> anova(fit,test="Chi")
```

#### Analysis of Deviance Table

Model: poisson, link: log

Response: counts

Terms added sequentially (first to last)

	Df	Deviance	Resid. Df	Resid. Dev	P(> Chi )
NULL			3	780.78	
chol	1	774.55	2	6.24	1.847e-170
personality	1	2.12	1	4.12	0.15
chol:personality	1	4.12	0	1.301e-13	0.04

As we have evidence against the independence hypothesis, it is of interest to explore the relationship between personality type and cholesterol level. The estimated odds ratio is:

$$\hat{\rho} = \frac{795 \times 208}{232 \times 886} = 0.804$$

Thus the odds that a personality type B subject has a high cholesterol level is estimated to be about 80 % of the odds for a personality type A subject. (in terms of level 1 rather than level 2, we could express this as the odds that a personality type A subject has a normal cholesterol level is estimated to be about 80 % of the odds for a personality type B subject.) A 95 % confidence interval for the log(odds ratio) is calculated as follows:

$$\begin{aligned} \log(0.804) \pm 1.96 \sqrt{\frac{1}{795} + \frac{1}{232} + \frac{1}{886} + \frac{1}{208}} \\ -0.2176 \pm 1.96 \times 0.1073 \\ (-0.4278, -0.0073) \end{aligned}$$

To convert this into a 95 % confidence interval for the odds ratio, we apply the exponential function to the limits:  $(\exp(-0.4278), \exp(-0.0073)) = (.652, .993)$ .

#### Using Poisson regression to get the odds ratio

In a  $2 \times 2$  table, there is a close relationship between the Poisson regression coefficient for the interaction and the odds ratio. In Example 19, the coefficient of the `chol*personality` interaction is -0.21757, as can be seen from the regression summary:

```
> summary(fit)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )	
(Intercept)	6.67834	0.03547	188.301	<2e-16	***
cholH	-1.23160	0.07462	-16.505	<2e-16	***
personalityB	0.10837	0.04885	2.218	0.0265	*
cholH:personalityB	-0.21757	0.10726	-2.028	0.0425	*

This is identical to the log of the odds ratio calculated directly from the table, as was done in Example 19. This is no coincidence: the regression coefficient of the interaction in a Poisson is always the estimated log odds ratio. The standard error for the log odds ratio from the summary table is 0.10726, which agrees with the previous figure. Thus, Poisson regression gives us an alternative way to estimate the odds ratio.

### 5.3.8 Odds ratios in $I \times J$ tables

We saw in the last section how to measure association between two factors if the table has two rows and two columns. What if there are more rows and columns? It turns out that to describe the association in an  $I \times J$  table, we need a *set* of odds ratios.

Table 5.16: Odds ratios in  $I \times J$  tables.

	Col 1	...	Col $j$	...
Row 1	$\pi_{11}$	...	$\pi_{1j}$	...
	.	...	.	...
Row $i$	$\pi_{i1}$	...	$\pi_{ij}$	...
	.	...	.	...

Suppose we select a typical cell in the table, say the  $i, j$  cell. Consider the  $2 \times 2$  table formed by taking the 1, 1 cell, the  $i, 1$  cell, the 1,  $j$ , cell and the  $i, j$  cell, as shown in Table 5.16. We can calculate the odds ratio for this table as

$$\rho_{ij} = \frac{\pi_{11}\pi_{ij}}{\pi_{i1}\pi_{1j}}.$$

We can then repeat this for all the cells (omitting the first row and column), and get a set of odds ratios. We illustrate with an example.

Table 5.17: National Opinion Research Center data.

Highest Degree	Religious Beliefs		
	Fundamentalist	Moderate	Liberal
Less than high school	178	138	108
High school or junior college	570	648	442
Bachelor or graduate	138	252	252

#### Example 22.

The National Opinion Research Centre at the University of Chicago conducts frequent surveys, including the General Social Survey, a very large survey conducted every two years. The data in Table 5.17 comes from the 1996 survey. In the table, 2726 respondents are classified by the two factors “Highest Degree” (with levels “Less than high school”, “High school or junior college”, “Bachelor or graduate”) and “Religious Beliefs” (with levels “Fundamentalist”, “Moderate”, “Liberal”). If we fit a Poisson regression model, a test for zero interaction has a  $\chi^2$  of 69.81 on 4 degrees of freedom, which gives a p-value of 1.675e-07, so that there is clear evidence of an association between education and religious belief. The R code and output is

```
> NORC.df = data.frame(counts = c(178,570,138,138,648,252,108,442,252),
  expand.grid(HighestDegree = c("L","H", "B"),
  ReligiousBeliefs=c("Fund","Mod","Lib")))
```

```
> NORC.df
  counts HighestDegree ReligiousBeliefs
1    178             L             Fund
2    570             H             Fund
3    138             B             Fund
4    138             L             Mod
5    648             H             Mod
6    252             B             Mod
7    108             L             Lib
8    442             H             Lib
9    252             B             Lib
```

```
> NORC.glm = glm(counts~HighestDegree*ReligiousBeliefs,
  family=poisson, data=NORC.df)
```

Analysis of Deviance Table

Model: poisson, link: log

Response: counts

Terms added sequentially (first to last)

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			8	1009.20	
HighestDegree	2	908.18	6	101.02	< 2.2e-16 ***
ReligiousBeliefs	2	31.20	4	69.81	1.675e-07 ***
HighestDegree:ReligiousBeliefs	4	69.81	0	0.00	2.488e-14 ***

To measure the association, we consider the four  $2 \times 2$  subtables shown in Figure 5.12, and compute the sample odds ratio for each one. There are now four odds ratios:

$$\hat{\rho}_{22} = 1.466, \quad \hat{\rho}_{23} = 1.278, \quad \hat{\rho}_{32} = 2.355, \quad \text{and} \quad \hat{\rho}_{33} = 3.009.$$

(The hats indicate sample estimates). Thus, the odds of having only a high school education to having less than high school are 1.466 times higher for religious moderates than for religious fundamentalists, and the odds of having a university education to having less than high school are 3.009 times higher for religious liberals than for religious fundamentalists.

The odds ratios above are just sample estimates. How can we get confidence intervals for the corresponding population odds ratios? As in the case of  $2 \times 2$  tables, there is a close relationship between the odds ratios and the coefficients in the Poisson regression we fitted above. Here is a part of the summary output from the regression, showing the estimates of the interaction terms:

Call:

```
glm(formula = counts ~ HighestDegree * ReligiousBeliefs, family = poisson,
  data = NORC.df)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	5.18178	0.07495	69.134	< 2e-16 ***
HighestDegreeH	1.16385	0.08586	13.555	< 2e-16 ***
HighestDegreeB	-0.25453	0.11342	-2.244	0.02483 *
ReligiousBeliefsMod	-0.25453	0.11342	-2.244	0.02483 *
ReligiousBeliefsLib	-0.49965	0.12197	-4.096	4.20e-05 ***
HighestDegreeH:ReligiousBeliefsMod	0.38278	0.12713	3.011	0.00260 **



	<b>Fund</b>	<b>Mod</b>		<b>Fund</b>	<b>Lib</b>
<b>L</b>	178	138	<b>L</b>	178	108
<b>H</b>	570	648	<b>H</b>	570	442
	<b>OR = 1.466</b>			<b>OR = 1.278</b>	

	<b>Fund</b>	<b>Mod</b>		<b>Fund</b>	<b>Lib</b>
<b>L</b>	178	138	<b>L</b>	178	108
<b>B</b>	138	252	<b>B</b>	138	252
	<b>OR = 2.355</b>			<b>OR = 3.009</b>	

Figure 5.12: The four  $2 \times 2$  sub-tables.

```

HighestDegreeB:ReligiousBeliefsMod 0.85671    0.15517    5.521 3.37e-08 ***
HighestDegreeH:ReligiousBeliefsLib 0.24533    0.13746    1.785 0.07430 .
HighestDegreeB:ReligiousBeliefsLib 1.10183    0.16153    6.821 9.03e-12 ***

```

Exponentiating these gives

```

> exp(coefficients(NORC.glm))
      (Intercept)                HighestDegreeH
      178.0000000                3.2022472
HighestDegreeB                ReligiousBeliefsMod
      0.7752809                0.7752809
ReligiousBeliefsLib HighestDegreeH:ReligiousBeliefsMod
      0.6067416                1.4663616
HighestDegreeB:ReligiousBeliefsMod HighestDegreeH:ReligiousBeliefsLib
      2.3553875                1.2780377
HighestDegreeB:ReligiousBeliefsLib
      3.0096618

```

so that the OR's can be calculated from the summary output. We can also calculate confidence intervals from the summary output: e.g. for  $\rho_{33}$ , the confidence interval for the interaction corresponding to `HighestDegreeB` and `ReligiousBeliefsLib` is  $1.10183 \pm 1.96 \times 0.16153$  or (0.7852, 1.4184). Exponentiating gives the confidence interval for the OR as  $(\exp(0.7852), \exp(1.4184))$  or (2.1929, 4.1306).

### 5.3.9 Simpson's paradox

Given a three-way table we may *collapse* the table over one of the factors to produce a two-way table in the other factors. In effect, we simply cross-classify the cases by the two remaining factors

ignoring the collapsing factor. The idea is that this allows us to focus on how the remaining factors are related. However, we must be careful in interpreting the results from a collapsed table in situations where the collapsed factor is related to the other factors. We may find that if we were to create a separate two-way table for each level of the collapsed factor that these tables give a very different picture to that given by the collapsed table.

The Florida murder data, Table 5.13, illustrates this point nicely. Suppose that we create two separate tables based on the Victim's race:

Victim=Black		
	Death penalty=Yes	Death penalty=No
Defendant=Black	13	195
Defendant=White	1	19

Victim=White		
	Death penalty=Yes	Death penalty=No
Defendant=Black	23	105
Defendant=White	39	265

The estimated odds ratio for the first table is 1.27 and for the second table is 1.49. Since both odds ratios are more than 1, this suggests that odds of getting the death penalty are higher for black defendants than for white defendants. Actually, our analysis in Example 15 indicates that the data in each of these tables is compatible with Defendants race and death penalty being independent in each case (i.e. the true odds ratio being 1). However there is certainly nothing in these two tables that would support the hypothesis that the odds of getting the death penalty is higher for white defendants.

Now suppose we collapse the table on Victim's race:

	Death penalty=Yes	Death penalty=No
Defendant=Black	36	300
Defendant=White	40	284

The estimated odds ratio is now 0.852 which suggests that the odds of getting the death penalty is *lower* for black defendants. Thus the collapsed table is suggesting the opposite conclusion to that suggested by the separate tables.

This apparent paradox (known as *Simpson's paradox*) is really just the difference between marginal association and conditional association that we saw in connection with scatterplots and coplots. The odds ratio between death penalty and race of defendant is the marginal odds ratio, which measures the association between the factors, with no other factors considered. It is a property of the joint distribution of death penalty and race of defendant.

The odds ratios in the separate tables (for white victims and black victims) measure the association between death penalty and race of defendant *conditional on victim's race* i.e. association in the conditional distributions. This will in general not be the same as the association in the unconditional distribution.

If, however, the victim's race (i.e. the conditioning factor) is in turn conditionally independent of one of death penalty or defendant's race, given the other, then the association in the conditional tables will be the same as in the marginal table. This is a special case of the following mathematical result:

*Suppose we have three factors  $A$ ,  $B$  and  $C$ , and suppose that  $C$  and  $A$  are conditionally independent given  $B$ . Then the odds ratio in the marginal  $AB$  table is the same as the odds ratios in the  $AB$  tables conditional on  $C$ . The same is true if  $C$  and  $B$  are conditionally independent given  $A$ .*

It turns out that death penalty is not independent of victims race, not even conditional on race of defendant. Murders involving white victims are more likely to lead to the death penalty. Since people tend to murder people of their own race, the collapsed table appears to show that whites get more death sentences. Using victim's race to create two separate tables, allows the true picture to emerge.

When the associations in the conditional and marginal tables are the same, we can collapse (add) the conditional tables and still get the same interpretation from the collapsed (i.e. marginal) table as that given by the conditional tables. Otherwise, collapsing may give a very different impression. In this case, collapsing over victim's race gave a false impression that the Florida courts are harder on white defendants than black.

### 5.3.10 Association graphs

In a contingency table with several factors, it can be quite complicated to sort out the pattern of association between the factors. The idea of an association graph can be helpful to illustrate the associations that may exist between the factors.

The graphs we consider here are not plots, but rather the mathematical version of a graph, which is a set of points or *vertices*. Pairs of points may be joined by *edges*. For example, in Figure 5.13 we show four different graphs, each with three vertices. Graphs are useful for describing

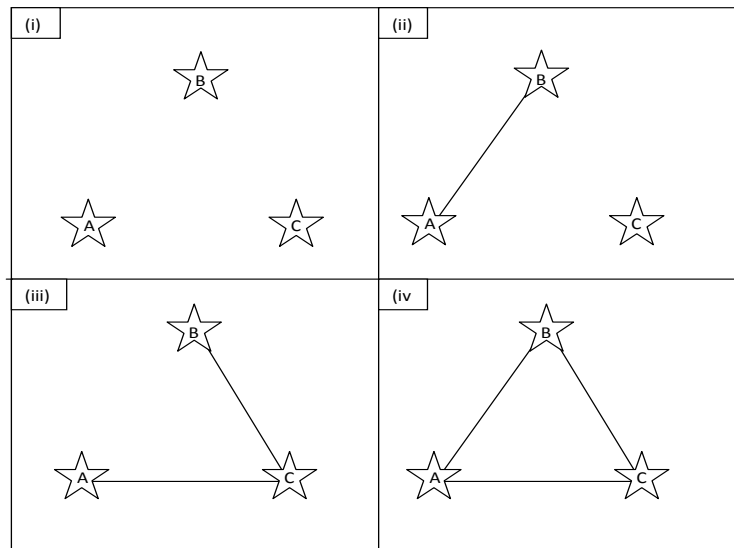


Figure 5.13: Four association graphs

contingency table models, particularly for *hierarchical* models. A model is hierarchical if whenever it includes an interaction with factors  $A_1, \dots, A_k$ , then it includes all main effects and interactions that can be formed from  $A_1, \dots, A_k$ . Thus, in R terms, if a model contains a term  $A : B : C$ , it must contain  $A : B$ ,  $A : C$  and  $B : C$ . Note that models written using the  $*$  operator will be hierarchical. For example, the model  $A * B * C + D$  is hierarchical.

We can represent a hierarchical model by a graph, by letting each vertex represents a factor. A pair of factors are joined by an edge if there is an interaction between them. Thus, Figure 5.13(i) represents a model where the factors  $A$ ,  $B$  and  $C$  are independent i.e. no interactions, Figure 5.13(ii) where  $A$  and  $B$  are independent of  $C$  (no  $AC$  or  $BC$  interaction), Figure 5.13(iii) a model where  $A$  and  $B$  are conditionally independent given  $C$  (no  $AB$  interaction). Note that the graph in Figure 5.13(iv) could represent either the model  $A * B * C$  or the model  $A * B + B * C + C * A$ .

The graphs are particularly useful in representing conditional independence. Suppose two factors  $A$  and  $B$  have no edge joining them directly, but there is a path from  $A$  to  $B$  going through vertices  $C, D, E$ . Then  $A$  and  $B$  will be conditionally independent given  $C, D$  and  $E$ .

## 5.4 Chapter summary

### 5.4.1 Main ideas

- The logistic regression model
- Interpreting regression coefficients
- Fitting the model
- Interpreting the output
- Grouped and ungrouped data
- Diagnosing model faults (see section summary for details)
- Contingency tables
- Goodness of fit
- Poisson regression
- Independence
- Odds ratios
- Association graphs
- Simpson's paradox

### 5.4.2 R functions used

`dummy.coef`

`gam`

`pchisq`

# Appendix A

## R details

In this Appendix we give some more details on R. We explain more about expressions, data objects, how to get data into R, how to write simple functions and a few other “non-statistical” aspects of R. This is only a brief survey; for more details consult the books recommended in Section A.11. The second Appendix (Appendix B) documents the R functions in the R330 package. This documentation is also available in the R on-line help.

### A.1 R expressions

We use R by typing in expressions, whose values are evaluated by the program. R then either stores the value of the expression (using the assignment operator), or prints it out on the screen. Values are stored in R by associating them with a variable name, which you can think of as a label for a piece of computer storage. Variable names are case sensitive, must begin with a letter, and contain only letters, digits or the symbols `.` and `_`. To store a value, we use the assignment operator `<-`, typed as two characters. Alternatively, we can use the equals sign (`=`). Thus, to store the value 10 in a variable `z`, we type

```
> z <-10
```

or

```
> z = 10
```

To see the number stored in a variable, just type the name:

```
> z
[1] 10
```

(the meaning of `[1]` is explained below). You can then use `z` in expressions:

```
> z + 2
[1] 12
```

Expressions often consist just of single functions, but more general expressions can be made using three ingredients: variables, functions and operators, such as `+` (plus) and `-` (minus). Here is a simple expression involving numbers and operators only:

```
> 3+5
[1] 8
```

Arithmetic expressions are built up in the usual way using the operators `+`, `-`, `*` (multiplication), `/` (division) and `^` (raising to a power). In addition, there are a wide variety of supplied mathematical functions, such as `log` (natural log), `exp` (exponential function), `sin` (sine) and `tan` (tangent). Thus if `z` has the value 9, the following expressions have the values shown:

```
> log(9)
```

```
[1] 2.197225
> exp(9)
[1] 8103.084
> tan(9)+9/20
[1] -0.002315659
```

A list of the R operators and mathematical functions is given in the on-line help. For example to get information on an operator, say `*`, or a function such as `log`, type

```
?"*"
```

or

```
?log
```

These operators have the usual *priorities*: in an expression, functions are evaluated before `*` and `/`, which in turn are done before `+` and `-`. Priorities can be overridden by parentheses: thus `tan(9) + 20/9` evaluates the tangent of 9, then adds the ratio of 20 and 9, while `tan(9+ 20)/9` evaluates `tan(29)` divided by 9.

### A.1.1 Character expressions

As well as expressions involving numbers, we can write expressions made up of strings of characters, which must be enclosed in double quotes (`"`). We can make data objects consisting of strings and store them:

```
> string1 <- "Happy"
> string2 <- "Christmas"
```

We can join strings together using the function `paste`:

```
> both <- paste(string1,string2)
> both
[1] "Happy Christmas"
```

Note how the blank is inserted automatically.

### A.1.2 Logical expressions

Logical expressions are used to compare data objects and evaluate to `TRUE` or `FALSE`. Thus `x == 2` is the expression “`x` equals 2”. It evaluates to `TRUE` if `x` has the value 2, and to `FALSE` otherwise. The other “logical operators are `!=` (not equals), `|` (or), `&` (and) and `!tind!` (not). Thus if `x` has the value 2, and `z` the value 9, we get

```
> x == 2
[1] TRUE
> x != 2
[1] FALSE
> (z==9) | (x==1) # either z equals 9 or x equals 1 (or both)
[1] TRUE
```

We can assign logical expressions to objects:

```
> expr1 <- x == 1
> expr1
[1] FALSE
> expr2 <- z == 9
> expr1 & expr2
[1] FALSE
> !(expr1&expr2)
[1] TRUE
```

## A.2 Data objects

If R could only operate on single numbers, it would be little more than a very non-portable calculator. Fortunately, a wide variety of data objects can be used in R. In this course we will use several of these: vectors, factors, matrices, data frames, and lists.

### A.2.1 Vectors

A vector is simply a collection of data values organised as a sequence. For example, the data object `BPD` we created in Section 3.11 was a vector. Vectors can be created by using the assignment operator `<-`, or the elements can be typed in using the keyboard, or read from an external data file. See Section A.5 below for more details.

The elements of vectors can be either numbers or character strings (which are enclosed in quotes) but we cannot mix numbers and strings in the same vector. Thus we can have vectors of numbers, or vectors of character strings. Vectors are useful for representing the columns of data matrices, and we use numeric vectors for numeric variables and vectors of character strings for categorical variables. For example, we can make a character vector `sex` to store the sexes of the 8 respondents in the dental survey data set (Table 1.2 in Chapter 1) by typing

```
sex <- c("M", "M", "F", "M", "F", "F", "M", "F")
```

Note the use of double quotes to surround each string.

Missing data is represented in R by the symbol `NA`. Thus if the age of one individual in the dental survey (see Table 1.2 of Chapter 1) was missing we get

```
> Age
[8] 37 23 45 NA 33 29 57 41
```

The function `length` is useful for counting the number of elements in a vector:

```
> length(BPD)
[1] 10
```

Sometimes it is useful to assign case labels to the elements of the vector, so we can match up a data value with the case the measurement comes from. The case labels can be either alphabetical strings (for example patients' names) or ID numbers. To assign labels, we can create a vector of labels using `c`. The elements of the vector can be either strings (if we use names) or numbers (if we use ID numbers). For the BPD data, suppose we want to use the babies' names as labels. We can create a vector `patient.labels` of names by typing

```
> patient.names <- c("Alastair", "Constance", "Peter", "Kathy",
+ "Ross", "Ilze", "Roland", "Lynne", "Charles", "Chris")
```

Note how we can break a long line by pressing the return key, and continuing on a new line. The R prompt changes from `>` to `+`, indicating that the line is not complete.

The labels can now be assigned to the elements of the vector `BPD` using the `names` function:

```
> names(BPD) <- patient.names
```

Now when we print out the vector on the computer screen both the elements and the labels are printed:

```
> BPD
Alastair Constance Peter Kathy Ross Ilze
      77      82      90      82      86      55
Roland   Lynne   Charles Chris
      80      87      79      81
```

Now we can see which element in the vector `BPD` corresponds to which infant.

If we don't assign our own labels, a vector will have sequence numbers `[1]`, `[2]`, ... and so on printed at the beginning of each line. This is why when single numbers are printed out, they are preceded by `[1]`, since single numbers are regarded as vectors having one element.

### Picking out elements: Subsetting

We can choose a subset of elements from a vector by supplying a set of indices in square brackets. If we supply a single index, we extract a single element. This gives us a way of referring to individual elements in a vector. Suppose that we wanted to look at the BPD of the first patient. We type

```
> BPD[1]
Alastair
77
```

If we supply a vector of indices, we get a new vector consisting of the corresponding elements of the vector BPD. This is useful for picking out subsets of cases. Suppose we wanted to pick out the BPD's of the female infants. We type

```
> BPD[c(2,4,6,8)]
Constance    Kathy    Ilze    Lynne
      82      82      55      87
```

since the female infants are the second, fourth, sixth and eighth cases. Negative indices delete elements rather than selecting them:

```
> BPD[c(-1,-3,-5,-7,-9,-10)]
Constance    Kathy    Ilze    Lynne
      82      82      55      87
```

Particularly useful is the operator `:` that allows us to construct a sequence of integers:

```
> 1:4
[1] 1 2 3 4
```

We can use this to pick out the first four elements of BPD:

```
> BPD[1:4]
Alastair Constance    Peter    Kathy
      77      82      90      82
```

We can also use a logical condition or a vector of logical values to select elements:

```
> is.female <- c(F,T,F,T,F,T,F,T,F,F)
> BPD[is.female]
Constance    Kathy    Ilze    Lynne
      82      82      55      87
```

The use of a logical condition to form a subset of a vector is illustrated in the next subsection.

### Vectors and Arithmetic

Arithmetic is done on vectors element by element:

```
> vector1 <- c(2, 4, 6, 8)
> vector1 + 2
[1] 4 6 8 10
> vector2 <- c(1, 3, 6, 8)
> vector1 + vector2
[1] 3 7 12 16
> vector1 * vector2
[1] 2 12 36 64
```

Logical operations are similar:

```
> vector1>2
[1] FALSE TRUE TRUE TRUE
```

This expression results in a vector of logical values – TRUE or FALSE, where the element of the vector is TRUE if the corresponding element of `vector1` is greater than 2, and FALSE otherwise.



This last example can be applied to extract data in a certain range from vectors. For example, to extract all the BPD's over 80mm, we can type

```
> BPD[BPD>80]
  Constance      Peter      Kathy      Ross      Lynne
      82         90         82         86         87
   Chris
      81
```

This works because the expression `BPD>80` gives a vector with `TRUE` in positions where BPD has elements over 80 and `FALSE` otherwise, and this vector picks out the appropriate elements from BPD when used as a vector of indices.

Functions that usually take a single numerical argument (such as the arithmetic functions `sin`, `exp` and so on) work element by element when applied to vectors. Thus if we want to transform the BPD measurements by taking logs, we type

```
> logBPD <- log(BPD)
> logBPD
  Alastair  Constance      Peter      Kathy      Ross
 4.343805  4.406719  4.499810  4.406719  4.454347

   Ilze      Roland      Lynne      Charles      Chris
 4.007333  4.382027  4.465908  4.369448  4.394449
```

to get the vector of logged BPD's.

As we saw above, there are functions which compute statistical summaries of the data in a vector: e.g. typing

```
> var(BPD)
[1] 92.1
```

calculates the variance of the data in the vector BPD. The simple statistical summaries are found by the functions `mean`, `min`, `max`, `median` and so on.

### A.2.2 Matrices

In R, a matrix is a rectangular array of data values organised by rows and columns. The data in a matrix must be all numeric, all character or all logical – we can't mix different types up in the same matrix.

#### Creating matrices

Table A.1 is a typical example of a two-way table obtained by classifying 400 skin cancer patients according to the type of cell abnormality and the location on the body of the cancer. The numbers in the table represent the numbers of patients having the various combinations of type of abnormality and location, so that for example, 22 out of the 400 patients had Hutchinson's melanomic freckle on the head or neck.

The first step is to type the data into a vector:

```
> cancer.data <- c(22,16,19,11,2,54,33,17,10,115,73,28)
```

We then use the R function `matrix` to construct the matrix, with arguments `cancer.data`, 3 and 4 to specify the number of rows and columns. We type

```
> cancer.matrix <- matrix(cancer.data, nrow=4, ncol=3)
> cancer.matrix
      [,1] [,2] [,3]
[1,]   22   2  10
[2,]   16  54 115
[3,]   19  33  73
[4,]   11  17  28
```

Table A.1: Skin cancer patients classified according to type and location of cancer.

Type	Location		
	Head and Neck	Trunk	Extremities
Hutchinson's melanomic freckle	22	2	10
Superficial melanoma	16	54	115
Nodular	19	33	73
Indeterminate	11	17	28

Note that when we typed in the elements of the vector, we typed the first column of the table, then the second, and so on. What if we had typed in the table row by row into a vector `cancer.byrows` say? We need to add an extra argument to the matrix function to cope with this. We now must type

```
> cancer.matrix <- matrix(cancer.byrows,nrow=4,ncol=3,byrow=TRUE)
```

to get the same result. Note that the function `matrix` takes more than one argument, and that these are expressed in an unfamiliar way, using equal signs. We will explain this below in Section A.8.1.

We can also create a matrix by pasting together rows and columns: suppose we had typed the rows of the cancer table into four separate vectors:

```
> row1 <- c(22,2,10)
> row2 <- c(16,54,115)
> row3 <- c(19,33,73)
> row4 <- c(11,17,28)
```

We can then use the R function `rbind` to join the rows together:

```
> cancer.matrix <- rbind(row1,row2,row3,row4)
> cancer.matrix
      [,1] [,2] [,3]
row1    22     2    10
row2    16    54   115
row3    19    33    73
row4    11    17    28
```

The function `cbind` does the same thing for columns.

Sometimes it is useful to label the rows and columns of a matrix. To assign labels, we use the functions `rownames` and `colnames`:

```
> rownames(cancer.matrix) <- c("Hutchinson's freckle",
+ "Superficial melanoma", "Nodular","Indeterminate")
> colnames(cancer.matrix) <- c("Head/neck","Trunk","Extremities")
> cancer.matrix
```

	Head/neck	Trunk	Extremities
Hutchinson's freckle	22	2	10
Superficial melanoma	16	54	115
Nodular	19	33	73
Indeterminate	11	17	28

**Matrix arithmetic**

Arithmetic operations on matrices are done elementwise. We illustrate on two small matrices **a** and **b**:

```
> a
      [,1] [,2]
row1    1    6
row2    3    8

> b
      col1 col2
[1,]    1    3
[2,]    2    4
> a*b
      [,1] [,2]
row1    1   18
row2    6   32
```

Arithmetic functions are applied element by element:

```
> log(a)
      [,1] [,2]
row1 0.000000 1.791759
row2 1.098612 2.079442
```

We can also perform matrix multiplications:

```
> a %*% b
      col1 col2
Row 1    13    27
Row 2    19    41
```

Note how the result has inherited the row labels from **a** and the column labels from **b**.

**Extracting submatrices**

We can pick out parts of a matrix by specifying a subset of rows and columns, just as for vectors. We can extract single elements, rows, columns or whole matrices:

```
> a[1,1]
[1] 1
> a[1,]
[1] 1 6
> a[,1]
      row1 row2
      1    3

> D
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> D[1:2,2:3]
      [,1] [,2]
[1,]    4    7
[2,]    5    8
```

**A.2.3 Lists**

A *list* is simply an ordered list of objects, which may be vectors or matrices (or lists!). Lists are more general than either vectors or matrices. All the elements of a vector or matrix must be of the same type, numbers or character strings, but the elements of a list need not be of the same type.

This feature is particularly useful when designing R functions. Functions in R can return only a single object, but often we want a function to produce a variety of information. For example, a function to do regression might produce regression coefficients and a vector of residuals. The solution is to have functions return a list, which can contain the desired information.

To be specific, suppose we have a function `reg` that takes two vectors `BW` and `BPD` say, and calculates a least squares line, returning the intercept, slope and residuals. The usual method is to have the function return these three separate objects as a list:

```
> reg.list <- reg(BPD,BW)
> reg.list
$intercept:
[1] -1903.000

$slope:
[1] 43.06008

$resids:
[1] -62.62577 -127.92617 197.59320 152.07383 19.83351
[6] 59.69598 -241.80601 -113.22657 81.25407 35.13391
```

We can extract the separate objects from the list using the special operator `$`:

```
> reg.list$intercept
[1] -1903.000
> reg.list$slope
[1] 43.06008
> reg.list$resids
[1] -62.62577 -127.92617 197.59320 152.07383 19.83351
[6] 59.69598 -241.80601 -113.22657 81.25407 35.13391
```

The names `intercept`, `slope` and `resids` are specific to the function `reg` and we have to know what they are in order to be able to extract the components of the list `reg.list`. See the on-line help for the names appropriate to each function. Alternatively, we can just type the name of the list to see the names of the components, as we did above.

## A.2.4 Data frames

Typically data matrices contain both categorical and numeric variables, and we often want to represent the value of categorical variables with character strings rather than numbers. The restriction that R matrices must have all elements of the same mode means that we can't use R matrices to represent data matrices containing both numeric and character values. For this reason, R has another data type, the **data frame**.

These are simply data matrices, whose columns may be of different modes, and which include case labels and variable names. The data frame we made in Section 1.2.3 looks like this:

```
> ultra
  BPD AC BW
1  77 248 1350
2  82 253 1500
3  90 265 2170
4  82 265 1780
5  86 260 1820
6  84 252 1740
7  80 230 1300
8  87 250 1730
9  79 242 1580
10 81 263 1620
```

The column labels are just the names of the variables. Usually, we can create data frames by reading in data from an external file, and this is described below in the section *Getting data into R*. Alternatively, we can create a data frame by joining together vectors representing the columns

of the data matrix. For example, suppose we have the vectors `v1`, `v2` and `v3` containing the columns of the dental data matrix:

```
> v1
[1] 37 23 45 61 33 29 57 41
> v2
[1] "m" "m" "f" "m" "f" "f" "m" "f"
> v3
[1] "Y" "Y" "Y" "N" "Y" "N" "N" "Y"
```

We can create a data frame `dental.df` using the `data.frame` function:

```
> dental.df<-data.frame(v1,v2,v3)
> dental.df
  v1 v2 v3
1 37  m  Y
2 23  m  Y
3 45  f  Y
4 61  m  N
5 33  f  Y
6 29  f  N
7 57  m  N
8 41  f  Y
```

Note that all the vectors must have the same length, but do not have to be of the same mode.

In this example the variable names are just the names of the vectors used to construct the data frame. More meaningful variable names can be assigned:

```
> names(dental.df)<-c("Age","Sex","Attitude")
> dental.df
  Age Sex Attitude
1  37  m          Y
2  23  m          Y
3  45  f          Y
4  61  m          N
5  33  f          Y
6  29  f          N
7  57  m          N
8  41  f          Y
```

These can also be assigned when we create the data frame. We could have typed

```
> dental.df <- data.frame(Age=v1,Sex=v2,Attitude=v3)
```

for the same result.

We need to draw a distinction between the columns of the data frame (which are referred to by their variable names), and the original vectors which we used to create the data frame. If we try to refer to `Age` as the name of an R object, we get an error:

```
> Age
Error: Object "Age" not found
```

However, it is often very convenient to be able to refer to the columns of the data frame as if they were vectors. We can do this if we “attach” the data frame:

```
> attach(dental.df)
> Age
[1] 37 23 45 61 33 29 57 41
```

We can then use `Age` as if it were an ordinary vector. Alternatively, refer to `Age` directly by typing

```
> Dental.df$Age
[1] 37 23 45 61 33 29 57 41
```

To assign case labels, we can use the `row.names` function. In the dental study, a useful case label might be an ID code for each respondent. Suppose these are in a character vector `respondentID`.

```
> respondentID
[1] "2639" "4243" "9450" "1298" "382"  "9084" "5647" "2897"
```

Then we can assign the case labels by typing

```
> row.names(dental.df) <- respondentID
> dental.df
      Age Sex Attitude
2639  37   m         Y
4243  23   m         Y
9450  45   f         Y
1298  61   m         N
 382  33   f         Y
9084  29   f         N
5647  57   m         N
2897  41   f         Y
```

### A.3 Getting data into R

We can create vectors using the function `c`:

```
> sample <- c(5,3,1,7,11)
> sample
[1] 5  3  1  7 11
```

Another method uses the `scan` function. To create a vector `v` containing the numbers 5, 3, 1, 7, 11, we type

```
> v<-scan()
```

We then type in the elements 5, 3, 1, 7, 11 of the vector `v`, separating the elements by white space (blanks or tabs, but not commas!). A blank line tells R that we are done.

Often we have a data matrix in an external disk file. How can we get it into R? We use the function `read.table`, to create a data frame, `dental.df` say. We type

```
> dental.df<-read.table("dental.dat")
> dental.df
  V1 V2 V3
1 37  m  Y
2 23  m  Y
3 45  f  Y
4 61  m  N
5 33  f  Y
6 29  f  N
7 57  m  N
8 41  f  Y
```

The default variable names and case labels shown here are automatically assigned. To assign our own case labels and variable names, we can use a method similar to that used in the `data.frame` function:

```
> dental.df<-read.table("dental.dat",col.names=c("Age",
+ "Sex","Attitude"),row.names=respondentID)
> dental.df
      Age Sex Attitude
2639  37   m         Y
4243  23   m         Y
9450  45   f         Y
```

1298	61	m	N
382	33	f	Y
9084	29	f	N
5647	57	m	N
2897	41	f	Y

Another method is to include the case labels and the variable names as the first column and row of the external file. Thus, if the file `dental.dat` is

	Age	Sex	Attitude
2639	37	m	Y
4243	23	m	Y
9450	45	f	Y
1298	61	m	N
382	33	f	Y
9084	29	f	N
5647	57	m	N
2897	41	f	Y

typing

```
> dental.df<-read.table("dental.dat")
```

produces the same result:

```
> dental.df
      Age Sex Attitude
2639  37  m         Y
4243  23  m         Y
9450  45  f         Y
1298  61  m         N
 382   33  f         Y
9084  29  f         N
5647  57  m         N
2897  41  f         Y
```

If there are no case labels in the external file, you have to type

```
> dental.df<-read.table("dental.dat", header=TRUE)
```

## A.4 Keeping track of data objects

All R data objects are stored in the computers memory, and disappear when you quit R unless they are saved. This is explained later in the “Saving the results of the R session” section.

We can see which objects are stored by using the functions `objects` or `ls`:

```
> objects()
[1] "BPD"      "BW"      "reg.names"
> ls()
[1] "BPD"      "BW"      "reg.names"
```

To delete objects, use the function `rm` (remove):

```
> rm(BPD,BW)
> ls()
[1] "reg.names"
```

## A.5 Editing data

R has a special spreadsheet-like data editor that will allow us to edit data frames, vectors or matrices, or create new ones. It is invoked by using the function `edit`. Details of its use may be found by using the on-line help: just type

```
> ?edit
```

and inspect the resulting display. We can also change elements by using assignment. Suppose we want to change the third element of the vector `BPD` from 90 to 95:

```
> BPD[3]
Peter
  90
> BPD[3] <- 95
> BPD[3]
Peter
  95
```

However, this method is not very convenient for more than one or two changes; using `edit` is more convenient.

## A.6 Getting online help

If you forget how to use a particular function, you can find useful information by typing `?` followed by the name of the function. Thus, to get information on how to use the function `plot`, we simply type

```
> ?plot
```

R will respond by displaying a brief description of the function and the optional arguments available and their use. R also provides a full on-line help system. If this window is not already open, you can open the help window from within your R session by pulling down the help menu and selecting “html help”. In these notes, we do not always give full details of the arguments of functions, and of the objects they return. Use the on-line help to find this information.

## A.7 Producing graphic images

One of the strengths of R is its wide range of graphics functions, which allow you to create a variety of statistical graphics. Graphics will appear in a separate “graphics window” on the computer screen. This window automatically appears when a plot command is issued.

### A.7.1 Drawing plots

Suppose we have made two R vectors `BPD` and `BW`, each having ten elements, which hold the values for biparietal diameter and birth weight for the 10 infants in Table 1.1. To explore the relationship between these variables, we can plot `BW` versus `BPD`<sup>1</sup>. Typing

```
> plot(BPD,BW)
```

produces the desired plot. Note that the plot function returns no value. Rather, it causes an action to be taken (drawing the plot).

We can enhance the plot in various ways, such as adding a title, and labelling the axes:

```
> plot(BPD,BW, xlab="bi-parietal diameter",ylab="birthweight",
+ main="Relationship between BPD and BW")
```

---

<sup>1</sup>In this book we follow the convention that “plotting  $y$  versus  $x$ ” means forming a scatter plot by plotting  $y$  up the vertical axis and  $x$  along the horizontal axis, i.e. plotting the points  $(x_i, y_i)$ ,  $i = 1, \dots, n$ .



This is in fact how Figure 1.2 in Chapter 1 was drawn. Note how we specified the title using the argument `main`.

There are a huge set of high level R functions for producing more complicated graphs, such as boxplots, stem and leaf plots and contour plots. Some of are discussed in the text. Also there are a large number of functions allowing the user to customise a particular plot. We can take our simple scatterplot,

```
> plot(BPD,BW, xlab="bi-parietal diameter",ylab="birthweight",
+      pch="+")
```

where the points are now plotted with the symbol `+`, and add a title and annotation

```
> title("Figure 1",sub="Relationship between birthweight and biparietal diameter")
```

We can add a fitted line

```
> abline(lsfrit(BPD,BW))
```

The graph can be further annotated with superimposed points, lines, text, and arrows.

```
> text(mean(BPD),mean(BW),adj=0,"Fitted line must pass through mean of the data")
```

The function `text` has arguments giving the x- and y- coordinates of the text - a string of characters.

Often you will want to plot more than one graph in a figure, or make the plotting region square. This is achieved by changing the default parameters of the `plot` command by using the command `par`. Thus to create an array of graphs in the graphics window arranged in a 2 by 2 array use the command

```
> par(mfrow=c(2,2),pty="s")
```

The first use of `plot` will produce in the upper left-hand corner of the graphics window. The three subsequent plots will appear in the upper right-hand, lower left-hand and lower right-hand corners. If you only wish to produce the first three graphs, print the graphics window after producing your third graph and then reinitialise the window with the command

```
> par(mfrow=c(2,2))
```

The default state of the window, with one rectangular graph, can be produced by typing

```
> par(mfrow=c(1,1),pty="m")
```

## A.8 More on functions

We have already briefly discussed functions at the beginning of Section A.1. Now we supply some more details.

### A.8.1 Arguments

Sometimes we don't want to provide all the arguments to a function. For example, the function `matrix` has four arguments: `x`, a vector containing the matrix elements, `nrow`, the number of rows, `ncol`, the number of columns, and `byrow`, which is set to `TRUE` if the matrix is to be filled up by rows rather than columns. Most of the time we don't need to supply values for all of the arguments. For example, the function is smart enough to figure out that if `x` is a vector of four elements, and `nrow` is 2, then `ncol` must be 2 also. Also, the default value of `byrow` is `FALSE`, so we only need to specify this argument if we want to fill up the matrix by rows.

However, if we don't specify all the arguments, we need some way of associating the correct values with the arguments, since order alone is now insufficient. If we type

```
> matrix(x,3)
```

R assumes that 3 is the value for `nrow`, since `nrow` is the second argument. What if we really want to set `ncol` to 3? The secret is to use the name of the argument: If we type

```
> matrix(x,ncol=3)
```

R knows that we want 3 columns.

## A.8.2 Writing functions

The functions available in R cover a very wide range of numerical, statistical and graphical procedures. If these are not enough, you can create your own! For example, suppose we want to define a new function `square` which takes the value of the argument and squares it. We use the special R function `function` to make new functions. We type

```
> square <- function(x){ x^2}
```

The result is a function called `square` that squares a number:

```
> square(3)
[1] 9
```

The function `function` assigns the definition of “squaring” to the “function object” `square` which is stored in memory. We can inspect the names of our own defined functions (just as we can the names of all our data objects) using the `ls` function described above in Section A.4.

We can inspect the function definition by typing the function’s name:

```
> square
function(x){ x^2}
```

The formal argument `x` in the parentheses in the function definition is not an R object, but merely serves to define the function. The R expression defining the function is enclosed in braces `{}`. When we use the function by typing for example

```
> zsqared <- square(z)
```

the expression `x^2` in the function definition is evaluated using the current value of `z` instead of `x`. The value of the expression (in this case the square of the value of `z`) is then assigned to the object `zsqared`. We call `z` the *actual* argument, used when the function is used, and `x` the *formal* argument, used when the function is defined. No matter what the function, the value of the actual argument will not be changed during execution of the function.

Here is another example of a function, which this time expects to have two arguments, a vector and a single number, and illustrates that a function definition may have more than one line. In this case, the last line of the function should be an expression, and the value of this expression is the value returned by the function. Suppose we want to calculate the moments about the mean of a set of data. If the data is  $x_1, \dots, x_n$ , the  $r$ th moment about the mean  $\bar{x}$  is given by the expression

$$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^r.$$

If we store the data  $x_1, \dots, x_n$  as the elements of an R vector `x`, one way to write the function is as follows. (The comments on each line after the symbol `#` are ignored by the computer.)

```
> moments <- function(x,r)
+ {
+ # this function calculates the rth moment
+ # about the mean. x is a vector containing
+ # the data. r is an integer indicating the
+ # moment required.
+
+   d <- (x-mean(x))^r
+ }
```

```

+ # The vector d contains deviations from the mean,
+ # each raised to the rth power.
+ # Now use the R function "mean" to find the mean of
+ # the elements of d. The value of
+ # this last expression is the one returned by
+ # the function "moments".
+
+   mean(d)
+ }
>

```

Several points are worth making. First, note how the usual prompt `>` changes to a `+` if the line is logically incomplete. In the function above the lines are incomplete because of the unmatched braces. After the closing brace, the prompt returns to `>`. Second, the assignments in the function definition create a *local variables*, `d`, which has a temporary life while the function is executing, and then disappears. The function of `d` is purely to make the function easier to understand. The function could have been written more compactly without local variables as

```

> moments <- function(x,r)
+ {
+ # this function calculates the rth moment about the mean
+ # x is a vector containing the data
+ # r is an integer indicating the moment required
+   mean((x-mean(x))^r)
+ }
>

```

Use of local variables is sometimes desirable for clarity. Comments also allow the function definition to be more easily understood. Finally, note the use of the R - supplied function `mean` to calculate the mean of the elements.

Suppose most of the time we want to use `moments` to calculate variances (i.e. second moments, where  $r = 2$ .) We can modify the function by supplying a default value for the second argument, so that typing the second argument is unnecessary if we want to calculate a variance. We replace the first line of the function definition by

```

> moments <- function(x,r=2)

```

Then to calculate a variance, we need only type

```

> moments(x)

```

For a third moment, we type

```

> moments(x, 3)

```

or

```

> moments(x, r=3)

```

### Typing in functions

We saw above how simple functions could be defined using the function `function` and typing in the definition directly at the terminal.

Typing in a long definition directly is not very satisfactory, since a typing error can cause the whole definition to become garbled. A better solution is to use text editor to prepare a file containing the text of the definition. We can use `notepad` to write our function and save the function definition in a text file. The file can be incorporated into R in several ways. If the file is called `function.text` say, then we can type

```

> source("function.text")

```

from within R. The `source` function tells R to stop looking for input from the keyboard, and start reading the function definition from the file `function.text`. When all the file has been read, input is again accepted from the keyboard. NOTE: the file must have its last line ended with a `return` character for R to recognise this as the end of the file.

A simpler alternative is simply to use the editor as a scratch pad to write the function, then cut and paste the definition into R. Functions can also be edited with the `edit` function.

## A.9 Saving the results of the R session

When you quit out of an R session (using the `q` command, the menu or by closing the window you will be prompted whether you want to save your session or not:

Save workspace image?

Clicking the “yes” button will save a copy of R containing all the objects that you have created, i.e. vectors, matrices, data frames etc. The next time you start R you can load this back in. Clicking “no” will mean that all objects you have created in this session will not be saved. Clicking “cancel” will cancel.

## A.10 R packages

One of the main strengths of R is that it is simple to add new functions and data to the basic structure. You can do this locally, by defining new functions as we described above. But what gives R its power is that so many functions written by many other people are available on-line through CRAN (the Comprehensive R Archive Network). Any user of R can bundle up a set of functions and data sets in the form of an “R package” and submit it to the network. Provided it passes certain tests, it is then made available to all R users.

You will already be familiar with the “s20x” package used in STATS 201 and STATS 208. The “R330” package is the equivalent for STATS 330. To download (install) packages, you select “Install package(s)” from the R “Packages” menu. This brings up a dialog asking you to select a “CRAN mirror”. This allows you to select the CRAN site of your choice (there is a site in most countries, including New Zealand). Select a country and click OK. A list of available packages appears. Select the ones you want (You can make multiple selections by holding down the Ctrl key as you click). For the R330 package, you will need to install the packages “s20x”, “leaps” and “rgl” as well as R330. Then, to make the packages available for use, type

```
> library(R330)
```

The R330 package includes many of the data sets used in this workbook. To access the data (say the data set `fatty.df`) you must type

```
> data(fatty.df)
```

You can then refer to this data frame in subsequent R code.

## A.11 How to get more information on R

The above material is not a complete reference to R. For more detail about particular functions, use the on-line help feature of R described in Section A.6.

There are many books describing the R language, some of which are listed in the bibliography. You might try looking at Paul Murrell’s book *An Introduction to Data Technologies*: in his Chapter 10 he gives a good introduction to the non-statistical parts of R. You might also look at *A Beginner’s Guide to R* by Zuur, Ieno and Meesters. Both these books are available on line from the University Library. *R in a Nutshell* by Adler is also good (available in the Library but not on line.) Additionally, there is an on-line guide available at

{[cran.r-project.org/doc/contrib/Paradis-rdebuts\\_en.pdf](http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf)}

## Appendix B

# R330 Documentation

---

`acc.df`

*Data from the Auckland City Council*

---

### Description

Data from the Auckland City Council where the aim was to predict the capital value from the rental value

### Usage

```
data(acc.df)
```

### Format

A data frame with observations on 96 properties having variables

`capital` the capital value of the property

`rental` the rental value of the property

### Examples

```
data(acc.df)
acc.lm<-lm(capital~rental,data=acc.df)
```

---

`acet.df`

*Acetylene Data*

---

### Description

Percentage conversions of n-heptane to acetylene

### Usage

```
data(acet.df)
```

**Format**

A data frame with 16 observations on the following 4 variables:

**percent.conv** percentage conversions of n-heptane to acetylene

**temp** temperature in degrees centergrade

**ratio** ratio of H2 to n-heptane

**contact** contact time in seconds

**Source**

DW Marquardt and RD Snee (1975). Ridge regression in practice. American Statistian, 28, 3-20

**Examples**

```
data(acet.df)
summary(lm(percent.conv~temp+ratio+contact, data=acet.df))
```

---

ad.df	<i>Advertising data</i>
-------	-------------------------

---

**Description**

A data set which looks at the relationship between the sales and the expenditure on sales over 36 months

**Usage**

```
data(ad.df)
```

**Format**

A data frame with 35 observations on the following 3 variables:

**sales** monthly sales

**spend** amount spent on advertising this month

**prev.spend** amount spent on advertising in the previous month

**Details**

We lose one observation when prev.spend was created

**Examples**

```
data(ad.df)
advert.lm<-lm(sales~spend+prev.spend,data=ad.df)
```

---

`added.variable.plots` *Draws an added variable plot for each independent variable*

---

## Description

Draws an added variable plot for each independent variable

## Usage

```
added.variable.plots(f, intercept = TRUE,...)
S3 method for class "lm"
added.variable.plots(f, intercept = TRUE,...)
## S3 method for class 'formula'
added.variable.plots(f, intercept = TRUE, data, subset, weights,
  na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE,
  qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)
```

## Arguments

<code>f</code>	An lm object or model formula
<code>intercept</code>	If TRUE (default) an intercept is fitted
<code>data</code>	A data frame, list or environment containing the variables in the model.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>weights</code>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of options, and is <code>na.fail</code> if that is unset. The ‘factory-fresh’ default is <code>na.omit</code> . Another possible value is NULL, no action. Value <code>na.exclude</code> can be useful.
<code>method</code>	the method to be used in fitting the model. The default method “ <code>glm.fit</code> ” uses iteratively reweighted least squares (IWLS): the alternative “ <code>model.frame</code> ” returns the model frame and does no fitting.
<code>x, y, qr, model</code>	For <code>glm</code> : logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For <code>glm.fit</code> : <code>x</code> is a design matrix of dimension $n * p$ , and <code>y</code> is a vector of observations of length <code>n</code> .
<code>singular.ok</code>	logical. If FALSE (the default in S but not in R) a singular fit is an error.
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>offset</code>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See <code>model.offset</code> .
<code>...</code>	additional arguments to be passed to the low level regression fitting functions see <code>lm</code> and <code>glm</code> help files

## Value

Returns no value but draws an added variable plot for each variable.

**Note**

This function redirects to other functions based on the type of object. eg `added.variable.plots.lm`, `added.variable.plots.formula`

**Author(s)**

Alan Lee, Blair Robertson

**Examples**

```
data(rubber.df)
rubber.lm<-lm(abloss~hardness+tensile,data=rubber.df)
par(mfrow=c(1,2))
added.variable.plots(rubber.lm)
```

---

allpossregs	<i>Calculates all possible regressions</i>
-------------	--

---

**Description**

Calculates all possible regressions for subset selection

**Usage**

```
allpossregs(f, best = 1, Cp.plot = TRUE, text.cex = 0.8, dp = 3,
            cv.rep = 50, nvmax = 20, ...)
## S3 method for class 'lm'
allpossregs(f, best = 1, Cp.plot = TRUE, text.cex = 0.8, dp = 3,
            cv.rep = 50, nvmax = 20, ...)
## S3 method for class 'formula'
allpossregs(f, best = 1, Cp.plot = TRUE, text.cex = 0.8, dp = 3,
            cv.rep = 50, nvmax = 20, data, subset, weights, na.action,
            method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
            singular.ok = TRUE, contrasts = NULL, offset, ...)
```

**Arguments**

<b>f</b>	an lm object or model formula
<b>best</b>	the number of models for each size (size=number of variables) to be printed
<b>Cp.plot</b>	print Cp plot? (TRUE=yes, FALSE=no)
<b>text.cex</b>	expansion factor for plot text
<b>dp</b>	number of decimal places
<b>cv.rep</b>	The number of random samplings when calculating the CV estimate of prediction error
<b>nvmax</b>	The maximum number of variables to be included in models.
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>weights</b>	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.



<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
<b>method</b>	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS): the alternative "model.frame" returns the model frame and does no fitting.
<b>x, y, qr, model</b>	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
<b>singular.ok</b>	logical. If FALSE (the default in S but not in R) a singular fit is an error.
<b>contrasts</b>	an optional list. See the contrasts.arg of model.matrix.default.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
<b>...</b>	additional arguments to be passed to the low level regression fitting functions see lm and glm help files

## Value

A matrix with columns labeled:

<b>rssp</b>	Residual Sum of Squares
<b>sigma2</b>	low values indicate better model
<b>adjRsqr</b>	adjusted R squared for the model. Big values indicate good model
<b>Cp</b>	Mallow's Cp measure of how well model predicts. Want small values
<b>AIC</b>	Akaike Information Criterion, estimate of the difference between the fitted model and actual model. Want small values
<b>BIC</b>	Bayesian Information Criterion, estimate of the posterior probability that fitted model is correct one. Want small values
<b>CV</b>	Cross-validation. Small values indicate good model.
<b>Variables</b>	states which variables were included for the regression (val=1 means included)

Rows represent the number of variables in the model

## Note

This function redirects to other functions based on the type of object. eg all.poss.regs.lm , all.poss.regs.formula

## Author(s)

Alan Lee, Blair Robertson

## Examples

```
data(fatty.df)
allpossregs(ffa ~ age + skinfold + weight, data = fatty.df, Cp.plot=TRUE)
```

---

births.df

*Risk factors for low birthweight*


---

## Description

Data were collected at Baystate Medical Center, Springfield, Mass. during 1986, as part of a study to identify risk factors for low-birthweight babies.

## Usage

```
data(births.df)
```

## Format

A data frame with 189 observations on the following 11 variables:

**id** Identificatin code

**low** low birthweight (defined as less than 2500g) 0 = No, 1 = Yes

**age** Age of mother (years)

**lwt** weight of mother at last menstrual period (pounds)

**race** Race (1 = white, 2 = black, 3 = other)

**smoke** smoking status during pregnancy (0 = No, 1 = Yes)

**ptl** History of premature labour (0 = None, 1 = one, 2 = two, etc.)

**ht** History of hypertension (0 = No, 1 = Yes)

**ui** Presence of Uterine Irritability (0 = No, 1 = Yes)

**ftv** Number of Physician Visits during the first trimester ( 0 = none, 1 = one, 2 = two, etc.)

**bwt** Birth weight (grams) (response)

## Source

Hosmer & Lemeshow, Applied Logistic Regression. pp 25-26.

## References

Hosmer, D.W. & Lemeshow, S.(2000), Applied Logistic Regression (2nd edition), John Wiley & Sons, New York.

## Examples

```
data(births.df)
births.lm<-lm(bwt~age*race*smoke*ui*ht+lwt*race*smoke*ui*ht,data=births.df)
anova(births.lm)
```

boxcoxplot

*Draws a Box-Cox plot*

## Description

Draws a plot of the Box-Cox profile likelihood.

## Usage

```
boxcoxplot(f, p = seq(-2, 2, length = 20), ...)
## S3 method for class 'lm'
boxcoxplot(f, p = seq(-2, 2, length = 20), ...)
## S3 method for class 'formula'
boxcoxplot(f, p = seq(-2, 2, length = 20), data, subset,
  weights, na.action, method = "qr", model = TRUE, x = FALSE,
  y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL,
  offset, ...)
```

## Arguments

<b>f</b>	an lm object or a model formula
<b>p</b>	a vector of powers, representing plotting positions along the horizontal axis
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>weights</b>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
<b>method</b>	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS); the alternative "model.frame" returns the model frame and does no fitting.
<b>x, y, qr, model</b>	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
<b>singular.ok</b>	logical. If FALSE (the default in S but not in R) a singular fit is an error.
<b>contrasts</b>	an optional list. See the contrasts.arg of model.matrix.default.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
<b>...</b>	graphical arguments passed to plot function

**Details**

The function draws a graph of the negative of the profile likelihood as a function of the transformation power  $p$ . The regression coefficients and the standard deviation of the errors have been profiled out. The indicated power is the value of  $p$  for which the function attains its minimum. It may on rare occasions be necessary to adjust the range of  $p$  (default is (2,2)).

**Value**

Returns no value but draws a plot of the Box-Cox profile likelihood.

**Note**

This function redirects to other functions based on the type of object. eg `boxcoxplot.formula`, `boxcoxplot.lm`

**Author(s)**

Alan Lee, Blair Robertson

**References**

Box, GEP and Cox, DR. (1964). An analysis of transformations. Journal of the Royal Statistical Society, Series B 26 (2): pp 211-252

**Examples**

```
data(educ.df)
boxcoxplot(educ~urban + percap + under18, data=educ.df[-50,])

data(wine.df)
boxcoxplot(price~year+temp+h.rain+w.rain,data=wine.df)
```

---

budworm.df

*Budworm data*

---

**Description**

Data from a small experiment on the toxicity to the tobacco budworm

**Usage**

```
data(budworm.df)
```

**Format**

A data frame with 12 observations on the following 4 variables:

**sex** the sex of the budworm  
**dose** amount of cypermethrin exposed to  
**r** number of budworms affected  
**n** total number of budworms

**Details**

The data come from an experiment on the toxicity to the tobacco budworm *Heliothis virescens* of doses of the pyrethroid trans-cypermethrin to which the moths were beginning to show resistance. Batches of 20 moths of each sex were exposed for three days to the pyrethroid and the number in each batch that were dead or knocked down was recorded.

## Source

Collette, D. (1991) Modelling Binary Data. Chapman and Hall, London. p 75

## References

Venables, W.N and Ripley, B. (2002) Modern Applied Statistics with S, Springer, New York.

## Examples

```
data(budworm.df)
bugs.glm<-glm(r/n~sex+dose,family=binomial,weight=n,data=budworm.df)
summary(bugs.glm)
```

---

cancer.df

*Death Rates for Child Cancer*

---

## Description

Data from study to investigate death rates for child cancer in the UK from 1951-1960 in Northumberland and Durham

## Usage

```
data(cancer.df)
```

## Format

A data frame with 8 observations on the following 5 variables:

**Cytology** a factor with levels L M

**Residence** type of residence either rural or urban

**Age** the age of the child classified as either 0-4 years or 5-14 years

**n** number of deaths

**pop** population

## Examples

```
data(cancer.df)
cancer.glm<-glm(n ~ Cytology*Residence*Age, family=poisson,
offset=log(pop/100000), data=cancer.df)
anova(cancer.glm, test="Chisq")
```

---

chd.df	<i>Coronary heart disease data</i>
--------	------------------------------------

---

**Description**

Shows the age of the subject and presence or absence of evidence of significant coronary heart disease.

**Usage**

```
data(chd.df)
```

**Format**

A data frame with 100 observations on the following 2 variables:

**age** age of subject in years

**chd** 0 indicates CHD absent, 1 indicates it is present

**Source**

Hosmer, and Lemeshow Applied Logistic Regression, pp 2-5

**References**

Hosmer, D. W., and Lemeshow, S. (2000). Applied Logistic Regression, Second edition, Wiley, New York.

**Examples**

```
data(chd.df)
chd.glm<-glm(chd~age,family=binomial,data=chd.df)
summary(chd.glm)
```

---

chem.df	<i>Yield of Chemical Process</i>
---------	----------------------------------

---

**Description**

The data was collected to see if how the yields from a particular chemical process were associated with higher or lower flows and conversion percentages

**Usage**

```
data(chem.df)
```

**Format**

A data frame with 44 observations on the following 4 variables:

**yield** yield

**conversion** conversion as a percentage

**flow** flow

**ratio** ratio

**Examples**

```
data(chem.df)
chem.lm<-lm(yield~.,data=chem.df)
summary(chem.lm)
```

---

cherry.df

*Girth, Height and Volume for Black Cherry Trees*


---

**Description**

This data set provides measurements of the girth, height and volume of timber in 31 felled black cherry trees. Note that girth is the diameter of the tree (in inches) measured at 4 ft 6 in above the ground.

**Usage**

```
data(cherry.df)
```

**Format**

A data frame with 31 observations on the following 3 variables:

**diameter** Tree diameter in inches

**height** Height in ft

**volume** Volume of timber in cubic ft

**Source**

Ryan, T. A., Joiner, B. L. and Ryan, B. F. (1976) The Minitab Student Handbook. Duxbury Press.

**References**

Atkinson, A. C. (1985) Plots, Transformations and Regression. Oxford University Press.

**Examples**

```
data(cherry.df)
cherry.lm =lm(volume~diameter+height,data=cherry.df)
new.df = data.frame(diameter=c(11,12),
                    height=c(85,90))
predict(cherry.lm,new.df)
```

---

chickwts.df

*Chicken Weights Data*


---

### Description

An experiment comparing 12 methods of feeding chickens was done independently in two replicates arranged in different houses

### Usage

```
data(chickwts.df)
```

### Format

A data frame with 24 observations on the following 4 variables:

**chickweight** weight gain

**protein** form of protien, either groundnut or soyabean

**protlevel** level of protein either 0, 1 or 2

**fish** level of fish solubles, either high or low

### Source

John, J.A. and Quenouille, M.H. (1977). Experiments: Design and Analysis, 2nd edn. London: Griffin.

### References

Cox, D. R. & Snell, E. J. (1981). Applied Statistics: Principles and Examples. Chapman and Hall, London.

### Examples

```
data(chickwts.df)
model1<-lm(chickweight~protein*protlevel*fish, data=chickwts.df)
summary(model1)
```

---

coag.df

*Blood Coagulation Data*


---

### Description

Experiment designed to see the effect of four different diets on a subject's blood coagulation time

### Usage

```
data(coag.df)
```

### Format

A data frame with 24 observations on each of the two variables

**coag** Blood coagulation time

**diet** Which diet they were on either A, B, C or D



## Source

Box, G.E.P., Hunter, J.S. Hunter, W.G. Statistics for experimenters, pp 165-197.

## References

Box, G.E.P., Hunter, J.S. Hunter, W.G. (1978). Statistics for experimenters, Wiley, New York.

## Examples

```
data(coag.df)
coag.lm <- lm(coag ~ diet, data = coag.df)
anova(coag.lm)
```

---

`cross.val`

*Calculates cross-validated estimates of prediction error*

---

## Description

For a logistic model, calculates cross-validated estimates of specificity, sensitivity and percentage correctly classified. For a Gaussian model, calculates a cross-validated estimate of prediction error.

## Usage

```
cross.val(f, nfold = 10, nrep = 20, ...)
## S3 method for class 'lm'
cross.val(f, nfold = 10, nrep = 20, ...)
## S3 method for class 'glm'
cross.val(f, nfold = 10, nrep = 20, ...)
## S3 method for class 'formula'
cross.val(f, nfold = 10, nrep = 20, family = gaussian,
  data, weights, subset, na.action, start = NULL, etastart,
  mustart, offset, control = list(...), model = TRUE,
  method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)
```

## Arguments

<code>f</code>	an lm object, a glm object or a model formula
<code>nfold</code>	number of parts data set divided into
<code>nrep</code>	number of random splits
<code>family</code>	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
<code>data</code>	A data frame, list or environment containing the variables in the model.
<code>weights</code>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.

<b>start</b>	starting values for the parameters in the linear predictor.
<b>etastart</b>	starting values for the linear predictor.
<b>mustart</b>	starting values for the vector of means.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See <code>model.offset</code> .
<b>control</b>	a list of parameters for controlling the fitting process. For <code>glm.fit</code> this is passed to <code>glm.control</code> .
<b>model</b>	a logical value indicating whether model frame should be included as a component of the returned value.
<b>method</b>	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS); the alternative "model.frame" returns the model frame and does no fitting.
<b>x, y</b>	For <code>glm</code> : logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For <code>glm.fit</code> : <code>x</code> is a design matrix of dimension $n * p$ , and <code>y</code> is a vector of observations of length <code>n</code> .
<b>contrasts</b>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<b>...</b>	additional arguments to be passed to the low level regression fitting functions see <code>lm</code> and <code>glm</code> help files

## Details

The object returned depends on the class of the input.

## Value

For a logistic model:

**Mean Specificity**  
false negatives  
**Mean Sensitivity**  
false positives  
**Mean Correctly classified**  
proportion correctly classified

For a Gaussian model, an estimate of the root mean squared error is returned.

## Note

This function redirects to other functions based on the type of object. eg `cross.val.glm` , `cross.val.formula`

## Author(s)

Alan Lee, Blair Robertson

## References

Bradley Efron and Robert Tibshirani (1993). An Introduction to the Bootstrap. Chapman and Hall, London.

**Examples**

```
data(fatty.df)
fatty.lm <- lm(ffa~age+weight+skinfold, data=fatty.df)
cross.val(fatty.lm)
#
data(drug.df)
cross.val(DFREE ~ NDRUGTX + factor(IVHX) + AGE + TREAT, family=binomial,
          data=drug.df)
```

cycles.df

*Cycles to failuer of worsted yarm***Description**

Data which looked at the number of cycles to failure of lengths of worsted yarm under cycles of repeated loading

**Usage**

```
data(cycles.df)
```

**Format**

A data frame with 27 observations on the following 4 variables:

**yarn.length** factor which takes lengths 250, 300 or 350mm, which were classified as low, med or high respectively

**amplitude** the cycle amplitudes were taken to be 8, 9 or 10mm, which were classified as low, med or high respectively

**load** loads taken to be 40, 45 or 50 grams, which were classified as low, med or high respectively

**cycles** number of cycles to failure

**Source**

Cox and Snell, Applied Statistics: Principles and Examples, pp 98-102

**References**

Cox, D. R. & Snell, E. J. (1981). Applied Statistics: Principles and Examples. Chapman and Hall, London.

**Examples**

```
data(cycles.df)
library(lattice)
dotplot(cycles~yarn.length|amplitude*load,xlab="Yarn length",
        ylab="Cycles to failure",data=cycles.df,
        strip=function(...)strip.default(...,strip.names=TRUE))
```

---

`diets.df`*Weight gains of rats*

---

**Description**

Data from an experiment on the weight gain of rats based on the source of protein and the amount

**Usage**

```
data(diets.df)
```

**Format**

A data frame with 60 observations on the following 3 variables:

**gain** amount of weight gained

**source** source of protein either beef, pork or cereal

**level** amount of protein given, high or low

**Source**

Snedecor, G.W., and Cochran, W.G., (1989), Statistical Methods.

**Examples**

```
data(diets.df)
plot.design(diets.df)
```

---

`drug.df`*Drug addiction data*

---

**Description**

Data from University of Massachusetts AIDS Research Unit IMPACT study, a medical study performed in the US in the early 90' s. The study aimed to evaluate two different treatments for drug addiction.

**Usage**

```
data(drug.df)
```

**Format**

A data frame with 575 observations on the following 9 variables:

**ID** Identification Code

**AGE** Age at enrollment

**BECK** Beck depression score

**IVHX** IV drug use history at admission (1=never, 2=previous, 3=recent)

**NDRUGTX** number of prior treatments

**RACE** subjects race (0 = white, 1 = other)

**TREAT** treatment duration (0 = short, 1 = long)

**SITE** treatment site (0 = A, 1 = B)

**DFREE** Remained drug free (1 = Yes, 0 = No) (response)

**Source**

Hosmer and Lemeshow, Applied Logistic Regression (2nd Ed), p28

**References**

Hosmer, D.W. and Lemeshow, S. (2000), Applied Logistic Regression (2nd Ed), Wiley, New York.

**Examples**

```
data(drug.df)
cross.val(DFREE ~ NDRUGTX + factor(IVHX) + AGE + TREAT, data = drug.df)
```

---

educ.df	<i>Educations expenditure data</i>
---------	------------------------------------

---

**Description**

Data set from 50 US states on education expenditure

**Usage**

```
data(educ.df)
```

**Format**

A data frame with 50 observations on the following 4 variables:

**urban** number of residents per 1000 in urban areas  
**educ** per capita expenditure on education (response)  
**percap** per capita income  
**under18** number of residents per 1000 under 18

**Examples**

```
data(educ.df)
educ.lm = lm(educ~urban + percap + under18, data=educ.df)
summary(educ.lm)
```

---

err.boot	<i>Calculates a bootstrap estimate of prediction error</i>
----------	--

---

**Description**

Calculates the training set prediction error and a bootstrap estimate of prediction error for the model in specified by a formula, lm object or glm object

## Usage

```
err.boot(f, B=500, ...)
## S3 method for class 'lm'
err.boot(f, B=500, ...)
## S3 method for class 'glm'
err.boot(f, B=500, ...)
## S3 method for class 'formula'
err.boot(f, B=500, family = gaussian, data, weights,
        subset, na.action, start = NULL, etastart, mustart, offset,
        control = list(...), model = TRUE, method = "glm.fit", x = FALSE,
        y = TRUE, contrasts = NULL, ...)
```

## Arguments

<b>f</b>	An lm object, a glm object or a model formula
<b>B</b>	number of bootstrap replications
<b>family</b>	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>weights</b>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
<b>start</b>	starting values for the parameters in the linear predictor.
<b>etastart</b>	starting values for the linear predictor.
<b>mustart</b>	starting values for the vector of means.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
<b>control</b>	a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.
<b>model</b>	a logical value indicating whether model frame should be included as a component of the returned value.
<b>method</b>	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS); the alternative "model.frame" returns the model frame and does no fitting.
<b>x, y</b>	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.  For glm.fit: x is a design matrix of dimension $n * p$ , and y is a vector of observations of length n.
<b>contrasts</b>	an optional list. See the contrasts.arg of model.matrix.default.
<b>...</b>	additional arguments to be passed to the low level regression fitting functions see lm and glm help files

**Value**

<code>\$err</code>	Training set estimate
<code>\$Err</code>	Bootstrap estimate

**Author(s)**

Alan Lee, Blair Robertson

**Examples**

```
data(drug.df)
err.boot(DFREE ~ NDRUGTX + factor(IVHX) + AGE + TREAT, data= drug.df)
```

---

<code>ethanol.df</code>	<i>Engine exhaust fumes from burning ethanol</i>
-------------------------	--

---

**Description**

Ethanol fuel was burned in a single-cylinder engine. For various settings of the engine compression and equivalence ratio, the emissions of nitrogen oxides were recorded.

**Usage**

```
data(ethanol.df)
```

**Format**

A data frame with 88 observations on the following 3 variables:

**NOx** Concentration of nitrogen oxides, NO and NO2, in micrograms per Joule

**C** Compression ratio of the engine

**E** Equivalence ratio - a measure of the richness of the air and ethanol fuel mixture.

**Source**

Brinkman, N.D. (1981) Ethanol Fuel: A Single - Cylinder Engine Study of Efficiency and Exhaust Emissions. SAE transactions, 90, 1410 - 1424.

**References**

Cleveland, William S. (1993) Visualizing Data. Hobart Press, Summit, New Jersey.

**Examples**

```
data(ethanol.df)
pairs20x(ethanol.df)
```

---

`evap.df`*Moisture evaporation data*

---

**Description**

The purpose was to see if the amount of evaporation could be predicted by the temperature humidity and wind speed

**Usage**

```
data(evap.df)
```

**Format**

A data frame with 46 observations on the following 11 variables:

`avst` average soil temperature over 24 hour period (x10)  
`minst` minimum soil temperature over 24 hour period (x10)  
`maxst` maximum soil temperature over 24 hour period (x10)  
`avat` average air temperature over 24 hour period (x10)  
`minat` minimum air temperature over 24 hour period (x10)  
`maxat` maximum air temperature over 24 hour period (x10)  
`avh` average humidity over 24 hour period (x10)  
`minh` minimum humidity over 24 hour period (x10)  
`maxh` maximum humidity over 24 hour period (x10)  
`wind` average wind speed over a 24 hour period (x100)  
`evap` amount of evaporation over 24 hour period

**Examples**

```
data(evap.df)
evap.lm<-lm(evap~avat+avh+wind,data=evap.df)
summary(evap.lm)
```

---

`fatty.df`*Fatty acid data*

---

**Description**

Data was collected to use physical measures to model a biochemical parameter in overweight children

**Usage**

```
data(fatty.df)
```

**Format**

A data frame with 20 observations on the following 4 variables:

`ffa` free fatty acid level in blood (response)  
`age` age (months)  
`weight` weight (pounds)  
`skinfold` skinfold thickness (inches)



## Examples

```
data(fatty.df)
fatty.lm<-lm(ffa~age+weight+skinfold,data=fatty.df)
```

---

funnel	<i>Plots for checking for unequal variances</i>
--------	---

---

## Description

Plots for checking for unequal variances

## Usage

```
funnel(f,...)
## S3 method for class 'lm'
funnel(f,...)
## S3 method for class 'formula'
funnel(f, data, subset, weights, na.action, method = "qr",
model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
contrasts = NULL, offset, ...)
```

## Arguments

<b>f</b>	an lm object or a model formula
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>weights</b>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
<b>method</b>	the method to be used in fitting the model. The default method ”glm.fit” uses iteratively reweighted least squares (IWLS); the alternative ”model.frame” returns the model frame and does no fitting.
<b>x, y, qr, model</b>	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
<b>singular.ok</b>	logical. If FALSE (the default in S but not in R) a singular fit is an error.
<b>contrasts</b>	an optional list. See the contrasts.arg of model.matrix.default.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
<b>...</b>	additional arguments to be passed to the low level regression fitting functions see lm and glm help files

**Value**

Prints the slope of the line of best fit for log std.errors vs log means. Returns (invisibly) the estimated variances of the observations.

**Note**

This function redirects to other functions based on the type of object. eg `funnel.lm` , `funnel.formula`

**Author(s)**

Alan Lee, Blair Robertson

**Examples**

```
data(educ.df)
educ50.lm = lm(educ~urban + percap + under18, data=educ.df, subset=-50)
funnel(educ50.lm)
#
funnel(educ~urban + percap + under18, data=educ.df, subset=-50)
```

---

HLstat

*Performs a Hosmer-Lemeshow test*


---

**Description**

Calculates and prints a  $\chi^2$  statistic and a p-value for the Hosmer-Lemeshow test.

**Usage**

```
HLstat(f,...)
## S3 method for class 'glm'
HLstat(f,...)
## S3 method for class 'formula'
HLstat(f, family = binomial, data = data, weights, subset,
       na.action, start = NULL, etastart, mustart, offset,
       control = list(...), model = TRUE, method = "glm.fit",
       x = FALSE, y = TRUE, contrasts = NULL, ...)
```

**Arguments**

<b>f</b>	a glm object or model formula
<b>family</b>	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>weights</b>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.

<b>start</b>	starting values for the parameters in the linear predictor.
<b>etastart</b>	starting values for the linear predictor.
<b>mustart</b>	starting values for the vector of means.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See <code>model.offset</code> .
<b>control</b>	a list of parameters for controlling the fitting process. For <code>glm.fit</code> this is passed to <code>glm.control</code> .
<b>model</b>	a logical value indicating whether model frame should be included as a component of the returned value.
<b>method</b>	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS); the alternative "model.frame" returns the model frame and does no fitting.
<b>x, y</b>	For <code>glm</code> : logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.  For <code>glm.fit</code> : <code>x</code> is a design matrix of dimension $n * p$ , and <code>y</code> is a vector of observations of length <code>n</code> .
<b>contrasts</b>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<b>...</b>	additional arguments to be passed to the low level regression fitting functions see <code>lm</code> and <code>glm</code> help files

## Value

HL stat the value of the Hosmer-Lemeshow test P-value an approximate p-value for the test

## Note

This function redirects to other functions based on the type of object. eg `HLstat.glm` , `HLstat.formula`

## Author(s)

Alan Lee, Blair Robertson

## Source

Hosmer and Lemeshow, Applied Logistic Regression (2nd Ed), p 147.

## References

Hosmer, D.W. and Lemeshow, S. (2000), Applied Logistic Regression (2nd Ed), Wiley, New York.

## Examples

```
data(drug.df)
drug.glm<-glm(DFREE ~ . - IVHX - ID + factor(IVHX), family = binomial,
              data = drug.df)
HLstat(drug.glm)
```

---

housing.df	<i>Housing conditions satisfaction</i>
------------	--

---

## Description

A data set investigating the satisfaction with housing conditions in Copenhagen

## Usage

```
data(housing.df)
```

## Format

A data frame with 18 observations on the following 4 variables:

**sat** Satisfaction of householders with their present housing circumstances, (High, Medium or Low, ordered factor).

**infl** Perceived degree of influence householders have on the management of the property (High, Medium, Low).

**cont** Contact residents are afforded with other residents, (Low, High).

**count** number in each category

## Source

Madsen, M. (1976). Statistical analysis of multiple contingency tables. Two examples. Scand J. Statist., 3,97-106

## References

Cox, D. R. & Snell, E. J. (1981). Applied Statistics: Principles and Examples. Chapman and Hall, London.

## Examples

```
data(housing.df)
housing.glm<-glm(count~sat*infl*cont, family=poisson, data=housing.df)
anova(housing.glm, test="Chisq")
```

---

influenceplots	<i>Draws plots of influence measures</i>
----------------	--

---

## Description

Draws plots of influence measures based on the family.

## Usage

```
influenceplots(f, cex.lab=0.7, ...)
## S3 method for class 'lm'
influenceplots(f, cex.lab=0.7, ...)
## S3 method for class 'glm'
influenceplots(f, cex.lab=0.7, ...)
## S3 method for class 'formula'
influenceplots(f, cex.lab=0.7, family = gaussian, data,
              weights, subset, na.action, start = NULL, etastart, mustart,
              offset, control = list(...), model = TRUE, method = "glm.fit",
              x = FALSE, y = TRUE, contrasts = NULL, ...)
```

## Arguments

<b>f</b>	a lm object, a glm object or a model formula
<b>cex.lab</b>	An expansion factor for plot labels
<b>family</b>	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>weights</b>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
<b>start</b>	starting values for the parameters in the linear predictor.
<b>etastart</b>	starting values for the linear predictor.
<b>mustart</b>	starting values for the vector of means.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
<b>control</b>	a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.
<b>model</b>	a logical value indicating whether model frame should be included as a component of the returned value.
<b>method</b>	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS); the alternative "model.frame" returns the model frame and does no fitting.
<b>x, y</b>	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.  For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
<b>contrasts</b>	an optional list. See the contrasts.arg of model.matrix.default.
<b>...</b>	additional arguments to be passed to the low level regression fitting functions see lm and glm help files

**Details**

For Gaussian models, the function plots the influence measures calculated by the R function `influence.measures`. These include the DFBETAS for each coefficient, DDFITS, COVRATIO, Cook's D, and the hat matrix diagonals. Set the plot layout accordingly with `par`. For logistic models, four plots are produced: index plots of the deviance and Pearson residuals, Cook's D and the leave-one-out deviance change.

**Value**

Draws the plots but returns no value.

**Note**

This function redirects to other functions based on the type of object. eg `influence.plots.lm` , `influence.plots.formula` , `influence.plots.glm`

**Author(s)**

Alan Lee, Blair Robertson

**References**

Chambers, J. M. (1992) Linear models. Chapter 4 of Statistical Models in S eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

**Examples**

```
data(educ.df)
educ.lm = lm(educ~urban + percap + under18, data=educ.df)
par(mfrow=c(2,4))
influenceplots(educ.lm)
#
influenceplots(educ~urban + percap + under18, data=educ.df)
```

---

`ingots.df`

*Unreadiness for rolling of metal ingots*

---

**Description**

An experiment testing metal ingots prepared with different soaking times and heats. The number which were not ready were counted

**Usage**

```
data(ingots.df)
```

**Format**

A data frame which shows a condensed table.

**heat** heating times for ingots either 7, 24, 27 or 51

**soak** soaking times for ingots either 1.0, 1.7, 2.2, 2.8 or 4.0

**notready** the number of ingots not ready for rolling

**total** the total number of ingots which were tested under that set of conditions

**Source**

D.R.Cox. (1970) The Analysis of Binary Data, p. 11

**References**

D.R.Cox. (1970) The Analysis of Binary Data, Chapman and Hall, London

**Examples**

```
data(ingots.df)
ingots.glm<-glm(cbind(notready, total-notready)~heat +
  soak, weight=total, family = binomial, data = ingots.df)
summary(ingots.glm)
```

---

kyphosis.df

*Data on Children who have had Corrective Spinal Surgery*

---

**Description**

The kyphosis data frame has 81 rows and 4 columns, representing data on children who have had corrective spinal surgery.

**Usage**

```
data(kyphosis.df)
```

**Format**

A data frame with 81 observations on the following 4 variables:

**Kyphosis** a factor with levels absent present indicating if a kyphosis (a type of deformation) was present after the operation.

**Age** in months

**Number** the number of vertebrae involved

**Start** the number of the first (topmost) vertebra operated on

**Source**

John M. Chambers and Trevor J. Hastie, Statistical Models in S, p 200.

**References**

John M. Chambers and Trevor J. Hastie eds. (1992) Statistical Models in S, Wadsworth and Brooks/Cole, Pacific Grove, CA.

**Examples**

```
data(kyphosis.df)
pairs20x(kyphosis.df)
```

---

<code>lizard.df</code>	<i>Lizard data</i>
------------------------	--------------------

---

**Description**

Site preferences of two species of lizard, grahami and opalinus

**Usage**

```
data(lizard.df)
```

**Format**

A data frame with 12 observations on the following 5 variables:

**length** perch lenght (short, long)  
**height** perch height (high, low)  
**time** time of day (early, late, mid)  
**r** number of grahami lizards  
**n** total number of lizards

**Source**

Schoener, T. W. (1970) Nonsynchronous spatial overlap of lizards in patchy habitats. *Ecology* 51, 408-418.

**References**

McCullagh, P. and Nelder, J. A. (1989.) *Generalized Linear Models* (2nd Edition). Chapman and Hall, London.

**Examples**

```
data(lizard.df)
plot.design(lizard.df, y=log(lizard.df$r
/(lizard.df$n-lizard.df$r)), ylab="mean of logits")
```

---

<code>metal.df</code>	<i>Metal Removal</i>
-----------------------	----------------------

---

**Description**

Data from an experiment to measure the rate of metal removal in a machining process on a lathe

**Usage**

```
data(metal.df)
```

**Format**

A data frame with 15 observations on the following 3 variables:

**hardness** hardness of the material being machined  
**setting** speed setting of the lathe (fast, medium or slow)  
**rate** rate of metal removal (response)



**Examples**

```
data(metal.df)
med <-ifelse(metal.df$setting=="medium", 1,0)
slow<-ifelse(metal.df$setting=="slow", 1,0)
summary(lm(rate~med + slow + hardness, data=metal.df))
```

mines.df

*Mining accident data***Description**

A data set with the number of accidents per mine in a 3 month period in 44 coal mines in West Virginia

**Usage**

```
data(mines.df)
```

**Format**

A data frame with 44 observations on the following 5 variables:

COUNT number of accidents (response)

INB inner burden thickness

EXTRP percentage of coal extracted from mine

AHS the average height of the coal seam in the mine

AGE the age of the mine

**Examples**

```
data(mines.df)
mines.glm<-glm(COUNT ~ INB + EXTRP + AHS + AGE,
               family=poisson, data=mines.df)
```

onions.df

*Onion growing data***Description**

From an onion growing experiment at the Hort Research station at Pukekohe. Conducted to compare the effects of different curing methods on the number of skins on individual onions at different stages of maturity

**Usage**

```
data(onions.df)
```

**Format**

A data frame with 300 observations on the following 5 variables:

**maturity** Maturity of the onion as a percentage (50,70,90,95,100)

**cure** method of curing (traditional, shears or partial)

**block** the area of land the onions were grown in (1,2,3,4)

**skins** the number of skins

**weight** a numeric vector

**Source**

C.M. Triggs, personal communication

**Examples**

```
data(onions.df)
onions.glm<-glm(skins ~ factor(block),
  family=poisson, weight=weight, data=onions.df)
```

---

plum.df

*Plum tree data*

---

**Description**

A study was conducted on the reproduction of plum trees by taking cuttings from older trees. Half the cuttings were planted immediately while the other half were bedded in sand until spring when they were planted. Two lengths of cuttings were used: long (12 cm) and short (6cm). A total of 240 cuttings were taken for each of the 4 combinations of planting time and cutting length and the number of cuttings that survived in each situation was recorded.

**Usage**

```
data(plum.df)
```

**Format**

A data frame with 4 observations on the following 4 variables:

**length** cutting length (long,short)

**time** planting time (spring, autumn)

**s** number that survived

**n** total number planted

**Examples**

```
data(plum.df)
plum.glm<-glm(cbind(s,n-s)~length*time, family=binomial, data=plum.df)
summary(plum.glm)
```

rats.df

*Rat growth data*

## Description

Each rat in the data set was measured on 11 dates expressed as days from start of the experiment. The purpose was to see if the growth rate was the same for each group.

## Usage

```
data(rats.df)
```

## Format

A data frame with 176 observations on the following 5 variables:

**growth** a numeric vector

**group** a numeric vector

**rat** a numeric vector

**change** a numeric vector

**day** days since the start of experiment

## Details

Hand and Crowder (1996) describe data on the body weights of rats measured over 64 days. These data also appear in Table 2.4 of Crowder and Hand (1990). The body weights of the rats (in grams) are measured on day 1 and every seven days thereafter until day 64, with an extra measurement on day 44. The experiment started several weeks before “day 1.” There are three groups of rats, each on a different diet.

## Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.3)

Crowder, M. and Hand, D. (1990), *Analysis of Repeated Measures*, Chapman and Hall, London.

Hand, D. and Crowder, M. (1996), *Practical Longitudinal Data Analysis*, Chapman and Hall, London.

## Examples

```
data(rats.df)
group.vec<-as.numeric(rats.df$group)
# convert group from factor to vector
plot(growth~day,type="n",
     data=rats.df,
     xlab="days elapsed",
     ylab="Weight (grams)",
     main="Rat Growth rates")
for(i in (0:15)){
  index<-(1:11) + i*11
  lines(rats.df$day[index],rats.df$growth[index],
        lty=group.vec[index[1]])
}
legend(45,400,paste("Group",1:3),lty=c(1,2,3))
```

---

reg3d	<i>3d plot of data</i>
-------	------------------------

---

**Description**

plots 3 variables on x,y,z axes and draws fitted plane

**Usage**

```
reg3d(data, wire = FALSE)
```

**Arguments**

<b>data</b>	a data frame, with the first three columns containing the values of x, y and z
<b>wire</b>	If TRUE draws a grid as a reference plane, if FALSE returns a solid reference plane

**Author(s)**

Alan Lee, Blair Robertson

**Examples**

```
data(fatty.df)
reg3d(fatty.df, wire=TRUE)

data(rubber.df)
reg3d(rubber.df)
```

---

ROC.curve	<i>draws an ROC curve</i>
-----------	---------------------------

---

**Description**

Draws an ROC curve and calculates the area under the curve

**Usage**

```
ROC.curve(f, ...)
## S3 method for class 'glm'
ROC.curve(f, ...)
## S3 method for class 'formula'
ROC.curve(f, family = gaussian, data, weights, subset, na.action,
  start = NULL, etastart, mustart, offset, control = list(...),
  model = TRUE, method = "glm.fit", x = FALSE, y = TRUE,
  contrasts = NULL, ...)
```

**Arguments**

<b>f</b>	a glm object or a model formula
<b>family</b>	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>weights</b>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
<b>start</b>	starting values for the parameters in the linear predictor.
<b>etastart</b>	starting values for the linear predictor.
<b>mustart</b>	starting values for the vector of means.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
<b>control</b>	a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.
<b>model</b>	a logical value indicating whether model frame should be included as a component of the returned value.
<b>method</b>	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS); the alternative "model.frame" returns the model frame and does no fitting.
<b>x, y</b>	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.  For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
<b>contrasts</b>	an optional list. See the contrasts.arg of model.matrix.default.
<b>...</b>	additional arguments to be passed to the low level regression fitting functions see lm and glm help files

**Details**

If the formula version is used, an error will occur unless family=binomial

**Value**

Returns no value but prints the area under the ROC curve and draws the curve

**Author(s)**

Alan Lee, Blair Robertson

**Examples**

```
data(drug.df)
ROC.curve(DFREE ~ NDRUGTX + factor(IVHX) + AGE + TREAT, family=binomial,
  data= drug.df)
```

---

 rubber.df

*Rubber Specimen Data*


---

**Description**

Thirty rubber specimens were rubbed with an abrasive metal.

**Usage**

```
data(rubber.df)
```

**Format**

A data frame with 30 observations on the following 3 variables:

**hardness** Hardness in degrees of Shore

**tensile** strength in kilograms per square centimetre

**abloss** the amount of material rubbed off in grams per horsepower-hour

**Source**

Chambers, J. M. et al. Graphical Methods for Data Analysis, p 379

**References**

Chambers, J. M. et al. (1983). Graphical Methods for Data Analysis. Duxbury Press: Boston.

**Examples**

```
data(rubber.df)
rubber.lm<-lm(abloss~hardness+tensile,data=rubber.df)
pred<-fitted.values(rubber.lm)
res<-residuals(rubber.lm)
plot(pred,res)
```

---

 salary.df

*Study of Supervisor Performance*


---

**Description**

A survey of the clerical employment of a large financial organisation included questions related to employee satisfaction with their supervisors, designed to determine the overall effectiveness of the supervisor

**Usage**

```
data(salary.df)
```

**Format**

A data frame with 31 observations on the following 6 variables:

- X1 Handles employee complaints
- X2 Does not allow special privileges
- X3 Opportunity to learn new things
- X4 Raises based on performances
- X5 Too critical of poor performances
- Y Overall rating of job being done by supervisor

**Source**

S. Chatterjee, A.S. Hadi and B. Price, Regression Analysis by Example, p56

**References**

S. Chatterjee, A.S. Hadi and B. Price, (2000). Regression Analysis by Example (3rd Ed), Wiley, New York.

**Examples**

```
data(salary.df)
salary.lm<-lm(Y~X1+X2+X3+X4+X5,data=salary.df)
resids<-residuals(salary.lm)
pred<-fitted.values(salary.lm)
plot(pred,resids,type="n")
ncases<-length(resids)
text(pred,resids,1:ncases)
```

---

sport.df

*Australian Institute of Sport*

---

**Description**

Data on 102 male and 100 female athletes collected at the Australian Institute of Sport, courtesy of Richard Telford and Ross Cunningham.

**Usage**

```
data(sport.df)
```

**Format**

A data frame with 158 observations on the following 5 variables:

- ID ID
- sex male or female
- sport Sport
- BMI Body mass index = weight/height<sup>2</sup>
- X.Bfat percentage body fat

**Source**

Richard Telford and Ross Cunningham, Australian National University.

Cook, R. D., and Weisberg, S. (1994). An Introduction to Regression Graphics. Wiley, New York.

**Examples**

```
data(sport.df)
library(lattice)
xyplot(X.Bfat~BMI|sport*sex,xlab="BMI",ylab="X.Bfat",data=sport.df)
```

---

stamford.df

*Maximum Daily Ozone Concentrations*


---

**Description**

Daily maximum ozone concentrations at Stamford, Connecticut and Yonkers, New York, during the period 1 May 1974 to 30 September 1974.

**Usage**

```
data(stamford.df)
```

**Format**

A data frame with 136 observations on the following 2 variables:

**days** denotes which day observation occurred on

**ozone** ozone in parts per billion

**Source**

Chambers, J. M. et al. Graphical Methods for Data Analysis. p346

**References**

Chambers, J. M. et al. (1983). Graphical Methods for Data Analysis. Duxbury Press: Boston.

**Examples**

```
data(stamford.df)
plot(stamford.df$days, stamford.df$ozone, xlab="Days", ylab="Ozone")
loess.stuff=loess(ozone~days, data=stamford.df, span=0.75)
lines(loess.stuff$x, loess.stuff$fitted)
```

---

test.lc

*tests hypothesis  $cc^Tb=c$* 


---

**Description**

Given a linear model with coefficient vector **b** tests the linear hypothesis  $cc^Tb=c$



**Usage**

```

test.lc(f, cc, c, ...)
## S3 method for class 'lm'
test.lc(f, cc, c, ...)
## S3 method for class 'glm'
test.lc(f, cc,c, ...)
## S3 method for class 'formula'
test.lc(f, cc, c, family = gaussian, data, weights, subset,
      na.action, start = NULL, etastart, mustart, offset,
      control = list(...), model = TRUE, method = "glm.fit",
      x = FALSE, y = TRUE, contrasts = NULL, ...)

```

**Arguments**

<b>f</b>	a glm object or a model formula
<b>cc</b>	a vector containing the coefficients of the linear combination
<b>c</b>	hypothetical value of the combination
<b>family</b>	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>weights</b>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
<b>start</b>	starting values for the parameters in the linear predictor.
<b>etastart</b>	starting values for the linear predictor.
<b>mustart</b>	starting values for the vector of means.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
<b>control</b>	a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.
<b>model</b>	a logical value indicating whether model frame should be included as a component of the returned value.
<b>method</b>	the method to be used in fitting the model. The default method “glm.fit” uses iteratively reweighted least squares (IWLS); the alternative “model.frame” returns the model frame and does no fitting.
<b>x, y</b>	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension $n * p$ , and y is a vector of observations of length n.
<b>contrasts</b>	an optional list. See the contrasts.arg of model.matrix.default.

... additional arguments to be passed to the low level regression fitting functions  
see `lm` and `glm` help files

### Value

`$est` gives estimate for `cc`  
`$std.err` gives standard error for `cc`  
`$df` degrees of freedom  
`$t.stat` gives the t stat for the hypothesis test  
`$p.val` gives p-value for the hypothesis test

### Note

Redirects based on the type of object

### Author(s)

Alan Lee, Blair Robertson

### Examples

```
data(cherry.df)
cherry.lm = lm(log(volume)~log(diameter)+log(height),data=cherry.df)
cc = c(0,1,1)
c = 3
test.lc(cherry.lm, cc, c)
```

---

traffic.df

*Highway accident rates*

---

### Description

Data gathered in the course of studying the relationship between accident rates on US highways and various characteristics of the highway

### Usage

```
data(traffic.df)
```

### Format

A data frame with 39 observations on the following 15 variables:

**obs** observation number  
**rate** The accident rate per million vehicle miles (response)  
**len** The length of the segment of highway in miles  
**adt** Average daily traffic count ('000)  
**trks** The percentage of the traffic that are trucks  
**slim** the speed limit in mph  
**lwd** the lane width in feet  
**shld** the shoulder width in feet  
**itg** number of freeway interchanges per mile  
**sigs** number of entrances controlled by lights per mile

**acpt** number of access points per mile  
**lane** number of lanes in each direction  
**fai** dummy variable, equal to 1 if an interstate highway, zero otherwise  
**pa** equal to 1 if a principal highway, 0 otherwise  
**ma** equal to 1 if a major highway, 0 otherwise

### Source

Carl Hoffstedt. This differs from the dataset highway in the alr3 package only by transformation of some of the columns.

### References

Fox, J. and Weisberg, S. (2011) An R Companion to Applied Regression, Second Edition, Sage.  
 Weisberg, S. (2005) Applied Linear Regression, Third Edition. Wiley, Section 7.2.

### Examples

```
data(traffic.df)
traffic.lm<-lm(rate~.,data=traffic.df)
summary(traffic.lm)
```

---

vapour.df

*Hydrocarbon data*


---

### Description

When petrol is pumped into a tank, hydrocarbon vapours are forced into the atmosphere. To reduce this significant source of air pollution, devices are installed to capture the vapour. A laboratory experiment was conducted in which the amount of vapour given off was measured.

### Usage

```
data(vapour.df)
```

### Format

A data frame with 125 observations on the following 5 variables:

**t.temp** initial tank temperature (degrees F)  
**p.temp** temperature of the dispensed petrol (degrees F)  
**t.vp** initial vapour pressure in tank (psi)  
**p.vp** vapour pressure of the dispensed petrol (psi)  
**hc** emitted dispensed hydrocarbons (g)(response)

### Examples

```
data(vapour.df)
vapour.lm<-lm(hc~ t.temp + p.temp + t.vp + p.vp, data=vapour.df)
summary(vapour.lm)
```

---

vaso.df	<i>vaso-constriction data</i>
---------	-------------------------------

---

### Description

Data from a study of reflex vaso-constriction (narrowing of the blood vessels) of the skin of the fingers

### Usage

```
data(vaso.df)
```

### Format

A data frame with 39 observations on the following 3 variables.

**Volume** volume of air breathed in

**Rate** rate of intake of breath

**Response** 1 = vaso-constriction occurs, 0 = doesn't occur

### Source

Finney, D. J. (1947). The estimation from individual records of the relationship between dose and quantal response. *Biometrika*, 34, 320-334.

### References

Pregibon, D. (1981) Logistic regression diagnostics. *Annals of Statistics*, 9,705-724.

### Examples

```
data(vaso.df)
plot(vaso.df$Rate, vaso.df$Volume, type="n", cex=1.2)
text(vaso.df$Rate, vaso.df$Volume, 1:39,
     col=ifelse(vaso.df$Response==1, "red", "blue"), cex=1.2)
text(2.3, 3.5, "blue: no VS", col="blue", adj=0, cex=1.2)
text(2.3, 3.0, "red: VS", col="red", adj=0, cex=1.2)
```

---

WB.test	<i>Performs a Weisberg-Bingham test for normality</i>
---------	---

---

### Description

Calculates and prints a  $\chi^2$  statistic and a p-value for the Weisberg-Bingham test. The p-value is calculated by simulation.

### Usage

```
WB.test(f, n.rep=1000, ...)
## S3 method for class 'lm'
WB.test(f, n.rep=1000, ...)
## S3 method for class 'formula'
WB.test(f, n.rep=1000, data, subset, weights, na.action,
       method = "qr", model = TRUE, x = FALSE, y = FALSE,
       qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)
```

**Arguments**

<b>f</b>	an lm object or model formula
<b>n.rep</b>	the number of simulations desired to compute the p-value. The default of 1000 should be adequate
<b>data</b>	A data frame, list or environment containing the variables in the model.
<b>subset</b>	an optional vector specifying a subset of observations to be used in the fitting process.
<b>weights</b>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
<b>method</b>	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS); the alternative "model.frame" returns the model frame and does no fitting.
<b>x, y, qr, model</b>	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension $n * p$ , and y is a vector of observations of length n.
<b>singular.ok</b>	logical. If FALSE (the default in S but not in R) a singular fit is an error.
<b>contrasts</b>	an optional list. See the contrasts.arg of model.matrix.default.
<b>offset</b>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
<b>...</b>	additional arguments to be passed to the low level regression fitting functions see lm and glm help files

**Value**

WB test stat gives value of Weisberg-Bingham test for normality

p gives the p-value of the test

**Note**

This function redirects to other functions based on the type of object. eg WB.test.lm , WB.test.formula

**Author(s)**

Alan Lee, Blair Robertson

**Examples**

```
data(cherry.df)
cherry.lm =lm(volume~diameter+height,data=cherry.df)
WB.test(cherry.lm)
```

---

`wine.df`*Bordeaux Wine data*

---

**Description**

A data set which attempted to assess the quality of various Bordeaux vintages based upon certain variables

**Usage**

```
data(wine.df)
```

**Format**

A data frame with 27 observations on the following 5 variables:

**year** year (1952-1980)

**price** Price (in 1980 US dollars, converted to an index with 1961=100)

**temp** average temp during the growing season (degrees Celcius)

**h.rain** total rainfall during harvest period (mm)

**w.rain** total rainfall over preceding winter (mm)

**Source**

The data are available at <http://www.liquidasset.com/winedata.html>

**References**

An article by Orly Ashenfelter is at <http://www.liquidasset.com/orley.htm> See also Orley Ashenfelter, David Ashmore, and Robert Lalonde, Bordeaux wine vintage quality and the weather. Chance Magazine, Fall 1995, pp.7-14

**Examples**

```
data(wine.df)
boxcoxplot(price~temp+h.rain+w.rain+year, data=wine.df)
```

# Appendix C

## Bibliography

- J Adler (2010). R in a Nutshell. O'Reilly.
- A Agresti (2002). Categorical Data Analysis, 2nd Ed, Wiley.
- AC Atkinson (1982). Plots, Transformations and Regression: A Introduction to Graphical methods of Diagnostic Residual Analysis. Oxford University Press.
- DA Belsley, E Kuh and RE Welsch (1980). Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. Wiley.
- GEP Box, WR Cousins, FR Hindsworth, H Heeny, M Milbourne, W Spendley and WL Stevens (1957). In OL Davies (Ed.) Statistical Methods in Research and Production, 3rd Ed. Oliver and Boyd, London.
- RJ Carroll, RJ and DHB Cline (1988). An asymptotic theory for weighted least squares with weights estimated by replication, *Biometrika* 75, 35–43.
- JM Chambers, WS Cleveland, B Kleiner and PA Tukey (1983). Graphical Methods for Data Analysis, Duxbury Press.
- JM Chambers and TJ Hastie (1992). Statistical Models in S, Wadsworth.
- S Chatterjee and AS Hadi (1988). Sensitivity Analysis in Linear Regression. Wiley.
- S Chatterjee AS Hadi and B Price, (2000). Regression Analysis by Example (3rd Ed), Wiley.
- WS Cleveland (1994). The Elements of Graphing Data (revised Ed), Hobart Press.
- WS Cleveland (1993). Visualizing Data, Hobart Press.
- Dianne Cook and Deborah F. Swayne (2007). Interactive and Dynamic Graphics for Data Analysis. Springer, New York.
- RD Cook (1977). Detection of influential observations in linear regression. *Technometrics*, 19, 15–18.
- RD Cook and S Weisberg (1982). Residuals and Influence in Regression, Chapman and Hall.
- RD Cook and S Weisberg (1999). Applied Regression Including Computing and Graphics, Wiley.
- DR Cox(1970). The Analysis of Binary Data, Methuen.

- Michael J. Crawley (2005). *Statistics: An Introduction using R*. Wiley.
- P Dalgaard (2002). *Introductory Statistics with R*. Springer.
- AJ Dobson (2002). *An Introduction to Generalized Linear Models* (2nd Ed), Chapman & Hall.
- NR Draper and H Smith (1998). *Applied Regression Analysis* (3rd Ed), Wiley.
- Bradley Efron and Robert J. Tibshirani (1993). *An Introduction to the Bootstrap*. Chapman & Hall, London.
- Brian Everitt and Torsten Hothorn (2006). *A Handbook of Statistical Analyses Using R*. Chapman & Hall/CRC.
- Julian J. Faraway (2006). *Extending Linear Models with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Chapman & Hall/CRC.
- J Fox (2002). *An R and S-Plus Companion to Applied Regression*, Sage Publications.
- Frank E. Harrell (2001). *Regression Modeling Strategies, with Applications to Linear Models, Survival Analysis and Logistic Regression*. Springer.
- T Hastie and RJ Tibshirani (1990). *Generalized Additive Models*. Chapman and Hall.
- T Hastie, R Tibshirani and J Friedman (2001). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer.
- Richard M. Heiberger and Burt Holland (2004). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer.
- Holloway, J.W. (1989). *A comparison of the toxicity of the pyrethroid trans-cypermethrin, with and without the synergist piperonyl butoxide, to adult moths from two strains of *Heliothis virescens**. University of Reading, Ph.D. thesis, Department of Pure and Applied Zoology.
- DW Hosmer and S Lemeshow (2000). *Applied Logistic Regression* (2nd Ed), Wiley.
- P Huber (1981). *Robust Statistics*. Wiley.
- DG Kleinbaum and M Klein, (2002). *Logistic Regression : a Self-Learning Text*. New York: Springer.
- John Maindonald and John Braun (2003). *Data Analysis and Graphics Using R*. Cambridge University Press.
- DW Marquardt and RD Snee (1975). Ridge regression in practice. *American Statistician*, 28, 3–20.
- S Menard (2002). *Applied Logistic Regression Analysis*. Thousand Oaks, Calif.: Sage Publications.
- DC Montgomery, EA. Peck and GG Vining (2001). *Introduction to Linear Regression Analysis* (3rd Ed), Wiley.
- P Murrell (2006). *R Graphics*. Chapman and Hall.
- P Murrell (2009). *Introduction to Data Technologies*. Chapman and Hall.
- D Pregibon (1981). Logistic Regression Diagnostics. *Annals of Statistics*, 9, 705-724. ]



- PJ Rousseeuw (1984). Least median of squares. *Journal of the American Statistical Association*, 79, 871–880.
- PJ Rousseeuw and AM Leroy (1987). *Robust Regression and Outlier Detection*. Wiley.
- WN Venables and BD Ripley (2004). *Modern Applied Statistics with S*, 4th Ed, Springer.
- WN Venables and DM Smith (2002). *Introduction to R*, Springer.
- John Verzani (2005). *Using R for Introductory Statistics*. Chapman & Hall/CRC.
- Simon N. Wood (2006). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC.
- S Weisberg, (1985). *Applied Linear Regression* (2nd Ed), Wiley.
- S Weisberg and C Bingham (1975). An approximate analysis of variance test for non-normality suitable for machine calculation, *Technometrics* 17, 133-134.
- AF Zuur, EN Ieno and EHWG Meesters (2009). *A Beginners Guide to R*. Springer.