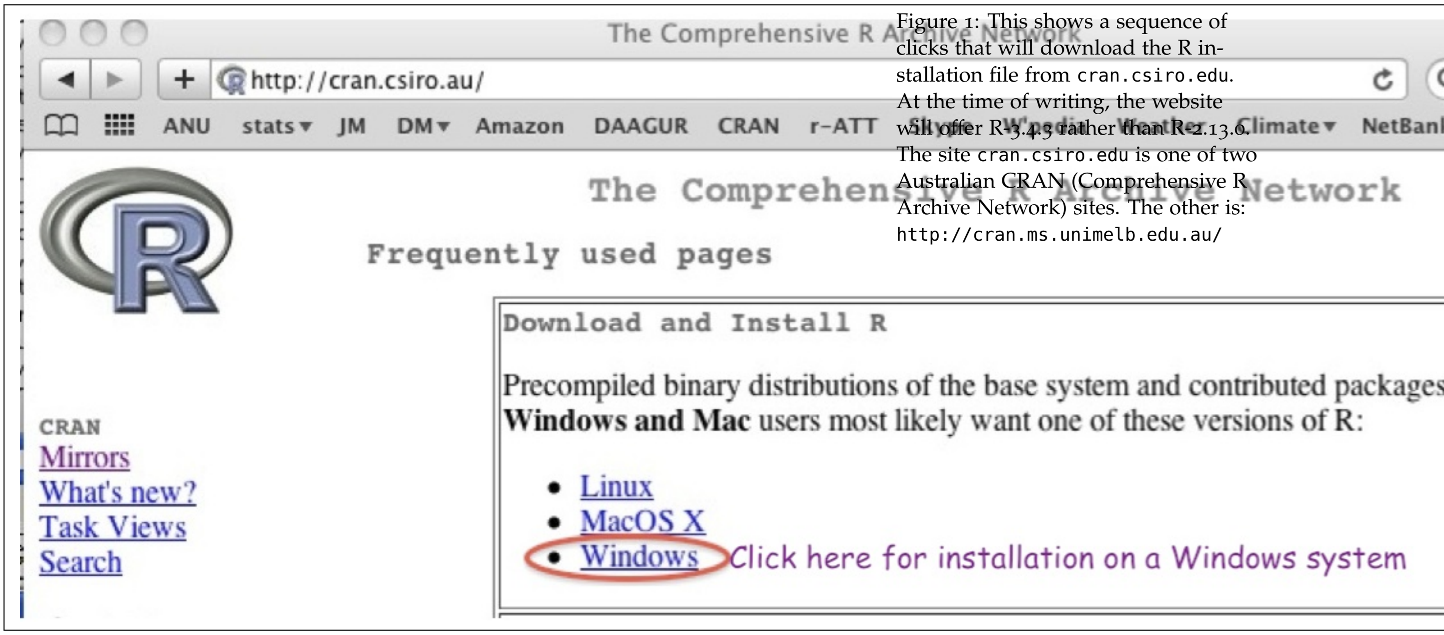
```
## Error in setwd(input_dir()):  cannot change working directory
```

```
## Warning in file(con, "r"):  cannot open file 'scripts/gettingStarted-code.R':
No such file or directory
## Error in file(con, "r"):  cannot open the connection
```

Installation of R

Click as indicated in the successive panels to download R for Windows from the web page `http://cran.csiro.au`:



The Comprehensive R Archive Network

Frequently used pages

CRAN
Mirrors
What's new?
Task Views
Search

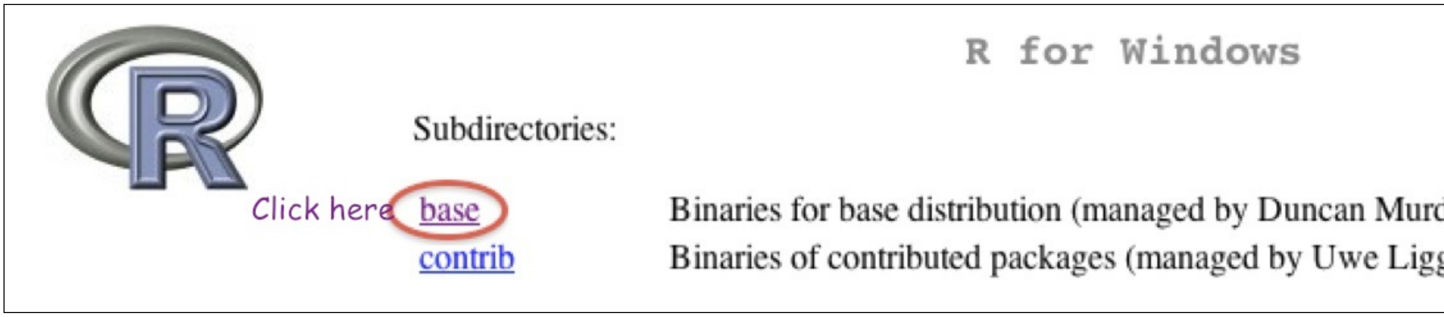
Download and Install R

Precompiled binary distributions of the base system and contributed packages **Windows and Mac** users most likely want one of these versions of R:

- Linux
- MacOS X
- Windows

Click here for installation on a Windows system

Figure 1: This shows a sequence of clicks that will download the R installation file from `cran.csiro.edu`. At the time of writing, the website will offer R-3.4.3 rather than R-2.13.0. The site `cran.csiro.edu` is one of two Australian CRAN (Comprehensive R Archive Network) sites. The other is: `http://cran.ms.unimelb.edu.au/`

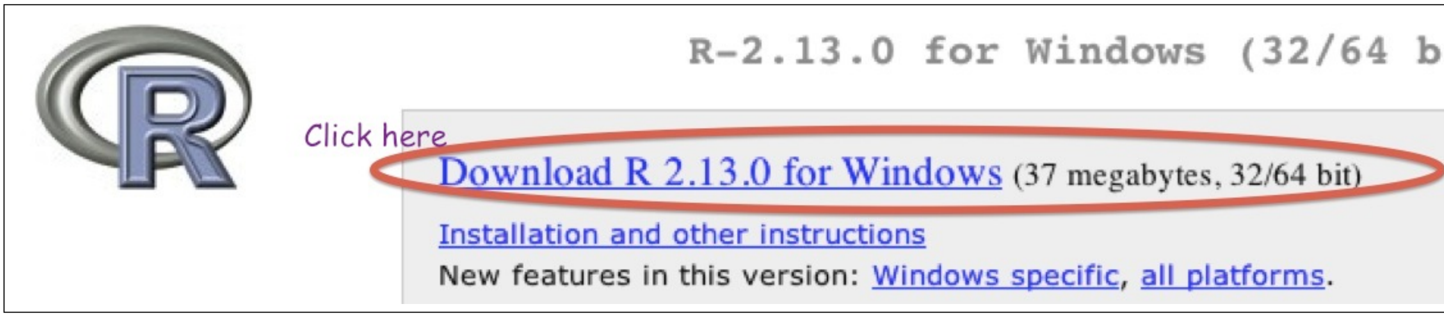


R for Windows

Subdirectories:

Click here

base	Binaries for base distribution (managed by Duncan Murd
contrib	Binaries of contributed packages (managed by Uwe Ligg



R-2.13.0 for Windows (32/64 b

Click here

Download R 2.13.0 for Windows (37 megabytes, 32/64 bit)

Installation and other instructions

New features in this version: Windows specific, all platforms.

Click on the downloaded file to start installation. Most users will want to accept the defaults. The effect is to install the R base system, plus recommended packages, with a standard “off-the-shelf” setup. Windows users will find that one or more desktop R icons have been created as part of the installation process.

Figure 2: On 64-bit Windows systems the default installation process creates two icons, one for 32-bit R and one for 64-bit R. Additional icons can be created as desired.

Depending on the intended tasks, it may be necessary to install further packages. Section 1.3 describes alternative ways to install packages.

An optional additional step is to install RStudio. RStudio has abilities that help in managing workflow, in navigating between projects, and in accessing R system information. See Section **??**.

Clicking on the RStudio icon to start a session will at the same time start R. RStudio has its own command line interface, where users can type R commands.

First steps

Readers who have RStudio running can type their commands in the RStudio command line panel.

Click on an R icon to start an R session. This opens an R command window, prints information about the installed version of R, and gives a command prompt.
The > prompt that appears on the final line is an invitation to start typing R commands:

Thus, type 2+5 and press the Enter key. The display shows:
 The result is 7. The output is immediately followed by the > prompt, indicating that R is ready for another command.

The [1] says, a little strangely, “first requested element will follow”. Here, there is just one element.

Try also:

The object result is stored in the *workspace*. The *workspace* holds objects that the user has created or input, or that were there at the start of the session and not later removed

Type ls() to list the objects in the workspace, thus:

Typing result on the command line has printed the value 7.

Technically, the *workspace* is one of a number of *databases* where objects may be stored.

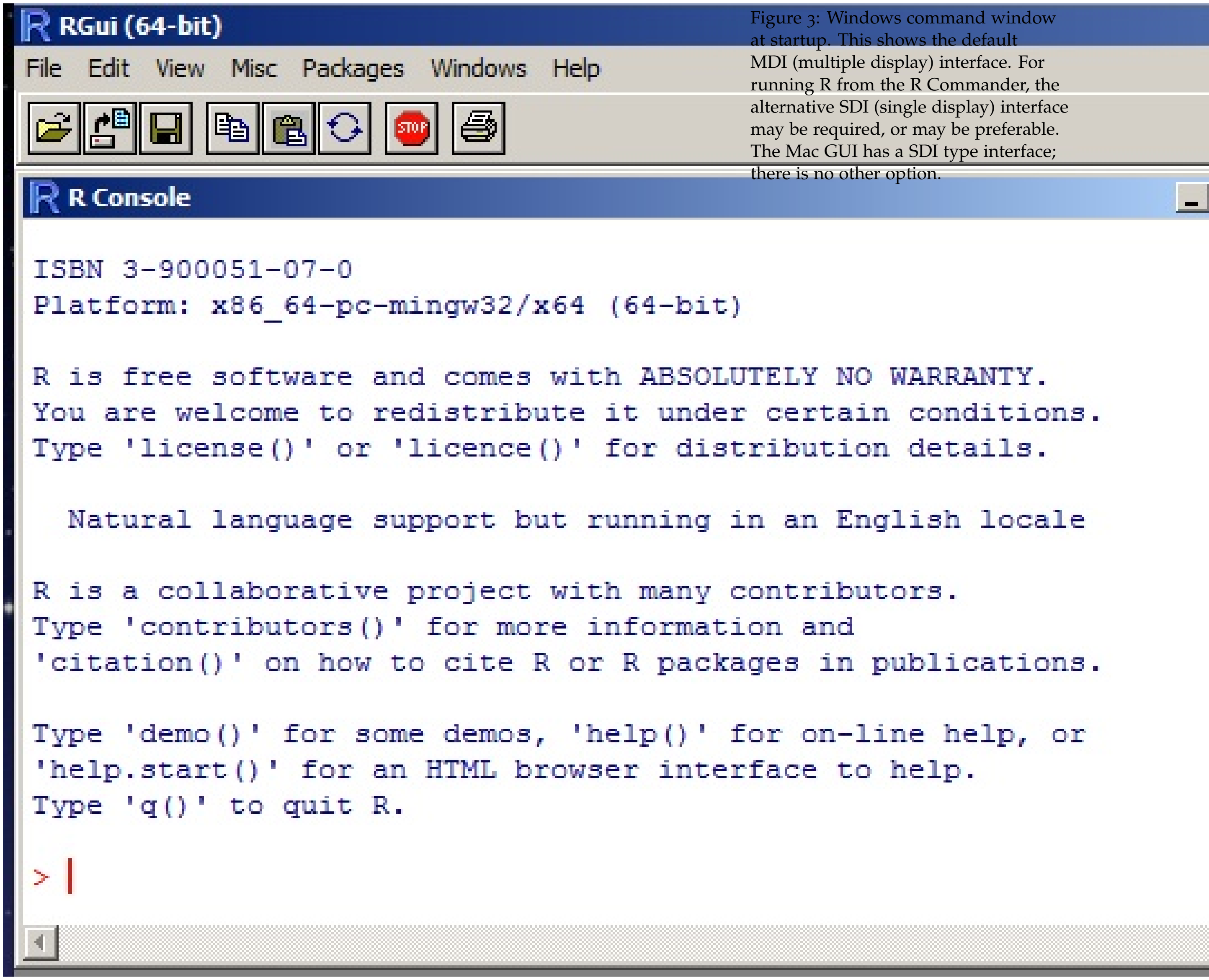
The object result was added to a previously empty workspace.

Figure 1.4 shows, with annotations, the screen as it appears following the above sequence of commands.

An R session is structured as a hierarchy of databases. Functions that were used or referred to above — such as ls() – are from a database or *package* that is part of the R system. Objects that the user has created or input, or that were there at the start of the session and not later removed, are stored in the *workspace*.

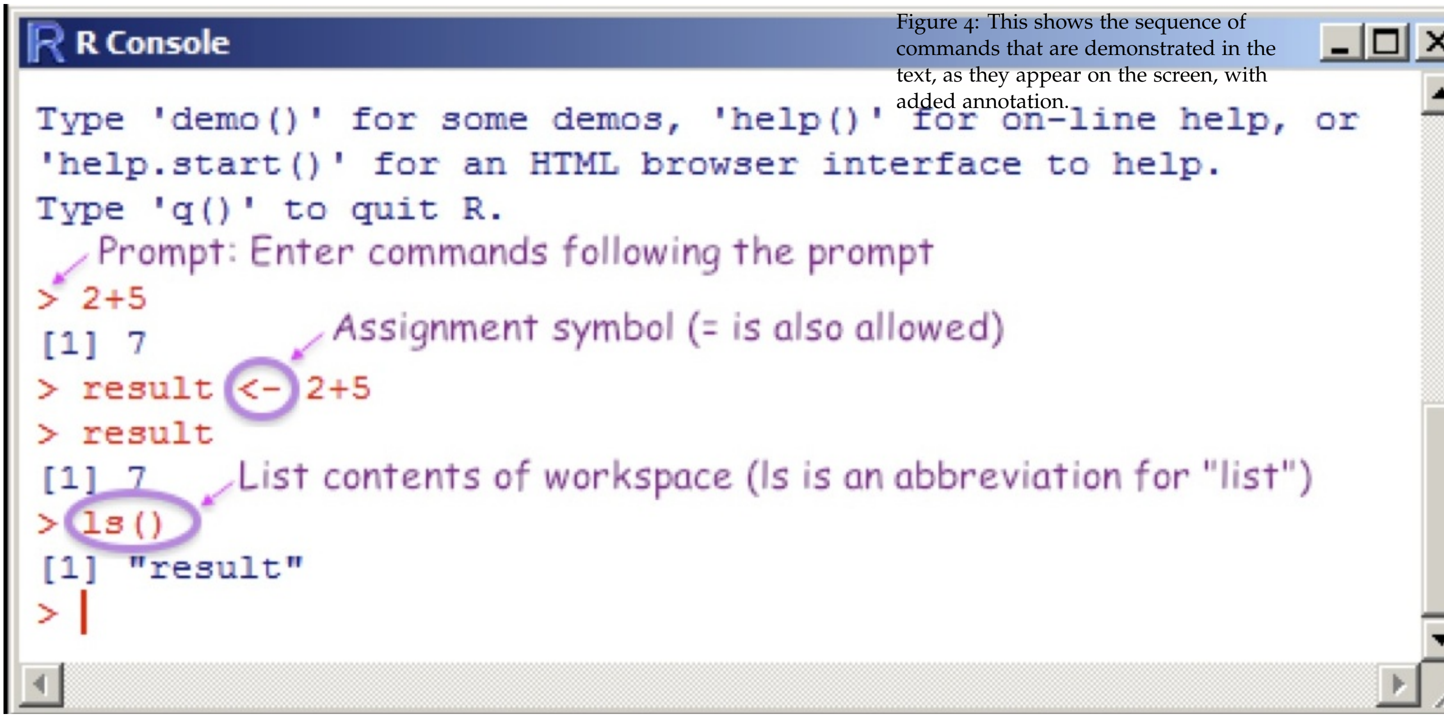
The workspace is the user’s database for the duration of a session. It is a volatile database, i.e., it will disappear if not explicitly saved prior to or at the end of the session.

Technically, the R system refers to the workspace as .Globalenv.



```
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Figure 3: Windows command window at startup. This shows the default MDI (multiple display) interface. For running R from the R Commander, the alternative SDI (single display) interface may be required, or may be preferable. The Mac GUI has a SDI type interface; there is no other option.



```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> 2+5
[1] 7
> result <- 2+5
> result
[1] 7
> ls()
[1] "result"
> 
```

Prompt: Enter commands following the prompt
Assignment symbol (= is also allowed)
List contents of workspace (ls is an abbreviation for "list")

Figure 4: This shows the sequence of commands that are demonstrated in the text, as they appear on the screen, with added annotation.

Points to note

Printing	Typing the name of an object (and pressing Enter) displays (prints) its contents.
Quitting	To quit, type q(), (not q)
Case matters	volume is different from Volume

Typing the name of an object (and pressing the Enter key) causes the printing of its contents, as above when result was typed. This applies to functions also. Thus type q() in order to quit, not q.[1] One types q() because this causes the function q to spring into action.

Upon typing q() and pressing the Enter key, a message will ask whether to save the workspace image.[2] Clicking Yes (usually the safest option) will save the objects that remain in the workspace – any that were there at the start of the session (unless removed or overwritten) and any that have been added since. The workspace that has been thus saved is automatically reloaded when an R session is restarted in the working directory to which it was saved.

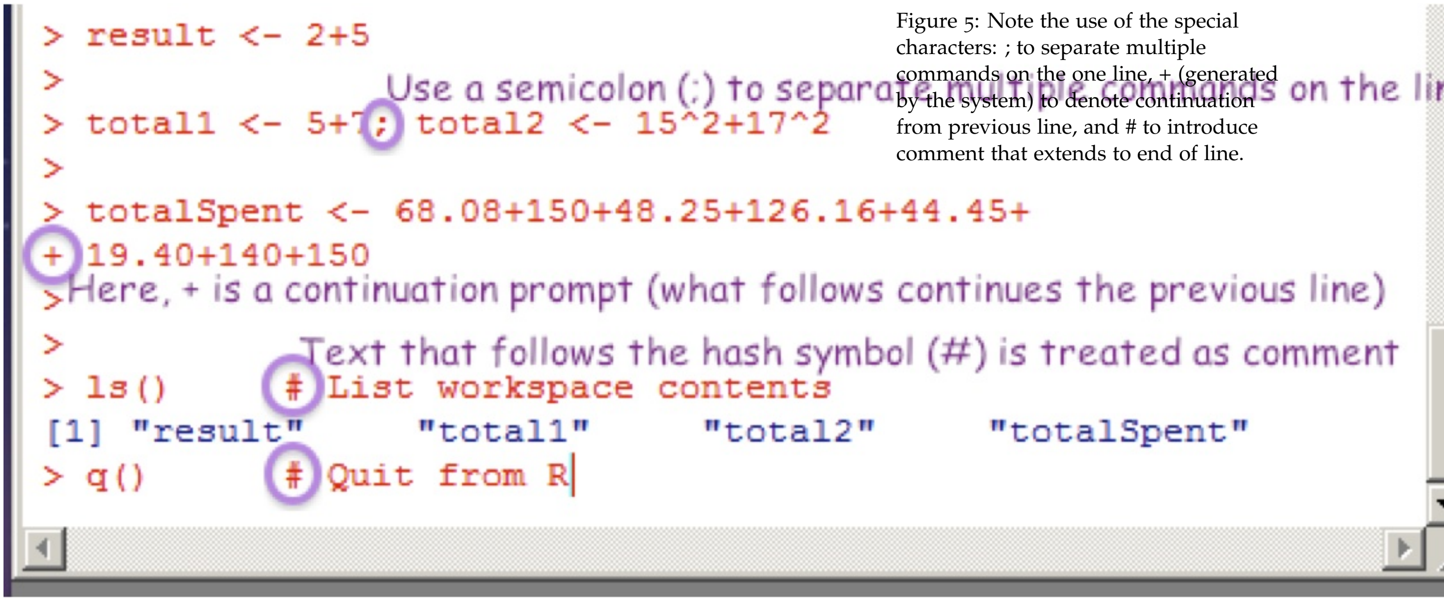
Note that for names of R objects or commands, case is significant. Thus Myr (millions of years, perhaps) differs from myr. For file names,[3] the operating system conventions apply.

Commands may, as demonstrated in Figure 1.5, continue over

[1] Typing q lists the code for the function.

[2] Such an *image* allows reconstruction of the workspace of which it forms an image!

[3] Under Windows case is ignored. For Unix case does distinguish. (Mac OS X Unix is a partial exception.)



```
> result <- 2+5
>
> total1 <- 5+7; total2 <- 15^2+17^2
>
> totalSpent <- 68.08+150+48.25+126.16+44.45+
+ 19.40+140+150
>
>
> ls()        # List workspace contents
[1] "result"      "total1"      "total2"      "totalSpent"
> q()         # Quit from R
```

Use a semicolon (;) to separate multiple commands on the li

Here, + is a continuation prompt (what follows continues the previous line)

Text that follows the hash symbol (#) is treated as comment

Figure 5: Note the use of the special characters: ; to separate multiple commands on the one line, + (generated by the system) to denote continuation from previous line, and # to introduce comment that extends to end of line.

more than one line. By default, the continuation prompt is +. As with the > prompt, this is generated by R, and appears on the left margin. Including it when code is entered will give an error!

Here is a command that extends over two lines:

```
> result <-
+ 2+5
```

Some further comments on functions in R

R is a functional language. Whenever a command is entered, this causes a function to run. Addition is implemented as a function, as are other such operations.

Common functions that users should quickly get to know include print(), plot() and help(). Above, we noted the function q(), used to quit from an R session.

Consider the function print(). One can explicitly invoke it to print the number 2 thus:

Objects on which the function will act are placed inside the round brackets. Such quantities are known as *arguments* to the function.

An alternative to typing print(2) is to type 2 on the command line. The function print() is then invoked implicitly:

Help information

Included on the information that appeared on the screen when R started up, and shown in Figures 1.4 and 1.5, were brief details on how to access R's built-in help information:

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
```

The shorthand ?plot is an alternative to typing help(plot).

Replace '?' by '??' for a wider search. This invokes the function help.search(), which looks for a partial match in the title or concept fields as well as in the name.

R has extensive built-in help information. Be sure to check it out as necessary. Section 1.8 has further details on what is available, beyond what you can get by using the help function.

Examples of use of ??:

The working directory

Associated with each session is a working directory where R will by default look for files. In particular:

- If a command inputs data from a file into the workspace and the path is not specified, this is where R will look for the file.
- If a command outputs results to a file, and the path is not specified, this is where R will place the file.
- Upon quitting a session, the "off-the-shelf" setup will ask whether to save an "image" of the session. Answering "Yes" has the result that the contents of the workspace are saved into a file, in the working directory, that has the name **.RData**.

Under Windows, if R is started by clicking on an R icon, the working directory is that specified in the Start in directory specified in the icon Preferences. Subsection **??** has details on how to specify the Start in directory for an icon.

When R finds a **.RData** file in the working directory at startup, that file will, in an off-the-shelf setup, be used to restore the workspace.

For regular day to day use of R, it is advisable to have a separate working directory for each different project. RStudio users will be asked to specify a working directory when setting up a new "project".

Installation of R Packages

Installation of R Packages (Windows & MacOS X)

Start R (e.g., click on the R icon). Then use the relevant menu item to install packages via an internet connection. This is (usually) easier than downloading, then installing.

For command line instructions to install packages, see below.

A fresh install of R packages is typically required when moving to a new major release (e.g., from a 3.0 series release to a 3.1 series release).

The functions that R provides are organised into packages. The packages that need to be installed, additional to those that come with the initial ready-to-run system, will vary depending on individual user requirements. The GUIs — MacOS X, Windows or Linux — make package installation relatively straightforward.

Installation of packages from the command line

To install the R Commander from the command line, enter:

The R Commander has a number of dependencies, i.e., packages that need to be installed for the R Commander to run. Graphics packages that are dependencies include *rgl* (3D dynamic graphics),

By default, a CRAN mirror is searched for the required package. Refer back to the introduction for brief comments on CRAN. Subsection **??** gives details of alternatives to CRAN. Note in particular the Bioconductor repository.

scatterplot3d, *vcd* (visualization of categorical data) and *colorspace* (generation of color palettes, etc).

Installation of Bioconductor packages

To set your system up for use of Bioconductor packages, type:

Additional packages can be installed thus:

See further `http://www.bioconductor.org/install/`.

For installation of Bioconductor packages from the GUI, see Subsection **??**.

Practice with R commands

Read c as “concatenate”, or perhaps as “column”.

Lists are widely used in R. A data frame is a special type of list, used to collect together column objects under one name.

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)
```

Data frame

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)
 )
```

Also: Arithmetic operations; simple plots; input of data.

Try the following

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

It turns out, surprisingly often, that logarithmic scales are appropriate 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 >:

The R language has the standard abilities for evaluating arithmetic and logical expressions. There are numerous functions that extend these basic arithmetic and logical abilities.

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.

Other relational operators are

< >= < <= == ! =

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:

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:

Images and perspective plots:

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

A Short R Session

We will work with the data set shown in Table 1.1:

	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 1: Weights and volumes, for six Australian travel books.

Entry of columns of data from the command line

The following enters data as numeric vectors:

Now store details of the books in the character vector description:

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

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

Listing the workspace contents

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

Use the argument pattern to specify a search pattern:

Operations with numeric vectors

Note also:

Here are the values of volume

To extract the final element of volume, do:

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

A note on functions

For the weight/volume calculation, two decimal places in the output is more than adequate accuracy. The following uses the function round() to round to two decimal places:

More simply, type:

Providing the arguments are in the defined order, they can as here be omitted from the function call.

Functions take *arguments* — these supply data on which they operate. For round() the arguments are 'x' which is the quantity that is to be rounded, and 'digits' which is the number of decimal places that should remain after rounding.

Use the function args() to get details of the named arguments:

Many functions, among them plot() that is used for Figure 1.6, accept unnamed as well as named arguments. The symbol '. . . ' is used to denote the possibility of unnamed arguments.

Tabulation

Use the function `table()` for simple numeric tabulations, thus:

If a '. . . ' appears, indicating that there can be unnamed arguments, check the help page for details.

A simple plot

Figure 1.6 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 1.6 is:

Figure 6: Weight versus volume, for six Australian travel books.

The axes can be labeled:

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

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

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" that appears at the top left of the screen, just under "Rgui" on the left of the blue panel header of the R window. Then click on "Stop locator".

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, ...

Data frames – Grouping columns of data

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

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"]

The following code groups the several columns of Table 1.1 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.

The vectors weight, volume and description were entered earlier, and (unless subsequently removed) can be copied directly into the data frame.

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.

The storage of character data as factors

Vectors of character, such as type, are by default stored in the data frame as *factors*. In the data as stored, "Guide" is 1 and "Roadmaps" is 2. Stored with the factor is an attribute table that interprets 1 as "Guide" and 2 as "Roadmaps".

While in most contexts 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:

For a matrix or array, users are restricted to the first and second of these alternatives. With a matrix travelmat use, e.g., travelmat[,4] or travelmat[,"weight"].

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*

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

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

Both these mechanisms look first for a data frame column with a needed name. The workspace is searched only if this fails.

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.

Subsection **??** will discuss the attaching of packages and image files.

Attachment of a data frame:

Input of Data from a File

The function read.table() is designed for input from a rectangular file into a data frame. There are several variants on this function — notably read.csv() and read.delim().

First use the function datafile() (*DAAG*) to copy from the *DAAG* package and into the working directory a data file that will be used for demonstration purposes.

This use of datafile(), avoiding use of the mouse to copy the file and the associated need to navigate the file system, is a convenience for teaching purposes.

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

The first two lines hold the column headings and first row, thus:

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

Observe that column 1, which has the row names, has no name.

The following reads the file into an R data frame:

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

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.

Alternatives to command line input include the R Commmander menu and the RStudio menu. These make it easy to check that data are being correctly entered.

If the first row of input gives column names, specify heading=TRUE. If the first row of input is the first row of data, specify heading=FALSE.

See help(read.table) for details of parameter settings that may need changing to match the data format.

Section **??** discusses common types of input errors.

Character vectors that are included as columns in data frames become, by default, factors.
Data input – points to note:

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

Sources of Help

This section enlarges on the very brief details in Subsection 1.2.3

Note also:
help.search()
apropos()
help.start()
RSiteSearch()

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.

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.

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

Searching for key words or strings

Use help.search() to look for functions that include a specific word or part of word in their alias or title. Thus, functions for operating on character strings are likely to have "str" or "char" in their name. Try

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

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

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.

Vignettes

Many packages have vignettes; these are typically pdf or (with version $\geq$ 3.0.0 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 in a package, call browseVignettes() with the package name

Vignettes are created from a Markdown or HTML or LaTeX document in which R code is embedded, surrounded by markup that controls what is to be done with the code and with any output generated. See Section **??**.

(as a character string) as argument. Thus, 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.

Searching for Packages

A good first place to look, for information on packages that relate to one or other area of knowledge, is the R Task Views web page, at: `http://cran.r-project.org/web/views/`. See also the website `http://crantastic.org/`, which has details on what packages are popular, and what users think of them.

Summary and Exercises

Summary

One use of R is as a calculator, to evaluate arithmetic expressions. Calculations can be carried out in parallel, across all elements of a vector at once.

The R Commander GUI can be helpful in getting quickly into use of R for many standard purposes. It may, depending on requirements, be limiting for serious use of R.

Use q() to quit from an R session. To retain objects in the workspace, accept the offer to save the workspace.

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).

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

The function help.start() starts a browser window from which R help information can be accessed.

Use the GUI interface in RStudio or R Commander to input rectangular files. Or, use read.table() or one of its aliases.

Aliases of read.table() include read.csv() and read.delim()

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().

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

For simple forms of scatterplot, use plot() and associated functions, or perhaps the *lattice* function xyplot().

Exercises

1. Use the following code to to place the file `bestTimes.txt` in the working directory:

   (a) Examine the file, perhaps using the function file.show(). Read the file into the workspace, with the name bestTimes.

   (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

   (c) Here are alternative ways to plot the data

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

Subsection **??** discusses the use of the function save().

2. Re-enter the data frame travelbooks.[4] Add a column that has the density (weight/volume) of each book.

3. The functions summary() and str() both give summary information on the columns of a data frames Comment on the differences in the information provided, when applied to the following data frames from the *DAAG* package:

   (a) nihills;

   (b) tomato.

4. Examine the results from entering:

   (a) ?minimum

   (b) ??minimum

   (c) ??base::minimum

   (d) ??base::min

   For finding a function to calculate the minimum of a numeric vector, which of the above gives the most useful information?

The notation base::minimum tells the help function to look in R's base package.

5. For each of the following tasks, applied to a numeric vector (numeric column object), find a suitable function. Test each of the functions that you find on the vector volume in Section 1.5:

   (a) Reverse the order of the elements in a column object;

   (b) Calculate length, mean, median, minimum maximum, range;

   (c) Find the differences between successive values.

[4] If necessary, refer back to Section 1.6 for details.