Data Analysis with R Laboratories – Sets of Exercises, with R Code

John Maindonald

July 25, 2006

Files that hold the R code are available from the web site
`http://www.maths.anu.edu.au/~johnm/courses/dm/`

The laboratory exercises make extensive use of datasets from the *DAAG* package. Make sure that it is installed.

For background to laboratory exercises I and II, see the document: *The R System – An Introduction and Overview*.

Laboratory exercises III – X assume the background knowledge that is covered in relevant parts of the document: *Statistical Perspectives on Data Mining* (referred to as SPDM).

Exercises can be grouped as follows:

Introduction to R

I: Simple Data Manipulation and Graphs (Weeks 1 & 2)

II: For Loops, Functions, and Data Exploration (Weeks 1 & 2)

Populations, Samples and Sampling Distributions

III: Populations and Samples – Theoretical and Empirical Distributions – Weeks 2 & 3

IV: Sampling Distributions, and the Central Limit Theorem (Week 3)

Linear Models

V: Linear Models in R (Week 4)

VI: Exploiting the Model Matrix (Week 5)

Data Mining Techniques

VII: The *rattle* package for R. (Week 6)
[The *rattle* package provides a graphical (GTK based) graphical user interface to R.]

Data Summary and Analysis

VIII: Data Summary; Traps for the Unwary (Week 7; August 29)

Models with a Complex Error Structure

VIIII: Multi-level Models (Week 8; September 19)

Discriminant Methods, Ordination & Clustering

IX: Discriminant Methods – Realistic Error Rate Estimation (Week 9)

X: Ordination & Clustering (Week 10)

XI: Data Exploration and Discrimination – Largish Dataset (Week 11)

Contents

I Data Input, Graphs, & Data Manipulation 5

1 Data Input 5

2 Scatterplots, Histograms and Dotplots 5

3 Factors 7

4 Sorting 7

5 Esoterica 8

II For loops, Functions, & Data Exploration 9

1 For loops 9

2 Functions 11

3 Data Exploration – A Further Exercise 14

III Populations and Samples – Theoretical and Empirical Distributions 15

1 Populations and Theoretical Distributions 15

2 Samples and Estimated Density Curves 16

3 Normal Probability Plots 17

4 Boxplots – Simple Summary Information on a Distribution 18

IV Sampling Distributions, and the Central Limit Theorem 19

1 Sampling Distributions 19

2 The Central Limit Theorem 22

V Linear Models in R 25

1 Linear and Other Models in R 25

2 Fitting Straight Lines to Data 25

3 Multiple Explanatory Variables 26

4 Errors in Variables 26

VI Exploiting the Model Matrix 29

1 A One-way classification – eggs in the cuckoo's nest 29

2 Regression Splines **31**

VII Data Summary – Traps for the Unwary 33

1 Multi-way Tables **33**

2 Weighting Effects – Example with a Continous Outcome **35**

VIII Multi-level Models 37

1 Description and Display of the Data **37**

2 Multi-level Modeling **39**

IX Discriminant Methods – Realistic Error Rate Estimation 43

1 rpart Analyses – the Combined `Pima` Dataset **43**

2 rpart Analyses – `Pima.tr` and `Pima.tr` **45**

3 Analysis Using *svm* **47**

4 Analysis Using *randomForest* **48**

5 Prior Proportions **48**

X Ordination & Clustering 51

XI Data Exploration and Discrimination – Largish Dataset 52

1 Data Input and Exploration **52**

2 Tree-Based Classification **54**

3 Use of `randomForest()` **55**

4 Further comments **56**

Part I
Data Input, Graphs, & Data Manipulation

1 Data Input

Exercise 1
Place the files **travelbooks.txt**, **molclock.txt**, **molclock1.txt**, **molclock2.txt** and **houses.txt** in your working directory. Use `read.table()` to read each of them into R. In each case, display the data frame and check that the data have been input correctly.
For a more challenging data input task, input the data from the file **bostonc.txt**.

2 Scatterplots, Histograms and Dotplots

Exercise 2
The data frame `rainforest` (*DAAG* package) has data on four different rainforest species. Use `table(rainforest$species)` to check the names and numbers of the species present. In the following, attention will be limited to the species Acmena *smithii*.
Here are two ways to plot histograms showing the distribution of the diameter at base height:

```
> library(DAAG)
> Acmena <- subset(rainforest, species == "Acmena smithii")
> hist(Acmena$dbh)
> hist(Acmena$dbh, prob = TRUE)
```

The density is a local estimate of the number per unit interval. The second plot is readily overlaid with a density plot, thus:

```
> hist(Acmena$dbh, prob = TRUE, xlim = c(0, 50))
> lines(density(Acmena$dbh, from = 0))
```

Why use the argument `from=0`? What is the effect of omitting it?

Exercise 3
Again using the Acmena *smithii* from the data frame `rainforest`, plot `wood` (wood biomass) vs `dbh` (diameter at breast height), trying both untransformed scales and logarithmic scales.

```
> Acmena <- subset(rainforest, species == "Acmena smithii")
> plot(wood ~ dbh, data = Acmena)
> plot(wood ~ dbh, data = Acmena, log = "xy")
```

Use of the argument `log="xy"` gives logarithmic scales on both the x and y axes. For purposes of adding additional features to the plot, note that logarithms to base 10 are used.
For the second plot, we add a line, thus:

```
> plot(wood ~ dbh, data = Acmena, log = "xy")
> abline(lm(log10(wood) ~ log10(dbh), data = Acmena))
> coef(lm(log10(wood) ~ log10(dbh), data = Acmena))
> coef(lm(log(wood) ~ log(dbh), data = Acmena))
```

Write down the equation that gives the fitted relationship between `wood` and `dbh`.

Exercise 4
The `orings` data frame gives data on the damage that had occurred in US space shuttle launches prior to the disastrous Challenger launch of January 28, 1986. Only the observations in rows 1, 2, 4, 11, 13, and 18 were included in the pre-launch charts used in deciding whether to proceed with the launch. Add a new column to the data frame that identifies rows that were included in the pre-launch charts. Now make three plots of `Total` incidents against `Temperature`:

(a) Plot only the rows that were included in the pre-launch charts.

(b) Plot all rows.

(c) Plot all rows, using different symbols or colors to indicate whether or not points were included in the pre-launch charts.

Comment, for each of the first two graphs, whether and open or closed symbol is preferable. For the third graph, comment on the your reasons for choice of symbols.

Use the following to identify rows that hold the data that were presented in the pre-launch charts:

```
> orings$Included <- logical(23)
> orings$Included[c(1, 2, 4, 11, 13, 18)] <- TRUE
```

The construct `logical(23)` creates a vector of length 23 in which all values are `FALSE`. The following are two possibilities for the third plot; can you improve on these choices of symbols and/or colors?

```
> plot(Total ~ Temperature, data = orings, pch = orings$included +
+     1)
> plot(Total ~ Temperature, data = orings, col = orings$Included +
+     1)
```

Exercise 5
Using the data frame `oddbooks`, use graphs to investigate the relationships between:

(a) weight and volume;

(b) density and volume;

(c) density and page area.

Exercise 6
Look up the help for the lattice function `dotplot()`.

(a) Compare the following:

```
> with(ant111b, stripchart(harvwt ~ site))
> library(lattice)
> stripplot(site ~ harvwt, data = ant111b)
```

Comment on the differences in syntax between the two graphics systems.

(b) Repeat the above plots, using whichever of the two graphics you prefer, but now with the data frame `ant111b`.

3 Factors

Exercise 7
Run the following code:

```
> gender <- factor(c(rep("female", 91), rep("male", 92)))
> table(gender)
> gender <- factor(gender, levels = c("male", "female"))
> table(gender)
> gender <- factor(gender, levels = c("Male", "female"))
> table(gender)
> rm(gender)
```

Explain the output from the final `table(gender)`.
The output is

```
gender
female   male
    91     92

> table(gender)
> gender <- factor(gender, levels = c("Male", "female"))
> table(gender)
> rm(gender)
```

4 Sorting

Exercise 8
Sort the rows in the data frame `Acmena` in order of increasing values of `dbh`.
[Hint: Use the function `order()`, applied to `age` to determine the order of row numbers required to sort rows in increasing order of age. Reorder rows of `Acmena` to appear in this order.]
To see why you might want to reorder the rows in this way, try

```
> plot(wood ~ dbh, data = Acmena)
> with(Acmena, lines(predict(loess(wood ~ dbh)) ~ dbh))
> plot(wood ~ dbh, data = Acmena)
> ord <- order(Acmena$dbh)
> with(Acmena[ord, ], lines(predict(loess(wood ~ dbh)) ~ dbh))
```

Other smoothing functions may return ordered x's, with the corresponding predcted y values, in their output. Try

```
> plot(wood ~ dbh, data = Acmena)
> with(Acmena, lines(lowess(wood ~ dbh)))
> plot(wood ~ dbh, data = Acmena)
> with(Acmena, lines(smooth.spline(wood ~ dbh, spar = 0.5)))
> with(Acmena, lines(smooth.spline(wood ~ dbh, df = 5), col = "red"))
> plot(wood ~ dbh, data = Acmena)
> with(Acmena, panel.smooth(dbh, wood))
```

For each of the functions just noted, what are the parameters that control the smoothness of the curve? What, in each case, is the default?

5 Esoterica

Exercise 9
Bored to tears by now? Here is something a bit different!
The binary arithmetic operators `+`, `-`, `*`, `/` and `^` are implemented as functions. (R is a functional language; albeit with features that compromise its purity as a member of this genre!) Try the following:

```
> 2 + 5
> 10 - 3
> 2/5
> (5 + 2) * (3 - 7)
```

Use this syntax to evaluate `1.25*(8-5)^3`
There are two other binary arithmetic operators – `%%` and `%/%`. Look up the relevant help page, and explain, with examples, what they do. Try

```
> (0:25)%/%5
> (0:25)%%5
```

Of course, the relational operators are also implemented as functions. Write code that demonstrates this.
Note also that `[` is implemented as a function. Try

```
> z <- c(2, 6, -3, NA, 14, 19)
> z[5]
> heights <- c(Andreas = 178, John = 185, Jeff = 183)
> heights[c("Jeff", "John")]
```

Rewrite these using the usual syntax.
Use this syntax to extract, from the data frame `possumsites` (*DAAG*), the altitudes for Byrangery and Conondale.

Part II
For loops, Functions, & Data Exploration

1 For loops

Exercise 1

(a) Create a `for` loop that, given a numeric vector, prints out one number per line, with its square and cube alongside.

(b) Look up `help(while)`. Show how use a `while` loop to achieve the same result.

(c) Show how to do achieve the same result without the use of an explicit loop.

Exercise 2
The following code uses a `for` loop to plot graphs that compare the relative population growth (here, by the use of a logarithmic scale) for the Australian states and territories.

```
> oldpar <- par(mfrow = c(2, 4))
> for (i in 2:9) {
+     plot(austpop[, 1], log(austpop[, i]), xlab = "Year", ylab = names(austpop)[i],
+         pch = 16, ylim = c(0, 10))
+ }
> par(oldpar)
```

Which Australian adminstration(s) showed the most rapid increase in the early years? Which showed the most rapid increase in later years?

Exercise 3
Here is code for the calculations of Exercise 2, but avoiding the use of a loop:

```
> oldpar <- par(mfrow = c(2, 4))
> invisible(sapply(2:9, function(i, df) plot(df[, 1], log(df[,
+     i]), xlab = "Year", ylab = names(df)[i], pch = 16, ylim = c(0,
+     10)), df = austpop))
> par(oldpar)
```

Run the code, and check that it does indeed give the same result.
[By wrapping the code in the function `invisible()`, printed output that gives no useful information is suppressed.]
Note that `lapply()` could be used in place of `sapply()`.

Note that there are several subtleties here:

(i) The first argument to `sapply()` can be either a list (which is, technically, a type of vector) or a vector. Here, we have supplied the vector 2:9

(ii) The second argument is a function. Here we have supplied an anonymous function that has two arguments. The argument `i` takes as its values, in turn, the sucessive elements in the first argument to `sapply`

(iii) Where as here the anonymous function has further arguments, they area supplied as additional arguments to `sapply()`. Hence the parameter `df=austpop`.

Exercise 4
A ramdom sample of 200 values from a normal distribution (with mean 0 and standard deviation 1) can be obtained thus:

```
> y <- rnorm(500)
```

Use the function `hist()` to show the distribution of values.
In the laboratory on distributions, repeated samples of size `n` (e.g., `n=4`, `n=9`) will be taken from such a distribution and the mean calculated for each such sample. For example, the following gives 500 means, each obtained from samples of size `n=4`:

```
> av <- numeric(500)
> for (i in 1:500) {
+     av[i] <- mean(rnorm(4))
+ }
```

Repeat the above calculation, with samples of sizes 9 and 25. For each of the sample sizes 4, 9 and 25, use the function `hist()` to show the distribution of values.

Exercise 5
Here is an alternative way to do the calculations of Exerecise 4. Code is given for samples of size 4:

```
> mat <- matrix(rnorm(500 * 4), nrow = 500)
> av <- apply(mat, 2, mean)
```

Explain why this is this equivalent to the code of Exercise 4.

Exercise 6
This exercise will investigate the relative times for different alternative ways to do a calculation. First, we will create both matrix and data frame versions of a largish data set.

```
> xxMAT <- matrix(runif(480000), ncol = 50)
> xxDF <- as.data.frame(xxMAT)
```

The function `system.time()` will provide timings. The first three numbers that are returned will be of interest; these are the user cpu time, the system cpu time, and the elapsed time. Repeat each calculation several times, and note whether there is variation between repeats. If there is, make the setting `options(gcFirst=TRUE)`, and see whether this leads to more consistent timings. NB: If your computer chokes on these calculations, reduce the dimensions of `xxMAT` and `xxDF`

(a) The following compares the times taken to increase element by 1:

```
> system.time(invisible(xxMAT + 1))[1:3]
> system.time(invisible(xxDF + 1))[1:3]
```

Exercise 6, continued

(b) Now compare the following alternative ways to calculate the means of the 50 columns:

```
> system.time(av1 <- apply(xxMAT, 2, mean))[1:3]
> system.time(av1 <- sapply(xxDF, mean))[1:3]
> system.time({
+     av2 <- numeric(50)
+     for (i in 1:50) av[i] <- mean(xxMAT[, i])
+ })[1:3]
> system.time({
+     av2 <- numeric(50)
+     for (i in 1:50) av[i] <- mean(xxDF[, i])
+ })[1:3]
> system.time({
+     colOFones <- rep(1, dim(xxMAT)[2])
+     av3 <- xxMAT %*% colOFones/dim(xxMAT)[2]
+ })[1:3]
```

(c) Pick one of the above calculations. Vary the number of rows in the matrix, keeping the number of columns constant, and plot each of user CPU time and system CPU time against number of rows of data.

Suggest why the calculation that uses matrix multiplication is so efficient, relative to the other options.

2 Functions

Exercise 7
The following function calculates the mean and standard deviation of a numeric vector.

```
> meanANDsd <- function(x) {
+     av <- mean(x)
+     sdev <- sd(x)
+     c(mean = av, sd = sdev)
+ }
```

Modify the function so that: (a) the default is to use `rnorm()` to generate 20 random normal numbers, and return the standard deviation for those; (b) if there are missing values, the mean and standard deviation are calculated for the remaining values.

Exercise 8
Write a function that does the calculations of Exercises 4 and 5 above. It should take as parameters the sample size (e.g., `n=4`), and the number of samples that should be taken (e.g., `numsamp=500`, and returns the vector of sample means.

Exercise 9

(a) Use `library(MASS)` to attach the *MASS* package. Look up the help page for the data frame `Pima.tr2`, and note the columns in the data frame.

Several of the columns have missing values. Determine the number of missing values in each column, thus:

```
> library(MASS)
> count.na <- function(x) sum(is.na(x))
> count.na(c(1, 5, NA, 5, NA, 8))
> sapply(Pima.tr2, count.na)
```

Write a function that does this last calculation, i.e., it takes a data frame as argument, and returns, for each column, the number of rows where values are missing. Apply this function both to the data frame `Pima.tr2` and to the data frame `cfseal` (*DAAG*).

(b) Modify this function so that it returns, in addition, the number of rows where one or more columns have missing values.

[Hint: Use `complete.cases()` to identify rows where there are no missing values.]

Exercise 10

Write a function that takes as argument an arbitrary numeric vector and returns: (a) the interquartile range; (b) the standard deviation. Run this function 50 times, with each of the following arguments:

(a) 20 randomly chosen values from a normal distribution;

(b) 40 randomly chosen values from a normal distribution;

(c) 20 randomly chosen values from a uniform distribution;

(d) 40 randomly chosen values from a uniform distribution.

In each case, plot the 50 values for the standard deviation against the 50 values of the interquartile range. On the plot, use the code `abline(0,0.75)` to draw the line $y = 0.75x$.
For data from a normal distribution, the standard deviation should be appromimately three quarters of the interquartile range. What happens when the distribution is uniform? What is the theoretical result?

Exercise 11
Data in the data frame `fumig` are from a series of fumigation trials, in which produce was exposed to the fumigant over a 2-hour time period. Concentrations in the chamber were measured at times 5, 10, 30, 60, 90 and 120 minutes. Two different formulae are in use for comparing the concentration-time (c-t) product that measures exposure to the fumigant, one using the the times and concentrations in the data, and the other using the times 15, 30, 60 and 120. The 15-minute concentration has to be estimated by interpolation. The following code does these calculations, and returns the two different estimates of the concentration-time (c-t) product.

```
> "calc.ct" <- function(df = fumig, times = c(5, 10, 30, 60, 90,
+     120), ctcols = 3:8) {
+     usualfac <- c(7.5, 12.5, 25, 30, 30, 15)
+     modfac <- c(20, 25, 30, 30, 15)
+     modtimes <- c(15, 30, 60, 120)
+     require(splines)
+     m <- dim(df)[1]
+     x1 <- times[-1]
+     conc15 <- numeric(m)
+     usualct <- numeric(m)
+     modct <- numeric(m)
+     for (i in 1:m) {
+         y <- unlist(df[i, ctcols])
+         y1 <- y[-1]
+         ct.lm <- lm(y1 ~ ns(x1, 4))
+         xy = data.frame(x1 = c(15, 30, 60, 120))
+         hat <- predict(ct.lm, newdata = xy)
+         conc15[i] <- hat[1]
+         usualct[i] <- sum(usualfac * y)/60
+         modct[i] <- sum(modfac * y1)/60
+     }
+     df <- cbind(usualct = usualct, modct = modct, df[, -ctcols],
+         estconc15 = conc15)
+     df
+ }
```

Examine the code, and the data frame `fumig` that is given as the default argument for the parameter `df`. Load or attach the data set `fumig`, and do the following"

(a) Run the function, with the default arguments, and note the output.

(b) Are fumigant concentration measurements noticeably more variable at some times than at others?

(c) Why was the first time omitted, in fitting the spline curve?

(d) Compare the two different calculations of the concentration-time (`ct`) sum – giving the estimates `usualct` (the 'usual' method) and `modct`) respectively. Is there any systematic bias, in using one method as opposed to the other?

3 Data Exploration – A Further Exercise

Exercise 12
Missing values are an issue for many data sets. Never cavalierly ignore their possible effect on results from an analysis. Ask: "Were observations where values of one or more variables are missing different in some important way from the rest of the data?"

(a) Split the data frame `Pima.tr2` into two data frames – the first consisting of rows where there are no missing values, and the second consisting of rows where there is one or more missing value. Here is how to do this:

```
> anymiss <- complete.cases(Pima.tr2)
> Pima.nomiss <- Pima.tr2[!anymiss, ]
> Pima.miss <- Pima.tr2[anymiss, ]
```

Calculate the mean values of columns other than `Type` for each of these two data frames. For `Type`, use `table()` to compare the relative numbers of the two types.

(b) Use the assignment `Pima.tr2$anymiss <- anymiss` to create a version of the data frame `Pima.tr2` that has `anymiss` as an additional column. Use strip plots to compare values of all columns except `Type`. Are there any columns where the distribution of differences seems shifted for the rows that have one or more missing values, relative to rows where there are no missing values?

(c) Density plots may be a better tool for comparing the distributions, Try the following, first with the variable `npreg` as shown, and then with each of the other columns except `Type`. Note that the comparison for `skin` is not very useful, though it may be educational. Why?

```
> densityplot(~npreg, groups = anymiss, data = Pima.tr2)
```

[For present purposes, it will be adequate to describe a density plot as a smoothed version of a histogram. Density plots, although like histograms open to misuse and misinterpretation, are in general preferable to histograms. Why? What are the traps?]

(d) If some differences are found that are greater than could be expected as a result of chance, what are the implications?
[The graphs are obviously an overly subjective basis for making this judgment. They are however a good start.]

Part III
Populations and Samples – Theoretical and Empirical Distributions

For background, see SPDM: Sections 5-7.

R functions that will be used in this laboratory include:

(a) `dnorm()`: Obtain the density values for the theoretical normal distribution;

(b) `pnorm()`: Given a normal deviate or deviates, obtain the cumulative probability;

(c) `qnorm()`: Given the cumulative probabilty. calculate the normal deviate;

(d) `sample()`: take a sample from a vector of values. Values may be taken without replacement (once taken from the vector, the value is not available for subsequent draws), or with replacement (values may be repeated);

(e) `density()`: fit an empirical density curve to a set of values;

(f) `rnorm()`: Take a random sample from a theoretical normal distribution;

(g) `runif()`: similar to `rnorm()`, but sampling is from a uniform distribution;

(h) `rt()`: similar to `rnorm()`, but sampling is from a t-distribution (the degrees of freedom must be given as the second parameter);

(i) `rexp()`: similar to `rnorm()`, but sampling is from an exponential distribution;

(j) `qqnorm()`: Compare the empirical distribution of a set of values with the empirical normal distribution.

1 Populations and Theoretical Distributions

Exercise 1

(a) Plot the density and the cumulative probability curve for a normal distribution with a mean of 2.5 and SD = 1.5.
[specify `mean=2.5` and `sd=1.5`]

(b) From the cumulative probability curve in (a), read off the area under the density curve between `x=-2` and `x=4`.

(c) Plot the density and the cumulative probability curve for a t-distribution with 3 degrees of freedom. Overlay, in each case, a normal distribution with a mean of 0 and SD=1.
[Replace `dnorm` by `dt`, and specify `df=10`]

(d) Plot the density and the cumulative probability curve for an exponential distribution with a rate parameter equal to 1 (the default). Repeat, with a rate parameter equal to 2. (When used as a failure time distribution; the rate parameter is the expected number of failures per unit time.)

By way of example, here is the code for (a):

```
> curve(dnorm(x, mean = 2.5, sd = 1.5), from = 2.5 - 3 * 1.5, to = 2.5 +
+     3 * 1.5)
> curve(pnorm(x, mean = 2.5, sd = 1.5), from = 2.5 - 3 * 1.5, to = 2.5 +
+     3 * 1.5)
```

2 Samples and Estimated Density Curves

Exercise 2
Use the function `rnorm()` to draw a random sample of 25 values from a normal distribution with a mean of 0 and a standard deviation equal to 1.0. Use a histogram, with `probability=TRUE` to display the values. Overlay the histogram with: (a) an estimated density curve; (b) the theoretical density curve for a normal distribution with mean 0 and standard deviation equal to 1.0. Repeat with samples of 100 and 500 values, showing the different displays in different panels on the same graphics page.

```
> par(mfrow = c(1, 3), pty = "s")
> x <- rnorm(50)
> hist(x, probability = TRUE)
> lines(density(x))
> xval <- pretty(c(-3, 3), 50)
> lines(xval, dnorm(xval), col = "red")
```

Exercise 3
Data whose distribution is close to lognormal are common. Size measurements of biological organisms often have this character. As an example, consider the measurements of body weight (`body`), in the data frame `Animals` (*MASS*). Begin by drawing a histogram of the untransformed values, and overlaying a density curve. Then

(a) Draw an estimated density curve for the logarithms of the values. Code is given immediately below.

(b) Determine the mean and standard deviation of `log(Animals$body)`. Overlay the estimated density with the theoretical density for a normal distribution with the mean and standard deviation just obtained.

Does the distribution seem normal, after transformation to a logarithmic scale?

```
> library(MASS)
> plot(density(Animals$body))
> logbody <- log(Animals$body)
> plot(density(logbody))
> av <- mean(logbody)
> sdev <- sd(logbody)
> xval <- pretty(c(av - 3 * sdev, av + 3 * sdev), 50)
> lines(xval, dnorm(xval, mean = av, sd = sdev))
```

Exercise 4
The following plots an estimated density curve for a random sample of 50 values from a normal distribution:

```
> plot(density(rnorm(50)), type = "l")
```

(a) Plot estimated density curves, for random samples of 50 values, from (a) the normal distribution; (b) the uniform distribution (`runif(50)`); (c) the t-distribution with 3 degrees of freedom. Overlay the three plots (use `lines()` in place of `plot()` for densities after the first).

(b) Repeat the previous exercise, but taking random samples of 500 values.

Exercise 5
There are two ways to make an estimated density smoother:

(a) One is to increase the number of samples, For example:

```
> plot(density(rnorm(500)), type = "l")
```

(b) The other is to increase the bandwidth. For example

```
> plot(density(rnorm(50), bw = 0.2), type = "l")
> plot(density(rnorm(50), bw = 0.6), type = "l")
```

Repeat each of these with bandwidths (`bw`) of 0.15, with the default choice of bandwidth, and with the bandwidth set to 0.75.

Exercise 6
Here we experiment with the use of `sample()` to take a sample from an empirical distribution, i.e., from a vector of values that is given as argument. Here, the sample size will be the number of values in the argument. Any size of sample is however permissible.

```
> sample(1:5, replace = TRUE)
> for (i in 1:10) print(sample(1:5, replace = TRUE))
> plot(density(log10(Animals$body)))
> lines(density(sample(log10(Animals$body), replace = TRUE)), col = "red")
```

Repeat the final density plot several times, perhaps using different colours for the curve on each occasion. This gives an indication of the stability of the estimated density curve with respect to sample variation.
Laboratory exercises 4 will pursue these ideas in more detail.

3 Normal Probability Plots

Exercise 7
Partly because of the issues with bandwidth and choice of kernel, density estimates are not a very effective means for judging normality. A much better tool is the normal probability plot, which works with cumulative probability distributions. Try

```
> qqnorm(rnorm(10))
> qqnorm(rnorm(50))
> qqnorm(rnorm(200))
```

For samples of modest to large sizes, the points lie close to a line.
The function `qreference()` (*DAAG*) takes one sample as a reference (by default it uses a random sample) and by default provides 5 other random normal samples for comparison. For example:

```
> library(DAAG)
> qreference(m = 10)
> qreference(m = 50)
> qreference(m = 200)
```

Exercise 8
The intended use of `qreference()` is to draw a normal probability for a set of data, and place alongside it some number of normal probability plots for random normal data. For example

```
> qreference(possum$totlngth)
```

Obtain similar plots for each of the variables `taill`, `footlgth` and `earconch` in the possum data. Repeat the exercise for males and females separately

Exercise 9
Use normal probability plots to assess whether the following sets of values, all from data sets in the `DAAG` package, have distributions that seem consistent with the assumption that they have been sampled from a normal distribution?

(a) the difference `heated - ambient`, in the data frame `pair65` (*DAAG*)?

(b) the values of `earconch`, in the `possum` data frame (*DAAG*)?

(c) the values of `body`, in the data frame `Animals` (*MASS*)?

(d) the values of `log(body)`, in the data frame `Animals` (*MASS*)?

4 Boxplots – Simple Summary Information on a Distribution

In the data frame `cfseal` (*DAAG*), several of the columns have a number of missing values. A relevant question is: "Do missing and non-missing rows have similar values, for columns that are complete?"

Exercise 10
Use the following to find, for each column of the data frame `cfseal`, the number of missing values:

```
sapply(cfseal, function(x)sum(is.na(x)))
```

Observe that for `lung`, `leftkid`, `rightkid`, and `intestines` values are missing in the same six rows. For each of the remaining columns compare, do boxplots that compare the distribution of values for the 24 rows that had no missing values with the distribution of values for the 6 rows that had missing values.

Here is code that can be used to get started:

```
present <- complete.cases(cfseal)
boxplot(age ~ present, data=cfseal)
```

Or you might use the lattice function and do the following:

```
present <- complete.cases(cfseal)
library(lattice)
present <- complete.cases(cfseal)
bwplot(present ~ age, data=cfseal)
```

Exercise 11
Tabulate, for the same set of columns for which boxplots were obtained in Exercise 2, differences in medians, starting with:

```
median(age[present]) - median(age[!present]))
```

Calculate also the ratios of the two interquartile ranges, i.e.

```
IQR(age[present]) - IQR(age[!present]))
```

Part IV
Sampling Distributions, and the Central Limit Theorem

For additional background, see SPDM: Section 8.

1 Sampling Distributions

The exercises that follow demonstrate the sampling distribution of the mean, for various different population distributions. More generally, sampling distributions of other statistics may be important.

Inference with respect to means is commonly based on the sampling distribution of the mean, or of a difference of means, perhaps scaled suitably. The ideas extend to the statistics (coefficients, etc) that arise in regression or discriminant or other such calculations. These ideas are important in themselves, and will be useful background for later laboratories and lectures.

Here, it will be assumed that sample values are independent. There are several ways to proceed.

- The distribution from which the sample is taken, although not normal, is assumed to follow a common standard form. For example, in the life testing of industrial components, an exponential or Weibull distribution might be assumed. The relevant sampling distribution can be estimated by taking repeated random samples from this distribution, and calculating the statistic for each such sample.

- If the distribution is normal, then the sample distribution of the mean will also be normal. Thus, taking repeated random samples is unnecessary; theory tells us the shape of the distribution.

- Even if the distribution is not normal, the Central Limit Theorem states that, by taking a large enough sample, the sampling distribution can be made arbitrarily close to normal. Often, given a population distribution that is symmetric, a sample of 4 or 5 is enough, to give a sampling distribution that is for all practical purposes normal.

- The final method [the "bootstrap"] that will be described is empirical. The distribution of sample values is treated as if it were the population distribution. The form of the sampling distribution is then determined by taking repeated random with replacement samples (bootstrap samples), of the same size as the one available sample, from that sample. The value of the statistic is calculated for each such bootstrap sample. The repeated bootstrap values of the statistic are used to build a picture of the sampling distribution.

  With replacement samples are taken because this is equivalent to sampling from a population in which each of the avaialble sample values is repeated an infinite number of times.

The bootstrap method obviously works best if the one available sample is large, thus providing an accurate estimate of the population distribution. Likewise, the assumption that the sampling distribution is normal is in general most reasonable if the one available sample is of modest size, or large. Inference is inevitably hazardous for small samples, unless there is prior information on the likely form of the distribution.

As a rough summary, I hazard the following comments:

- Simulation (repeated resampling from a theoretical distribution or distributions) is useful
  - as a check on theory (the theory may be approximate, or of doubtful relevance)
  - where there is no adequate theory
  - to provide insight, especially in a learning context.

- The bootstrap (repeated resampling from an empirical distribution or distributions) can be useful

- when the sample size is modest and uncertainty about the distributional form may materially affect the assessment of the shape of the sampling distribution;
- when standard theoretical models for the population distribution seem unsatisfactory.

The idea of a sampling distribution is wider than that of a sampling distribution of a statistic. It can be useful to examine the sampling distribution of a graph, i.e., to examine how the shape of a graph changes under repeated bootstrap sampling.

Exercise 1
First, take a random sample from the normal distribution, and plot the estimated density function:

```
> y <- rnorm(100)
> plot(density(y), type = "l")
```

Now take repeated samples of size 4, calculate the mean for each such sample, and plot the density function for the distribution of means:

```
> av <- numeric(100)
> for (i in 1:100) {
+     av[i] <- mean(rnorm(4))
+ }
> lines(density(av), col = "red")
```

Repeat the above: taking samples of size 9, and of size 25.

Exercise 2
It is also possible to take random samples, usually with replacement, from a vector of values, i.e., from an empirical distribution. This is the bootstrap idea. Again, it may of interest to study the sampling distributions of means of different sizes. Consider the distribution of heights of female Adelaide University students, in the data frame `survey` (*MASS* package). The following takes 100 bootstrap samples of size 4, calculating the mean for each such sample:

```
> library(MASS)
> y <- na.omit(survey[survey$Sex == "Female", "Height"])
> av <- numeric(100)
```

Repeat, taking samples of sizes 9 and 16. In each case, use a density plot to display the (empirical) sampling distribution.

Exercise 3
Repeat exercise 1 above: (a) taking values from a uniform distribution (replace `rnorm(4)` by `runif(4)`); (b) from an exponential distribution with rate 1 (replace `rnorm(4)` by `rexp(4, rate=1)`).
[As noted above, density plots are not a good tool for assessing distributional form. They are however quite effective, as here, for showing the reduction in the standard deviation of the sampling distribution of the mean as the sample size increases. The next exercise but one will repeat the comparisons, using normal probability plots in place of density curves.]

Exercise 4
The final exercise of Assignment 2 compared density plots, for several of the variables, between rows that had one or more missing values and those that had no missing values. We can use the bootstrapping idea to ask how apparent differences stand up against repeated simulation.
The distribution for `bmi` looked as though it might have shifted a bit, for data where one or more rows was missing, relative to other rows. We can check whether this apparent shift is consistent under repeated sampling. Here again is code for the graph for `bmi`

```
> library(MASS)
> library(lattice)
> complete <- complete.cases(Pima.tr2)
> completeF <- factor(c("oneORmore", "none")[as.numeric(complete) +
+     1])
> Pima.tr2$completeF <- completeF
> densityplot(~bmi, groups = completeF, data = Pima.tr2, auto.key = list(columns = 2))
```

Here is code for taking one bootstrap sample from each of the two categories of row, then repeating the density plot.

```
> rownum <- seq(along = complete)
> allpresSample <- sample(rownum[complete], replace = TRUE)
> NApresSample <- sample(rownum[!complete], replace = TRUE)
> densityplot(~bmi, groups = completeF, data = Pima.tr2, auto.key = list(columns = 2),
+     subset = c(allpresSample, NApresSample))
```

Wrap these lines of code in a function. Repeat the formation of the bootstrap samples and the plots several times. Does the shift in the distribution seem consistent under repeating sampling?

Exercise 5
More commonly, one compares examines the displacement, under repeated sampling, of one mean relative to the other. Here is code for the calculation:

```
> twot <- function(n = 99) {
+     complete <- complete.cases(Pima.tr2)
+     rownum <- seq(along = complete)
+     d2 <- numeric(n + 1)
+     d2[1] <- with(Pima.tr2, mean(bmi[complete], na.rm = TRUE) -
+         mean(bmi[!complete], na.rm = TRUE))
+     for (i in 1:n) {
+         allpresSample <- sample(rownum[complete], replace = TRUE)
+         NApresSample <- sample(rownum[!complete], replace = TRUE)
+         d2[i + 1] <- with(Pima.tr2, mean(bmi[allpresSample],
+             na.rm = TRUE) - mean(bmi[NApresSample], na.rm = TRUE))
+     }
+     d2
+ }
> dens <- density(twot(n = 999))
> plot(dens)
> sum(d2 < 0)/length(d2)

[1] 0.199
```

Those who are familiar with t-tests may recognize the final calculation as a bootstrap equivalent of the t-test.

Exercise 6
The range that contains the central 95% of values of `d2` gives a 95% confidence (or coverage) interval for the mean difference. Given that there are 1000 values in total, the interval is the range from the 26th to the 975th value, when values are sorted in order of magnitude, thus:

```
> round(sort(d2)[c(26, 975)], 2)

[1] -0.94  2.47
```

Repeat the calculation of `d2` and the calculation of the resulting 95% confidence interval, several times.

2 The Central Limit Theorem

Theoretically based t-statistic and related calculations rely on the assumption that the sampling distribution of the mean is normal. The Central Limit Theorem assures that the distribution will for a large enough sample be arbitrarily close to normal, providing only that the population distribution has a finite variance. Simulation of the sampling distribution is especially useful if the population distribution is not normal, providing an indication of the size of sample needed for the sampling distribution to be acceptably close to normal.

Exercise 7
The function `simulateSampDist()` allows investigation of the sampling distribution of the mean, for an arbitrary population distribution and sample size. Figure 1 shows sampling distributions for samples of sizes 4 and 9, from a normal population. The function call is

```
> page <- "http://www.maths.anu.edu.au/~johnm/courses/dm/math3346/functions/"
> load(url(paste(page, "simulateSampDist.RData", sep = "")))
> load(url(paste(page, "plotSampDist.RData", sep = "")))
> simulateSampDist(numINsamp = c(4, 9))
> plotSampDist(sampvalues = sampvalues, graph = "density", titletext = NULL)
```

Experiment with sampling from normal, uniform, exponential and t_2-distributions. What is the effect of varying the value of `numsamp`?
[To vary the kernel and/or the bandwidth used by `density()`, just add the relevant arguments in the call to to `simulateSampDist()`, e.g. `sampdist(numINsamp=4, bw=0.5)`. Any such additional arguments (here, `bw`) are passed via the `...` part of the parameter list.]

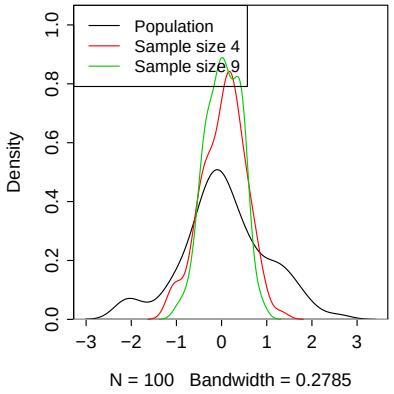

Figure 1: Empirical density curve, for a normal population and for the sampling distributions of means of samples of sizes 4 and 9 from that population.

Exercise 8
The function `simulateSampDist()` has an option (`graph="qq"`) that allows the plotting of a normal probability plot. Alternatively, by using the argument `graph=c("density","qq")`, the two types of plot appear side by side, as in Figure 2. Figure 2 is an example of its use.

```
> sampvalues <- simulateSampDist()
> plotSampDist(sampvalues = sampvalues, graph = c("density", "qq"))
```

In the right panel, the slope is proportional to the standard deviation of the distribution. For means of a sample size equal to 4, the slope is reduced by a factor of 2, while for a sample size equal to 9, the slope is reduced by a factor of 3.
Comment in each case on how the spread of the density curve changes with increasing sample size. How does the qq-plot change with increasing sample size? Comment both on the slope of a line that might be passed through the points, and on variability about that line.

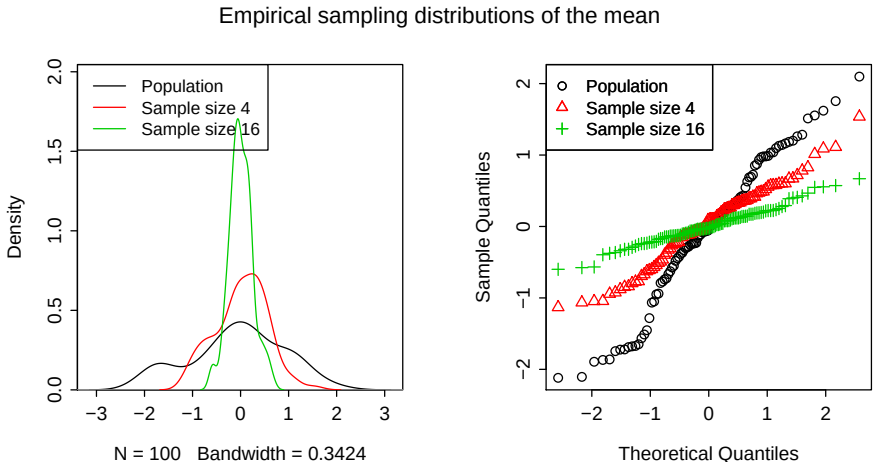

Figure 2: Empirical density curves, for a normal population and for the sampling distributions of means of samples of sizes 4 and 9, are in the left panel. The corresponding normal probability plots are shown in the right panel.

Exercise 9
How large is "large enough", so that the sampling distribution of the mean is close to normal? This will depend on the population distribution. Obtain the equivalent for Figure 2, for the following populations:

(a) A t-distribution with 2 degrees of freedom
[`rpop = function(n)rt(n, df=2)`]

(b) A log-normal distribution, i.e., the logarithms of values have a normal distribution
[`rpop = function(n, c=4)exp(rnorm(n)+c)`]

(c) The empirical distribution of heights of female Adelaide University students, in the data frame `survey` (*MASS* package). In the call to `simulateSampDist()`, the parameter `rpop` can specify a vector of numeric values. Samples are then obtained by sampling with replacement from these numbers. For example:

```
> library(MASS)
> y <- na.omit(survey[survey$Sex == "Female", "Height"])
> sampvalues <- simulateSampDist(y)
> plotSampDist(sampvalues = sampvalues)
```

How large a sample seems needed, in each instance, so that the sampling distribution is approximately normal – around 4, around 9, or greater than 9?

Part V
Linear Models in R

For background, see SPDM: Sections 9-11.

1 Linear and Other Models in R

The base R system and the various R packages provide, between them, a huge range of model fitting abilities. In this laboratory, the major attention will be on the model fitting function is `lm()`, where the `lm` stands for linear model. An aim is to encourage an expansive view of linear models. Linear models can do a great deal more than fit straight lines and planes in a high-dimensional space.

R's implementation of linear models uses a symbolic notation, described in (Wilkinson & Rogers , 1973), that gives a straightforward means for describing elaborate and intricate models. A very similar notation is used throughout the modeling functions in R, with modification as required. Important ideas are *basis function*, *factor*, *interaction*, and *model formula*.

2 Fitting Straight Lines to Data

Exercise 1
In each of the data frames `elastic1` and `elastic2`, fit straight lines that show the dependence of `distance` on `stretch`. Plot the two sets of data, using different colours, on the same graph. Add the two separate fitted lines. Also, fit one line for all the data, and add this to the graph.

Exercise 2
In the data set pressure (*datasets*), the relevant theory is that associated with the Claudius-Clapeyron equation, by which the logarithm of the vapor pressure is approximately inversely proportional to the absolute temperature. Transform the data in the manner suggested by this theoretical relationship, plot the data, fit a regression line, and add the line to the graph. Does the fit seem adequate?
[For further details of the Claudius-Clapeyron equation, search on the internet, or look in a suitable reference text.]

Exercise 3
Run the function `plot.intersalt()`; data frame `intersalt`. Data are population average values of blood pressure, from 52 different studies. Is the fitted line reasonable? Or is it a misinterpretation of the data? Suggest alternatives to regression analysis, for getting a sense of how these results should be interpreted? What are the populations where levels are very low? What is special about these countries?
[The function `plot.intersalt()`, and the data frame `intersalt`, are included in `http://www.maths.anu.edu.au/~johnm/r/misc-data/misc.RData`]

3 Multiple Explanatory Variables

Exercise 4
For the data frame `oddbooks` (*DAAG*),

(a) Add a further column that gives the density.

(b) Use the function `pairs()`, or the *lattice* function `splom()`, to display the scatterplot matrix. Which pairs of variables show evidence of a strong realtionship?

(c) In each panel of the scatterplot matrix, record the correlation for that panel. (Use `cor()`) to calculate correlations).

(d) Fit the following regression relationships:

  (a) `log(weight)` on `log(thick)`, `log(height)` and `log(breadth)`.

  (b) `log(weight)` on `log(thick)` and `0.5*(log(height) + log(breadth))`. What feature of the scatterplot matrix suggests that this might make sense to use this form of equation?

(e) Take the fitted regression equation (take whichever of the two forms of equation seems preferable) and rewrite it in a form that as far as possible separates effects that arise from changes in the linear dimensions from effects that arise from changes in linear dimensions.

4 Errors in Variables

Exercise 5
Run the accompanying function `errorsINx()` several times. Comment on the results. The underlying relationship between y and x is the same in all cases. The error in x is varied, from values of x that are exact to values of x that have random errors added with a variance that is twice that of the variance of x.

4.1 Functions used

```
errorsINx <-
    function(mu = 20, n = 100, a = 15, b = 2.5, sigma = 12.5,
           timesSigma=(1:5)/2.5){
    mat <- matrix(0, nrow=n, ncol=length(timesSigma)+2)
    x0 <- mu*exp(rnorm(n,1,0.15))
    y <- a + b*x0+rnorm(n,0,sigma)
    mat[, length(timesSigma)+2] <- y
    mat[,1] <- x0
    sx <- sd(x0)
    k <- 1
    for(i in timesSigma){
      k <- k+1
      x1 <- x0+rnorm(n, 0, sx*i)
      mat[, k] <- x1
    }
    mat
  }

oldpar <- par(mar=c(3.6,3.1,1.6,0.6), mgp=c(2.5,0.75,0),
              oma=c(1,1,0.6,1),
              mfrow=c(2,3), pty="s")
mu <- 20; n <- 100; a <- 15; b <- 2.5; sigma <- 12.5; timesSigma<-(1:5)/2.5
```

```
mat <- errorsINx(mu = 20, n = 100, a = 15, b = 2.5, sigma = 12.5,
                 timesSigma=(1:5)/2.5)
beta <- numeric(dim(mat)[2]-1)
sx <- sd(mat[,1])
y <- mat[, dim(mat)[2]]
for(j in 1:length(beta)){
  xj <- mat[,j]
  plot(y ~ xj, xlab="", ylab="", col="gray30")
  if(j==1)
    mtext(side=3, line=0.5, "No error in x") else{
    xm <- timesSigma[j-1]
    mtext(side=3, line=0.5, substitute(tau == xm*s[z], list(xm=xm)))
    }
  if(j>=4)mtext(side=1, line=2, "x")
  if(j%%3 == 1)mtext(side=2, line=2, "y")
  errors.lm <- lm(y ~ xj)
  abline(errors.lm)
  beta[j] <- coef(errors.lm)[2]
  bigsigma <- summary(errors.lm)$sigma
  print(bigsigma/sigma)
  abline(a, b, lty=2)
}
print(round(beta, 3))

plot.intersalt <-
function (dset = intersalt1, figno = 2)
{
    oldpar <- par(oma = c(6.5, 0, 0, 0), mar = par()$mar - c(0,
        0, 3.5, 0))
    on.exit(par(oldpar))
    lowna <- c(4, 5, 24, 28)
    plot(dset$na, dset$bp, pch = 15, ylim = c(50, 85),
         xlab = "Median sodium excretion (mmol/24hr)",
         ylab = "Median diastolic BP (mm Hg)", type = "n")
    points(dset$na[-lowna], dset$bp[-lowna], pch = 16)
    points(dset$na[lowna], dset$bp[lowna], pch = 1, lwd = 3)
    u <- lm(bp ~ na, data = dset)
    abline(u$coef[1], u$coef[2])

    figtxt <-
       paste("Fig. ", figno, ": Plot of median blood pressure versus salt",
             "\\n(measured by sodium excretion) for 52 human",
             "\\npopulations. Four results (open circles) are for",
             "\\nnon-industrialised societies with very low salt intake,",
             "\\nwhile other results are for industrialised societies.",
       sep = "")
    mtext(side = 1, line = 6, figtxt, cex = 1.1, adj = 0, at = -20)
}
```

Part VI
Exploiting the Model Matrix

For background, see SPDM: Sections 12-13.

It is straightforward to model qualitative effects. In R, factors, with one level for each qualitative category, are typically used to account for qualitative effects. Additonally, and perhaps more surpringingly, it is straightforward to use linear models to model curvilinear effects,

1 A One-way classification – eggs in the cuckoo's nest

This demonstrates the use of linear models to fit qualitative effects.

Like many of nature's creatures, cuckoos have a nasty habit. They lay their eggs in the nests of other birds. First, let's see how egg length changes with the host species. This will use the graphics function `stripplot()` from the *lattice* package. The data frame `cuckoos` is in the *DAAG* package.

```
> par(mfrow = c(1, 2))
> library(DAAG)
> library(lattice)
> names(cuckoos)[1] <- "lngth"
> table(cuckoos$species)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
           14            45            15            16            15
         wren
           15

> stripplot(species ~ lngth, data = cuckoos)
> cuckoo.strip <- stripplot(breadth ~ lngth, groups = species,
+     data = cuckoos, auto.key = list(columns = 3))
> print(cuckoo.strip)
> par(mfrow = c(1, 1))
```

Now estimate the means for each of the groups. We will use two forms of the model. The first gives the species means directly, as parameters for the model. In the second parameterization, the first parameter estimate is the mean for the first species, while later estimates are differences from the first species.

Exercise 1
In examining how cuckoo egg length varies between host species, the following forces all parameters to be species means

```
> lm(lngth ~ -1 + species, data = cuckoos)

Call:
lm(formula = lngth ~ -1 + species, data = cuckoos)

Coefficients:
specieshedge.sparrow   speciesmeadow.pipit   speciespied.wagtail
               23.11                 22.29                 22.89
        speciesrobin     speciestree.pipit           specieswren
               22.56                 23.08                 21.12
```

The following makes the first species the baseline

```
> lm(lngth ~ species, data = cuckoos)

Call:
lm(formula = lngth ~ species, data = cuckoos)

Coefficients:
        (Intercept)  speciesmeadow.pipit  speciespied.wagtail
           23.11429             -0.82095             -0.22762
       speciesrobin    speciestree.pipit          specieswren
           -0.55804             -0.03429             -1.99429
```

Reconcile the two sets of results.

Exercise 2
Use the `termplot()` function to show the effects of the different factor levels. Be sure to call the function with `partial.resid=TRUE` and with `se=TRUE`. Does the variability seem similar for all host species?

Exercise 3
Obtain the standard deviation for each species:

```
> attach(cuckoos)

> sapply(split(lngth, species), sd)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
    1.0494373     0.9195849     1.0722917     0.6821229     0.8800974
         wren
    0.7542262
```

[The function `split()` splits the lengths into six sublists, one for each species. The function `sapply()` applies the function calculation to the vectors of lengths in each separate sublist.]
Check that the SD seems smallest for those species where the SD seems, visually, to be smallest.

Exercise 4
Obtain, for each species, the standard error of the mean. This is obtained by dividing the standard deviation by the square root of the number of values:

```
> sdev <- sapply(split(lngth, species), sd)
> n <- sapply(split(lngth, species), length)
> sdev/sqrt(n)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
    0.2804739     0.1370836     0.2768645     0.1705307     0.2272402
         wren
    0.1947404
```

Alternatively, create a function that calculates the standard error of the mean in a single calculation:

```
> se <- function(x) sd(x)/sqrt(length(x))
> sapply(split(lngth, species), se)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
    0.2804739     0.1370836     0.2768645     0.1705307     0.2272402
         wren
    0.1947404
```

This can be done in a single line of code. The function `se()` can be inserted as an anonymous function. In this case, it does not need a name; the function definition is inserted where the function name would otherwise appear:

```
> sapply(split(lngth, species), function(x) sd(x)/sqrt(length(x)))

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
    0.2804739     0.1370836     0.2768645     0.1705307     0.2272402
         wren
    0.1947404
```

2 Regression Splines

Exercise 5
The following are based on the `geophones` data frame.

(a) Plot `thickness` against distance.

(b) Try the following:

```
> plot(thickness ~ distance, data = geophones)
> with(geophones, lines(lowess(distance, thickness)))
> with(geophones, lines(lowess(distance, thickness, f = 0.2), col = "red"))
```

Which of the two fitted curves best captures the pattern of change?

(c) Now fit a natural regression spline. Try for example:

```
> library(splines)
> hat <- with(geophones, fitted(lm(thickness ~ ns(distance, 4))))
> with(geophones, lines(distance, hat, col = 3))
```

Experiment with different choices for the number of degrees of freedom. How many degrees of freedom seem needed to adequately capture the pattern of change?

Part VII
Data Summary – Traps for the Unwary

For background, see SPDM: Section 3.

1 Multi-way Tables

	Small (<2cm)			Large (>=2cm)			Total	
	open	ultrasound		open	ultrasound		open	ultrasound
yes	81	234	yes	192	55	yes	273	289
no	6	36	no	71	25	no	77	61
Success rate	93%	87%		73%	69%		78%	83%

Table 1: Outcomes for two different types of surgery for kidney stones. The overall success rates (78% for open surgery as opposed to 83% for ultrasound) favor ultrasound. Comparison of the success rates for each size of stone separately favors, in each case, open surgery.

Exercise 1
Table 1 illustrates the potential hazards of adding a multiway table over one of its margins. Data are from a study (Charig, 1986) that compared outcomes for two different types of surgery for kidney stones; A: `open`, which used open surgery, and B: `ultrasound`, which used a small incision, with the stone destroyed by ultrasound. The data can be entered into R, thus:

```
stones <- array(c(81,6,234,36,192,71,55,25), dim=c(2,2,2),
                dimnames=list(Sucess=c("yes","no"),
                              Method=c("open","ultrasound"),
                              Size=c("<2cm", ">=2cm")))
```

(a) Determine the success rate that is obtained from combining the data for the two different sizes of stone. Also determine the success rates for the two different stone sizes separately.

(b) Use the following code to give a visual representation of the information in the three-way table:

```
mosaicplot(stones, sort=3:1)
   # Re-ordering the margins gives a more interpretable plot.
```

Annotate the graph to show the success rates?

(c) Observe that the overall rate is, for open surgery, biased toward the open surgery outcome for large stones, while for ultrasound it is biased toward the outcome for small stones. What are the implicatiosn for the interpretation of these data?

[Without additional information, the results are impossible to interpret. Different surgeons will have preferred different surgery types, and the prior condition of patients will have affected the choice of surgery type. The consequences of unsuccessful surgery may have been less serious than for ultrasound than for open surgery.]

The relative success rates for the two different types of surgery, for the two stone sizes separately, can be calculated thus:

```
> stones[1, , ]/(stones[1, , ] + stones[2, , ])
```

To perform the same calculation after adding over the two stone sizes (the third dimension of the table), do

```
> stones2 <- stones[, , 1] + stones[, , 2]
> stones2[1, ]/(stones2[1, ] + stones2[2, ])
```

Seatbelt	Airbag	Fatalities	Occupants
seatbelt	airbag	8626	4871940
none	airbag	10650	870875
seatbelt	none	7374	2902694
none	none	20550	1952211

Table 2: Number of fatalities, by use of seatbelt and presence of airbag.

Exercise 2
Each year the National Highway Traffic Safety Administration in the USA collects, using a random sampling method, data from all police-reported crashes in which there is a harmful event (people or property), and from which at least one vehicle is towed. The data in Table 2 are a summary of a subset of the 1997-2002 data, as reported in Meyer and Finney (2005). The data from which this table was created can be obtained from the website
`http://www.stat.uga.edu/~mmeyer/airbags.htm`.
Determine fatality rates: (1) for seatbelt use, irrespective of airbag use; (2) for airbag use, irrespective of seatbelt use; (3) for airbag use, for each category of seatbelt use.

```
> CDS <- data.frame(Seatbelts = c("seatbelt", "none", "seatbelt",
+     "none"), Airbags = c("airbag", "airbag", "none", "none"),
+     dead = c(8626, 10650, 7374, 20550), total = c(4871940, 870875,
+         2902694, 1952211))
> with(CDS, sum(dead)/sum(total))
> print("Total counts for seatbelt/none")
> seatsum <- aggregate(CDS[, 3:4], by = list(CDS$Seatbelts), FUN = sum)
> with(seatsum, dead/total)
> print("Total counts for seatbelt/none")
> bagnum <- aggregate(CDS[, 3:4], by = list(CDS$Airbags), FUN = sum)
> with(bagnum, dead/total)
> print("By airbag use, for the separate seatbelt categories")
> by(CDS, list(CDS$Seatbelts), FUN = function(x) x$dead/x$total)
> print("Alternative")
> sapply(split(CDS, CDS$Seatbelts), function(x) x$dead/x$total)
```

Exercise 3
Load the contents of **cdsDeltaV.RData**. Plot the fatality rate against deltaV, separately for seatbelt/none, with the two separate categories of airbag use on the same graph.

```
> xyplot(mrate ~ deltaV | Seatbelts, groups = Airbags, type = "b",
+     data = cdsDeltaV)
```

	baclofen	placebo
females	15	7
males	3	16

Table 3: Numbers of males and females on the two treatments, in a trial that investigated the effect of pentazocine on post-operative pain (VAS scores).

	min	mbac	mpl	fbac	fpl
2	10.00	1.76	1.76	2.18	2.55
3	30.00	1.31	1.65	3.48	4.15
4	50.00	0.05	0.67	3.13	3.66
5	70.00	−0.57	−0.25	3.03	2.05
6	90.00	−1.26	−0.50	2.08	0.61
7	110.00	−2.15	−2.22	1.60	0.34
8	130.00	−1.65	−2.18	1.38	0.67
9	150.00	−1.68	−2.86	1.76	0.76
10	170.00	−1.68	−3.23	1.06	0.39

Table 4: The table shows, separately for males and females, the effect of pentazocine on post-operative pain (average VAS scores), with (mbac and fbac) and without (mpl and fpl) preoperatively administered baclofen.

.

2 Weighting Effects – Example with a Continous Outcome

Exercise 4
The data in Table 3, in the data frame `gaba`, are from Gordon et al (1995).
[The data frame `gaba`, and functions `plot.gaba()` and `compare.gaba()` that use these data for their displays, are included in the image file
`http://www.maths.anu.edu.au/~johnm/r/misc-data/misc.RData`]
Table 4 has a tabular summary of the outcome of the trial to which Table 3 relates.

(a) What do you notice about the relative numbers on the two treatments?

(b) For each treatment, obtain overall weighted averages at each time point, using the numbers in Table 3 as weights. (These should be the numbers you would get if you divided the total over all patients on that treatment by the total number of patients.) This will give columns `avbac` and `avplac` that can be added the data frame.

(c) Plot `avbac` and `avplac` against time, on the same graph. On separate graphs, repeat the comparisons (a) for females alone and (b) for males alone. Which of these graphs make a correct and relevant comparison between baclofen and placebo (albeit both in the presence of pentazocine)?

Part VIII
Multi-level Models

For background, see SPDM: Sections 14-15.

1 Description and Display of the Data

This laboratory will work with data on corn yields from the Caribbean islands of Antigua and St Vincent. Data are yields from packages on eight sites on the Caribbean island of Antigua. They are a summarized version of a subset of data given in Andrews and Herzberg 1985, pp.339-353.

For the present analysis, the chief interest will be in yields for the standard treatment, here identified as `111`, for the Antiguan data, held in the data frame `ant111b`. As will be described below, two possible predictions are:

(a) Predictions for new packages of land in one of the existing sites.

(b) Predictions for new packages in a new site.

The accuracies for the second type of prediction may be much less accurate than for the first type.

We will however start by examining plots, for the treatment `111`, from the combined data for the two islands. This information for the separate islands is summarized in the datasets `ant111b` and `vince111b` in the *DAAG* package.

A first step is to combine common columns of `ant111b` and `vince111b` into the single data frame `corn111b`.

```
> library(lattice)
> library(DAAG)
> corn111b <- rbind(ant111b[, -8], vince111b)
> corn111b$island <- c("Antigua", "StVincent")[corn111b$island]
```

- The following plot uses different panels for the two islands:

```
> corn.strip1 <- stripplot(site ~ harvwt | island, data = corn111b,
+     xlab = "Harvest weight")
```

- The following plot uses different panels for the two islands, but allows separate ("free" = no relation) vertical scales for the two plots.

```
> corn.strip2 <- stripplot(site ~ harvwt | island, data = corn111b,
+     xlab = "Harvest weight", scale = list(y = list(relation = "free")))
```

- The following uses a single panel, but uses different colours (or, on a black and white device, different symbols) to distinguish the two islands. Notice the use of `auto.key` to generate an automatic key:

```
> corn.strip3 <- stripplot(site ~ harvwt, data = corn111b, groups = island,
+     xlab = "Harvest weight", auto.key = list(columns = 2))
```

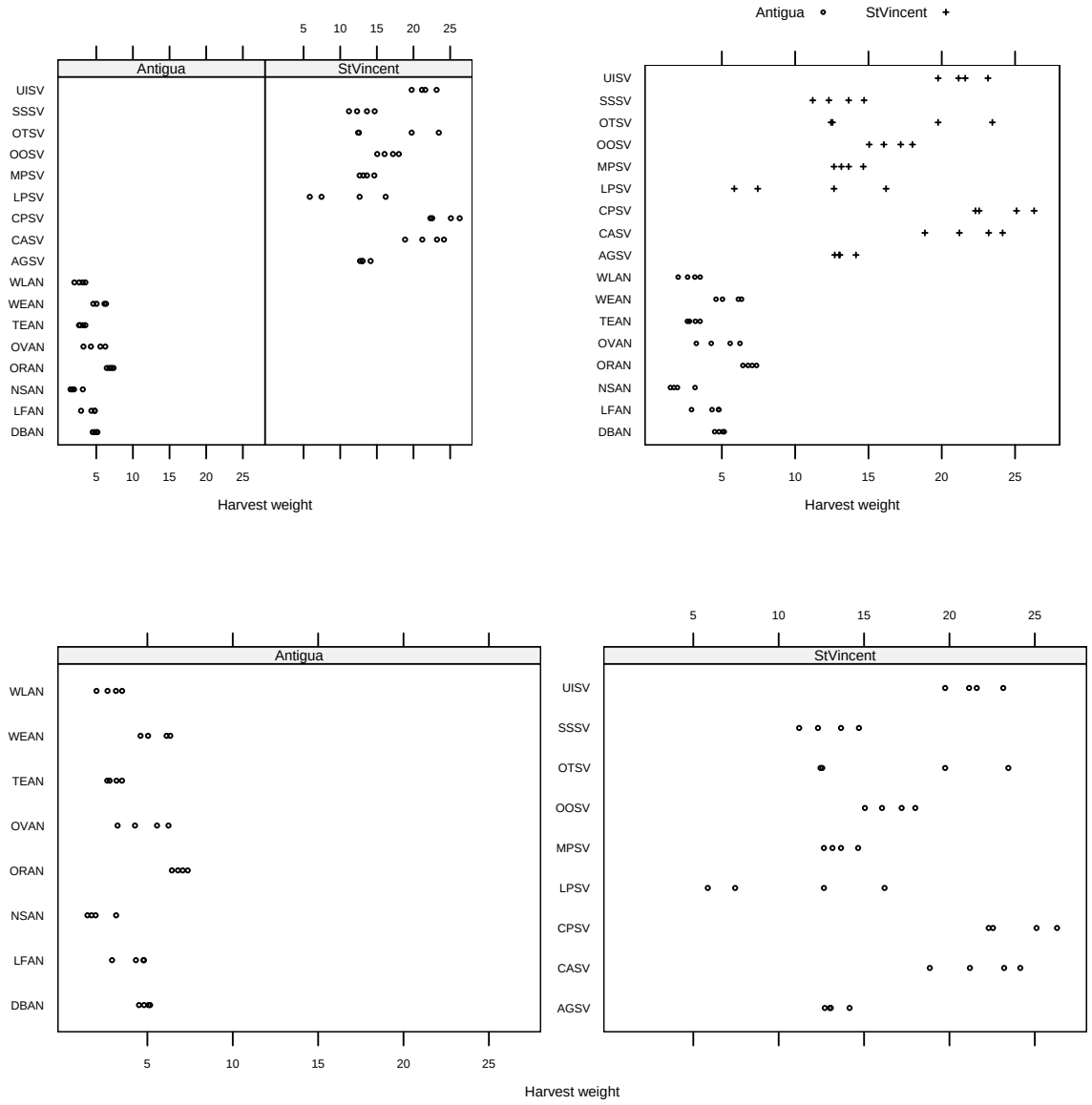

Figure 3: Yields for the four packages of corn on sites on the islands of Antigua and St Vincent.

Next, we will obtain package means for the Antiguan data, for all treatments.

```
> attach("Antigua.RData")

> with(antigua, antp <<- aggregate(harvwt, by = list(site = site,
+     package = package, trt = trt), FUN = mean))
> names(antp)[4] <- "harvwt"
```

Notice the use of the version `<<-` of the assignment symbol to ensure that assignment takes place in the workspace.

Now plot mean harvest weights for each treatment, by site:

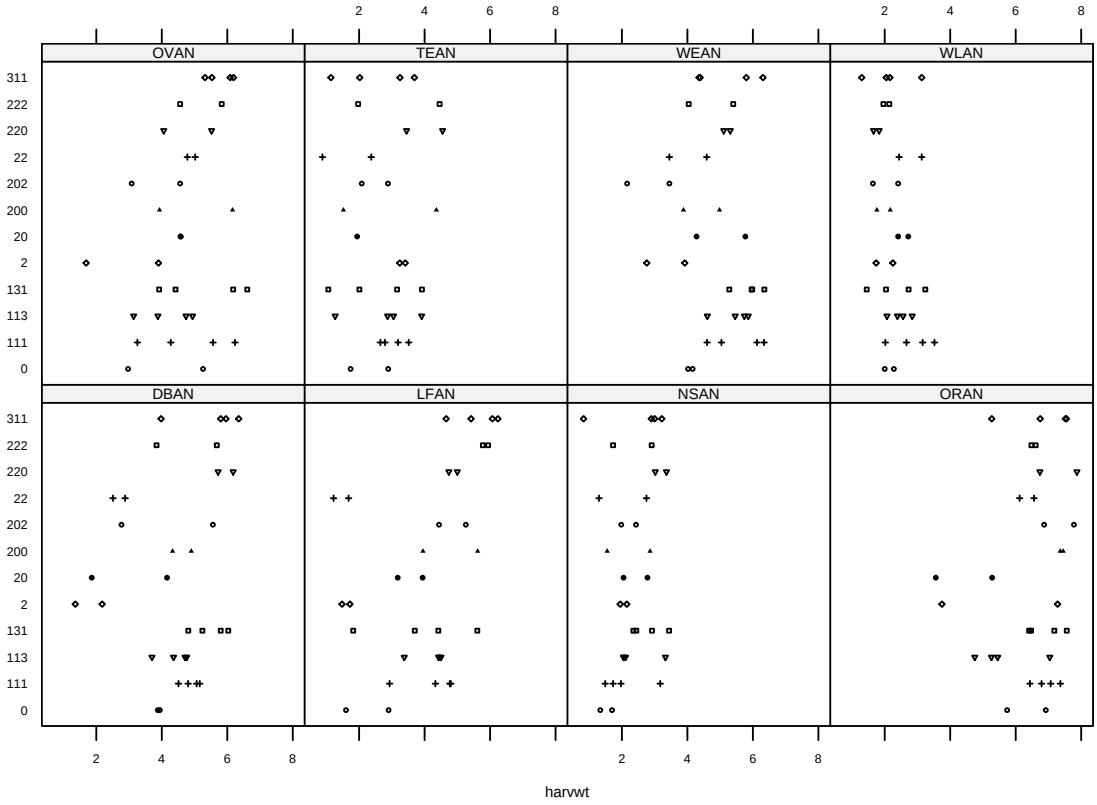

Figure 4: Yields for the four packages of corn on each of eight sites on the island of Antigua.

2 Multi-level Modeling

***Analysis using *lme*:** The modeling command takes the form:

```
> library(nlme)
> ant111b.lme <- lme(fixed = harvwt ~ 1, random = ~1 | site, data = ant111b)
```

The only fixed effect is the overall mean. The argument `random = ~1|site` fits random variation between sites. Variation between the individual units that are nested within sites, i.e., between packages, are by default treated as random. Here is the default output:

```
> options(digits = 4)
> ant111b.lme

Linear mixed-effects model fit by REML
  Data: ant111b
  Log-restricted-likelihood: -47.21
  Fixed: harvwt ~ 1
(Intercept)
      4.292

Random effects:
 Formula: ~1 | site
        (Intercept) Residual
```

```
StdDev:       1.539     0.76

Number of Observations: 32
Number of Groups: 8
```

Notice that *lme* gives, not the components of variance, but the standard deviations (`StdDev`) which are their square roots. Observe that, according to *lme*, $\widehat{\sigma_B^2} = 0.76^2 = 0.578$, and $\widehat{\sigma_L^2} = 1.539^2 = 2.369$. Those who are familiar with an analysis of variance table for such data should note that *lme* does not give the mean square at any level higher than level 0, not even in this balanced case.

The take-home message from this analysis is:

- For prediction for a new package at one of the existing sites, the standard error is 0.76
- For prediction for a new package at a new site, the standard error is $\sqrt{1.539^2 + .76^2} = 1.72$
- For prediction of the mean of n packages at a new site, the standard error is $\sqrt{1.539^2 + 0.76^2/n}$

Where there are multiple levels of variation, the predictive accuracy can be dramatically different, depending on what is to be predicted. Similar issues are arise in repeated measures contexts, and in time series. Repeated measures data has multiple profiles, i.e., many small time series.

2.1 The Attitudes to Science Data Set

These data are from in the *DAAG* package for R. The data are measurements of attitudes to science, from a survey where there were results from 20 classes in 12 private schools and 46 classes in 29 public (i.e. state) schools, all in and around Canberra, Australia. Results are from a total of 1385 year 7 students. The variable `like` is a summary score based on two of the questions. It is on a scale from 1 (dislike) to 12 (like). The number in each class from whom scores were available ranged from 3 to 50, with a median of 21.5.

There are three variance components:

```
Between schools 0.00105
Between classes 0.318
Between students 3.05
```

The between schools component can be neglected. The variance for a class mean is $0.318 + 3.05/n$, where n is the size of the class. The two contributions are about equal when n =10.

2.2 *Additional Calculations

We return again to the corn yield data.

Is variability between packages similar at all sites?:

```
> if (dev.cur() == 2) invisible(dev.set(3))
> vars <- sapply(split(ant111b$harvwt, ant111b$site), var)
> vars <- vars/mean(vars)
> qqplot(qchisq(ppoints(vars), 3), 3 * vars)
```

Does variation within sites follow a normal distribution?:

```
> qqnorm(residuals(ant111b.lme))
```

What is the pattern of variation between sites?

```
> locmean <- sapply(split(log(ant111b$harvwt), ant111b$site), mean)
> qqnorm(locmean)
```

The distribution seems remarkably close to normal.

Fitted values and residuals in *lme*: By default fitted values account for all random effects, except those at level 0. In the example under discussion `fitted(ant111b.lme)` calculates fitted values at level 1, which can be regarded as estimates of the site means. They are not however the site means, as the graph given by the following calculation demonstrates:

```
> hat.lm <- fitted(lm(harvwt ~ site, data = ant111b))
> hat.lme <- fitted(ant111b.lme)
> plot(hat.lme ~ hat.lm, xlab = "Site means", ylab = "Fitted values (BLUPS) from lme")
> abline(0, 1, col = "red")
```

The fitted values are known as BLUPs (Best Linear Unbiased Predictors). Relative to the site means, they are pulled in toward the overall mean. The most extreme site means will on average, because of random variation, be more extreme than the corresponding "true" means for those sites. There is a theoretical result that gives the factor by which they should be shrunk in towards the true mean.

Residuals are by default the residuals from the package means, i.e., they are residuals from the fitted values at the highest level available. To get fitted values and residuals at level 0, enter:

```
> hat0.lme <- fitted(ant111b.lme, level = 0)
> res0.lme <- resid(ant111b.lme, level = 0)
> plot(res0.lme, ant111b$harvwt - hat0.lme)
```

Part IX
Discriminant Methods – Realistic Error Rate Estimation

1 rpart Analyses – the Combined `Pima` Dataset

Note the rpart terminology:

size	Number of leaves	Used in plots from `plotcp()`
`nsplit`	Number of splits = size - 1	Used in `printcp()` output
`cp`	Complexity parameter	Appears as CP in graphs and printed output. A smaller `cp` gives a more complex model, i.e., more splits.
rel error	Resubstitution error measure	Multiply by baseline error to get the corresponding absolute error measure. In general, treat this error measure with scepticism.
xerror	Crossvalidation error estimate	Multiply by baseline error to get the corresponding absolute error measure.

After loading the *MASS* package, type `help(Pima.tr)` to get a description of these data. They are relevant to the investigation of conditions that may pre-dispose to diabetes.

The `Pima` data frame combines `Pima.tr` and `Pima.te` from the *MASS* package, thus:

```
> library(e1071)

Loading required package: class

> library(MASS)
> Pima <- rbind(Pima.tr, Pima.te)
```

Exercise 1: Fit an rpart model to the `Pima` data:

```
> library(rpart)
> Pima.rpart <- rpart(type ~ ., data = Pima, method = "class")
> plotcp(Pima.rpart)
```

The formula `type` $\sim$ `.` has the effect of using as explanatory variables all columns of `Pima` except `type`. The parameter cp is a complexity parameter; it penalizes models that are too complex. A small penalty leads to more splits. Note that cp, at this initial fit, has to be small enough so that the minimum of the cross-validated error is attained.

Try this several times. The result will vary somewhat from run to run. Why?

One approach is to choose in a manner that makes the model on average best, i.e., choose the absolute minimum of the cross-validated error. If the fitting has been repeated several times, the cross-validation errors can be averaged over the separate runs.

A more cautious approach is to choose a model where there is some modest certainty that the final split is giving a better than chance improvement. For this, choose the smallest number of splits so that upper SE limit lies under the dotted line. For example, in one of my runs, this happened for cp=0.017, with size=10. The resulting tree can be cut back to this size with:

```
> Pima.rpart10 <- prune(Pima.rpart, cp = 0.016)
```

The value of cp is chosen to be less than 0.017, but more than the value that led to a further split.

Plot the tree, thus:

```
> plot(Pima.rpart10)
> text(Pima.rpart10)
```

Note also the printed output from

```
> printcp(Pima.rpart)

Classification tree:
rpart(formula = type ~ ., data = Pima, method = "class")

Variables actually used in tree construction:
[1] age   bmi   glu   npreg ped

Root node error: 177/532 = 0.33271

n= 532

        CP nsplit rel error  xerror     xstd
1 0.265537      0   1.00000 1.00000 0.061400
2 0.056497      1   0.73446 0.83051 0.058272
3 0.025424      3   0.62147 0.74011 0.056141
4 0.020716      5   0.57062 0.72881 0.055849
5 0.014124      9   0.48588 0.71751 0.055552
6 0.011299     11   0.45763 0.70621 0.055249
7 0.010000     14   0.42373 0.71186 0.055401
```

Get the absolute cross-validated error by multiplying the root node error by xerror. With nsplit=9 (tree of size 10 leaves), this is, in the run I did, 0.3327*0.6836. (The accuracy is obtained by subtracting this from 1.0, i.e., about 77%.)

Where it is not clear from the graph where the minimum (or the minimum+SE, if that is used) lies, it will be necessary to resort to use this printed output for that purpose. It may be necessary to use a value of `cp` that is smaller than the default in the call to `rpart()`, in order to be reasonably sure that the optimum (according to one or other criterion) has been found.

Exercise 2a: Working with the initial `Pima.rpart` model, plot on the same graph:

(1) the resubstitution error rate (.3327 times the rel error column, and subtract from 1.0) against nsplit.

(2) the cross-validated error rate (.3327 times the xerror column, and subtract from 1.0) against nsplit.

You can get these columns thus:

```
> errmat <- printcp(Pima.rpart)

Classification tree:
rpart(formula = type ~ ., data = Pima, method = "class")

Variables actually used in tree construction:
[1] age   bmi   glu   npreg ped

Root node error: 177/532 = 0.33271

n= 532

        CP nsplit rel error  xerror     xstd
1 0.265537      0   1.00000 1.00000 0.061400
2 0.056497      1   0.73446 0.83051 0.058272
```

```
3 0.025424      3   0.62147 0.74011 0.056141
4 0.020716      5   0.57062 0.72881 0.055849
5 0.014124      9   0.48588 0.71751 0.055552
6 0.011299     11   0.45763 0.70621 0.055249
7 0.010000     14   0.42373 0.71186 0.055401

> colnames(errmat)

[1] "CP"        "nsplit"    "rel error" "xerror"    "xstd"

> resub.err <- 1 - 0.3327 * errmat[, "rel error"]
> cv.err <- 1 - 0.3327 * errmat[, "xerror"]
```

Exercise 2b: Prune the model back to give the optimal predictive error rate. How does the error rate vary with the observed value of `type`? Examine the confusion matrix: (The following assumes that 12 leaves, i.e., `cp` less than about 0.013 and greater than 0.011, is optimal.)

```
> Pima.rpart <- prune(Pima.rpart, cp = 0.012)
> hat <- predict(Pima.rpart, type = "class")
> tab <- table(Pima$type, hat)
> print(1 - tab[1, ]/sum(tab[1, ]))

        No        Yes
0.06760563 0.93239437

> print(1 - tab[2, ]/sum(tab[2, ]))

       No       Yes
0.6779661 0.3220339
```

of `type`?

2 rpart Analyses – `Pima.tr` and `Pima.tr`

Exercise 3 These exercises will use the two data sets `Pima.tr` and `Pima.te`, and carrying out a variation on the calculations of Exercise 2.

What are the respective proportions of the two types in the two data sets?

Exercise 3a: Repeat Exercise 2, but now use the data frame `Pima.tr` to determine the model, and comparing cross-validation accuracies from calculations with `Pima.tr` with accuracies when `Pima.te` is used as the test data:

```
> trPima.rpart <- rpart(type ~ ., data = Pima.tr, method = "class")
> plotcp(trPima.rpart)
> printcp(trPima.rpart)

Classification tree:
rpart(formula = type ~ ., data = Pima.tr, method = "class")

Variables actually used in tree construction:
[1] age bmi bp  glu ped

Root node error: 68/200 = 0.34

n= 200
```

```
        CP nsplit rel error  xerror     xstd
1 0.220588      0   1.00000 1.00000 0.098518
2 0.161765      1   0.77941 1.02941 0.099197
3 0.073529      2   0.61765 0.86765 0.094844
4 0.058824      3   0.54412 0.85294 0.094370
5 0.014706      4   0.48529 0.70588 0.088822
6 0.010000      7   0.44118 0.76471 0.091224
```

as above, e.g.

```
> trPima.rpart <- prune(trPima.rpart, cp = 0.02)
```

Is it still true that the error rate is similar both for `No`'s and `Yes`'s. How does the expected error rate vary with the proportion of `No`'s in the target population?

Exercise 3b: Refit the model. Choose values of `cp` that will give the points on the graph obtained from `plotcp(trPima.rpart)`. Suitable values of `cp` are those that appear on the graph given by `plotcp()`. These (except for the first; what happens there?) are the geometric means of the successive pairs that are printed, and can be obtained thus:

```
> trPima.rpart <- rpart(type ~ ., data = Pima.tr, method = "class")
> cp.all <- printcp(trPima.rpart)[, "CP"]

Classification tree:
rpart(formula = type ~ ., data = Pima.tr, method = "class")

Variables actually used in tree construction:
[1] age bmi bp  glu ped

Root node error: 68/200 = 0.34

n= 200

        CP nsplit rel error  xerror     xstd
1 0.220588      0   1.00000 1.00000 0.098518
2 0.161765      1   0.77941 0.97059 0.097791
3 0.073529      2   0.61765 0.86765 0.094844
4 0.058824      3   0.54412 0.85294 0.094370
5 0.014706      4   0.48529 0.70588 0.088822
6 0.010000      7   0.44118 0.70588 0.088822

> n <- length(cp.all)
> cp.all <- sqrt(cp.all * c(Inf, cp.all[-n]))
> nsize <- printcp(trPima.rpart)[, "nsplit"] + 1

Classification tree:
rpart(formula = type ~ ., data = Pima.tr, method = "class")

Variables actually used in tree construction:
[1] age bmi bp  glu ped

Root node error: 68/200 = 0.34

n= 200

        CP nsplit rel error  xerror     xstd
```

```
1 0.220588      0   1.00000 1.00000 0.098518
2 0.161765      1   0.77941 0.97059 0.097791
3 0.073529      2   0.61765 0.86765 0.094844
4 0.058824      3   0.54412 0.85294 0.094370
5 0.014706      4   0.48529 0.70588 0.088822
6 0.010000      7   0.44118 0.70588 0.088822
```

Observe that the `nsize` is one greater than the number of splits.

Prune back successively to these points. In each case determine the cross-validated error for the training data and the error for test data. Plot both these errors against the size of tree, on the same graph. Are they comparable? The following will get you started:

```
> tr.cverr <- printcp(trPima.rpart)[, "xerror"] * 0.34

Classification tree:
rpart(formula = type ~ ., data = Pima.tr, method = "class")

Variables actually used in tree construction:
[1] age bmi bp  glu ped

Root node error: 68/200 = 0.34

n= 200

        CP nsplit rel error  xerror     xstd
1 0.220588      0   1.00000 1.00000 0.098518
2 0.161765      1   0.77941 0.97059 0.097791
3 0.073529      2   0.61765 0.86765 0.094844
4 0.058824      3   0.54412 0.85294 0.094370
5 0.014706      4   0.48529 0.70588 0.088822
6 0.010000      7   0.44118 0.70588 0.088822

> n <- length(cp.all)
> trPima0.rpart <- trPima.rpart
> te.cverr <- numeric(n)
> for (i in n:1) {
+     trPima0.rpart <- prune(trPima0.rpart, cp = cp.all[i])
+     hat <- predict(trPima0.rpart, newdata = Pima.te, type = "class")
+     tab <- table(hat, Pima.te$type)
+     te.cverr[i] <- 1 - sum(tab[row(tab) == col(tab)])/sum(tab)
+ }
```

of splits.

3 Analysis Using *svm*

Exercise 4: Compare the result also with the result from an SVM (Support Vector Machine) model. For getting predictions for the current test data, it will be necessary to use the `newdata` argument for `predict()`. Follow the prototype

```
> library(e1071)
> library(MASS)
> trPima.svm <- svm(type ~ ., data = Pima.tr)
> hat <- predict(trPima.svm, newdata = Pima.te)
> tab <- table(Pima$type, hat)
```

```
> 1 - sum(tab[row(tab) == col(tab)])/sum(tab)
> 1 - tab[1, ]/sum(tab[1, ])
> 1 - tab[2, ]/sum(tab[2, ])
```

4 Analysis Using *randomForest*

Exercise 5: Repeat the previous exercise, but now using `randomForest()`

```
> library(randomForest)

randomForest 4.5-15
Type rfNews() to see new features/changes/bug fixes.

> Pima.rf <- randomForest(type ~ ., data = Pima.tr, xtest = Pima.te[,
+     -8], ytest = Pima.te$type)
> Pima.rf

Call:
 randomForest(formula = type ~ ., data = Pima.tr, xtest = Pima.te[,      -8], ytest = Pima.te$type)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 27.5%
Confusion matrix:
     No Yes class.error
No  110  22   0.1666667
Yes  33  35   0.4852941
                Test set error rate: 23.19%
Confusion matrix:
     No Yes class.error
No  192  31   0.1390135
Yes  46  63   0.4220183
```

Note that OOB = Out of Bag Error rate, calculated using a cross-validation type approach with the training data, i.e., with the data specified by the `data` parameter. Notice that `randomForest()` is set up to give, following the one function call, an assessment of predictive error for a completely separate set of test data that has had no role in training the model. Where such a test set is available, this provides a reassuring check that `randomForest()` is not over-fitting.

(a): The function `tuneRF()` can be used for such limited tuning as `randomForest()` allows. Look up `help(tuneRF()`, and run this function in order to find an optimal value for the parameter `mtry`. Then repeat the above with this optimum value of `mtry`, and again compare the OOB error with the error on the test set.

(b): Comment on the difference between `rpart()` and `randomForest()`: (1) in the level of automation; (2) in error rate for the data sets for which you have a comparison; (3) in speed of execution.

(c): You might also try logistic regression on these data.

5 Prior Proportions

Exercise 6: Analysis with and without specification of a prior

```
> library(MASS)
> rp1 <- rpart(W.Hnd ~ Age + Smoke + Height + Sex + Exer, data = survey)
```

What prior has been implicitly assumed?

```
> rp2 <- rpart(W.Hnd ~ Age + Smoke + Height + Sex + Exer, data = survey,
+     parms = list(prior = c(0.9, 0.1)))
```

Compare the error rate for `rp1` with that for `rp2`. Is it fair to compare them?

It would be helpful to have the confusion matrices for `rp1` and `rp2`, i.e., the two-way tables that give the error rate according to whether `W.Hnd` was `Left` or `Right`.

Part X
Ordination & Clustering

Part XI
Data Exploration and Discrimination – Largish Dataset

Note: These computations are at the limit, or beyond, what a machine with 512MB memory and running Microsoft Windows is able to handle. A reboot may be necessary in order to get the calculations to run. If you cannot get calculations to run at all, try taking every second observation in the relevant data frames, and working with these reduced datasets.

One a 512MB Mac system running OS X, calculations have run without problem. The same is almost certainly true on a Unix or Linux system (I have not tested this),

The data used in this laboratory gives forest cover type (seven different types), for 581,012 sites in the United States. There are 54 columns of explanatory features (variables or dummy variables that code for qualitative effects). The 55th column holds the cover type, given as an integer in the range 1 to 7. Data, stored in the R image file **covtype.RData**, are available from
`http://www.maths.anu.edu.au/~johnm/r/misc-data/`
[To obtain the data as a text file, go to the UCI Machine Learning Repository at
`http://www.ics.uci.edu/~mlearn/MLRepository.html`

1 Data Input and Exploration

As available from the repository, data are comma delimited, and (assuming that the file has been placed in the working directory; if accessible from another location, the path must be included) can be read in with

```
covtype <- read.csv("covtype.data", header=FALSE)
```

For purposes of this laboratory, data have been placed in the image file **covtype.RData**.

The image file `covtype.RData` holds all 581,012 records. The first 11340 records have been sampled in some systematic way from an original total data set, for use as a training set. The next 3780 records, again sampled in some systematic way from the total data, were used as test data. Below, the first 11340 records will be extracted into the data frame `covtrain`, while the next 3780 records will be extracted into the data frame `covtest`.

1.1 Extraction of subsets of interest

The following extracts these data, plus a systematic sample of every 50th record, taken right through the 565,982 records that remain after the training and test sets have been extracted (if these calculations prove troublesome on your system, skip them and obtain the data sets from a web page or from a CD). It is assumed that the file **covtype.RData** is in your working directory; if it is elsewhere you must of course include the path:

```
> attach("covtype.RData")
> covtrain <- covtype[1:11340, ]
> covtest <- covtype[11340 + (1:3780), ]
> every50th <- seq(from = 11340 + 3780 + 1, to = 581012, by = 50)
> covsample <- covtype[every50th, ]
```

> **Note:** These computations may, on some systems, be unreasonably time-consuming. On my 1.25GHz G4 Powerbook with 512MB of RAM, the elapsed time for extraction of `covtrain` was 74 seconds while that for extraction of `covtest` was 35 seconds (try, e.g., `system.time(covtest <- covtype[11340+(1:3780),])` – the third of these numbers is the elapsed time). On less well endowed systems, the calculations may take an unreasonable

amount of time, or may not run at all. I would be glad to have feedback on experience with such systems.

The reason that these apparently simple operations are so time-consuming is that because data are stored columnwise, extraction of records that lie within specific ranges requires substantial manipulation within memory.

An alternative, for the input of `covtrain` and `covtest`, is:

```
covtrain <- read.csv("covtype.data", header=FALSE, nrows=11340)
covtest <- read.csv("covtype.data", header=FALSE, skip=11340, nrows=3780)
```

(Use these on Windows machines with less than 512MB of random access memory.)

Question: Which of the following is preferable?

```
every50th <- seq(from=11340+3780+1, to=581012, by=50)
every50th <- seq(from=15121, to=581012, by=50)
every50th <- 15121+(0:((581012-15121) %/% 50))*50
```

Which is more consistent with notions of literate programming? Is computational efficiency a consideration?
(The symbol `%/%` is the integer division operator. Try, e.g., `11 %/% 3`, `12 %/% 3`, etc. Try also `11 %% 3`, `12 %% 3`, which give the remainders after division.)

1.2 Image files

Having extracted these data, they can be saved to image files, which can then be attached so that they are available as required:

```
> save(covtrain, file = "covtrain.RData")
> rm(covtrain)
> save(covtest, file = "covtest.RData")
> rm(covtest)
> save(covsample, file = "covsample.RData")
> rm(covsample)
```

Now attach these image files, making `covtrain`, `covtest` and `covsample` available as required:

```
> attach("covtrain.RData")
> attach("covtest.RData")
> attach("covsample.RData")
```

1.3 Data exploration

Next, we extract some basic statistical information about these data:

```
> options(digits = 3)
> tab.all <- table(covtype$V55)
> tab.train <- table(covtrain$V55)
> tab.test <- table(covtest$V55)
> tab.sample <- table(covsample$V55)
> tab.all/sum(tab.all)
> tab.sample/sum(tab.sample)
> tab.train/sum(tab.train)
> tab.test/sum(tab.test)
```

What is interesting is that the proportions of the different cover types, in both the training and the test data, are equal. That is not the case for the data as a whole, and in this respect the "training" and "test" data are untypical.

The above suggests that, in forming the training and test data, observations were taken from the original main body of data until cover types 3-7 were largely "used up". It might be suspected that the proportions of the different cover types will vary systematically as one moves through data. The following function, which models the occurrence of the specified forest cover as a function of distance (as a proportion) through the data, can be used to check this:

```
> library(splines)
> runningprops <- function(df = covtrain, type = 1) {
+     n <- dim(df)[1]
+     propthru <- (1:n)/(n + 1)
+     occurs <- as.integer(df$V55 == type)
+     print(table(occurs))
+     cov.glm <- glm(occurs ~ bs(propthru, 6), family = binomial)
+     hat <- predict(cov.glm, type = "response")
+     cbind(propthru, hat)
+ }
> hat.train <- runningprops()
> hat.test <- runningprops(df = covtest)
> hat.sample <- runningprops(df = covsample)
> print(range(c(hat.train[, 2], hat.test[, 2], hat.sample[, 2])))
```

Next, plot this information:

```
> plot(hat.train[, 1], hat.train[, 2], ylim = c(0, 0.65))
> lines(hat.test[, 1], hat.test[, 2], col = 2)
> lines(hat.sample[, 1], hat.sample[, 2], col = 3)
```

What does this plot suggest?

Exercise 1: Repeat the above plots, but now for forest cover type 2.

Exercise 2: Another way to estimate `hat` would be as a moving average of values of the variable `occurs` in the above function. Write a function to calculate moving averages, with the window `wid` as one of its parameters.

Exercise 3: What other preliminary explorations of these data might be useful?

2 Tree-Based Classification

Now fit a tree-based model for `covtrain`:

```
> library(rpart)
> train.rpart <- rpart(V55 ~ ., data = covtrain, cp = 1e-04, method = "class")
> train.rpart <- prune(train.rpart, cp = 0.0048)
> trainhat.train <- xpred.rpart(train.rpart, cp = 0.0048)
> testhat.train <- predict(train.rpart, newdata = covtest, type = "class")
> samplehat.train <- predict(train.rpart, newdata = covsample,
+     type = "class")
```

Next, we will define a function that calculates error rates for each different cover type:

```
> errs.fun <- function(obs, predicted) {
+     tab <- table(obs, predicted)
+     grosserr <- 1 - sum(tab[row(tab) == col(tab)])/sum(tab)
+     errs <- 1 - tab[row(tab) == col(tab)]/apply(tab, 1, sum)
+     names(errs) <- paste(1:length(errs))
+     print("Overall error rate (%)")
+     print(round(100 * grosserr, 1))
+     print("Error rate (%), broken down by forest cover type")
+     print(round(100 * errs, 2))
+     cat("\n")
+     invisible(errs)
+ }
```

Now apply the function, first to `trainhat.train`, then to `testhat.train`, and finally to `samplehat.train`:

```
> errs.fun(covtrain$V55, trainhat.train)
> errs.fun(covtest$V55, testhat.train)
> errs.fun(covsample$V55, samplehat.train)
```

Exercise 4: Explain: calculations:

- What data have been used, in each case, to test the model?
- Explain the notation used for the different sets of fitted values (`trainhat.train`, `testhat.train`, `samplehat.train`.)
- Why is it that, although the three sets of error rates are relatively similar when broken down by cover type, are there are major differences in the overall error rate?
- Which, if any, of these overall error rates is it reasonable to quote in practical application of the results? If none of them seem appropriate, what error rate do you consider should be quoted?

Exercise 5: Do the following calculation and comment on the results:

```
> sample.rpart <- rpart(V55 ~ ., data = covsample, cp = 1e-04,
+     method = "class")
> sample.rpart <- prune(sample.rpart, cp = 0.001)
> samplehat.sample <- xpred.rpart(sample.rpart, cp = 0.001)
> samplehat.sample <- factor(samplehat.sample, levels = 1:7)
> trainhat.sample <- predict(sample.rpart, newdata = covtrain,
+     type = "class")
> testhat.sample <- predict(sample.rpart, newdata = covtest, type = "class")
> errs.fun(covtrain$V55, trainhat.sample)
> errs.fun(covtest$V55, testhat.sample)
> errs.fun(covsample$V55, samplehat.sample)
```

3 Use of `randomForest()`

For use of `randomForest()`, calculations speed up greatly if explanatory variables are input as columns of a matrix, while (for classification) the response variable is input as a vector of class factor.

Exercise 6: Run the following computations, and interpret the output, comparing it with the relevant output from `rpart()`. Suggest why the error rates may be so much lower than those from `rpart()`:

```
> library(randomForest)
> xcovtrain <- as(covtrain[, 1:54], "matrix")
> ycovtrain <- covtrain[, 55]
> xsampletrain <- as(covsample[, 1:54], "matrix")
> ysampletrain <- covsample[, 55]
> ycovtrain <- factor(covtrain[, 55])
> ysampletrain <- factor(covsample[, 55])
> covtrain.rf <- randomForest(x = xcovtrain, y = ycovtrain, xtest = xsampletrain,
+     ytest = ysampletrain)
> covtrain.rf
```

4 Further comments

This has been a preliminary exploration of these data. The model that is optimal changes as one moves through the data. This can be verified by selecting successsive subsets of perhaps 10,000 successive observations at various points through the data (e.g.: 1-10,000, 101,000-110,000, ...), and comparing: (1) gross error rate for that subset as calculated by using `xpred.rpart()` and `errs.fun()`, with (2) the gross error rate when `train.rpart()` or (more appropriately) `sample.rpart()` is used determine fitted values for that subset.

Thus, after the first 15,120 (11,340 + 3780) records, a reasonable guess is that the order of records reflects an ordering of the data according to geographical location. The ordering holds information that can be used, even without knowledge of more exact geographical locations, to get improved model predictions.

References

Charig, C. R., 1986. Comparison of treatment of renal calculi by operative surgery, percutaneous nephrolithotomy, and extracorporeal shock wave lithotripsy. *British Medical Journal*, 292:879–882.

Gordon, N. C. et al.(1995): 'Enhancement of Morphine Analgesia by the $GABA_B$ against Baclofen'. *Neuroscience 69: 345-349.*

Intersalt Cooperative Research Group. 1988. Intersalt: an international study of electrolyte excretion and blood pressure: results for 24 hour urinary sodium and potassium excretion. British Medical Journal 297: 319-328.

Maindonald, J. H.; Waddell, B. C.; and Petry, R. J., 2001. Apple cultivar effects on codling moth (Lepidoptera: Tortricidae) egg mortality following fumigation with methyl bromide. *Postharvest Biology and Technology*, 22:99–110.

Meyer, M.C. and Finney, T. (2005): 'Who wants airbags?'. *Chance* 18:3-16.

Wilkinson, G. N. & Rogers, C. E. 1973. *Symbolic description of models in analysis of variance.* Applied Statistics 22: 392-399.