4

The R Working Environment

Object	Objects can be data objects, function objects, formula objects, expression objects, . . . Use `ls()` to list contents of current workspace.
Workspace	User's "database", where the user can make additions or changes or deletions.
Working directory	Default directory for reading or writing files. Use a new working directories for a new project.
Image files	Use to store R objects, e.g., workspace contents. (The expected file extension is **.RData** or **.rda**)
Search list	`search()` lists 'databases' that R will search. `library()` adds packages to the search list

Important R technical terms include *object*, workspace*, working directory*, *image file*, *package*, *library*, *database* and *search list*.

Use the relevant menu. or enter `save.image()` on the command line, to store or back up workspace contents. During a long R session, do frequent saves!

4.1 The Working Directory and the Workspace

Each R session has a *working directory* and a workspace. By default R looks in the *working directory* for files, and saves files that are output to it.

The workspace is, in R technical language, a "database" that holds all the objects that are under direct user control. It holds objects that the user has created or input, or that were there at the start of the session and not later removed.

The workspace is at the base of a list of "databases", called the search list, that controls access to packages, objects in other directories, etc.

The workspace changes as objects are added or deleted or modified. It disappears at the end of the session, but a copy or "image" can and usually should be kept. Upon quitting from R (type `q()`, or use the relevant menu item), users are asked whether they wish to save the current workspace. It will be saved, with the name **.RData**, into the current working directory. When an R session is next started in that working directory, R looks for a workspace **.RData**, and if found reloads it.

Setting the Working Directory

When a session is started by clicking on a Windows icon, the icon's Properties specify the Start In directory. The default choice, usually an R installation directory, is not satisfactory for long-term use, and should be changed.[1]

[1] When a Unix or Linux command starts a session, the default is to use the current directory.

It is good practice to use a separate working directory, and associated workspace or workspaces, for each different project. On Windows systems, copy an existing R icon, rename it as desired, and change the Start In directory to the new working directory. The working directory can be changed[2] once a session has started, either from the menu (if available) or from the command line. Changing the working directory from within a session requires a clear head; it is usually best to save one's work, quit, and start a new session.

[2] To make a complete change to a new workspace, first save the existing workspace, and type `rm(list=ls(all=TRUE)` to empty its contents. Then change the working directory and load the new workspace.

4.2 Work and Data Maintenance

4.2.1 Maintenance of R scripts

Note the importance of maintaining a transcript from which work done during the session, including data input and manipulation, can as necessary be reproduced. Where calculations are quickly completed, this can be re-executed each time a new session is started, to get to the point where the previous session left off.

Note again RStudio's abilities for managing and keeping R scripts.

4.2.2 Saving and retrieving R objects

Where computations are time-consuming, users will be advised to save (back up) the current workspace image from time to time. The command `save.image()`) saves everything in the workspace, by default into a file named **.RData** in the working directory. Or, depending on the implementation, click on the relevant menu item.

Before making major changes in the workspace, it may be sensible to archive the contents of the current workspace, e.g., into a file with the name **archive.RData**. Specify

In place of **archive**, it might be better to use, e.g.,the date when the file was created, e.g.:

```
fnam ← "2014Feb1.RData"
save.image(file=fnam)
```

Before saving the workspace, perhaps on quitting the session, consider use of `rm()` to remove objects that are no longer required. Saving the workspace image will then save everything that remains.

Use `save()` to save one or more named objects into an image file. The following demonstrate the explicit use of `save()` and `load()` commands:

The function `save.image()` calls `save()`, in order to do its task.

```
volume ← c(351, 955, 662, 1203, 557, 460)
weight ← c(250, 840, 550, 1360, 640, 420)
save(volume, weight, file="books.RData")
  # Can save many objects in the same file
rm(volume, weight)       # Remove volume and weight
```

```
load("books.RData")      # Recover the saved objects
```

An alternative to `load("books.RData")` is `attach("books.RData")`. This makes the objects available, but in a *database* that is separate from the user's workspace.

Two further possibilities are:

- Use `dump()` to save one or more objects in a text format. For example:

```
volume ← c(351, 955, 662, 1203, 557, 460)
weight ← c(250, 840, 550, 1360, 640, 420)
dump(c("volume", "weight"), file="volwt.R")
rm(volume, weight)
source("volwt.R")        # Retrieve volume & weight
```

- Use txttwrite.table() to write a data frame to a text file.

4.3 Packages and System Setup

For download or installation of R or CRAN packages, use for preference a local mirror. In Australia `http://cran.csiro.au` is a good choice. The mirror can be set from the Windows or Mac GUI. Alternatively (on any system), type `chooseCRANmirror()` and choose from the list that pops up.

Packages	Packages are structured collections of R functions and/or data and/or other objects.
Installation of packages	Most users will install R from CRAN binaries. Binaries include 'recommended' packages. Install other packages, as required,
`library()`	Use to attach a package, e.g., `library(DAAG)` Once attached, a package is added to the list of "databases" that R searches for objects.
`attach()`	Attach data frames or image files.

An R installation is structured as a library of packages.

- All installations should have the base packages (one of them is called *base*). These provide the infrastructure for other packages.
- Binaries that are available from CRAN sites include, also, all the recommended packages.
- Other packages can be installed as required, from a CRAN mirror site, or from another repository.

A number of packages are by default attached at the start of a session. To attach other packages, use `library()` as required.

To discover which packages are attached, enter one of:

```
search()
sessionInfo()
```

4.3.1 Installation of R packages

Section 1.3 described the installation of packages from the internet. Note also the use of `update.packages()` or its equivalent from the

GUI menu. This identifies packages for which updates are available, offering the user the option to proceed with the update.

The function `download.packages()` allows the downloading of packages for later installation. The menu, or `install.packages()`, can then be used to install the packages from the local directory. For command line installation of packages that are in a local directory, call `install.packages()` with `pkgs` giving the files (with path, if necessary), and with the argument `repos=NULL`.

Arguments are a vector of package names and a destination directory `destdir` where the latest file versions will be saved as **.zip** or (MacOS X) **.tar.gz** files.

If for example the binary **DAAG_1.11.zip** has been downloaded to **D:\tmp\**, it can be installed thus

```
install.packages(pkgs="D:/DAAG_1.11.zip",
                 repos=NULL)
```

On Unix and Linux systems, gzipped tar files can be installed using the shell command:

R CMD INSTALL xx.tar.gz

(replace xx.tar.gz by the file name.)

On the R command line, be sure to replace the usual Windows backslashes by forward slashes.

Use `.path.package()` to get the path of a currently attached package (by default for all attached packages).

4.3.2 The search path: library() and attach()

The R system maintains a *search path* (or list) that determines, at any time in a session, where and in what order to look for objects. The *search path* determinines the sequence of *databases* where R looks for objects (functions or data) that may be required.

To get a snapshot of the search path, here taken after starting up and entering `library(MASS)`, type:

Packages other than *MASS* were attached at startup.

```
search()
```

```
 [1] ".GlobalEnv"        "package:MASS"
 [3] "tools:RGUI"        "package:stats"
 [5] "package:graphics"  "package:grDevices"
 [7] "package:utils"     "package:datasets"
 [9] "package:methods"   "Autoloads"
[11] "package:base"
```

If the process runs from RStudio, `"tools:rstudio"` will appear in place of `"tools:RGUI"`. Technically, these are "databases". Database 1, where R looks first, is the user workspace, called `".GlobalEnv"`. If the object is not in database 1, it looks in database 2, and so on.

For more detailed information that has version numbers of any packages that are additional to base packages, type:

```
sessionInfo()
```

Attachment of image files

Section 2.3 described the attaching of a data frame (or list object).

It is also possible to attach an R image file, thus:

Objects that are attached, whether data frames or workspaces or packages (using `library()`), are added to the search list.

```
attach("books.RData")
```

The session then has access to objects in the file **books.RData**.

Technically, the file becomes a further "database" on the search list, separate from the workspace.

Note that if the object is modified, the modified copy becomes part of the workspace.

In order to detach such a database, proceed thus:

```
detach("file:books.RData")
```

Alternatively type `search()`, note the number that gives the position of the database on the search list, and supply that number as an argument to `detach()`.

4.3.3 **Where does the R system keep its files?*

Type `R.home()` to see where the R system has stored its files.

```
R.home()
```

```
[1] "/Library/Frameworks/R.framework/Resources"
```

Note that R expects (and displays) either a single forward slash or double backslashes, where Windows would show a single backslash.

Notice that the path appears in abbreviated form. Type `normalizePath(R.home())` to get the more intelligible result

```
[1] "C:\\Program Files\\R\\R-2.15.2"
```

By default, the command `system.file()` gives the path to the base package. For other packages, type, e.g.

```
system.file(package="DAAG")
```

```
[1] "/Library/Frameworks/R.framework/Versions/3.0/Resources/library/DAAG"
```

To get the path to a file **viewtemps.RData** that is stored with the *DAAG* package in the **misc** subdirectory, type:

```
system.file("misc/viewtemps.RData", package="DAAG")
```

4.3.4 *Option Settings*

Type `help(options)` to get full details of option settings. There are a large number. To change to 60 the number of characters that will be printed on the command line, before wrapping, do:

```
options(width=60)
```

To display the setting for the line width (in characters), type:

```
options()$width
```

```
[1] 61
```

The printed result of calculations will, unless the default is changed (as has been done for most of the output in this document) often show more significant digits of output than are useful. The following demonstrates a global (until further notice) change:

```
sqrt(10)
```

```
[1] 3.162
```

Use `signif()` to affect one statement only. For example
`signif(sqrt(10),2)`
NB also the function `round()`.

```
opt <- options(digits=2) # Change until further notice,
                         # or until end of session.
sqrt(10)
```

```
[1] 3.2
```

```
options(opt)             # Return to earlier setting
```

Note that `options(digits=2)` expresses a wish, which R will not always obey!

Rounding will sometimes introduce small inconsistencies!

For example:

```
round(sqrt(85/7), 2)
```

```
[1] 3.48
```

```
round(c(sqrt(85/7)*9, 3.48*9), 2)
```

```
[1] 31.36 31.32
```

4.4 Summary and Exercises

4.4.1 Summary

Each R session has a working directory, where R will by default look for files or store files that are external to R.

User-created R objects are added to the workspace, which is at the base of a search list, i.e., a list of "databases" that R will search when it looks for objects.

It is good practice to keep a separate workspace and associated working directory for each major project. Use script files to keep a record of work.

At the end of a session an image of the workspace will typically (respond "y" when asked) be saved into the working directory.

Before making big changes to the workspace, it may be wise to save the existing workspace under a name (e.g., `Aug27.RData`) different from the default `.RData`.

The search path determines the order of search for objects that are accessed from the command line, or that a function requires and are not in the functions environment.

Note also the use of `attach()` to give access to objects in an image (**.RData** or **.rda**) file. Include the name of the file (optionally preceded by a path) in quotes.

R has an extensive help system. Use it!

4.4.2 Exercises

1. Read the data that is stored in the file **molclock1.txt** into the data frame `molclock`.[3]. Use the function `save()` to save the data into an R image file. Delete the data frame `molclock`, and check that you can recover the data by loading the image file.

[3] With the package `DAAG` attached, typing `datafile()` will store **molclock1.txt**, **molclock2.txt**, and also **travelbooks.txt**, in your working directory