Couse of R language syntax

© Vojtech Janousek 2016–2022

Here we give an overview of the syntax and basic usage of the R language. We describe how to start and stop an R session, obtain help and find documentation. An account of the main object types and their manipulation in direct (i.e. command line) mode is followed by text on arguably the strongest point—graphics. The short course is closed by a section on writing true R programs (batch mode).

The text is largely adopted from Appendix A to the monograph on R modelling by Janoušek et al. (2016) that is, in turn, partly based on the pdf manual *An Introduction to R*, available from the `Help` menu.

Apart from the help system and pdf manuals included with the R distribution, the reader can find additional information in an increasing number of books and web sites.

Importantly, information concerning the current state of the R project, binaries, source codes and documentation are obtainable from the R project website (http://www.r-project.org) and the Comprehensive R Archive Network (CRAN) (http://cran.r-project.org). CRAN, which is mirrored at many servers worldwide, also provides a distribution channel for user-contributed packages that add new functionality to the core of the R system.

References

Adler J (2012) R in a nutshell. O'Reily, Sebastopol

Crawley MJ (2007) The R book. John Wiley & Sons, Chichester

Janoušek V, Moyen JF, Martin H, Erban V, Farrow C (2016) Geochemical Modelling of Igneous Processes – Principles and Recipes in R Language. Bringing the Power of R to a Geochemical Community. Springer-Verlag, Berlin (doi: 10.1007/978-3-662-46792-3). Appendix A to this monograph is freely downloadable from here.

Murrell P (2019) R Graphics. Chapman & Hall/CRC, London

Maindonald J, Braun J (2003) Data analysis and graphics using R. Cambridge University Press, Cambridge

R syntax in a nutshell – Part I

1.1 Basic operations

1.1.1 Starting and terminating the R session

In Windows, double clicking the file 'RGUI.EXE' or the associated shortcut opens the `R Console`, a text window for the entry of commands and display of textual output. The system prints a number of messages, the most important of which is the last line with a prompt, showing that the R environment is awaiting commands. Apart from the R Console, one or more windows for graphical output and help pages may be displayed.

To end an R session, one can invoke the menu item `File|Exit` or type `q()`. Alternatively, one can terminate the session by closing the R Console window.

1.1.2 Seeking help and documentation

The R environment provides help in several forms—including plain text (displayed in the Console) or HTML (viewed using a web browser). A browsable HTML help window can be obtained from the menu `Help|R language (html)` or by `help.start()`.

`help(plot)` gives text help on a function called 'plot'; `?plot` is an equivalent, a shortcut.

`apropos("plotting")` lists all commands related to 'plot', `example(plot)` shows examples of correct usage of 'plot'.

The R language is distributed together with PDF documents, which can be invoked from the menu `Help|Manuals`. At this stage the most appropriate will be 'An Introduction to R' which outlines the basics of the R language.

The R Journal, an open access refereed journal, features short to medium length articles on various R applications, R programming and add-on packages. Moreover, there exist e-mail discussion groups dedicated to R. Further information, including the archive of the e-mail discussion groups, is available on CRAN.

Lastly, there exist numerous blogs dedicated to the R project, most of them accessible from the R-bloggers web page (http://www.r-bloggers.com). A useful starting point to search for a topic is RSeek (http://www.rseek.org).

1.2 Fundamental objects of the R language

1.2.1 Commands

The R environment can be utilized in **direct mode**, typing commands straight into the Console and getting an immediate response. Alternatively, the whole R code (a text file) can be run at once in **batch mode**.

The commands (functions) in R are entered either individually, or are separated by a semicolon(`;`). More complex blocks of statements are enclosed in braces `{}`. Each command is followed by brackets with parameters (or empty, if none are required, i.e.

default values are desired). Typing just the command name returns a listing of the function's definition.

 The R language is case sensitive. Commands are typed in lowercase and the environment distinguishes between lower and upper case letters in variable names. The latter cannot start with a digit and may contain any symbols apart from the hash mark (`#`). It is also important to note that several words are reserved by the system and cannot be used as variable names. For details, see `?Reserved`.

Direct mode

In the simplest case, commands are entered and the result displayed:

In [1]:
```
(15+6)*3
```

63

The numerical values can be assigned to a named variable. Perhaps confusingly, R does not use `=` as an assignment operator. Instead, an arrow is implemented, corresponding to a combination of `<` (less than) and `-` (minus) characters.

In [2]:
```
x <- 5
```

In direct mode, the content of a variable is displayed by typing its name:

In [3]:
```
x
```

5

If an incomplete command is entered, R displays a prompt "+" giving the opportunity to add what is needed.

```
> (15+6
+ )*3
```

Batch mode

A sequence of R commands or program can also be prepared beforehand in the form of plain text (ASCII) file. This can be written by any editor (e.g., Notepad). Often, the file may need to be exported as a Plain/ASCII text. However, the most useful are text editors designed for programmers as they often provide extra functionality such as line numbering, syntax checking/highlighting and brackets matching (WinEdt, Tinn-R, Notepad++, etc.)

Commonly used suffixes for R program scripts are `r` or `R`. The script can be run from the menu `File|Source R code` or using the command `source`:

```
source("myprogram.r")
```

The program can be stopped by pressing the `Esc` key or from the menu `Misc|Stop current computation`. Optional comments start by a hash mark:

```
# My comment
```

Note that in batch mode the content of a variable must be displayed using the commands `print` or `cat`.

1.2.2 Handling objects in memory

R stores data using a variety of object types including vectors, arrays, matrices (two-dimensional arrays), factors, data frames, lists and functions.

Object	Characteristics	Possible modes	Can have several modes?
Vector	one-dimensional collection of elements	numeric, character, complex, logical	No
Factor	each element is set to one of several discrete values, so-called levels (a categorical variable)	numeric, character	No
Array	multidimensional collection of elements	numeric, character, complex, logical	No
Matrix	two-dimensional array	numeric, character, complex, logical	No
Data frame	like a matrix but every column can be of a different mode<	numeric, character, complex, logical	Yes
List	consists of an ordered collection of objects (components) that can be of various modes (recursive, i.e. list can itself consist of lists)	numeric, character, complex, logical, function, expression, formula	Yes
Function	fundamental piece of code, typically dedicated to a single task, with defined input parameters (arguments) and output values	-	-

In order to display the current list of user objects, use the menu `Misc|List objects` or function:

`ls()`
Unnecessary objects can be removed with the function `rm`:

`rm(x,y,junk)`
All user R objects can be deleted using the menu item `Misc|Remove all objects`.

All the objects are stored in memory, and not automatically saved to disc. When quitting the R environment, you are given the opportunity to save the current workspace, i.e. all the objects created during the given session. If you accept, they are written to a file `.RData` in the current directory and reloaded automatically the next time R is started. Different R sessions may therefore be maintained in different directories.

1.3 Numeric vectors

1.3.1 Assignment

Assignment of several items to a vector is done using the combine function `c`:

In [4]:
```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
y <- c(x, 0, x,x,x,x,x)
```

Printing content of an object (`head` gives just a handful of items in the beginning of the

given object, usualy a rather large one).

In [5]:
```
print(y)
head(y)
```

```
 [1] 10.4  5.6  3.1  6.4 21.7  0.0 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4
[16] 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4
[31] 21.7
```

10.4 · 5.6 · 3.1 · 6.4 · 21.7 · 0

1.3.2 Vector arithmetic

For vectors, calculations are made using basic arithmetic operators: `+ - * / ^`. The use of these operators for two vectors of the same length is intuitive. In other cases, the elements of the shorter vector are recycled as often as necessary. For instance, for the x vector defined above, we can calculate:

In [6]:
```
x
x^3
```

10.4 · 5.6 · 3.1 · 6.4 · 21.7

1124.864 · 175.616 · 29.791 · 262.144 · 10218.313

1.3.3 Names

Each vector may have an attribute `names` (the lengths of the vector itself and its names must match!). For instance:

In [7]:
```
x <- c(3,15,27)
names(x) <- c("Opx","Cpx","Pl")
x
```

Opx: 3 **Cpx:** 15 **Pl:** 27

1.3.4 Generating regular sequences

Regular sequences with step 1 or -1 can be generated using the colon operator (`:`). This operator has the highest priority within an expression, higher than the other ones (`+-*/^`).

In [8]:
```
1:9
9:1
```

1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9

9 · 8 · 7 · 6 · 5 · 4 · 3 · 2 · 1

In [9]:
```
1:9*2
1:(9*2)
```

2 · 4 · 6 · 8 · 10 · 12 · 14 · 16 · 18

1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 · 15 · 16 · 17 · 18

seq(from, to, by)

Yields a sequence of numbers with an arbitrary step (`by`):

In [10]:
```
seq(30, 22, -2)
```

30 · 28 · 26 · 24 · 22

rep(x, times)

Repeats the argument x specified number of `times` :

In [11]:
```
x <- c(3,9)
rep(x, 5)
```

3 · 9 · 3 · 9 · 3 · 9 · 3 · 9 · 3 · 9

1.3.5 Functions to manipulate numeric vectors

The R language contains a number of functions. Only those most important for manipulation of numeric vectors are presented in the following Table. Information about others and the whole range of available parameters can be found in the R documentation.

Function	Meaning
`abs(x)`	absolute value
`sqrt(x)`	square root
`log(x)`	natural logarithm
`log10(x)`	common (base 10) logarithm
`log(x,base)`	logarithm of the given base
`exp(x)`	exponential function
`sin(x)` , `cos(x)` , `tan(x)`	trigonometric functions
`min(x)`	minimum
`max(x)`	maximum
`which.min(x)`	index of the minimal element of a vector
`which.max(x)`	index of the maximal element of a vector
`range(x)`	range of elements in `x` ; equals to `c(min(x),max(x))`
`length(x)`	number of elements (= length) of a vector
`rev(x)`	reverses the order of elements in a vector
`sort(x)`	sorts elements of a vector (ascending)
`rev(sort(x))`	sorts elements of a vector (descending)
`round(x,n)`	rounds elements of a vector to `n` decimal places
`sum(x)`	sum of the elements of a vector
`mean(x)`	mean of the elements of a vector
`prod(x)`	product of the elements of a vector

1.4 Character vectors

Character vectors are collections of text strings, i.e. sequences of characters delimited by the double quote symbol, e.g., `"granite"`. They use backslash as the escape sequence for special characters, including `"\t"` – tab and `"\n"` – new line.

paste(x, y,..., sep = "")

Merges two (or more) character vectors, one by one, the elements being separated by string `sep`:

In [12]:
```
paste("Gabbro","olivine and","pyroxene",sep=" with ")
```

'Gabbro with olivine and with pyroxene'

substring(x, first, last)

Extracts a part of vector x starting at position `first` and ending at `last`:

In [13]:
```
x <- c("Plagioclase","Biotite","Muscovite")
substring(x,1,4)
```

'Plag' · 'Biot' · 'Musc'

strsplit(x, split)

Splits the strings in `x` into substrings based on the presence of `split`. Returns a list (see below):

In [14]:
```
x <- c("Plagioclase","K-feldspar")
strsplit(x,"a")
```

1. 'Pl' · 'giocl' · 'se'
2. 'K-feldsp' · 'r'

1.5 Logical vectors

Logical vectors consist of elements that can attain only two logical values: `TRUE` or `FALSE`. These can be abbreviated as `T` and `F`, respectively.

In the R language, the symbol name `F` is reserved as an abbreviation for logical `FALSE`. For this reason, it is not advisable to name any variable `F`. The same applies, of course, to `T`.

1.5.1 Logical operators

The logical vectors are typically produced by comparisons using operators < (smaller than) <= (smaller or equal to) > (greater than) >= (greater or equal to) == (equals to) != (does not equal to). For instance:

In [15]:
```
x <- c(1,12,15,16,13,0)
x > 13
```

FALSE · FALSE · TRUE · TRUE · FALSE · FALSE

The result can be assigned to a vector of the mode *logical*:

In [16]:
```
temp <- x>13
temp
```

FALSE · FALSE · TRUE · TRUE · FALSE · FALSE

Boolean arithmetic combines two or more logical conditions using the logical operators: `and(&)` , `or(|)` , `not(!)` with or without brackets:

In [17]:
```
x <- c(1,12,15,16,13,0)
c1 <- x>10
c2 <- x<15
c1
c2
```

FALSE · TRUE · TRUE · TRUE · TRUE · FALSE

TRUE · TRUE · FALSE · FALSE · TRUE · TRUE

In [18]:
```
c1 & c2 # logical "and"
```

FALSE · TRUE · FALSE · FALSE · TRUE · FALSE

In [19]:
```
c1 | c2 # logical "or"
```

TRUE · TRUE · TRUE · TRUE · TRUE · TRUE

In [20]:
```
!c1 # negation
```

TRUE · FALSE · FALSE · FALSE · FALSE · TRUE

1.5.2 Missing values (NA, NaN)

Within R, missing data are represented by a special value `NA` (not available). Some operations which give no meaningful result in the circumstances will return its special form, `NaN` (not a number). Moreover, division by zero gives +∞ (`Inf`).

In [21]:
```
sqrt(-15)
```

```
Warning message in sqrt(-15):
"NaNs produced"
```

NaN

In [22]:
```
1/0
```

Inf

is.na(x)

Tests for the presence of missing values in each of the elements of the vector `x` :

In [23]:
```
x <- c(5,9,-4,12,-6,-7)
!is.na(sqrt(x))
```

```
Warning message in sqrt(x):
"NaNs produced"
```

TRUE · TRUE · FALSE · TRUE · FALSE · FALSE