2
An Overview of R

Demonstrations	`demo()` # List available demonstrations `demo(graphics)` # Demonstrations of base graphics
Help	The main help function is `help()`. Note `help(help)` `help(plot)` `# Show the help page for plot()`

Column objects	`width = c(11.3, 13.1, 20, 21.1, 25.8, 13.1)`
	`height = c(23.9, 18.7, 27.6, 28.5, 36, 23.4)`
	Read the symbol `c` as "concatenate", or perhaps "column".

Dataframe	A data frame is a list of column objects (all of the same length)
	`widheight <- data.frame(`
	`width = c(11.3, 13.1, 20, 21.1, 25.8, 13.1),`
	`height = c(23.9, 18.7, 27.6, 28.5, 36, 23.4)`
	`}`

Other topics	Arithmetic operations; simple plots; input of data from a file.

2.1 Practice with R commands

The R language has the standard abilities for evaluating arithmetic and logical expressions. A wide variety of functions extends these basic arithmetic and logical abilities.

Try the following

```
2+3          # Simple arithmetic
```

```
[1] 5
```

```
1:5          # The numbers 1, 2, 3, 4, 5
```

```
[1] 1 2 3 4 5
```

```
mean(1:5)
```

```
[1] 3
```

```
sum(1:5)     # Sum the numbers 1, 2, 3, 4, 5
```

```
[1] 15
```

```
(2:4)^9    # 2^9 (2 to the power of 9), 3^9, 4^9
```

```
[1]    512  19683 262144
```

In addition to `log()`, note `log2()` and `log10()`:

A change by a factor of 2 is a one unit change on a log2 scale. A change by a factor of 10 is a one unit change on a log10 scale.

```
log2(c(0.5, 1, 2, 4, 8))
```

```
[1] -1  0  1  2  3
```

```
log10(c(0.1, 1, 10, 100, 1000))
```

```
[1] -1  0  1  2  3
```

It turns out, surprisingly often, that logarithmic scales are apprapriate for one or other type of graph. Logarithmic scales focus on relative change — by what factor has the value changed?

The following uses the relational operator >:

Other relational operators are
< >= < <= == !=

```
(1:5) > 2  # Returns FALSE FALSE  TRUE  TRUE  TRUE
```

```
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

Functions

Functions have a central role in the use of the R system. Common functions that R users should quickly get to know include `print()`, `plot()` and `help()`.

Use the function `help()` to get information on other functions. Thus, type `help(plot)` to view the help page for `plot()`.

Demonstrations

Demonstrations can be highly helpful in learning to use R's functions. The following are some of demonstrations that are available for graphics functions:

Images and perspective plots:

```
demo(image)
demo(persp)
```

For the following, the *vcd* package must be installed:

```
library(vcd)
demo(mosaic)
```

```
demo(graphics)   # Type <Enter> for each new graph
library(lattice)
demo(lattice)
```

Especially for `demo(lattice)`, it pays to stretch the graphics window to cover a substantial part of the screen. Place the cursor on the lower right corner of the graphics window, hold down the left mouse button, and pull.

The following lists available demonstrations:

```
## List demonstrations in attached packages
demo()
## List demonstrations in all installed packages
demo(package = .packages(all.available = TRUE))
```

2.2 A Short R Session

We will work with the data set shown in Table 2.1:

	thickness	width	height	weight	volume	type
Aird's Guide to Sydney	1.30	11.30	23.90	250	351	Guide
Moon's Australia handbook	3.90	13.10	18.70	840	955	Guide
Explore Australia Road Atlas	1.20	20.00	27.60	550	662	Roadmaps
Australian Motoring Guide	2.00	21.10	28.50	1360	1203	Roadmaps
Penguin Touring Atlas	0.60	25.80	36.00	640	557	Roadmaps
Canberra - The Guide	1.50	13.10	23.40	420	460	Guide

Table 2.1: Weights and volumes, for six Australian travel books.

Entry of columns of data from the command line

Data may be entered from the command line, thus:

Read c as "concatenate", or perhaps as "column". It joins elements together into a vector, here numeric vectors.

```
volume ← c(351, 955, 662, 1203, 557, 460)
weight ← c(250, 840, 550, 1360, 640, 420)
```

Now store the descriptions in the character vector `description`:

```
description ← c("Aird's Guide to Sydney",
 "Moon's Australia handbook",
 "Explore Australia Road Atlas",
 "Australian Motoring Guide",
 "Penguin Touring Atlas", "Canberra – The Guide")
```

The end result is that objects `volume`, `weight` and `description` are stored in the workspace.

Listing the workspace contents

Use the function `ls()` to examine the current contents of the workspace.

```
ls()
```

```
[1] "description" "oldopt"      "volume"
"weight"
```

```
ls(pattern="ume")   # Names that include "ume"
```

```
[1] "volume"
```

```
ls(pattern="^des")  # Names that start with "des"
```

```
[1] "description"
```

Operations with `vectors`

Here are the values of `volume`

```
volume
```

```
[1]  351  955  662 1203  557  460
```

To extract the final element of `volume`, do:

```
volume[6]
```

```
[1] 460
```

For the ratio of weight to volume, i.e., the density, do:

```
round(weight/volume, 2)
```

```
[1] 0.71 0.88 0.83 1.13 1.15 0.91
```

A simple plot

Figure 2.1 plots `weight` against `volume`, for the six Australian travel books. Note the use of the graphics formula `weight ~ volume` to specify the $x-$ and $y-$variables. It takes a similar from to the "formulae" that are used in specifying models, and in the functions `xtabs()` and `unstack()`.

Code for Figure 2.1 is:

```
## Code
plot(weight ~ volume, pch=16, cex=1.5)
  # pch=16: use solid blob as plot symbol
  # cex=1.5: point size is 1.5 times default
## Alternative
plot(volume, weight, pch=16, cex=1.5)
```

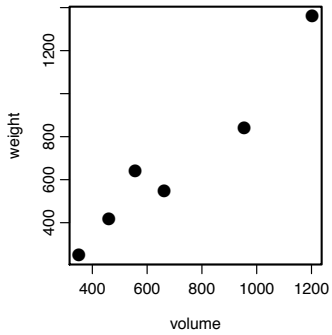

Figure 2.1: Weight versus volume, for six Australian travel books.

The axes can be labeled:

```
plot(weight ~ volume, pch=16, cex=1.5,
     xlab="Volume (cubic mm)", ylab="Weight (g)")
```

Interactive labeling of points (e.g., with species names) can be done interactively, using `identify()`:

Use `text()` for non-interactive labeling of points.

```
identify(weight ~ volume, labels=description)
```

Then click the left mouse button above or below a point, or on the left or right, depending on where you wish the label to appear. Repeat for as many points as required.

On most systems, the labeling can be terminated by clicking the right mouse button. On the Windows GUI, an alternative is to click on the word "Stop" (then on "Stop locator") that appears at the top left of the screen, just under "Rgui" on the left of the blue panel header of the R window.

Formatting and layout of plots

There are extensive abilities that may be used to control the formatting and layout of plots, and to add features such as special symbols, fitted lines and curves, annotation (including mathematical annotation), colors and so on.

2.3 Data frames – Grouping columns of data

Data frames	Store data that have a cases by columns layout.
Creating data frames	Enter from the command line (small datasets) Or: Use `read.table()` to input from a file.
Columns of data frames	`travelbooks$weight` or `travelbooks[, 4]` or `travelbooks[, "weight"]`

Data frames are pervasive in R. Most datasets that are included with R packages are supplied as data frames.

The following code groups the several columns of Table 2.1 together, under the name `travelbooks`. It is tidier to have matched columns of data grouped together into a data frame, rather than separate objects in the workspace.

```
## Group columns together into a data frame
travelbooks ← data.frame(
   thickness = c(1.3, 3.9, 1.2, 2, 0.6, 1.5),
   width = c(11.3, 13.1, 20, 21.1, 25.8, 13.1),
   height = c(23.9, 18.7, 27.6, 28.5, 36, 23.4),
   weight = weight,  # Use values entered earlier
   volume = volume,  # Use values entered earlier
   type = c("Guide", "Guide", "Roadmaps", "Roadmaps",
            "Roadmaps", "Guide"),
   row.names = description
)
## Remove objects that are not now needed.
rm(volume, weight, description)
```

The vectors `weight`, `volume` and `description` were entered earlier, and (unless subsequently removed) need not be re-entered.

It is a matter of convenience whether the description information is used to label the rows, or alternatively placed in a column of the data frame. Vectors of character, such as `type`, are by default stored as factors. In the data as stored, `"Guide"` is replaced by 1 and `"Roadmaps"` by 2. Stored with the factor is the information that 1 is `"Guide"` and 2 is `"Roadmaps"`.

While there are many contexts where factors and character vectors are interchangeable, there are important exceptions.

Accessing the columns of data frames

The following are alternative ways to extract the column `weight` from the data frame:

```
travelbooks[, 4]
travelbooks[, "weight"]
travelbooks$weight
travelbooks[["weight"]]   # Reference as a list.
```

There are several mechanisms that avoid repeated reference to the name of the data frame. The following are alternative ways to plot `weight` against `volume`:

1. Use the parameter `data`*, where available, in the function call*

```
plot( weight ~ volume, data=travelbooks)
```

Most modeling functions and many plotting functions accept a `data` argument.

2. Use `with()`*: Take columns from specified data frame*

```
## Take columns from the specified data frame
with(travelbooks, plot(weight ~ volume))
```

With both of these mechanisms, columns of the data frame are taken in preference to any object of the same name that may happen to be in the workspace.

A third option, usually best avoided, is to use `attach()` to add the data frame to the search list. In this case, names in the workspace take precedence over column names in the attached data frame – not usually what is wanted if there are names in common.

Attachment of a data frame:

```
attach(travelbooks)
plot( weight ~ volume)
detach(travelbooks)
 ## Detach when not required
```

Subsection 4.3.2 will discuss the attaching of packages and image files.

2.4 Input of Data from a File

The function `read.table()` is designed for input from a file into a data frame. As an example, observe input of the data in Table 2.1. The *DAAG* package has a function `datafile()` that is convenient to use to place into the working directory this and/or several other files that can be used for demonstrating input of data from a file.[1]

[1] *DAAG* must be installed.

The first two lines hold the column headings[2] and first row of data, thus:

[2] The first column holds the row names, and has no header.

	thickness	width	height	weight	volume	type
Aird's Guide to Sydney	1.30	11.30	23.90	250	351	Guide
...						

First store the file in the working directory, `datafile()` using the function from the *DAAG* package.

Use of `datafile()` to place one or more files in the working directory is a convenience for teaching purposes. The same command is used, independent of the setup on individual computers, to place the file in the working directory.

```
## Place the file in the working directory
## NB: DAAG must be installed
library(DAAG)      # Attach the DAAG package
```

```
datafile("travelbooks")
```

```
Data written to file: travelbooks.txt
```

Use `dir()` to check that the file is indeed in the working directory:

```
dir()              # List files in working directory
```

The following reads the file into an R data frame:

Row 1 has column names. Column 1 has row names.

```
## Input the file to the data frame travelbooks
travelbooks ← read.table("travelbooks.txt",
                         header=TRUE, row.names=1)
```

The assignment places the data frame in the workspace, with the name `travelbooks`. The first seven columns are numeric. The character data in the final column is by default stored as a factor.

Data input – points to note:

- The R Commmander GUI can be helpful for data input. It displays entry boxes for input settings that users may find it expedient to change.
- Use the parameter `heading` to control whether (`heading=TRUE`) or not (`heading=FALSE`) the first row of input is used for row names.
- See `help(read.table)` for details of parameter settings that may need changing to match the data format.
- Character vectors that are included as columns in data frames become, by default, factors.

Section 7.4 discusses common types of input errors.

Character vectors and factors can often, but by no means always, be treated as equivalent.

2.5 Help Information

Note also:
`help.search() # see below`
`apropos()`

```
help()              # Help for the help function
help(plot)          # Show the help page for plot
?plot               # Shorthand for help(plot)
example(plot)       # Run examples from help(plot)
```

Examples that are included on help pages

All functions have help pages. Most help pages include examples, which can be run using the function `example()`. Be warned that, even for relatively simple functions, some of the examples may illustrate non-trivial technical detail.

To work through the code for an example, look on the screen for the code that was used, and copy or type it following the command line prompt. Or get the code from the help page.

Access to help resources from a browser screen

Type `help.start()` to display a screen that gives a browser interface to R's help resources. Note especially Frequently Asked Questions and Packages. Under Packages, click on base to get information on base R functions. Standard elementary statistics functions are likely to be found under stats, and base graphics functions under graphics.

Official R manuals include An Introduction to R, a manual on Writing R Extensions, and so on.

Also available, after clicking on a package name, is a link User guides, package vignettes and other documentation. Click to get details of any documentation that is additional to the help pages.

Vignettes

Many packages have vignettes; these are typically pdf or (with version 3.0.x of R) HTML files that give information on the package or on specific aspects of the package. To get details of vignettes that are available for one or other package, call `browseVignettes()` with the package name (in character string form) as argument. Thus, to get details of the vignettes that are available for the *knitr* package, enter `browseVignettes(package="knitr")`.

The browser window that appears will list the vignettes, with the option to click on links that, in most cases, offer a choice of one of PDF and HTML, source, and R code.

Abilities in the *knitr* package and elsewhere allow the creation of HTML vignettes from HTML or R Markdown *source* files, and of PDF vignettes from LaTeX. A Markdown or HTML or LaTeX source document is enhanced with markup in which R code is embedded. See Chapter 3.

Searching for key words or strings

Use `help.search()` to look for functions that include a specific word in their alias or title. For example, to look for a function for bar plots, try

```
help.search("bar")
??"bar"          # Shorthand for help.search("bar")
```

This draws attention to the function `barplot()`. Type in `help(barplot)` to see the help page, and/or `example(barplot)` to run the examples.

Functions for operating on character strings are likely to have "str" or "char" in their name. Try

```
help.search("str", package="base")
help.search("char", package="base")
```

By default, all available installed packages are searched. Limiting the search, here to `package="base"`, will often give more manageable and useful output.

The function `RSiteSearch()` searches web-based resources, including R mailing lists, for the text that is given as argument.

2.6 *Summary and Exercises*

2.6.1 *Summary*

- Useful help functions are `help()` (for getting information on a known function) and `help.search()` (for searching for a word that is used in the header for the help file).
- The function `help.start()`, starts a browser window from which R help information can be accessed.

NB also: Use `apropos()` to search for functions that include a particular text string as part of their name.

- Use `read.table()`, or an alias such as `read.csv()`, to input rectangular files. As an alternative, consider use of the R Commander GUI.
- Data frames collect together under one name columns that all have the same length. Columns of a data frame can be any mix of, among other possibilities: logical, numeric, character, or factor.
- The function `with()` attaches a data frame temporarily, for the duration of the call to `with()`.
- For simple forms of scatterplot, use `plot()` and associated functions, or perhaps the *lattice* function `xyplot()`.

Use `with()` in preference to the `attach()` / `detach()` combination.

2.6.2 *Exercises*

1. Use the function `datafile()` (*DAAG* or *DAAG*), with the argument `file=bestTimes`, to place the file **bestTimes.txt** into the working directory.[3]

   (a) Examine the file. (Include the path if the file is not in the working directory.)

   ```
   ## Input from file that is in working directory
   datafile("bestTimes")
   ## file.show("bestTimes.txt")
   bestTimes ← read.table("bestTimes.txt")
   ```

   (b) The `bestTimes` file has separate columns that show hours, minutes and seconds. Use the following to add the new column `Time`, then omitting the individual columns as redundant

   ```
   bestTimes$Time ← with(bestTimes,
                         h*60 + min + sec/60)
     # Time in minutes
   names(bestTimes)[2:4]    # Check column names
   ```

   ```
   [1] "h"   "min" "sec"
   ```

   ```
   bestTimes ← bestTimes[, -(2:4)]
                          # omit columns 2:4
   ```

   (c) Here are alternative ways to plot the data

   ```
   plot(Time ~ Distance, data=bestTimes)
   ## Now use a log scale
   plot(log(Time) ~ log(Distance), data=bestTimes)
   plot(Time ~ Distance, data=bestTimes, log="xy")
   ```

   (d) Now save the data into an image file in the working directory

   ```
   save(bestTimes, file="bestTimes.RData")
   ```

   For further explanation of the function `save()`, see the next chapter.

[3] Alternatively, copy it from the web page `http://www.maths.anu.edu.au/~johnm/datasets/text/` and place it in the working directory.