Introduction to the R statistical language

Pierre Legendre
Département de sciences biologiques
Université de Montréal

August 2005; May 2006;
May, July, Nov. 2007; Feb., May 2008; Feb., May 2009;
Jan., Sept. 2011; Jan., Sep. 2012; April 2013; Sep. 2014;

R is a statistical and graphical analysis package, freely available for several platforms (Windows, Mac OS X, Linux). R was created in 1990 by Ross Ihaca and Robert Gentleman at the *University of Auckland*. R is a dialect of the S language developed in 1976 by John Chambers and colleagues at *AT&T Bell Laboratories*. The S language is also available in the commercial implementation S-PLUS, distributed since 1993 by *Insightlful* (formerly *MathSoft*). R became freeware in 1995.

R can be seen as both a programming language and a package of statistical functions. The basic implementation of R contains a great number of statistical and graphical functions to compute means and variances, for example, and draw histograms and the like. Besides that, many researchers have been developing advanced functions that are now available to all users of R. These functions are usually grouped into packages that are freely available on the Internet at the R project site (http://www.r-project.org/), on developers' homepages, or in publications (e.g., electronic supplements of the journal *Ecology*). Given its flexibility and the fact that it is multi-platform and **free**, R has become one of the main statistical packages used by ecologists, statisticians, as well as scientists in many other fields.

1. Installing R on a computer

• Go to http://cran.r-project.org/ on the Internet. The site is hosted by the *University of Technology* (*Technische Universität*) in Vienna. You may choose to go to one of the mirror computers of CRAN.

• Choose a version in the section "*Download and Install R*". Follow the instructions.

(a) For **Windows**-operating machines, click on "Download R for Windows". Choose "base", then click on "Download R 3.1.1 for Windows" to download the executable file "R-3.1.1-win.exe". If you want the graphics windows to be unattached to and independent of the R console (which I recommend), click "Yes" to *Customize the startup options*, then click "SDI (separate windows)" for *Display Mode*.

(b) For **MacOS X**, click on "Download R for (Mac) OS X", then click on "R-3.1.1-maverick.pkg" to install R. Lower in the page, click on the hyperlink "tools" to access another page where you can download the disk-images **gfortran-4.2.3.pkg**. Versions of R are also available for Linux and other Unix platforms.

• Additional packages are found in the subdirectory "Packages" of the "Software" directory in the left-hand margin of the page. These informations may change with time.

Manuals are available in section *Documentation => Manuals => Contributed documentation*, including the guide *R for Beginners* written by Emmanuel Paradis (Université Montpellier II). The internal documentation of R is in English.

2. The R language and its commands

A window, the R console, appears on the computer screen when R is started. Lines of command are typed in that window; they are executed when hitting "Return".

The contents of the R window can be saved to a file, for instance "RConsole.txt". **More often**, R users save their commands and results as the work progresses, by copying and pasting them to a text file. Any portion of that text file can be copied back into the R window, one or several lines at a time, to run the commands again.

Giving a name to an object

Objects used in this short course will be limited to the types "vectors", "matrices", and "data frames" (a kind of matrix that may contain several types of vectors). In R, a number is a vector containing a single element.

To carry out statistical analyses, we will use functions of the R language. Some of these functions compute statistical parameters of data vectors, like the mean or variance, whereas others plot graphs.

Function "c" ("combine") allows the combination of numbers to form a vector. For example, to construct a vector "vec1" containing numbers 1, 12, and 7, the command is the following:

```
> vec1 <- c(1, 12, 7)
```

The ">" at the beginning of the line is R's prompt character. It indicates that R is waiting for a command from the user. Operator "assign", represented by "<" followed by "-", is used to assign a name to an object or the result of a calculation:

```
> vec1 <- c(1, 12, 7)
```

with or without spaces. One can also use "=" for assignment:

```
> vec1= c(1, 12, 7)
```

It is important to distinguish the name assigned to an object from the function used to compute that object. Suppose, for example, that one wants to compute the mean of the values in a vector (object) called "toto"; "toto" is the name assigned to the vector in the current R run. We can compute the mean of the values in object "toto" as follows:

```
> mean.of.these.values <- mean(toto)
```

"mean" is an R function; it computes the mean of the values in object "toto" given as argument to "mean". The result is stored into a new object called "mean.of.these.values". The arguments of a function are written in parentheses; that includes the names of objects used by functions.

Important : "toto" and "mean1" are names arbitrarily chosen by the user. They designate objects in the current R run. On the contrary, "read.table" and "mean" are functions that perform specific tasks. Their names cannot be changed.

Object names **must begin with a letter**; they can contain letters, numbers (0-9), and periods (.), but no space. For object names, R distinguishes capital and small letters; so, x and X can be used to name different objects.

Information on the functions of R

Information on the functions of R can be obtained on-line in two different ways:

> ?name-of-the-function

A window appears on the screen, containing a description of the function. For example, to obtain information about the function “mean”, type:

> ?mean

If the exact name of an R function is not known, type “help.search”. This command searches the R packages in your computer and produces a list of the R functions whose descriptions contain the given expression or term; the packages in which they are found are stated. For example, type:

> help.search("median")

Notes and comments

It is important to add notes and comments to the text file containing the saved commands, in order to remember later the objectives of the calculations or the details of the algorithm. A comment line starts with “#”. Example:

> # Computing a multiple regression

When they are copied back into the R window, the lines or sections of lines beginning with “#” are not executed.

Importing a data file to be analysed by R

One could type the data to be analysed in the R window, but it is more practical to have them written in advance in a text file or a spreadsheet page, whose content is imported into the current R run. The function “read.table” imports data **from a text file** and creates an object of type “data.frame”. (For very large tables, use “scan” instead of “read.table”.) Example:

> toto <- read.table("name-of-the-text-file", header=TRUE, row.names=1)

“toto” is the name given to the **data frame** in the R run; “name-of-the-text-file” is the name of the text file on your computer’s hard disk, including the extension if any (beware: MS Windows does not necessarily show extensions in file lists).

Before importing data, one must tell R where the data file is located on your hard disk. Go to the “File”, “Tools”, or “Misc.” menu (depending on the version of R) and click on "Change Working Directory". Navigate in your file list to find the name of the folder containing the file to be imported. In some versions of R, you will have to get inside the folder to turn it into your active directory.

It is easier to import files that are not in the “Working Directory” by the following command:

> toto <- read.table(file.choose())

That command opens a dialogue box, which allows one to select the file to be read.

Most data files contain object (row) names and column headers. You have to tell R how the file is organized in that respect. In the “read.table” command above, the argument “header” refers to column (variable) names; “header = TRUE” indicates that there are column names in the first row of the data file to be imported; “header = FALSE” indicates that there are no column names in the data file. The argument “row.names” indicates in which column the row (object) names are found, if any; if row names are in column 8, write “row.names=8”.

By default, values in any line are separated by spaces or tab characters and the decimal mark is the period (.). If the values are separated by other markers, for example a $ signs, specify this by sep="$". If the decimal mark is the comma, add an argument indicating that: dec = "," . Many other options covering a wide range of situations are described in the help file:

> ?read.table

The arguments "header" and "row.names" are not required if the file contains no row or column identifiers, or if the file is organized exactly as follows:

• object names are found in column 1;

• there are column headers, **except for column 1 which contains the object names**. A "tab" may be present, or not, before the identifier of the first column. The "read.table" command has been programmed to recognize files that contain one element fewer in row 1 than in all other rows (that line is recognized to contain the column headers), and import them without any further specification.

For example, the following text file:

	Spec1	Spec2	Spec3	Spec4	Spec5	Spec6	Depth(m)	Coral	Sand	Other
Site1	1	0	0	0	0	0	1	0	1	0
Site2	0	0	0	0	0	0	2	0	1	0
Site3	0	1	0	0	0	0	3	0	1	0
Site4	11	4	0	0	8	1	4	0	0	1
Site5	11	5	17	7	0	0	5	1	0	0
Site6	9	6	0	0	6	2	6	0	0	1
Site7	9	7	13	10	0	0	7	1	0	0
Site8	7	8	0	0	4	3	8	0	0	1
Site9	7	9	10	13	0	0	9	1	0	0
Site10	5	10	0	0	2	4	10	0	0	1

will be imported correctly by the command:

> reef <- read.table("fich.txt")

without having to specify "header" and "row.names". Yet, one must always check that the data have been imported correctly into the R window. Type the object name to print the data file:

> reef

Be careful with large files: typing the name of a file will print the whole file to your R window. The alternative command

> head(reef)

only prints the first few lines of a data file (default: the first 6 lines).

Data frames, matrices, or vectors can be edited, and their contents modified, by the command:

> fix(name-of-the-R-object) # for example: fix(reef)

One can save an R object (e.g., "toto") on disk as a text file with the "write.table" command. Example:

> write.table(toto, file="file.name.txt")

3. R packages

R allows users to perform calculations using pre-programmed functions. These functions are available in packages. Some packages come with the basic R installation; others can be downloaded from the Internet. The command:

```
> .Library
```

prints the access path to the packages already loaded into your computer.

Before using a function from a package present in the computer, that package must be attached to the current R run. Go to the **Packages** menu and click on **Load package…** (Windows) or **Package Manager** (OS X) to activate one or several packages. To load a single package, one can type:

```
> library(name-of-the-package)      # for example:      library(MASS), or
> require(name-of-the-package)      # for example:      require(MASS)
```

In that example, "MASS" is the name of an R package written by Venables and Ripley, corresponding to the contents of their book *Modern Applied Statistics with S*. This package, which was first written for the S language, has been translated to R by Prof. Brian Ripley.

As mentioned above, one can learn about packages containing pre-programmed functions of one's interest and available on the hard disk by typing "help.search". For example:

```
> help.search("RDA")
```

will inform users that package "vegan" contains a function for canonical redundancy analysis (RDA).

Additional packages like "vegan" can be downloaded from the Web. More than 3000 packages are presently available on the CRAN site. Other packages are found on researchers' Web pages or in appendices of scientific papers.

• Windows clients: go to the **Packages** menu => **Install package(s)**.

• MacOS X clients: go to the **Package & Data** menu and click on **Package Installer** => **CRAN (binaries)** => **Get list**. Click on the box "install dependencies" in order to automatically install other necessary packages while installing your R packages.

Choose from the list of available packages.

=> The following packages will be used in the practicals of this course: 'ade4', 'ape', 'cclust', 'cluster', 'geoR', 'labdsv', 'mapdata', 'maps', 'mvpart', 'rgl', 'spam', 'spdep', and 'vegan'.

After you finish installing these packages from the CRAN site, download the 'AEM', 'packfor'[1] and 'PCNM' packages from http://r-forge.r-project.org/R/?group_id=195, and 'rdaTest' from http://numericalecology.com. Do not decompress these files; keep them as .zip (Windows) or .tgz (MacOS X). To install these packages from your hard disk:

• Windows clients: go to the **Packages** menu => **Install package(s) from local zip files**.

• MacOS X clients: go to the **Package & Data** menu and click on **Package Installer**; select **Local Source Package** and click the button **Install**.

---

[1] MacOS X users need to install the Fortran compiler to get 'packfor' to run. Instructions are found in section "**1. Installing R on a computer**" on page 1.

Some R-language functions, found in the "Practicals_in_R" folder available on the course's Web page, will also be used during the course. To load a simple R function:

• Windows clients: go to the **File** menu => **Source R Code...**

• MacOS X clients: go to the **Files** menu => **Source File...**

Your own R functions can be loaded using the same procedure.

4. Statistical analyses in R

Nearly all contemporary statistical procedures have been programmed for R. For example, have a look at the functions available in the "stat" package, a list of which is found in the INDEX file of that package. Computations are run by calling these pre-programmed functions. The help file associated with each function gives examples of its use.

Examples of applications to basic statistics are provided in the first few pages of the file "*Practicals_in_R.pdf*" available in the folder of the short course.

5. Saving results

To save the contents of the R window, go to **File** => **Save** or **Save as…** Save the file in your working directory, not on the desktop. **A more common way** is to copy-and-paste selected results into a text file. In most cases, there is no point in saving files of intermediate results as these can easily be recomputed by running the saved list of R commands.

One can save graphical results of R in several formats, depending on the OS: pdf, metafile, postscript, jpeg, etc. in Windows; pdf in MacOSX. Activate the graphics window by clicking in it, then go to **File** => **Save** or **Save as…** Other formats are available by typed commands. Descriptions of these formats can be obtained by typing:

```
> ?Devices
```

6. Practical advice

Rather than saving the contents of the R window at the end of a session, it is preferable to copy the command lines and relevant results to text files during the course of the work. Use one of the free ASCII (= text) editor such as EditPad Lite <http://www.editpadpro.com/editpadlite.html> and Notepad++ <http://notepad-plus.sourceforge.net/uk/site.htm> for Windows or TextWrangler for MacOS X <http://www.barebones.com/products/textwrangler/download.html>. For Windows machines, the best free ASCII editor is Tinn-R <http://www.sciviews.org/Tinn-R/index.html>; it allows users to write and edit R code and submit it directly to the R console.

Write many comments into your work file. This will allow you to understand, later, what you have done. If your lines of comments begin with "#", you will be able to run all your commands by copying-and-pasting part of your text file, comments included, into the R window. Avoid accents, spaces and commas in the names of objects and variables in data tables: R cannot deal with these characters.

7. Operators of the R language

Table 1. Operators of the R language. Some of them are shown with a space for clarity; write them without space in the R language.

Operator	Function
Arithmetic operators	
+	Addition
-	Subtraction
*	Multiplication (for vectors and matrices, this operator is the Hadamard product, or the product element pair by element pair)
/	Division (for matrices, this operator produces a division element pair by element pair)
% / %	Integer division of two numbers
% %	Modulo (remainder of the integer division of two numbers)
% * %	Scalar product of two matrices
^	Exponent
Comparison operators	
>	Larger than
<	Smaller than
> =	Larger than or equal to
< =	Smaller than or equal to
= =	Equal (logical operator)
! =	Unequal, different (logical operator)
Logical operators	
& *or* &&	Logical “and”
| *or* ||	Logical “or”
!	Logical “not”
Assignment operators	
<- *or* =	Assign a value to an object name