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

John Maindonald

August 14, 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:

Contents

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 `vince111b`.

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 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 5
Here is an alternative way to do the calculations of Exercise 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)
> for (i in 1:100) av[i] <- mean(sample(y, 4, replace = TRUE))
```

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
+ }
> d2 <- twot(n = 999)
> dens <- density(d2)
> plot(dens)
> sum(d2 < 0)/length(d2)

[1] 0.173
```

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] -0.87  2.29
```

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 = "")))
> sampvalues <- 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.]

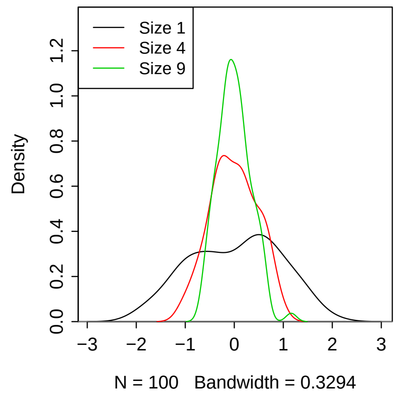

Figure 1: Empirical density curve, for a normal population and for the sampling distributions of means of samples of sizes 4 and 9 from that population.

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.

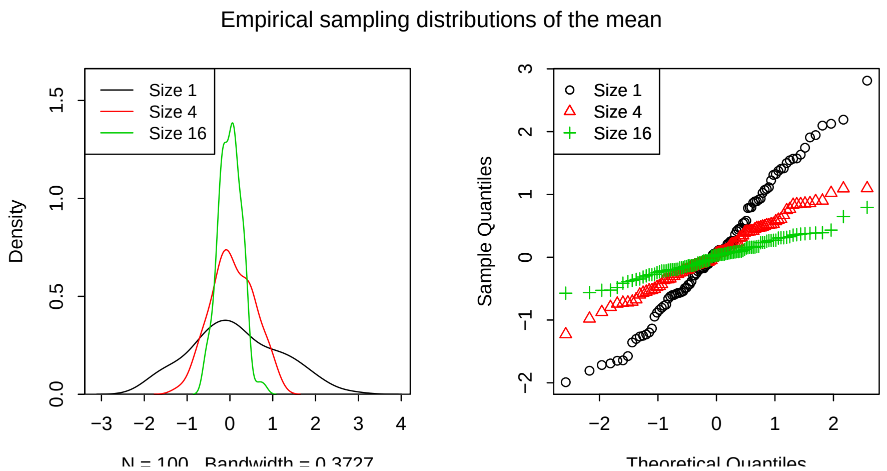

Figure 2: Empirical density curves, for a normal population and for the sampling distributions of means of samples of sizes 4 and 9, are in the left panel. The corresponding normal probability plots are shown in the right panel.

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?