Practice with R – Laboratory Exercises

John Maindonald

December 27, 2006

Laboratory exercises make extensive use of datasets from the *DAAG* package. Make sure that it is installed. Data that are not included in the *DAAG* package, are available from

`http://www.maths.anu.edu.au/~johnm/datasets/`

References

MAINDONALD, J.H. AND BRAUN, W.J., $2^n d$ edn, 2007. *Data Analysis and Graphics Using R – An Example-Based Approach.* Cambridge University Press.
`<URL:http://www.maths.anu.edu.au/~johnm/r-book.html>`
[This is aimed at practicing scientists who have some modest statistical sophistication, and at statistical practitioners. It demonstrates the use of the R system for data analysis and for graphics.]

MAINDONALD, J.H. 2006A. *Statistical Perspectives on Data Mining* (referred to as SPDM).
`http://www.maths.anu.edu.au/~johnm/courses/dm`

MAINDONALD, J.H. 2006B. *The R System – An Introduction and Overview.*
`http://www.maths.anu.edu.au/~johnm/courses/dm`

Contents

Part I
Data Input, Graphs, & Data Manipulation

1 Data Input

Exercise 1
The files **molclock.txt**, **molclock1.txt** and **molclock2.txt** are available from `http://www.maths.anu.edu.au/~johnm/datasets/text/` Download each of these files, into your working directory. In each case, examine their contents. From the R command line you can do this using, e.g., `readLines("molclock.txt")`. Alternatively, you can use R's script editor (under Windows, go to File | Open script...), or use another editor such as the Windows tinn-R editor that is designed to interface to R.
Use `read.table()` to read each of them into R. Check carefully whether you need `header=TRUE`. Then display the data frame and check that the data have been input correctly.

Exercise 2
For the next exercise, it will be handy to use the `paste()` function. Compare the following:

```
> paste("Leo", "the", "lion")

[1] "Leo the lion"

> paste("a", "b")

[1] "a b"

> paste("a", "b", sep = "")

[1] "ab"
```

Given

```
> webpage <- "http://wwwmaths.anu.edu.au/~johnm/datasets/text/"
```

paste the file name onto the end to give

```
"\http://wwwmaths.anu.edu.au/~johnm/datasets/text/molclock.txt"
```

Exercise 3
Files can be input directly from a web page. Try

```
> webpage <- "http://wwwmaths.anu.edu.au/~johnm/datasets/text/"
> molclock <- read.table(url(paste(webpage, "molclock.txt", sep = "")))
```

Use this approach to input the file **travelbooks.txt** that is available from this same web page.
The function `read.csv()` is a variant of `read.table()` that is designed to read in comma delimited files such as may be obtained from Excel. Use this function to read in the file **houses.csv** that is availble from this same web page.

Exercise 4
For a more challenging data input task, input the data from the file **bostonc.txt**. You can create this by attaching the *DAAG* package (enter `library(DAAG)`) and entering

```
> datafile("bostonc")
```

2 Subsets of Dataframes

Exercise 5
Use `head()` to check the names of the columns, and the first few rows of data, in the data frame `rainforest` (*DAAG*). Use `table(rainforest$species)` to check the names and numbers of each species that are present in the data. The following extracts the rows for the species *Acmena smithii*

```
> Acmena <- subset(rainforest, species == "Acmena smithii")
```

The following extracts the rows for the species `Acacia mabellae` and `Acmena smithii`

```
> AcSpecies <- subset(rainforest, species %in% c("Acacia mabellae",
+     "Acmena smithii"))
```

(a) From the data frame `ais` (*DAAG*), extract the data for the rowers.

(b) From the data frame `ais` (*DAAG*), extract the data for the rowers, the netballers and the tennis players.

3 Scatterplots

Exercise 6
Using the Acmena data 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 7
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 8
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.

4 Information about the Columns of Data Frames

Exercise 9
Functions that may be used to get information about data frames include `str()`, `dim()`, `row.names()` and `names()`. Try each of these functions with the data frames `allbacks`, `ant111b` and `tinting` (all in *DAAG*).
For getting information about the class of each column use e.g.

```
> sapply(ant111b, class)
```

or

```
> unlist(sapply(ant111b, class))
```

This applies the function `class()` to each column of the data frame.
For each of these data frames, use `table()` to tabulate the number of values for each level.

5 Factors

Exercise 10
Investigate the use of the functions `as.character()` and `unclass()` with a factor argument. Comment on their use in the following code.

```
> par(mfrow = c(1, 2), pty = "s")
> plot(weight ~ volume, pch = unclass(cover), data = allbacks)
> plot(weight ~ volume, data = allbacks, type = "n")
> with(allbacks, text(weight ~ volume, labels = as.character(cover)))
> par(mfrow = c(1, 1))
```

[The setting `mfrow=c(1,2)` gives side by side plots. The setting `pty="s"` gives a square plotting region.]

Exercise 11
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)
```

6 Stripplots (base graphics) and Stripcharts (*lattice*)

Exercise 12
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 `vince111b`.

7 Sorting

Exercise 13
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.]

```
> Acmena <- subset(rainforest, species == "Acmena smithii")
> ord <- order(Acmena$dbh)
> acm <- Acmena[ord, ]
```

Sort the row names of `possumsites` (*DAAG*) into alphanumeric order. Reorder the rows of `possumsites` in order of the row names.

Part II
For loops, Functions & Data Exploration

```
> library(DAAG)
```

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 to use a `while` loop to achieve the same result.

(c) Show how to 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
A ramdom sample of 500 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 4
Here is an alternative way to do the calculations of Exercise 3. 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.

2 Functions

Exercise 5
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; (b) if there are missing values, the mean and standard deviation are calculated for the remaining values.

Exercise 6
Write a function that does the calculations of Exercises 3 (and 4) above for an arbitrary choice of sample size `n` in place of `n=4`), and for an arbitary number of samples `numsamp` in place of `numsamp=500`. It should return the vector of sample means.

Exercise 7

(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.]

3 Data Exploration – Distributions of Data Values

Exercise 8
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 9
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?

Exercise 10

(a) Density plots may be better than the strip plots of Exercise 9 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?]

(b) 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
Practice with R – Further Exercises

Exercises that are more technical or challenging are marked with an asterisk.

```
> library(DAAG)
```

1 Tabulation

Exercise 1
Tabulate the number of observations in each of the different districts in the data frame `rockArt` (*DAAGxtras*). Create a factor `groupDis` in which all `District`s with less than 5 observations are grouped together into the category `other`.

```
> library(DAAGxtras)
> groupDis <- as.character(rockArt$District)
> tab <- table(rockArt$District)
> le4 <- rockArt$District %in% names(tab)[tab <= 4]
> groupDis[le4] <- "other"
> groupDis <- factor(groupDis)
```

2 Smooth Curves

Exercise 2
The following compares three different smoothing functions. Comment on the different syntax and, in the case of `lowess()`, the different default output that is returned. Why, for the smooth obtained using `lowess()`, is it necessary to sort data in order of values of `dbh`? (Try omitting the ordering, and observe the result.)

```
> Acmena <- subset(rainforest, species == "Acmena smithii")
> plot(wood ~ dbh, data = Acmena)
> ord <- order(Acmena$dbh)
> with(Acmena[ord, ], lines(predict(loess(wood ~ dbh)) ~ dbh))
> 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?

*Exercise 3**
Here is yet another way to add a smooth curve.

```
> plot(wood ~ dbh, data = Acmena)
> with(Acmena, lines(smooth.spline(wood ~ dbh, spar = 0.5)))
> with(Acmena, lines(smooth.spline(wood ~ dbh, df = 4), col = "red"))
```

Experiment with different choices of `df`. What choice gives the same smoothness as for `spar=0.5`

3 Avoiding For Loops

*Exercise 4**
Here is code for the calculations that compare the relative population growth rates for the Australian states and territories (Exercise 2 of Laboratory Exercises 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 as the use of an explicit loop. [By wrapping the code in the function `invisible()`, printed output that gives no useful information can be 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`.

4 Different Ways to Do a Calculation – Timings

Exercise 5
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 each element by 1:

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

Exercise 5, 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.

5 Functions

Exercise 6

(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 7**
Data in the data frame `fumig` (*DAAGxtras*) 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.

```
> "calcCT" <- 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`.
Attach the `DAAGxtras` package, 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?

Exercise 8
A workspace includes objects `possum1`, `possum2`, ...`possum5`. The folowing shows how to get the size of one of these objects one at a time.

```
> possum1 <- rnorm(10)
> object.size(possum1)

[1] 152
```

The names of the objects can be obtained with

```
> nam <- ls(pattern = "^possum")
```

To get the sizes from the names that are held in `nam`, do

```
> sapply(nam, function(x) object.size(get(x)))
```

Create objects `possum2`, ...`possum5`, and enter this command. Explain the successive steps in the computation.
[Hint: Compare `class(possum1)` with `class("possum1")`, and `object.size(possum1)` with `object.size("possum1")`]

Exercise 9
The function `ls()` lists, by default, the names of objects in the current environment. If used from the command line, it lists the objects in the workspace. If used in a function, it lists the names of the function's local variables. To get a listing of the contents of the workspace, do the following

```
> workls <- function() {
+     a <- 0
+     ls(name = ".GlobalEnv")
+ }
> workls()

 [1] "Acmena"      "Pima.miss"   "Pima.nomiss" "anymiss"     "betterls"
 [6] "calcCT"      "count.na"    "delI"        "flexisub"    "groupDis"
[11] "heights"     "i"           "le4"         "nam"         "oldflexi"
[16] "oldpar"      "ord"         "possum1"     "subI"        "tab"
[21] "test"        "vv"          "workls"      "xx"          "xxDF"
[26] "xxMAT"       "z"           "zerodists"   "zz"
```

[If `ls(name=".GlobalEnv")` is replaced by `ls()`, the function lists the names of its local variables.]

Write a function that calculates the sizes of all objects in the workspace, then listing the names and sizes of the largest ten objects.

6 Data Exploration – A Further Exercise

Exercise 10
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?

```
> library(lattice)
> 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.]

7 Esoterica

*Exercise 11**
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`

Exercise 11, continued*
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.