Exercises for

Modern Applied Statistics with S-Plus

Second edition

by

W. N. Venables and B. D. Ripley

Springer (1997). ISBN 0-387-98214-0

Selectable links are in this colour.

Exercises and Further Exercises

For convenience we reproduce the exercises from the text as well as further exercises.

Chapter 2

2.1 How would you find the index(es) of specified values within a vector? For example, where is the hill race (in `hills`) with a climb of 2100 feet? Answer

2.2 The column `ftv` in data frame `birthwt` counts the number of visits. Reduce this to a factor with levels 0, 1 and ‘2 or more’. [Hint: manipulate the `levels`, or investigate functions `cut` and `merge.levels`.] Answer

2.3 Write a simple function to compute the median absolute deviation (used in robust statistics; see Chapter 8) median$|x - \mu|$ with default μ the sample median. Compare your answer with the S-PLUS function `mad`. Answer

2.4 Suppose `x` is an object with named components and `out` is a character string vector. How would you make a new object obtained from `x` by *excluding* any components whose names are in `out`? Answer

Further exercises

2.5 Given a matrix `X` of distinct rows and a vector `w` of the number of times that each row should occur, reconstruct the original matrix. Answer

2.6 From a question of Daniel Svozil.

‘I calculated a cross-correlation matrix. I want to print only members of this matrix that are larger than 0.90 and I want to include dimnames in the answer.’ Answer

2.7 Andrew McCulloch asked:

‘I have a large data frame (5000 observations) and I would like the cases where a variable indicating ethnic group is in (1,3,4,6,7).’ Answer

Chapter 3

3.1 The data frame `survey` contains the results of a survey of 237 first-year Statistics students at Adelaide University. For a graphical summary of all the variables, use `plot(survey)`. Note that this produces a dotchart for factor variables, and a normal scores plot for the numeric variables.

One component of this data frame, `Exer`, is a factor object containing the responses to a question asking how often the students exercised. Produce a barchart of these responses. Use `table` and `pie` or `piechart` to create a pie chart of the responses. Do you like this better than the bar plot? Which is more informative? Which gives a better picture of exercise habits of students? The `pie` function takes an argument `names` which can be used to put labels on each pie slice. Redraw the pie chart with labels. Alternatively, you could add a legend to identify the slices.

You might like to try the same things with the `Smoke` variable, which records responses to the question "How often do you smoke?" Note that `table` and `levels` ignore missing values; if you wish to include non-respondents in your chart use `summary` to generate the values, and `names` on the summary object to generate the labels. Answer

3.2 Make a plot of petal width *vs* petal length of the `iris` data for a partially sighted audience, identifying the three species. You will need to double the annotation size, thicken the lines and change the layout to allow larger margins for the larger annotation. Answer

3.3 Plot $\sin(x)$ against x, using 200 values of x between $-\pi$ and π, but do not plot any axes yet (use parameter `axes=F` in the call to `plot`.) Add a y axis passing through the origin using the 'extended' style and horizontal labels. Add an x axis with tick-marks from $-\pi$ to π in increments of $\pi/4$, twice the usual length. Answer

3.4 Cleveland (1993) recommends that the aspect ratio of line plots is chosen so that lines are 'banked' at $45°$. By this he means that the averaged absolute value of the slope should be around $\pm 45°$. Write a function to achieve this for a time-series plot, and try it out on the `sunspots` dataset. (Average along the arc length of the curve. See the function `banking` for Cleveland's solution.)Answer

3.5 The Trellis function `splom` produces a complete matrix of scatterplots, as does the basic plotting functions `pairs`, but in earlier versions of S-PLUS `pairs` only plotted the lower triangle of the matrix. Write a function to emulate the earlier behaviour. (HINT: look at `pairs.default`. The graphics parameter `mfg` may be useful.) Answer

Further exercise

3.6 *Ternary* plots are used for compositional data (Aitchison, 1986) where there are three components whose proportions add to one. These are represented by a point in an equilateral triangle, where the distances to the sides add to a constant.

Write an S function to plot a matrix of compositions on a ternary diagram. Apply this to the dataset `Skye` on the composition of rocks on the Isle of Skye in Scotland. Answer

Chapter 4

4.1 Write a function to determine which integers (less than 10^{10}) in a given vector are prime. Answer

4.2 Given two vectors `a` and `b`, compute `s`, where `s[i]` is the number of `a[j] <= b[i]`, for `a` much longer than `b`. Then consider `a[j] >= b[i]`. (Based on a question of Frank Harrell.) Answer

4.3 Implement `print` and `plot` method functions for the class `"lda"` of Chapter 13. (Perhaps the `plot` method should give a barchart of the singular values or plot the data on the first few linear discriminants, identifying the groups and marking their means.) Answer

4.4 Write an S function to increment its argument object. [Not as easy as it looks: based on an example of David Lubinsky, *Statistical Science* **6**, p. 356.] Answer

4.5 Write a function that generates the all possible subsets of size `r` from a set of size `n` successively rather than simultaneously. Given one subset it should produce the next in lexicographic order, or report that no further subsets exist in some convenient way that can be trapped in a program. Answer

4.6 For each row of a matrix `X`, we want to find which row of a matrix `M` is nearest in the sense of Euclidean distance. Use `apply` to find out. (You may assume that there is a unique nearest row.) Answer

4.7 For designed experiments it is often useful to predict the response at each combination of levels of the treatment factors. Write a function `expand.factors` which takes any number of factor arguments and returns a data frame with the factors as columns and each combination of levels occurring as exactly one row.

Try a direct solution without using `expand.grid`. Answer

Further exercises

4.8 The data frame `OME` has columns `Correct` and `Trials` giving the number of correct responses out of that number of trials. Generate a new data frame with a row for each individual trial and a new column `Resp` giving the response (correct or not) for that trial. Answer

4.9 Is there a vectorized way to derive partial irregular sums of vector elements? For instance, for a vector `v <- c(1:100)` we might want to create a new vector

```
new <- c(sum(v[1:8]), sum(v[9:25]), sum(v[26:50]),
         sum(v[51:100]))
```

In reality the vectors are long (more than 10000 elements). Answer

4.10 Write a function to generate n samples from a multinomial distribution with probability vector p. Answer

4.11 Write an S function to convert file paths between MS-DOS-style (with separator \\) and Unix-style (with /). Answer

4.12 Write a function to convert a call (as returned by `sys.call` and `match.call`) to a character string in a readable form. (Using `as.character` gives a list of strings with information loss.) Does your answer work with a formula (which is of mode `call`)? Answer

Chapter 5

5.1 Experiment with our dataset `galaxies`. How many modes do you think there are in the underlying density?

Further exercises

5.2 Write functions to produce QQ-plots for a gamma and a Weibull distribution. Note that unlike the normal QQ-plot, the shape parameters may need to be estimated.

5.3 The ideas used in bandwidth selection for kernel density estimation which are implemented in `width.SJ` can also be applied to the choice of bin width in a histogram (Wand, 1997). Implement such a bin-width estimator in S-PLUS. Answer

5.4 Rice (1995, p.390) gives the following data (Natrella, 1963) on the latent heat of the fusion of ice (*cal/gm*):

```
Method A: 79.98 80.04 80.02 80.04 80.03 80.03 80.04 79.97
          80.05 80.03 80.02 80.00 80.02
Method B: 80.02 79.94 79.98 79.97 79.97 80.03 79.95 79.97
```

(a) Assuming normality, test the hypothesis of equal means, both with and without making the assumption of equal variances.

Compare the result with a Wilcoxon/Mann-Whitney nonparametric two-sample test.

(b) Inspect the data graphically in various ways, for example boxplots, QQ-plots and histograms.

(c) Fit a one-way analysis of variance and compare it with your $t-$ test.

Chapter 6

Programming exercises

6.1 How do you obtain the standard prediction and confidence intervals for a linear model fitted by `lm`? Answer

6.2 How can we we add a confidence or prediction region to an existing plot of a simple linear regression?

As an example, add a prediction region to Figure 6.1. Answer

6.3 Write a function to fit a linear model by generalized least squares, that is to minimize

$$(\boldsymbol{y} - X\boldsymbol{\beta})^T W(\boldsymbol{y} - X\boldsymbol{\beta})$$

for a given symmetric positive definite matrix W, or given $\Sigma = W^{-1}$. Answer

6.4 Implement a ridge regression (Brown, 1994, Sen & Srivastava, 1990) function in S. Answer

Data analysis exercises

6.5 The data frame `rubber` in the library `MASS` gives 30 measurements of rubber loss under accelerated testing together with the hardness and tensile strength of the rubber itself. Explore the data in `brush`, then fit linear and quadratic regressions of `loss` on `hard` and `tens`. Select a suitable submodel of the quadratic model, and inspect the fitted surface by a perspective plot.

6.6 Analyse the data on CPU performance in data frame `cpus` and compare your model with that fitted by Ein-Dor & Feldmesser (1987). See also Ripley (1994).

6.7 Criminologists are interested in the effect of punishment regimes on crime rates. This has been studied using aggregate data on 47 states of the USA for 1960, available in data frame `UScrime` (Ehrlich, 1973, Vandaele, 1978, Raftery, 1995). The response variable is the rate of crimes in a particular category per head of population. There are 15 explanatory variables; most of these and the response variable have been rescaled to convenient numbers.

(a) Analyse these data. In your report pay particular attention to how your model was selected.

(b) Comment on the effect of the last two explanatory variables in relation to the criminologists' interest in the effect of punishment.

(c) Comment critically on the assumptions needed to draw conclusions from aggregate studies such as this.

6.8 Susan Prosser collected data on the concentration of a chemical GAG in the urine of 314 children aged from zero to seventeen years. The data are in data frame `GAGurine`. Analyse these data, and produce a chart to help a paediatrican to assess if a child's GAG concentration is 'normal'.

6.9 The Janka hardness data in data frame `janka` gives the density (`Dens`) and hardness (`Hard`) of a sample of Australian Eucalypt hardwoods. The problem is to build a prediction equation for hardness in terms of density.

6.10 The `Cars93` data frame gives data on 93 new car models on sale in the USA in 1993. Use this dataset to predict fuel consumption from the remaining variables. (HINT: The fuel consumption is in miles per US gallon. In metric units fuel consumption is expressed in litres/100km, a reciprocal scale.)

6.11 The data in Table 6.1 (from Scheffé, 1959, and in data frame `genotype`)

Table 6.1: The rat `genotype` data.

	Foster mother			
Litter	A	B	I	J
A	61.5	55.0	52.5	42.0
	68.2	42.0	61.8	54.0
	64.0	60.2	49.5	61.0
	65.0		52.7	48.2
	59.7			39.6
B	60.3	50.8	56.5	51.3
	51.7	64.7	59.0	40.5
	49.3	61.7	47.2	
	48.0	64.0	53.0	
		62.0		
I	37.0	56.3	39.7	50.0
	36.3	69.8	46.0	43.8
	68.0	67.0	61.3	54.5
			55.3	
			55.7	
J	59.0	59.5	45.2	44.8
	57.4	52.8	57.0	51.5
	54.0	56.0	61.4	53.0
	47.0			42.0
				54.0

refer to rat litters which were separated from their natural mothers at birth and given to foster mothers to rear. The rats were classified into one of four genotypes, `A`, `B`, `I` and `J`. The response is the litter average weight gain, in grams, over the time of the study. The aim is to test whether the litters' and mothers' genotypes act additively and if this may be retained to test for differences in litter and mother genotype effects.

Chapter 7

New exercises

7.1 Analyse the `menarche` dataset on the proportions of female children in Warsaw at various ages during adolescence who have reached menarche (Milicer & Szczotka, 1966) using both logit and probit links.

7.2 Knight & Skagen (1988) collected the data shown in the table during a field study on the foraging behaviour of wintering Bald Eagles in Washington State, USA. The data concern 160 attempts by one (pirating) Bald Eagle to steal a chum salmon from another (feeding) Bald Eagle. The abbreviations used are

L = 'large' S = 'small'; A = 'adult' I = 'immature'

Report on factors which explain the success of the pirating attempt, and give a prediction formula for the probability of success.

Number of successful attempts	Total number of attempts	Size of pirating Eagle	Age of pirating Eagle	Size of feeding Eagle
17	24	L	A	L
29	29	L	A	S
17	27	L	I	L
20	20	L	I	S
1	12	S	A	L
15	16	S	A	S
0	28	S	I	L
1	4	S	I	S

7.3 The following data are part of a survey by Dr Mutch of low weight births in Scotland between 1981 and 1988. The table refers to 661 children with birth weights between 650g and 1749g all of whom survived for at least one year. The variables of interest are:

Cardiac: Mild heart problems of the mother during pregnancy.

Comps: Gynaecological problems during pregnancy.

Smoking: Mother smoked at least one cigarette per day during the first 6 months of pregnancy.

BW: Was the birth weight less than 1250g?

Cardiac		Yes				No			
Comps		Yes		No		Yes		No	
Smoking		Yes	No	Yes	No	Yes	No	Yes	No
BW	Yes	10	25	12	15	18	12	42	45
	No	7	5	22	19	10	12	202	205

Analyse this table.

7.4 A survey was made of bicycle and other traffic in the neighbourhood of the Berkeley campus of the University of California in 1993 (Gelman *et al.*, 1995, p. 91). Sixty city streets were selected at random, with a stratification into three levels of activity and whether or not the street had a marked bicycle lane. The counts observed in one hour are shown in the table: for two of the streets the data were lost.

Type of street	Bike lane?		Counts									
Residential	yes	bikes	16	9	10	13	19	20	18	17	35	55
		other	58	90	48	57	103	57	86	112	273	64
Residential	no	bikes	12	1	2	4	9	7	9	8		
		other	113	18	14	44	208	67	29	154		
Side	yes	bikes	8	35	31	19	38	47	44	44	29	18
		other	29	415	425	42	180	675	620	437	47	462
Side	no	bikes	10	43	5	14	58	15	0	47	51	32
		other	557	1258	499	601	1163	700	90	1093	1459	1086
Main	yes	bikes	60	51	58	59	53	68	68	60	71	63
		other	1545	1499	1598	503	407	1494	1558	1706	476	752
Main	no	bikes	8	9	6	9	19	61	31	75	14	25
		other	1248	1246	1596	1765	1290	2498	2346	3101	1918	2318

Report on these data, paying particular attention to the effects of bicycle lanes.

7.5 To study the relative survival capacities of two species of native and exotic snails, here labelled A and B, groups of 20 animals were held in controlled laboratory conditions for periods of 1, 2, 3 or 4 weeks. At the end of the period the animals were checked for whether or not they had survived, but as the check itself is a destructive process a longitudinal study with the same animals was not possible. The groups were held in chambers where the temperature and relative humidity were held fixed at three and four levels respectively. There were thus $2 \times 4 \times 3 \times 4 = 96$ groups laid out in a complete factorial design.

The data are shown in Table 7.2, where each entry is the number who did not survive out of the 20 test animals. The data set is also available as the data frame `snails` in library `MASS`. Variable `Species` is a two-level factor but treat the other stimulus variables as quantitative.

(a) Fit separate logistic regression models on exposure, relative humidity and temperature for each species, that is a logistic regression of the form `Species/(Exposure + Rel.Hum + Temp)`.

(b) Fit parallel logistic regressions for the two species on the three stimulus variables and show that it may be retained when tested within the separate regressions model.

Table 7.2: The snail mortality data

		Species A				Species B			
		Exposure				Exposure			
Rel. Hum.	Temp. (° C)	1	2	3	4	1	2	3	4
60.0%	10	0	0	1	7	0	0	7	12
	15	0	1	4	7	0	3	11	14
	20	0	1	5	7	0	2	11	16
65.8%	10	0	0	0	4	0	0	4	10
	15	0	1	2	4	0	2	5	12
	20	0	0	4	7	0	1	9	12
70.5%	10	0	0	0	3	0	0	2	5
	15	0	0	2	3	0	0	4	7
	20	0	0	3	5	0	1	6	9
75.8%	10	0	0	0	2	0	1	2	4
	15	0	0	1	3	0	0	3	5
	20	0	0	2	3	0	1	5	7

(c) There are no deaths for either species for the 1 week exposure time. This suggests a quadratic term in `Exposure` might be warranted. Repeat the analysis above including such a quadratic term.

(d) Because deaths are so sparse a residual analysis is fairly meaningless. Nevertheless look at the residuals to see how they appear for this kind of data set.

(e) Is there a significant difference between the survival rates of the two species? Describe qualitatively how the probability of death depends upon the stimulus variables. Summarise your conclusions.

7.6 An experiment was performed in Sweden in 1961–2 to assess the effect of speed limits on the motorway accident rate (Svensson, 1981). The experiment was conducted on 92 days in each year, matched so that day j in 1962 was comparable to day j in 1961. On some days the speed limit was in effect and enforced, while on other days there was no speed limit and cars tended to be driven faster. The speed limit days tended to be in contiguous blocks.

The data set is given in the data frame `Traffic` with factors `year`, `day` and `limit` and the response is the daily traffic accident count, `y`.

Fit Poisson log-linear models and summarize what you discover.

You might assume `day` occurs as a main effect only (fitting models with interaction terms involving factors of 92 levels may take some time and memory!), but assess if an interaction between `limit` and `year` is needed.

Check if the deviance residuals provide any hint of irregular behaviour.

7.7 The data given in data frame `Insurance` consist of the numbers of policyholders n of an insurance company who were exposed to risk, and the numbers of car insurance claims made by those policyholders in the third quarter of 1973 (Baxter *et al.*, 1980; Aitkin *et al.*, 1989). The data are cross-classified by `District` (4 levels), `Group` of car (4 levels), and `Age` of driver (4 ordered levels). The other variables in the data frame are the numbers of `Holders` and `Claims`.

The relevant model is taken to be a Poisson log-linear model with offset $\log n$.

(a) Fit an initial model with all terms present up to the three-way interaction, that is

```
Claims ~ District*Group*Age - District:Group:Age
         + offset(log(Holders))
```

(b) Using `stepAIC`, or otherwise, prune the model of unjustified terms and report your findings.

Present your results as a table of estimated claim rates per policy holder for each category of holder.

(c) It is not strictly valid to regard such data as having the obvious binomial distribution, since some policyholders may make multiple claims. Nevertheless it should be a reasonable approximation. Repeat the analysis with a binomial model and compare the outcomes on estimated claim rates (or in this case, estimated probabilities of making a claim).

Chapter 9

9.1 For the weight loss example compare the negative exponential model with quadratic and cubic polynomial regression alternative models, in particular check the behaviour of each model under extrapolation into the future. Answer

9.2 Fit the negative exponential weight loss model in the 'goal weight' form, equation (9.5), for the three goal weights, $w_0 = 110$, 100 and 90 kg. Plot the profiles and compare the local curvature measures. Answer

9.3 The model used in connection with the Stormer data may also be expressed as a generalized linear model. To do this we write

$$\frac{\beta_1 v}{w - \beta_2} = \frac{1}{\gamma_1 z_1 + \gamma_2 z_2}$$

where $\gamma_1 = 1/\beta_1$, $\gamma_2 = \beta_2/\beta_1$, $z_1 = w/v$ and $z_2 = -1/v$. This has the form of a generalized linear model with inverse link. Fit the model in this form using a `quasi` family with `inverse` link and `constant` variance function.

Back-transform the estimated coefficients and show that they agree with the values obtained using the non-linear regression approach.

Also compute the estimated standard errors and verify that they also agree with the values obtained directly by the non-linear regression approach.

Finding the standard errors is more challenging. We need first to find the variance matrix from the generalized linear model. The large sample variance matrix for $\widehat{\boldsymbol{\beta}}$ is related to that for $\widehat{\boldsymbol{\gamma}}$ by $\operatorname{Var}\left[\widehat{\boldsymbol{\beta}}\right] = \boldsymbol{J}\operatorname{Var}\left[\widehat{\boldsymbol{\gamma}}\right]\boldsymbol{J}^T$ where $\boldsymbol{J}$ is the Jacobian matrix of the inverse of the parameter transformation:

$$\boldsymbol{J} = \begin{bmatrix} \partial\beta_1/\partial\gamma_1 & \partial\beta_1/\partial\gamma_2 \\ \partial\beta_2/\partial\gamma_1 & \partial\beta_2/\partial\gamma_2 \end{bmatrix} = \begin{bmatrix} -1/\gamma_1^2 & 0 \\ -\gamma_2/\gamma_1^2 & 1/\gamma_1 \end{bmatrix}$$

(To achieve close agreement you may need to tighten the convergence criteria for the `glm` fit, for example by setting `eps=1.0e-10`.) Answer

9.4 *Stable parameters*

Ross (1970) has suggested using *stable parameters* for non-linear regression, mainly to achieve estimates which are as near to uncorrelated as possible. It turns out that in many cases stable parameters also define a coordinate system within the solution locus with a small curvature.

The idea is to use the means at p well-separated points in sample space as the parameters. Writing the regression function in terms of the stable parameters is often intractable, but in the case of a negative exponential decay model of the type we considered for the weight loss data it is possible if the points are chosen equally spaced.

If the three mean parameters, μ_i are chosen at x–points $x_0 + i\delta_x$, $i = 0, 1, 2$, show that the model may be written explicitly as:

$$\eta = \frac{\mu_0\mu_2 - \mu_1^2}{\mu_0 - 2\mu_1 + \mu_2} + \frac{(\mu_0 - \mu_1)^2}{\mu_0 - 2\mu_1 + \mu_2}\left(\frac{\mu_1 - \mu_2}{\mu_0 - \mu_1}\right)^{(x - x_0)/\delta_x}$$

Fit the negative exponential decay model to the weight loss data using this parametrization and choosing, say, $x_0 = 40$ days and $\delta_x = 80$ days. Look at the characteristics of the fit, including the correlations between the parameter estimates. Explain in heuristic terms why they are relatively low.

Examine the profiles of the fit and check for straightness. Also look at the local relative curvature measures[1]. Verify numerically that the intrinsic curvature is unchanged (as it must be), but that the parametric curvature is much reduced. Answer

9.5 *Heteroscedastic regression models*

A common heteroscedastic regression model specifies that the observations have constant coefficient of variation, that is $Y \sim \mathrm{N}(\mu, \theta\mu^2)$ where $\theta > 0$ and μ depends on regressor variables according to some linear model perhaps with a link function such as $\mu = \exp\eta$. Write a function to fit such models and try it out using the Quine data. Compare with the negative binomial models fitted in Section 7.4, page 243ff.

[1] For this rather complicated-looking model specification, using `deriv3` to produce a model function with `gradient` and `hessian` attributes may cause memory overflow problems on some machines.

New exercises

9.6 A deterministic relationship between pressure and temperature in saturated steam can be written as

$$\text{Pressure} = \alpha \exp\left(\frac{\beta T}{\gamma + T}\right)$$

where T is the temperature, considered the determining variable. Data collected to estimate the unknown parameters α, β and γ is contained in the data frame `steam`.

(a) Fit this model as a non-linear regression assuming additive errors in the pressure scale. Devise a suitable method for arriving at initial values.

(b) Fit the model again, this time taking logarithms of the relationship above and assuming that the errors are additive in the log(pressure) scale, (and hence multiplicative on the original scale).

(c) Which model do you consider is better supported on the basis of model checks?

9.7 Sarah Hogan collected data on the 'binaural hearing' ability of children with a history of otitis media with effusion (OME). Some of the data (and a description of the problem) are in data frame `OME`. Fit a suitable non-linear model, and assess if there is a change in ability with age and OME status.

(a) The suggested model is a logistic curve that ranges from 0.5 at low noise levels (when the response is effectively a guess) to 1.0 at high noise levels. Then the most important parameter will be the noise level `L75` at which the child has a 75% success rate. The amount of data on each child is small, so fit a model with a common slope but a separate `L75` for each child, and analyse the fitted parameters by age and group. [You may want to look up the function `nlsList`.]

(b) Consider a linear model for `L75` on age, and differences between the OME groups, for each type of noise stimulus. Assess the significance of your results via standard errors and/or F-tests. Answer

9.8 Write a function to fit a gamma distribution to n observations by maximum likelihood. Answer

9.9 McLachlan & Jones (1988) (see also McLachlan & Krishnan, 1997, pp. 73ff) give the following grouped data on red blood cell volume, in 18 equally spaced bins of width 7.2 fl, starting at 21.6 fl.

```
Set 1: 10 21 51 77 70 50 44 40 46 54 53 54 44 36 29 21 16 13
Set 2:  9 32 64 69 56 68 88 93 87 67 44 36 30 24 21 14  8  7
```

McLachlan and Jones fit a mixture of two normal densities on log scale by an involved method using the EM algorithm. Fit this model directly to each set of data by a small modification of the approach in Section 9.7.

Chapter 10

10.1 Add the fitted lines for the final model for the `petrol` data to Figure 10.2.

10.2 Find a way to plot the `Sitka` data which facilitates comparison of the growth curves for the two treatment groups.

Add to your plot the fitted mean growth curve and some 95% confidence intervals.

10.3 Consider how to explore the assumptions made for the `lme` model for the `Sitka` data. Are the `plot` methods for `lme` objects helpful in this? In particular, how would you check the adequacy of the $AR(1)$ model for ϵ_{ij} ? How much is this likely to matter?

10.4 The object `Sitka89` contains the 1989 data on the same 79 Sitka trees measured on 8 days in 1989. Analyse the 1989 data separately, and then in conjunction with the 1988 data.

10.5 Investigate restricting the covariance matrix of the random effects in model `R.nlme3`.

10.6 For the rabbit data, the PGB was administered in increasing doses at 10 minute intervals. Consider how to allow for serial correlation between measurements in an `nlme` fit and investigate if this might have any effect on the estimates and conclusions. (It may help to start by considering each treatment separately.)

10.7 Consider models that use a multiplicative effect of treatment on the asymptote `A` in the `Rabbit` example, and explore the assumption of a constant residual variance.

10.8 The default estimation method for `lme` is REML but the default method for `nlme` is ML. Why do you suppose this is the case? Investigate the difference changing from ML to REML makes in both the small but complex `Rabbit` data set and the larger but simpler `Sitka` set.

Note that the estimation method can be changed using `update`:

```
fmRML <- update(fmML, est.method="RML",
           start=list(fixed=fixed.effects(fmML),
                      random=random.effects(fmML)))
```

but even starting from this initial value set the calculations can still be quite lengthy. For serially correlated structures such as `sitka.nlme` you may also want to set an initial value for `alpha`.

New exercise

10.9 Exercise 7 discussed non-linear models for an auditory perception experiment. The analysis there did not take account of the differences between individual subjects. Repeat the analysis using non-linear mixed-effects models.

[The models are much easier to specify with `nlme`, but the fitting process is *much* slower. Exercise 8 may help.] Answer

Chapter 11

New exercises

11.1 This data frame `gilgais` was collected on a line transect survey in gilgai territory in New South Wales, Australia. Gilgais are natural gentle depressions in otherwise flat land, and sometimes seem to be regularly distributed. The data collection was stimulated by the question: are these patterns reflected in soil properties? At each of 365 sampling locations on a linear grid of 4 meters spacing, samples were taken at depths 0–10 cm, 30–40 cm and 80–90 cm below the surface. pH, electrical conductivity and chloride content were measured on a 1:5 soil:water extract from each sample.

Produce smoothed maps of the measurements.

11.2 Exercise 8 considered linear regression for the GAG in urine data in data frame `GAGurine`. Consider using a non-linear or smooth regression for the same task.

Chapter 13

New exercises

13.1 Data frame `biopsy` contains data on 699 biopsies of breast tumours, which have been classified as benign or malignant (Mangasarian & Wolberg, 1990). The nine variables on each biopsy are a rating (1 to 10) by the coordinating physician; ratings on one variable are missing for some biopsies.

Analyse these data. In particular, investigate the differences in the two types of tumour, find a rule to classify tumours based solely on the biopsy variables, and assess the accuracy of your rule. [You may need some of the ideas of Chapter 17.]

13.2 Data frame `UScereals` describes sixty-five commonly available breakfast cereals in the USA, based on the information available on the mandatory food label on the packet. The measurements are normalized to a serving size of one American cup.

(i) Is there any way to discriminate among the major manufacturers by cereal characteristics, or do they each have a balanced portfolio of cereals?

(ii) Are there interpretable clusters of cereals?

(iii) Can you describe why cereals are displayed on high, low or middle shelves?

Chapter 15

15.6.1 Use the information gained in the analysis of `beav1` to refine the analysis for `beav2`.

15.6.2 15.6.2If you have access to the S-PLUS module S+SPATIALSTATS, consider how to apply the spatial linear model function `slm` to this problem. Answer

15.6.3 Consider the problem of estimating the effect of seat belt legislation on road accident casualties in the UK considered by Harvey & Durbin (1986). The data (from Harvey, 1989) are in the series `drivers`.

New exercises

15.1 Dataset `austres` is a quarterly series of the number of Australian residents from March 1971 to March 1994. It comes from Brockwell & Davis (1996) who analyse the percentage quarterly changes and consider a fractionally differenced noise model. Explore suitable models in S-PLUS. Answer

15.2 Repeat exercise 11.1 as a time series problem.

15.3 Write a function to fit a spatial autoregression (Ripley, 1981) of the form

$$Y = \theta HY + X\beta + N(0, \sigma^2)$$

by maximum likelihood. (This will be a limited version of the `slm` function of S+SPATIALSTATS.) Apply this to the beaver data example of Section 15.6.

(For fixed θ this is an example of exercise 3, since Y has a multivariate normal distribution with mean $X\beta$ and variance matrix $\sigma^2(S^TS)^{-1}$ where $S = I - \theta H$. Given the eignedecomposition of H, the profile log-likelihood can be written as a function of β alone, and directly optimized over β.) Answer

Chapter 16

New exercises

16.1 Repeat exercise 11.1 as a spatial statistics problem.

Answers to Selected Exercises

Chapter 2

2.1. One way is to use

```
find.val <- function(x, val) seq(along=x)[x==val]
row.names(hills)[find.val(hills$climb, 2100)]
```

although in most cases it is easier to subscript directly by a logical vector, for example `row.names(hills)[hills$climb==2100]`.

2.2. Three solutions are:

```
res <- factor(ftv); levels(res)[-(1:2)] <- "2 or more"
res <- cut(ftv, c(-1,0,1, 10))
     levels(res) <- c("0", "1", "2 or more")
merge.levels(factor(ftv), c(1,2,3,3,3,3))
```

where the first is explained in the help page for `merge.levels`.

2.3. We used

```
mad <- function(y, mu=median(y))
           median(abs(as.vector(y)-mu))
```

where `as.vector` strips off the name attribute which `median` retains in some versions of S-PLUS. (Note that the system function by default calculates $1.4826 \times$ median$|x - \mu|$ which is a consistent estimator of the standard deviation for a Gaussian model.)

2.4. The idea is to find any indices where the strings in `out` match the names of `x` and to use their negatives as an index vector. Matching is such a common problem there is a general function, `match`, to do it.

```
x.in <- x[-match(out, names(x), nomatch=0)]
```

Note the use of `nomatch=0` to generate a zero index (and hence no action) if some string in `out` is not the name of any component in `x`.

This solution relies on the uniqueness of the names of the object (which is not guaranteed in all instances), since `match` will find only the first match. An alternative approach is to match the names in `out` and use logical indexing, by

```
x.in <- x[match(names(x), out, nomatch=0) == 0]
```

2.5. All that is needed is

```
X[rep(1:nrow(X), w), ]
```

2.6. Try this:

```
r <- cor(X)
rc <- format(r)
rc[r < 0.9] <- ""
print(rc, quote=F)
```

Note the use of `format` to get consistent formating of the entries; `format` could also be used to prune the number of significant digits, if required.

2.7. The following solution extends exercise 2.4.
Use `match` and the indexing capabilities. If the data frame is `df` and the variable is `ethnic` the subset you want is

```
df[match(df$ethnic, c(1,3,4,6,7), nomatch=0) > 0, ]
```

The function `is.element` of S-PLUS 4.0 implements this idea as a function.

```
is.element <-
    function(el, set) !is.na(match(el, set, nomatch = NA))
df[is.element(df$ethnic, c(1,3,4,6,7)), ]
```

This is an alternative way to use `match` which has the advantage here of working even when the first argument is empty.

Chapter 3

3.1. To create a barchart of `Exer` we just use `plot(Exer)`, or

```
barplot(table(Exer), names=names(table(Exer)))
```

(Try them to see the differences.) For a pie chart, we need to tabulate the frequencies first:

```
exer.freq <- table(Exer)
exer.freq
 Freq Some None
  115   98   24
```

The command `pie(exer.freq)` will now create a pie chart, but to add labels to the slices we use the `names` argument

```
pie(exer.freq, names=levels(Exer))
```

Adding a legend is accomplished by using `legend` with the `fill` argument:

```
legend(locator(1), names(exer.freq), fill=1:3)
```

For the `Smoke` variable a slightly different approach is needed if we wish to include the missing value in the plot.

```
smoke.freq <- summary(Smoke)
smoke.freq
 Heavy Regul Occas Never NA's
    11    17    19   189    1
```

Since the missing value represents such a small proportion of the data, we highlight it with `explode=5` (because `NA's` is the fifth category) so it is not lost in the pie:

```
pie(smoke.freq, names=names(smoke.freq), explode=5)
legend(locator(1), names(smoke.freq), fill=1:5)
```

Alternatives using Trellis graphics are

```
barchart(~ exer.freq, main="Exercise frequency")
piechart(~ exer.freq, main="Exercise frequency")
piechart(~ smoke.freq, explode = 5)
```

Adding legends and other annotations is left as a further exercise for the reader.

3.2. This is straightforward once the layout is adjusted. We just increased the sizes of the margins which are to hold text.

```
ir <- rbind(iris[,,1], iris[,,2], iris[,,3])[, 3:4]
irs <- c(rep("S", 50), rep("C", 50), rep("V", 50))
par(mar=c(7,7,7,5)) # more space on label sides
plot(ir, type="n", cex=2, lwd=2, tck=-0.02)
title("The Iris Data", cex=2)
text(ir, irs, col=c(rep(2,50), rep(3,50), rep(4, 50)))
```

On-screen the title size is limited by the displayable fonts under the `motif` driver (and probably others).

3.3. Our solution was

```
x <- seq(-pi, pi, length=200)
plot(x, sin(x), type="l", axes=F, ylab="", main="sin(x)")
axis(2, pos=0, yaxs="e", las=1)
axis(1, pos = -1.1, at = pi*seq(-1, 1, 1/4), tck = -0.02,
   labels = c("-Pi", "-3Pi/4", "-Pi/2", "-Pi/4", "0",
              "Pi/4", "Pi/2", "3Pi/4", "Pi"))
```

3.4. We apply this to the annual mean sunspot series. The function `bank` (based on `banking`) computes an aspect ratio `ar` such that the average slope of the line segments `(dx, ar*dy)` is $45°$. We adjust the aspect ratio by reducing the size of one of the axes of the plot.

```
sunsp <- aggregate(sunspots, 1, mean)

bank <- function(dx, dy, iter = 20, tol = 0.5)
{
  dx <- abs(dx)
  dy <- abs(dy)[dx > 0]
  dx <- dx[dx > 0]
  mad <- median(dy/dx)
  ar <- ifelse(mad > 0, mad, 1)
  radians.to.angle <- 180/pi
  for(i in 1:iter) {
    distances <- sqrt(dx^2 + (ar * dy)^2)
    orientations <- atan(ar * dy, dx)
    avg <- (radians.to.angle * sum(orientations *
              distances))/sum(distances)
    if(abs(45 - avg) < tol) break
    ar <- ar * (45/avg)
  }
  ar
}

bankplot <- function(x, y, type="l", ...)
{
  dx <- diff(x/diff(range(x)))
  dy <- diff(y/diff(range(y)))
  ar <- bank(dx, dy)
  pin <- par("pin")
  ar <- ar/(pin[2]/pin[1])
  oldpar <- par(pin=pin*c(1,ar)/max(1,ar))
  on.exit(par(oldpar))
  plot(x, y, type=type, ...)
}

bankplot(time(sunsp), sunsp)
```

3.5. The following function is based closely on `pairs.default`. We use `mfg` to choose which panel to fill in a $n \times n$ grid. As we only ever write to the panels, we need to clear the plot first with a call to `frame`.

```
mypairs <- function(x, labels = dimnames(x)[[2]],
                     panel = points, ...)
{
  doaxis <- function(which, dolabel = T)
    axis(which, outer = T, line = -0.5, labels = dolabel)
  setup <- function(x, y, ...)
    .S(plot(range(x[!is.na(x)]), range(y[!is.na(y)]),
            type = "n", axes = F,  ...), "plot")
  x <- as.matrix(x)
  if(is.character(panel)) panel <- get(panel, mode="function")
  n <- ncol(x)
  oldpar <- par("oma", "mar", "cex", "tck", "mfg", "mgp",
                "mex", "mfrow")
  oldcex <- par("cex")
  CEX <- oldcex * max(7.7/(2 * n + 3), 0.6)
  par(mfrow = c(n, n), mgp = c(2, 0.80, 0), oma = rep(3, 4),
      mar = rep(0.5, 4), tck = -0.03/n)
  on.exit({par(oldpar)})
  par(cex = CEX)
  frame()
  if(length(labels) < n)
    labels <- paste(deparse(substitute(x)),
                    "[,", 1:n, "]", sep = "")
  if(par("pty") == "s") {
    dif <- diff(par("fin"))/2
    if(dif > 0) par(omi = c(dif*n, 0, dif*n, 0) + par("omi"))
    else par(omi = c(0, -dif*n, 0, -dif*n) + par("omi"))
  }
  for(i in 1:n)
    for(j in 1:i) {
      par(mfg = c(i,j,n,n))
      setup(as.vector(x[, j]), as.vector(x[, i]), ...)
      box()
      if(i == n && j < n) doaxis(1)
      if(j == 1 && i > 1) doaxis(2)
      if(i > j) {
          panel(as.vector(x[, j]), as.vector(x[, i]), ...)
      } else {
        par(usr = c(0, 1, 0, 1))
        text(0.5, 0.5, labels[i], cex = 1.5 * CEX)
      }
    }
  invisible()
}
```

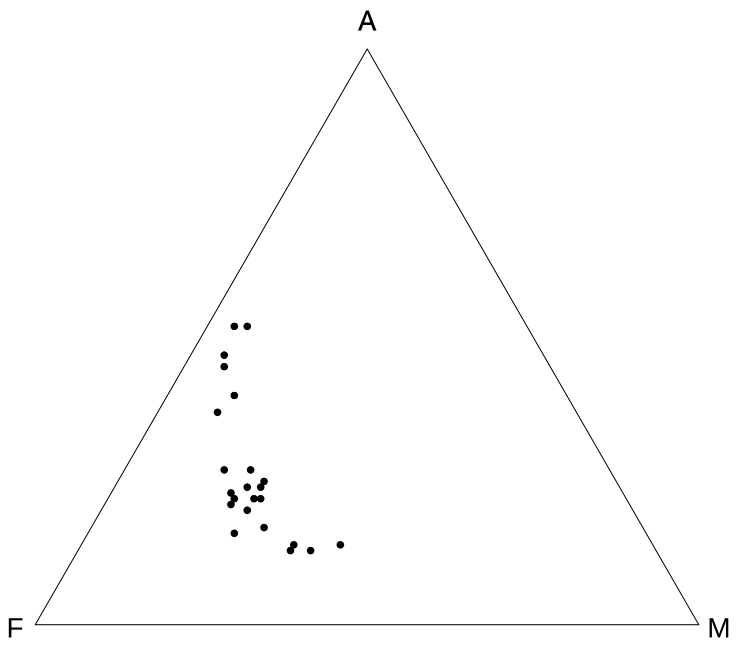
3.6. As a precaution, we rescale the entries in `X` to sum to one.

```
  ternary <- function(X, pch = par("pch"), lcex = 1,
                      add = F, ord = 1:3, ...)
  {
    if(any(X) < 0) stop("X must be non-negative")
    s <- drop(X %*% rep(1, ncol(X)))
    if(any(s<=0)) stop("each row of X must have a positive sum")
    if(max(abs(s-1)) > 1e-6) {
      warning("row(s) of X will be rescaled")
      X <- X / s
    }
    X <- X[, ord]
    s3 <- sqrt(1/3)
    if(!add)
    {
      oldpty <- par("pty")
      on.exit(par(pty=oldpty))
      par(pty="s")
      plot(c(-s3, s3), c(0.5-s3, 0.5+s3), type="n", axes=F,
           xlab="", ylab="")
      polygon(c(0, -s3, s3), c(1, 0, 0), density=0)
      lab <- NULL
      if(!is.null(dn <- dimnames(X))) lab <- dn[[2]]
      if(length(lab) < 3) lab <- as.character(1:3)
      eps <- 0.05 * lcex
      text(c(0, s3+eps*0.7, -s3-eps*0.7),
           c(1+eps, -0.1*eps, -0.1*eps), lab, cex=lcex)
    }
    points((X[,2] - X[,3])*s3, X[,1], ...)
  }
```

This labels the vertices clockwise from the top, but other conventions are possible by altering the argument `ord`. For example, we can reproduce Fig. 1.9 of Aitchison (1986) by

```
    ternary(Skye/100, ord=c(1,3,2))
```

as



Chapter 4

4.1. A simple function to test all potential divisors on each element in turn is

```
is.prime0 <- function(x)
{
   is.prime1 <- function(x)
    if(x <= 10) c(T,T,T,F,T,F,T,F,F,F)[x] else {
           !any(x %% c(2, seq(3,sqrt(x),2)) == 0)}
   if(any(x)  <= 0 || sum(x - floor(x)) > 0)
       error("not all positive integers")
   sapply(x, is.prime1)
}
```

However, most of the divisions will be unnecessary, so we first screen out by small divisors:

```
is.prime <- function(x)
{
   is.prime1 <- function(x)
    if(x <= 10) c(T,T,T,F,T,F,T,F,F,F)[x] else {
           !any(x %% c(2, seq(3,sqrt(x),2)) == 0)}
   is.prime2 <- function(x)
    if(x <= 10) c(T,T,T,F,T,F,T,F,F,F)[x] else {
           !any(x %% c(2, seq(3,min(sqrt(x),29),2)) == 0)}
   if(any(x)  <= 0 || sum(x - floor(x)) > 0)
       error("not all positive integers")
   ind <- sapply(x, is.prime2)
   ind[ind] <- sapply(x[ind], is.prime1)
   ind
}
```

Sieve methods could be employed, but `is.prime` is probably fast enough for most practical purposes.

4.2. First we generate some test data and a naive solution, to test the answers:

```
set.seed(777)
a <- round(runif(10000), 2)
b <- round(runif(50), 2)
res <- integer(length(b))
for (i in seq(along=b)) res[i] <- sum(a <= b[i])
res
```

Then we time this and some alternatives

```
unix.time( for (i in seq(along=b)) res[i] <- sum(a <= b[i]) )
[1] 3.80 0.13 4.00 0.00 0.00
sapply(b, function(b) sum(a <= b))
as.vector(rep(1, length(a)) %*% outer(a, b, "<="))
```

Using `outer` used 20 Mb, and so was slow (32 secs).

The next ideas depend on having sorted `b`, which has negligible cost. The first idea is based on using a series of sieves, the second uses `cut`, and the last two will be familiar to students of two-sample rank tests. (They rely on the ordering of ties being preserved. We could have (did) try `rank`, but this treats ties incorrectly for this purpose and is slower.)

```
sieve <- function(a, b)
{
   l <- sort.list(b); ll <- sort.list(l)
   res <- integer(length(b))
   x <- a
   for (i in seq(along=b)) {
      ind <- x > b[l[i]]
      x <- x[ind]
      res[i] <- length(x)
   }
   (length(a) - res)[ll]
}
usecut <- function(a, b)
{
   l <- sort.list(b); ll <- sort.list(l)
   breaks <-  sort(c(b[l], range(a,b)+c(-1,1)))
   as.vector(cumsum(table(cut(a, breaks)))[-(1+length(b))])[ll]
}
interleave <- function(a, b)
{
   l <- sort.list(b); ll <- sort.list(l)
   x <- c(a, b[l])
   ind <- c(rep(1, length(a)), rep(0, length(b)))
   ind <- ind[order(x)]
   cumsum(ind)[ind == 0][ll]
}
viarank <- function(a, b)
{
   l <- sort.list(b); ll <- sort.list(l)
   x <- sort.list(sort.list(c(a, b[l])))
   (x[(length(a)+1):(length(a)+length(b))] - 1:length(b))[ll]
}
```

The order of merit varies with the lengths of the vectors. Timings in CPU seconds on a SparcStation IPC were:

length(a)	length(b)	naive	sapply	sieve	usecut	interleave	viarank
10^4	50	4.0	4.0	3.0	1.7	1.0	1.3
10^5	20	8.3	15.8	13.8	14.0	10.7	15.5
500	250	1.5	1.5	1.6	0.68	0.13	0.13

If the inequality is reversed, we can change the inequality in the first four solutions. For the last three there are difficulties with the handling of ties, most easily addressed by changing the signs of the vectors and using `rev` on the answer.

4.3. The `print` method function in library `MASS` is

```
print.lda <- function(object, ...)
{
   if(!is.null(cl <- object$call)) {
     names(cl)[2] <- ""; cat("Call:\n"); dput(cl)
   }
   cat("\nPrior probabilities of groups:\n")
   print(object$prior, ...)
   cat("\nGroup means:\n")
   print(object$means, ...)
   cat("\nCoefficients of linear discriminants:\n")
   print(object$scaling, ...)
   svd <- object$svd
   names(svd) <- dimnames(object$scaling)[[2]]
   if(length(svd) > 1) {
      cat("\nProportion of trace:\n")
      print(round(svd^2/sum(svd^2), 4),  ...)
   }
   invisible(object)
}
```

Possible `plot` methods are

```
plot1.lda <- function(x, ...)
{
  x$sdev <- x$svd
  invisible(screeplot.princomp(x, ...))
}

plot.lda <- function(obj, xlab="First LD", ylab="Second LD",
  data = get(obj$call$data), ...)
{
  if(!is.null(Terms <- obj$terms)) {
    x <- model.matrix(delete.response(Terms), data)
    m <- model.frame(obj$call$formula, data)
    g <- model.extract(m, "response")
  } else {
    xname <- eval(substitute(obj$call$x))
    x <- as.matrix(eval(parse(text=as.name(xname)),
                        sys.parent()))
    gname <-  eval(substitute(obj$call$grouping))
    g <- eval(parse(text=as.name(gname)), sys.parent())
  }
  ld <- x %*% obj$scaling[,1:2]
  eqscplot(ld, type="n", xlab=xlab, ylab=ylab)
  text(ld, as.character(g))
  ld.means <- obj$means %*% obj$scaling[, 1:2]
  points(ld.means[,1], ld.means[,2], pch = 3, cex = 4)
}
```

Take care when calling `plot.lda` from within a function, as the `sys.parent` code may not be appropriate.

4.4. The S language has no pointers, so we have to arrange for `x <- x+1` to be evaluated in the correct frame. The function

```
incr <- function(x)
   eval(substitute(x <- x+1), local=sys.parent())
```

(Lubinsky's solution) will work, but will leave a new name-value pair in the frame of the calling function (or the working directory if called from the top level). It is exactly equivalent to replacing `incr(a)` by `a <- a + 1`.

We interpret the question to ask that the value of the actual object matching `x` be incremented. To do that, we have to establish where `x` was matched, and perform the evaluation in that frame. The possible choices are the parent frame, frame 1, frame 0 and the search path, but we can only assume that database 1 is writable, so if `x` was found anywhere on the search path we assign in database 1.

```
incr <- function(x)
{
   dx <- deparse(substitute(x))
   if(exists(dx, frame=sys.parent()))
       assign(dx, x+1, sys.parent())
   else if(exists(dx, frame=1)) assign(dx, x+1, 1)
   else if(exists(dx, frame=0)) assign(dx, x+1, 0)
   else assign(dx, x+1, where=1)
   invisible(NULL)
}
```

4.5. A recursive way to do this is to include `v[1]` with all subsets of size `r-1` from `v[-1]`, and to generate all subsets of size `r` from `v[-1]`:

```
subsets <- function(r, n, v = 1:n)
   if(r <= 0) NULL else
   if(r >= n) v[1:n] else
   rbind(cbind(v[1], Recall(r - 1, n - 1, v[-1])),
        Recall(r, n - 1, v[-1]))
```

A follow-up exercise is to write a function that generates the subsets successively rather than simultaneously; given one subset it should produce the next in lexicographic order.

4.6. Look at the functions `predict.qda` and `predict.lda`, which perform this calculation. Since for vectors $\boldsymbol{x}$ and $\boldsymbol{m}$,

$$\|\boldsymbol{x} - \boldsymbol{m}\|^2 = \|\boldsymbol{x}\|^2 + \|\boldsymbol{m}\|^2 - 2\boldsymbol{m}^T\boldsymbol{x}$$

we can ignore the first term in the comparison and only compute the other two. Finally, the function

```
which.is.min <- function(x) seq(length(x))[x == min(x)]
```

will pick out the minimum if used with `apply`.

4.7. There is a lazy way to do this using the system function `expand.grid`:

```
expand.factors <- function(...)
{
    dfr <- expand.grid(lapply(list(...), levels))
    names(dfr) <- as.character(match.call()[-1])
    dfr
}
```

This assumes the function will be called with simple factor names as the actual arguments. A more illuminating way to do this uses a pair of nested `for` loops:

```
expand.factors <- function(...)
{
  lev <- lapply(list(...), levels)
  df <- NULL
  for(i in lev)  {
      df0 <- df; df <- NULL
      for(j in i)
        df <- rbind(df, cbind(df0, j))
  }
  df <- as.data.frame(df)
  names(df) <- as.character(match.call()[-1])
  df
}
```

A recursive solution is possible, but probably no more elegant. As a follow-up exercise you might add some error protection and flexibility. For example, how should arguments that are not factors be handled?

4.8. We can generate most of the data frame using answer 5, by

```
OMEf <- OME[rep(1:nrow(OME), OME$Trials),]
```

To generate the `Resp` column is slightly trickier: we used

```
attach(OME)
OMEf$Resp <- unlist(lapply(1:length(Trials), function(i)
   c(rep(1, Correct[i]), rep(0, Trials[i] - Correct[i]))))
OMEf <- OMEf[, -match(c("Correct", "Trials"), names(OMEf))]
```

It is possible to fully vectorize this, for example by

```
OMEf$Resp <- rep(rep(c(1,0), length(Trials)),
                 t(cbind(Correct, Trials-Correct)))
```

where the matrix transpose is a ‘trick’ to interleave the two vectors. This approach is significantly faster (0.21 secs *versus* 18 secs on a SparcStation IPC), but the thinking time was much longer.

Another approach is to generate all the 1’s before the 0’s, then sort on the row names.

```
R <- c(rep(1, sum(Correct)), rep(0, sum(Trials-Correct)))
id <- c(rep(seq(along=Trials), Correct),
        rep(seq(along=Trials), Trials - Correct))
OMEf$Resp <- R[sort.list(id)]
```

This took 0.45 secs.

4.9. Here are some solutions with timings on a SparcStation IPC; `rowsum` is a builtin function.

```
>  # 2000 partial sums
> partial.sum.sizes <- trunc(abs(rnorm(2000, 50, 10)))
> grouping <- rep(1:2000, partial.sum.sizes)
> length(grouping)
[1] 98985          # length of the data vector
> data <- rnorm(length(grouping))

# comparison of execution times on SparcStation IPC
> unix.time( tapply(data, grouping, sum) )
[1] 18.13  0.87 21.00  0.00  0.00
> unix.time( unlist(lapply(split(data, grouping), sum)) )
[1] 5.96 0.20 7.00 0.00 0.00
> unix.time( rowsum(data, grouping) )
[1] 216.97   2.66 240.00   0.00   0.00
> unix.time(
   diff(c(0, cumsum(data)[cumsum(partial.sum.sizes)])) )
[1] 0.93 0.17 1.00 0.00 0.00
```

Note that using `split` and `lapply` is faster than using `tapply`, but vectorizing is the fastest, and also used least memory. Using the builtin function is in this case by far the least efficient, and used 14 Mb of memory.

4.10. A one-line solution is

```
rmultinomial0 <- function(n, p)
                   table(sample(seq(along=p), n, T, p))
```

However, if speed is important `table` is rather slow, and we can call `tabulate` directly. For example,

```
rmultinomial1 <- function(reps=1, n, p)
{
  x <- matrix(sample(seq(along=p), reps*n, T, p), nrow = reps)
  assign("k", length(p), frame = 1)
  apply(x, 1, function(x) tabulate(x, k))
}
```

will generate `reps` simulations from our multinomial distribution. Actually, `apply` itself is slow, and we can improve this using `tabulate` alone.

```
rmultinomial2 <- function(reps=1, n, p)
{
  tab <- tabulate(sample(length(p), reps*n, T, p) +
                 length(p) * (1:reps - 1),
                 nbins = reps * length(p))
  dim(tab) <- c(length(p), reps)
  matrix(tab, length(p), reps)
}
```

The difference can be marked: for 1000 replications these methods took 13, 20 and 6 seconds respectively. The gain in using `tabulate` is squandered in using `apply`. It is relatively straightforward to extend `rmultinomial2` so that n can vary between replications.

4.11. One direction, from Unix to MS-DOS, is easy on a Windows version of S-PLUS; use the internal code for `dos_sed` found in the functions `win3` and `dos`. But this only applies in one direction, and not on Unix. The name of that code gives a clue; seasoned Unix programmers would immediately call `sed` via `unix`. There do exist versions of `sed` that run under Windows, but they cannot be assumed to be present. So we need a solution in S.

The character vectors in S are handled as units, so the most effective way to solve this exercise is to 'explode' the path into a vector of single characters, manipulate this and use `paste` to re-assemble the path.

```
path.d2u <- function(path) {
  nc <- nchar(path)
  tmp <- substring(path, 1:nc, 1:nc)
  tmp[tmp == "\\"] <- "/"
  paste(tmp, collapse="")
}
```

This exploits a vectorization of `substring` that goes a little beyond its documented actions. The changes for `path.u2d` should be obvious.

There is a system function `AsciiToInt` that takes a character string and returns a numeric vector of the ASCII positions of the characters. This is often useful for this type of calculation (see the functions in our library `helpfix`, but unfortunately there is no converse function.

4.12. The `print` methods such as `print.lm` print the call in a suitable form, but they use `dput`, an internal function that will only output to a file. So one solution is to write to a file and scan the result back in.

```
call2char <- function(x)
{
  file <- tempfile("call2char")
  on.exit(unlink(file))
  dput(x, file=file)
  paste(scan(file = file, what = "", sep = "\n"),
              collapse = "")
}
```

We have to take care, as the printed version could cover more than one line. Sometimes if the `print` method produces the output you want as a character vector the only simple thing to do is to use `sink` and `scan`.

An alternative is to note that a call object is a list, the first component being the function name and remaining components the arguments, with component names the argument names (if they are named). We can unravel this structure by

```
call2char <- function(x)
{
  args <- x[-1]
  argnames <- names(args)
  havenames <- argnames != ""
  args[havenames] <- paste(argnames[havenames], args[havenames],
                           sep = " = ")
  paste(x[[1]], "(",
        paste(unlist(args), collapse=", "),
        ")", sep="")
}
```

The final paste needs to be of a vector and not a list to have the desired effect, hence the use of `unlist`.

These solutions are instructive, but in this case are overkill as using `deparse(x)` or `as.character(as.name(x))` will produce the desired character string[1]. The second version of `call2char` may be used as a template for other styles of printing a call.

The same ideas work for printing a formula; these have mode `call` (to the function `"~"`) but class `"formula"`. Thus component 2 is the left-hand side and component 3 the right-hand side. These components may themselves be of mode `name` or mode `call`. For example

```
> as.character(as.name(x ~ y + z))
[1] "x ~ y + z"
> (x ~ y + z)[[2]]
x
> (x ~ y + z)[[3]]
y + z
> mode((x ~ y + z)[[3]])
[1] "call"
> unlist((x ~ y + z)[[3]])
[1] "+" "y" "z"
> mode((x ~ y + z)[[2]])
[1] "name"
> mode((log(x) ~ y + z)[[2]])
[1] "call"
```

[1] provided it is not too long for `as.name` which will truncate it, apparently to the current setting of `options("width")`. `deparse` may give a vector of character strings, one for each line.

Chapter 5

5.2. The answers if the shape parameters are known are easy using `ppoints`.

```
qqgamma <- function(x, shape, ...)
  plot(qgamma(ppoints(x), shape), sort(x), ...)
qqweibull <- function(x, shape, ...)
  plot(qweibull(ppoints(x), shape), sort(x), ...)
```

To fit a gamma we can use the function `gamma.mle1` of the answer to exercise 8, by

```
qqgamma <- function(x, shape = gam.mle(x),
  xlab = paste("Quantiles of gamma(",
               format(shape, digits=3), ")", sep=""),
  ylab = deparse(substitute(x)), ...)
{
  gam.mle <- function(x) gamma.mle1(x)$alpha
  plot(qgamma(ppoints(x), shape), sort(x),
       xlab=xlab, ylab=ylab, ...)
}
```

For a Weibull we can fit using `survreg`, converting from its parametrization to a more standard one.

```
qqweibull <- function(x, shape = wei.shape(x),
  xlab = paste("Quantiles of Weibull(",
                format(shape, digits=3), ")", sep=""),
  ylab = deparse(substitute(x)), ...)
{
  wei.shape <- function(x) exp(-survreg(Surv(x) ~ 1)$parms)
  plot(qweibull(ppoints(x), shape), sort(x),
       xlab=xlab, ylab=ylab, ...)
}
```

It is possible to avoid estimating the shape parameter in this case, as a QQ-plot of any Weibull against a Weibull(1,1) is a straight line *on a log-log scale*. Thus it is possible to assess the fit of a Weibull (of any shape) by `qqweibull(x, 1, log="xy")`. In any case, a log-log plot is desirable for small values (less than 0.5) of the shape parameter as those distributions have a very long right tail.

It is easy to produce Trellis versions of these plots using `qqmath`, with a common shape parameter across panels.

5.3. See the function `dpi` in Wand's library `KernSmooth`.

Chapter 6

6.1. Here is an example for the `hills` dataset of how to find the confidence interval for the fit at each data point.

```
hills.lm <- lm(time ~ dist + climb, data=hills)
hills.pred <- predict(hills.lm, se.fit = T)
hills.ci <- pointwise(hills.pred, coverage = 0.95)
```

The prediction interval is a little trickier. The simplest idea is to add s^2 to the squared standard errors returned by `predict`, noting that s^2 has in fact been stored already.

```
hills.s <- summary(hills.lm)$sigma
hills.pred$se.fit <- sqrt(hills.pred$se.fit^2 +
                       hills.pred$residual.scale^2)
hills.ci <- pointwise(hills.pred, coverage = 0.95)
```

6.2. Most of the work was done in the previous exercise. We will try this out on the data for male cats.

```
Mcats <- cats[cats$Sex == "M", ]
cats.lm <- lm(Hwt ~ Bwt, data=Mcats)
attach(Mcats)
plot(Bwt, Hwt)

conflines.lm <- function(obj, coverage = 0.95, pred = F, ...)
{
# Check for simple linear regression
  xnames <- attr(obj$terms,"term.labels")
  if(length(xnames) != 1)
    stop("First argument is not a simple linear fit")
# Work out the range of the existing plot.
  ux <- par("usr")[1:2]
  xp <- seq(ux[1], ux[2], length = 100)
  newdf <- data.frame(xp)
  names(newdf) <-  xnames
  pr <- predict(obj, newdf, se.fit = T)
  if(pred) {
    pr$se.fit <- sqrt(pr$se.fit + pr$res^2)
  }
  ci <- pointwise(pr, coverage = coverage)
  lines(xp, ci$lower, ...)
  lines(xp, ci$upper, ...)
}
conflines.lm(cats.lm)
conflines.lm(cats.lm, pred=T, lty=2)
```

Figure 6.1 is a Trellis plot, so we cannot add information to it; rather we have to create a new Trellis plot by adding to the panel function. We could do this by operating on the data for each panel, but we will illustrate a more general solution, which allows the pooling of standard errors between the sexes.

```
cats.lm <- lm(Hwt ~ Sex/Bwt - 1, data=Cats)
pr <- predict(cats.lm, se=T)
Cats.ci <- pointwise(pr)
pr$se.fit <- sqrt(pr$se.fit + pr$res^2)
Cats.ti <- pointwise(pr)

prepanel.cats <- function(x, y, subscripts, ...)
{
  xlim <- range(x)
  ylim <- range(y, Cats.ti$fit[subscripts],
                Cats.ti$upper[subscripts],
                Cats.ti$lower[subscripts])
  list(xlim = xlim, ylim = ylim,
  dx = diff(xlim), dy = diff(ylim))
}
panel.cats <- function(x, y, subscripts, ...)
{
  panel.xyplot(x, y, cex = 0.5)
  ord <- order(x)
  lines(x[ord], Cats.ci$fit[subscripts][ord])
  lines(x[ord], Cats.ci$upper[subscripts][ord], lty=3)
  lines(x[ord], Cats.ci$lower[subscripts][ord], lty=3)
  lines(x[ord], Cats.ti$upper[subscripts][ord], lty=2)
  lines(x[ord], Cats.ti$lower[subscripts][ord], lty=2)
}
xyplot(Hwt ~ Bwt | Sex, Cats, aspect = "xy",
  prepanel = prepanel.cats,
  panel = panel.cats,
  xlab = "Body weight (kg)", ylab = "Heart weight (gm)",
  strip = function(...) strip.default(..., style = 1)
)
```

The `prepanel` function is needed both to ensure that the tolerance bands fall inside the display and to allow the slopes of the fitted lines to be used in setting the aspect ratio.

6.3. We choose to use an eigendecomposition of W, as it is more stable than a Choleski factorization, and also makes it easier to use the same code for W or Σ. Let $W = UDU^T$. Then

$$(\boldsymbol{y}-X\boldsymbol{\beta})^T W(\boldsymbol{y}-X\boldsymbol{\beta}) = (\boldsymbol{y}-X\boldsymbol{\beta})^T UDU^T(\boldsymbol{y}-X\boldsymbol{\beta}) = \|D^{1/2}U^T(\boldsymbol{y}-X\boldsymbol{\beta})\|^2$$

so we can regress $A\boldsymbol{y}$ on AX where $A = D^{1/2}U^T$. If $W = \Sigma^{-1}$ we can take the eigendecomposition of Σ and replace D by D^{-1}. We modify `lm` as necessary.

```
lm.gls <- function(formula, data, W, subset, na.action,
        inverse = F, method = "qr",
        model = F, x = F, y = F, contrasts = NULL, ...)
{
  call <- match.call()
  m <- match.call(expand = F)
  m$W <- m$inverse <- m$method <- m$model <- m$x <-
    m$y <- m$contrasts <- m$... <- NULL
  m[[1]] <- as.name("model.frame")
  m <- eval(m, sys.parent())
  if(method == "model.frame") return(m)
  Terms <- attr(m, "terms")
  Y <- model.extract(m, response)
  X <- model.matrix(Terms, m, contrasts)
  n <- nrow(X)
  if(any(dim(W) != c(n, n))) stop("dim(W) is not correct")
  eW <- eigen(W, T)
  d <- eW$values
  if(any(d <= 0)) stop("W is not positive definite")
  A <- diag(d^ifelse(inverse, -0.5, 0.5)) %*% t(eW$vector)
  fit <- lm.fit(A %*% X, A %*% Y, method, ...)
  fit$terms <- Terms
  fit$call <- call
  if(model) fit$model <- m
  if(x) fit$x <- X
  if(y) fit$y <- Y
  attr(fit, "na.message") <- attr(m, "na.message")
  class(fit) <- c("lm.gls", class(fit))
  fit
}
```

Our task is not over, since we need to be able to do something useful with the output. However, much of the `print` and `summary` methods for class `"lm"` are based on the stored results for the transformed problem and so are approximately correct. The fitted values and residuals are not simply related to the original problem.

We can test this with an example from Section 15.6. There we fitted a regression with autoregressive errors, and the covariance matrix for AR(1) errors is proportional to $(\alpha^{|i-j|})$.

```
alpha <- 0.8255; n <- 100
arow <- c(1, alpha^(1:n))
B <- matrix(c(rep(arow, n-1),1), n,n, byrow = T)
B[lower.tri(B)] <- 0
B <- B + t(B) - diag(n)
beav.gls <- lm.gls(temp ~ activ, W = B , inverse = T)
> summary(beav.gls)

Call: lm.gls(formula = temp ~ activ, W = B, inverse = T)
```

```
Coefficients:
                Value Std. Error t value Pr(>|t|)
(Intercept)  37.166    0.091    408.776    0.000
      activ   0.669    0.098      6.809    0.000
```

This is reasonably consistent with the results of Section 15.6, where we noted appreciable end effects.

An important special case is for W a diagonal matrix. As a extension of the exercise modify `lm.gls` to allow the user to specify this case by supplying a vector of weights in `W` rather than a matrix. Note that `lm` can handle this can by the use of (case) weights.

6.4. Recall what ridge regression does (Brown, 1994, Sen & Srivastava, 1990). Instead of fitting $X\beta$ to Y by least squares, it solves $[X^TX+\lambda I]\beta = X^TY$. (The case $\lambda = 0$ is the least-squares solution, but the ridge constant λ is positive in ridge regression.) Suppose X is an $n\times p$ matrix. Then the ridge regression problem is equivalent to the regression of Y' on X' where

$$X' = \begin{bmatrix} X \\ \sqrt{\lambda}I \end{bmatrix}, \qquad Y' = \begin{bmatrix} Y \\ 0 \end{bmatrix}$$

Thus we can implement ridge regression by adding p imaginary observations of 0, with $\sqrt{\lambda}$ as the value of the ith regressor and the others zero, for $i = 1, \dots, p$. Conventionally ridge regression is applied to the data with the mean removed and scaled so that the columns of X have constant length. (Any intercept term must then be removed.)

There is another approach that is more efficient if we need multiple values of λ, for example to plot a ridge trace or to choose λ by cross-validation. Let $X = U\Lambda V^T$ be the singular-value decomposition of X. Then $[X^TX+\lambda I]\beta = X^TY$ may be rewritten as $V[\Lambda^2+\lambda]V^T\beta = V\Lambda U^TY$ and hence $V^T\hat\beta_\lambda = \Lambda/(\Lambda^2+\lambda)U^TY = \Lambda^2/(\Lambda^2+\lambda)V^T\beta_{LS}$. We implement this for a vector of values of λ, and compute some statistics to help choose λ, from Brown (1994, pp. 63–64).

```
lm.ridge <- function(formula, data, subset, na.action,
    lambda = 0, model = F, x = F, y = F, contrasts = NULL, ...)
{
  call <- match.call()
  m <- match.call(expand = F)
  m$model <- m$x <- m$y <- m$contrasts <-
    m$... <- m$lambda <- NULL
  m[[1]] <- as.name("model.frame")
  m <- eval(m, sys.parent())
  Terms <- attr(m, "terms")
  Y <- model.extract(m, response)
  X <- model.matrix(Terms, m, contrasts)
  n <- nrow(X); p <- ncol(X)
  if(Inter <- attr(Terms, "intercept"))
```

```
    {
      Xm <- apply(X[, -Inter], 2, mean)
      Ym <- mean(Y)
      p <- p - 1
      X <- X[, -Inter] - rep(Xm, rep.int(n, p))
      Y <- Y - Ym
    } else Ym <- Xm <- NA
    Xscale <- drop(rep(1/n, n) %*% X^2)^0.5
    X <- X/rep(Xscale, rep.int(n, p))
    Xs <- svd(X)
    rhs <- t(Xs$u) %*% Y
    d <- Xs$d
    lscoef <-  Xs$v %*% (rhs/d)
    lsfit <- X %*% lscoef
    resid <- Y - lsfit
    s2 <- sum(resid^2)/(n - p - Inter)
    HKB <- (p-2)*s2/sum(lscoef^2)
    LW <- (p-2)*s2*n/sum(lsfit^2)
    k <- length(lambda)
    div <- d^2 + rep(lambda, rep.int(p,k))
    a <- (d*rhs)/div
    dim(a) <- c(p, k)
    coef <- Xs$v %*% a
    dimnames(coef) <- list(names(Xscale), format(lambda))
    GCV <- apply((Y - X %*% coef)^2, 2, sum)/
           (n-apply(matrix(d^2/div,p), 2, sum))^2
    structure(list(coef = drop(coef), scales = Xscale,
       Inter = Inter, lambda = lambda, ym = Ym, xm = Xm,
       GCV = GCV, kHKB = HKB, kLW = LW), class="ridgelm")
  }

  print.ridgelm <- function(obj)
  {
    scaledcoef <- t(as.matrix(obj$coef / obj$scales))
    if(obj$Inter) {
      inter <- obj$ym - scaledcoef %*% obj$xm
      scaledcoef<- cbind(Intercept=inter, scaledcoef)
    }
    print(drop(scaledcoef))
  }

  choose <- function(obj)
  UseMethod("choose")

  choose.ridgelm <- function(obj)
  {
    cat("modified HKB estimator is", format(obj$kHKB), "\n")
    cat("modified L-W estimator is", format(obj$kLW), "\n")
    GCV <- obj$GCV
    if(length(GCV) > 0) {
      k <- seq(along=GCV)[GCV==min(GCV)]
      cat("smallest value of GCV  at",
          format(obj$lambda[k]), "\n")
    }
  }

  plot.ridgelm <- function(obj)
  {
    matplot(obj$lambda, t(obj$coef), type = "l")
  }
```

We can apply this to the celebrated Longley data, get a ridge trace and some estimates of λ.

```
longley <- data.frame(y = longley.y, longley.x)
lm.ridge(y ~ ., longley)
plot(lm.ridge(y ~ ., longley,
              lambda = seq(0,0.1,0.001)))
choose(lm.ridge(y ~ ., longley,
              lambda = seq(0,0.1,0.0001)))
modified HKB estimator is 0.0042754
modified L-W estimator is 0.032295
smallest value of GCV  at 0.0028
```

There is only a little evidence for the necessity to use ridge regression here, but it can be seen as an alternative to variable selection.

Chapter 9

9.1. The code from the First Edition follows.

```
attach(wtloss)
plot(Days, Weight, xlab= "days", ylab ="weight (kg)",
   xlim=c(0,730), ylim=c(70, 200))
xx <- seq(0, 730, 10)
lines(xx, 81.37+ 102.68 * 2^(-xx/141.91))
wtloss.quad <- lm(Weight ~ poly(Days, 2))
lines(xx, predict.gam(wtloss.quad, data.frame(Days=xx)), lty=2)
wtloss.cub <- lm(Weight ~ poly(Days, 3))
lines(xx, predict.gam(wtloss.cub, data.frame(Days=xx)), lty=3)
legend(locator(1), c("exponential", "quadratic", "cubic"),
    lty=1:3)
```

Note the use of `predict.gam` to get valid predictions.

9.2. To find the curvatures we need to compute the Hessian, so `expn2` is computed with `deriv3` rather than `deriv`.

```
expn2 <- deriv3(~ b0 + b1*((w0 - b0)/b1)^(x/d0),
         c("b0","b1","d0"), function(b0, b1, d0, x, w0) {})
wtloss.init <- function(obj, w0) {
  p <- coef(obj)
  d0 <-  - log((w0 - p["b0"])/p["b1"], 2) * p["th"]
  c(p[c("b0", "b1")], d0 = as.vector(d0))
}
for(w0 in c(110, 100, 90)) {
    fm <- nls(Weight ~ expn2(b0, b1, d0, Days, w0),
              wtloss, start = wtloss.init(wtloss.gr, w0))
    print(plot(profile(fm)))
    print(rms.curv(fm))
  }
```

9.3. [From the fourth printing of the First Edition.]

```
> attach(stormer,1)
> z1 <- Wt/Viscosity
> z2 <- -1/Viscosity
> detach(1,save="stormer")
> attach(stormer)
> storm.gm <- glm(Time ~ z1 + z2 - 1,
     family=quasi(link=inverse, variance=constant),
     data=stormer, trace=T, eps=1.0e-10)
GLM    linear loop 1: deviance = 860.92
GLM    linear loop 2: deviance = 825.06
GLM    linear loop 3: deviance = 825.05
GLM    linear loop 4: deviance = 825.05
> g <- coef(storm.gm)
> b <- coef(storm.fm)
> b0 <- c(1/g[1], g[2]/g[1])
> cbind(b,b0)
         b      b0
z1 29.4013 29.4013
z2  2.2182  2.2183
```

To find the standard errors we used

```
> J <- matrix(c(-1/g[1]^2, -g[2]/g[1]^2, 0, 1/g[1]), 2, 2)
> J %*% vcov(storm.gm) %*% t(J)
         [,1]     [,2]
[1,]  0.83820 -0.56055
[2,] -0.56055  0.44292
```

Note that to achieve agreement to this accuracy we had to tighten the convergence criteria for the `glm` fit by setting `eps=1.0e-10`. With the default convergence criteria there is agreement to about 3 significant digits.

9.4. [From the First Edition.]
We fit the model using the stable parametrization. Good initial values are always easy to find by estimating the mean at the required points by an approximating linear model.

```
> stab <- deriv3(~ ((u0*u2-u1^2) +
    (u0-u1)^2 *((u1-u2)/(u0-u1))^((x-40)/80))/(u0-2*u1+u2),
    c("u0","u1","u2"), function(x, u0, u1, u2) NULL)
> mu <- predict(lm(Weight ~ Days+Days^2, data=wtloss),
    newdata=data.frame(Days=c(40,120,200)))
> names(mu) <- paste("u", 0:2, sep="")
> wtloss.st <- nls(Weight ~ stab(Days, u0, u1, u2),
    start=mu, data=wtloss, trace=T)
43.3655 : 166.18 138.526 119.742
39.2447 : 165.834 138.515 120.033
> rms.curv(wtloss.st)
Parameter effects: c^theta x sqrt(F) = 0.0101
        Intrinsic: c^iota  x sqrt(F) = 0.0101

> summary(wtloss.st)$correlation
         u0      u1       u2
u0  1.00000 0.43675 -0.11960
u1  0.43675 1.00000  0.25806
u2 -0.11960 0.25806  1.00000
> plot(profile(wtloss.st))
```

9.7. Some of the children were tested at more than one age, so first we generate unique IDs for each experiment.

```
aa <- factor(OME$Age)
ab <- 10*OME$ID + unclass(aa)
ac <- unclass(factor(ab))
OME$UID <- as.vector(ac)
OME$UIDn <- OME$UID + 0.1*(OME$Noise=="incoherent")
rm(aa, ab, ac)
```

Our first model is least-squares fitting to the success probabilities.

```
fp1 <- deriv(~ 0.5 + 0.5/(1 + exp(-(x-L75)/scal)),
            c("L75", "scal"),
            function(x,L75,scal) NULL)
```

The effective range of a logistic is about ± 3 times `scal`, so by inspecting the data we can choose initial values of `L75` as 45 and `scal` as 3. It seems appropriate to analyse the two types of noise stimulus separately, at least initially.

```
> nls(Correct/Trials ~ fp1(Loud, L75, scal),
      data=OME[OME$Noise=="coherent",],
      start=c(L75=45, scal=3))
    L75   scal
```

```
  47.993 1.2594
> nls(Correct/Trials ~ fp1(Loud, L75, scal),
      data=OME[OME$Noise=="incoherent",],
      start=c(L75=45, scal=3))
    L75   scal
  38.866 2.1702
```

This suggests fixing on `scal = 2`, and fitting a separate `L75` for each experiment[2]. We used `nlsList`, and allow that a small proportion of fits will fail.

```
OMEi <- OME
parameters(OMEi) <- list(L75=45)
fp2 <- deriv(~ 0.5 + 0.5/(1 + exp(-(x-L75)/2)),
            "L75", function(x,L75) NULL)
OMEi.nls <- nlsList(Correct/Trials ~ fp2(Loud, L75),
   data = OMEi, cluster = ~UIDn, control = list(maxiter=100))
tmp <- sapply(OMEi.nls, function(X)
              {if(is.null(X)) NA else as.vector(X$param)})
OMEif <- data.frame(UID = round(as.numeric((names(tmp)))),
         Noise = rep(c("coherent", "incoherent"), 110),
         L75 = as.vector(tmp))
OMEif$Age <- OME$Age[match(OMEif$UID, OME$UID)]
OMEif$OME <- OME$OME[match(OMEif$UID, OME$UID)]
```

This provides a data frame of the result of each experiment to which we can apply standard linear models. (The precise results will vary by platform, and it may be necessary to exclude 'silly' values such as -39 dB.) For example, we can consider if `L75` varies linearly with `Age` by

```
options(contrasts=c("contr.treatment", "contr.poly"))
summary(lm(L75 ~ Noise/Age, data=OMEif, na.action=na.omit))
```

and if the `OME` groups (only defined at ages 30 and 60 months) differ by

```
summary(lm(L75 ~ Noise/(Age + OME), data=OMEif,
           subset=Age >=30 & Age <= 60,
           na.action=na.omit, singular.ok=T), cor=F)
```

The analysis so far does not take the varying number of trials into account. We can do a weighted least-squares analysis by, for example

```
fpl75 <-
deriv(~ sqrt(n)*(r/n - 0.5 - 0.5/(1 + exp(-(x-L75)/scal))),
      c("L75", "scal"), function(r,n,x,L75,scal)NULL)
nls(0 ~ fpl75(Correct, Trials, Loud, L75, scal),
    data=OME[OME$Noise=="coherent",],
    start=c(L75=45, scal=3))
```

[2] In principle it would be better to fit a combined `nls` model with a separate `L75` for each level of `UIDn` and a common value of `scal`. This can be specified by `Correct/Trials ~ 0.5 +0.5/(1 + exp(-(Loud - L75[UIDn])/scal))` but failed to converge.

```
    L75   scal
 47.798 1.2962
nls(0 ~ fpl75(Correct, Trials, Loud, L75, scal),
    data=OME[OME$Noise=="incoherent",],
    start=c(L75=45, scal=3))
    L75   scal
 38.553 2.0781

fpl75age <- deriv(~ sqrt(n)*(r/n -  0.5 - 0.5/
                     (1 + exp(-(x-L75-slope*age)/scal))),
   c("L75", "slope", "scal"),
   function(r,n,x,age,L75,slope,scal) NULL)
OME.nls1 <- nls(0 ~ fpl75age(Correct, Trials, Loud, Age,
                             L75, slope, scal),
    data=OME[OME$Noise=="coherent",],
    start=c(L75=45, slope=0, scal=2))
    L75     slope   scal
 48.682 -0.028716 1.2596
sqrt(diag(vcov(OME.nls1)))
[1] 0.61093 0.01666 0.17565

OME.nls2 <-nls(0 ~ fpl75age(Correct, Trials, Loud, Age,
                            L75, slope, scal),
               data=OME[OME$Noise=="incoherent",],
               start=c(L75=45, slope=0, scal=2))
   L75    slop    scal
 41.73 -0.10006 1.9796
sqrt(diag(vcov(OME.nls2)))
[1] 0.495592 0.013484 0.244558
```

and similarly for the individual fits. It would also be possible to extract standard errors for the individual `L75` estimates from the results of `nlsList`.

9.8. Let us write the gamma density as

$$f(x;\lambda,\alpha) = \lambda^\alpha x^{\alpha-1} e^{-\lambda x}/\Gamma(\alpha) \qquad \text{on } [0,\infty)$$

Then the log-likelihood is

$$L(\lambda,\alpha) = \sum_i \Big[\alpha\log\lambda + (\alpha-1)\log x_i - \lambda x_i - \log\Gamma(\alpha)\Big]$$

Reasonable initial estimates are given by the moment estimators $\mu = \alpha/\lambda, \sigma^2 = \alpha/\lambda^2$ so $\hat\lambda = \overline{x}/s^2, \hat\alpha = \overline{x}^2/s^2$. Thus a first approach might be

```
gamma.mle0 <- function(x)
{
  nloglik <- function(theta, x)
     - (theta[2] - 1)*sum(log(x)) + theta[1]*sum(x) -
     length(x) * (theta[2]*log(theta[1]) - lgamma(theta[2]))
```

```
    xbar <- mean(x)
    lambda0 <- xbar/var(x); alpha0 <- xbar*lambda0
    res <- nlminb(c(lambda0, alpha0), nloglik, lower=c(0,0), x=x)
    list(lambda = res$par[1], alpha = res$par[2],
         loglik = -res$objective)
}
```

Such a function has been posted to `S-news`, but it can be improved in a number of ways. The sufficient statistic $(\sum x_i, \sum \log x_i)$ is computed many times. The range for the parameters is not really $[0,\infty)$ but $(0,\infty)$, and we would do better to take $\theta = (e^\lambda, e^\alpha)$; at the very least we should give a lower limit at which `nloglik` can be evaluated. We could use gradient information in the calculation, but if we compute derivatives we find $\hat\lambda = \alpha/\overline{x}$ for given α, so we can reduce the problem to maximizing

$$L(\hat\lambda(\alpha), \alpha) = n\alpha \log \alpha/\overline{x} + (\alpha - 1)\sum \log x_i - n\alpha^2/\overline{x} - n \log\Gamma(\alpha)$$

We can easily find the derivative, but for one-dimensional optimization problems it is not particularly helpful, and `optimize` cannot make use of it.

```
gamma.mle1 <- function(x)
{
    nloglik <- function(alpha, n, xbar, st)
      -(n*alpha*log(alpha/xbar) + (alpha - 1)*st
        - n*alpha - n*lgamma(alpha))

    xbar <- mean(x); n <- length(x); st <- sum(log(x))
    alpha0 <- xbar^2/var(x)
    res <- optimize(nloglik, lower=alpha0/3, upper=alpha0*3,
                    n=n, xbar=xbar, st=st)
    alpha <- res$min
    list(lambda = alpha/xbar, alpha = alpha,
         loglik = -res$objective)
}
```

We minimize minus the log likelihood because `optimize` does not work correctly when maximizing (at least in S-PLUS 3.x).

```
> set.seed(123)
> xg <- rgamma(500, 1.4)
> unix.time(gamma.mle0(xg))
[1] 5.34 0.20 6.00 0.00 0.00
> unix.time(gamma.mle1(xg))
[1] 0.75 0.09 1.00 0.00 0.00
> gamma.mle1(xg)
$lambda:
[1] 0.93358
$alpha:
[1] 1.3737
$loglik:
[1] -678.95
```

An alternative approach using a Newton algorithm is given in the function `gamma.shape.glm` in library `MASS`.

Chapter 10

10.9. We have to use the expanded data frame `OMEf` created in answer 8, as it is not sensible to weight mixed models. We change the parametrization of `scal` to ensure it remains positive: we allow a random effect on log scale for this parameter.

```
fp2 <- deriv(~ 0.5 + 0.5/(1 + exp(-(x-L75)/exp(lsc))),
             c("L75", "lsc"),
             function(x, L75, lsc) NULL)
G1.nlme <- nlme(Resp ~ fp2(Loud, L75, lsc),
     fixed = list(L75 ~ Age, lsc ~ .),
     random = list(L75 ~ ., lsc ~ .),
     cluster=~UID, data=OMEf[OMEf$Noise=="coherent",],
     start = list(fixed=c(L75=c(48, -0.03), lsc=0)), verbose=T)
summary(G1.nlme)$fixed
                     Value Approx. Std.Error z ratio(C)
L75.(Intercept)  48.158851          0.684821   70.32327
        L75.Age  -0.025208          0.020704   -1.21755
            lsc   0.157546          0.167722    0.93932

G2.nlme <- nlme(Resp ~ fp2(Loud, L75, lsc),
     fixed = list(L75 ~ Age, lsc ~ .),
     random = list(L75 ~ ., lsc ~ .),
     cluster=~UID, data=OMEf[OMEf$Noise=="incoherent",],
     start = list(fixed=c(L75=c(41, -0.1), lsc=0)), verbose=T)
summary(G2.nlme)$fixed
                    Value Approx. Std.Error z ratio(C)
L75.(Intercept)  41.73298          0.466101   89.5364
        L75.Age  -0.10012          0.012682   -7.8944
            lsc   0.68361          0.116088    5.8887
```

The results are remarkably similar to those by weighted least squares in answer 7. In the case of `G2.nlme` this is not surprising as the estimates of the variances of the random effects are effectively zero. For `G1.nlme` the variances are reasonable but the estimation of the correlation is -0.99858.

Chapter 15

15.6.2.15.6.2 We treat the AR(1) model as a simultaneous spatial autoregression (SAR) although it could also be handled as a conditional autoregression (CAR) process. Note that unlike `arima.mle` this computes an exact (not conditional) likelihood.

```
module(spatial)
beav2.nbr1 <- spatial.neighbor(2:100, 1:99, nregion=100)
slm(temp ~ activ, data = beav2, cov.family = SAR,
   spatial.arglist = list(neighbor=beav2.nbr1))
Coefficients:
              Value Std. Error t value Pr(>|t|)
(Intercept)  36.586   0.126    290.197   0.000
      activ   0.476   0.126      3.774   0.000
rho =  0.99521
Residual standard error: 0.12621 on 97 degrees of freedom

slm(temp ~ activ, data = beav2, subset = 6:100,
   cov.family = SAR,
   spatial.arglist = list(neighbor=beav2.nbr1))

Coefficients:
              Value Std. Error t value Pr(>|t|)
(Intercept)  37.252   0.082    453.049   0.000
      activ   0.596   0.094      6.322   0.000

rho =  0.849
Residual standard error: 0.12067 on 92 degrees of freedom
```

15.1. We start by creating the quarterly percentage differences. Fitting by `ar` suggests that AR(4) and AR(6) models are almost equally good. Thus we try fractional differencing without an AR component and with AR(4) and AR(6) components. Unfortunately the likelihoods are not comparable between `arima.mle` and `arima.fracdiff`, and it seems the latter cannot be used with a specified degree d of (fractional) differencing. Since with the fractional ARIMA(6, d,0) model the estimate $\hat{d} \approx 0$ we can guess that AR(6) has AIC approximately $-2 \times 122.7 + 2 \times 6 = -233.4$ and the I(d) has an AIC of $-2 \times 116.47 + 2 = -231.1$. Thus we would choose the ARIMA(6,0,0) model. This differs from the conclusions of Brockwell & Davis (1996), but with such a short series end-effects may be important.

```
> y <- diff(austres)/austres * 100
> ar(y)
$order:
[1] 6

$ar:
          [,1]
[1,]  0.422690
[2,]  0.081845
[3,]  0.124695
[4,]  0.232673
[5,] -0.016759
[6,] -0.199008
```

```
$var.pred:
          [,1]
[1,] 0.0040826

$aic:
 [1] 30.89342  1.77094  2.48377  1.45398  0.53235  1.55605
 [7]  0.00000  1.94228  3.26542  5.07989  5.44845  6.32720
[13]  8.13424  6.68541  5.62113  7.61780  9.52565 11.23940
[19] 13.23219 14.39861

> arima.mle(y-mean(y), model=list(ar=rep(0,6)), n.cond=6)$aic
[1] -218.41
> arima.mle(y-mean(y), model=list(ar=rep(0,4)), n.cond=6)$aic
[1] -218.31
> arima.mle(y-mean(y), model=list(ar=0), n.cond=6)$aic
[1] -215.28

> arima.fracdiff(y-mean(y), model=list(d=0, ar=rep(0,6)))
$model:
$model$d:
[1] 4.583e-05

$model$ar:
[1]  0.3964324  0.1059762  0.1618604  0.2528531 -0.0094455
[6] -0.2029133
    ....
$loglik:
[1] 122.7

> arima.fracdiff(y-mean(y), model=list(d=0))
$model:
$model$d:
[1] 0.43245
    ....
$loglik:
[1] 116.47
```

15.3. We modify `lm.gls` (from answer 3). Computing the (possibly complex) eigenvalues of W in advance makes this a slightly easier calculation. We choose a sensible range for β, and try to ensure that the profile log-likelihood is not evaluated at the extremes of that range.

```
SARfit <- function(formula, data, H, subset, na.action,
        beta.range = c(-Inf, Inf),
        model = F, x = F, y = F, contrasts = NULL, ...)
{
  SAR.fit <- function(beta, n, X, Y, H, d)
    {
      A <- diag(n) - beta* H
```

```
        fit <- lm.fit(A %*% X, A %*% Y)
        RSS <- sum(fit$residual^2)
        log(RSS/n) - 2 * Re(sum(log(1-beta*d)))/n
      }
    call <- match.call()
    m <- match.call(expand = F)
    m$H <- m$beta.range <- m$model <- m$x <-
      m$y <- m$contrasts <- m$... <- NULL
    m[[1]] <- as.name("model.frame")
    m <- eval(m, sys.parent())
    Terms <- attr(m, "terms")
    Y <- model.extract(m, response)
    X <- model.matrix(Terms, m, contrasts)
    n <- nrow(X)
    if(any(dim(H) != c(n, n))) stop("dim(H) is not correct")
    eH <- eigen(H)
    d <- eH$values
    ad <- 1/max(abs(d))
    res <- optimize(SAR.fit, lower=max(-ad, beta.range[1]) + 1e-6,
                    upper=min(ad, beta.range[2]) - 1e-6,
                    n=n, X=X, Y=Y, H=H, d=d)
    beta <- res$minimum
    if(res$message != "normal termination")
      warning(paste("Optimize gave", res$message,
                    "as reason for termination"))
    A <- diag(n) - beta * H
    fit <- lm.fit(A %*% X, A %*% Y, ...)
    fit$terms <- Terms
    fit$call <- call
    if(model) fit$model <- m
    if(x) fit$x <- X
    if(y) fit$y <- Y
    fit$betahat <- beta
    attr(fit, "na.message") <- attr(m, "na.message")
    class(fit) <- c("lm.gls", class(fit))
    fit
  }
```

We can apply this to the beaver data by

```
H <- matrix(0, 100, 100)
H[cbind(1:99, 2:100)] <- 1
SARfit(temp ~ activ, data=beav2, H)
```

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data*. London: Chapman and Hall. 2, 21

Aitkin, M., Anderson, D., Francis, B. and Hinde, J. (1989) *Statistical Modelling in GLIM*. Oxford: Oxford University Press. 10

Baxter, L. A., Coutts, S. M. and Ross, G. A. F. (1980) Applications of linear models in motor insurance. In *Proceedings of the 21st International Congress of Actuaries*, pp. 11–29. Zurich: . 10

Brockwell, P. J. and Davis, R. A. (1996) *Introduction to Time Series and Forecasting*. New York: Springer. 15, 43

Brown, P. J. (1994) *Measurement, Regression and Calibration*. Oxford: Oxford University Press. 5, 34

Cleveland, W. S. (1993) *Visualizing Data*. Summit, NJ: Hobart Press. 2

Ehrlich, I. (1973) Participation in illegitimate activities: a theoretical and empirical investigation. *Journal of Political Economy* **81**, 521–565. 5

Ein-Dor, P. and Feldmesser, J. (1987) Attributes of the performance of central processing units: A relative performance prediction model. *Communications of the ACM* **30**, 308–317. 5

Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. (1995) *Bayesian Data Analysis*. London: Chapman & Hall. 8

Harvey, A. C. (1989) *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press. 15

Harvey, A. C. and Durbin, J. (1986) The effects of seat belt legislation on British road casualties: A case study in structural time series modelling (with discussion). *Journal of the Royal Statistical Society series A* **149**, 187–227. 15

Knight, R. L. and Skagen, S. K. (1988) Agonistic asymmetries and the foraging ecology of bald eagles. *Ecology* **69**, 1188–1194. 7

Mangasarian, O. L. and Wolberg, W. H. (1990) Cancer diagnosis via linear programming-cancer diagnosis via linear programming. *SIAM News* **23**, 1, 18. 14

McLachlan, G. J. and Jones, P. N. (1988) Fitting mixture models to grouped and truncated data via the EM algorithm. *Biometrics* **44**, 571–578. 12

McLachlan, G. J. and Krishnan, T. (1997) *The EM Algorithm and Extensions*. New York: Wiley. 12

Milicer, H. and Szczotka, F. (1966) Age at menarche in Warsaw girls in 1965. *Human Biology* **B38**, 199–203. 7

Natrella, M. (1963) *Experimental Statistics*. Washington, DC: NBS Handbook 91. 4

Raftery, A. E. (1995) Bayesian model selection in social research. In *Sociological Methodology 1995*, ed. P. V. Marsden, pp. 111–196. Oxford: Blackwells. 5

Rice, J. A. (1995) *Mathematical Statistics and Data Analysis*. Second Edition. Belmont, CA: Duxbury Press. 4

Ripley, B. D. (1981) *Spatial Statistics*. New York: John Wiley and Sons. 15

Ripley, B. D. (1994) Neural networks and flexible regression and discrimination. In *Statistics and Images 2*, ed. K. V. Mardia, volume 2 of *Advances in Applied Statistics*, pp. 39–57. Abingdon: Carfax. 5

Ross, G. J. S. (1970) The efficient use of function minimization in non-linear maximum-likelihood estimation. *Applied Statistics* **19**, 205–221. 11

Scheffé, H. (1959) *The Analysis of Variance*. New York: John Wiley and Sons. 6

Sen, A. and Srivastava, M. (1990) *Regression Analysis*. New York: Springer. 5, 34

Svensson, Å. (1981) On the goodness-of-fit test for the multiplicative Poisson model. *Annals of Statistics* **9**, 697–704. 9

Vandaele, W. (1978) Participation in illegitimate activities: Ehrlich revisited. In *Deterrence and Incapacitation*, eds A. Blumstein, J. Cohen and D. Nagin, pp. 270–335. Washington: US National Academy of Sciences. 5

Wand, M. P. (1997) Data-based choice of histogram bin width. *American Statistician* **52**, 59–64. 4