List of R Resources

Marcus W. Beck[1] Todd D. O'Brien[2]

[1]ORISE, USEPA NHEERL Gulf Ecology Division
Email: beck.marcus@epa.gov

[2]NOAA/NMFS COPEPOD Project
Email: todd.obrien@noaa.gov

R Resources

All of the course material is available on the website:
http://copepod.org/nerrs-swmp-workshop/

- Pre-workshop toolkit
- Training modules - presentations, scripts, datasets
- SWMP cookbook and supplementary materials

The SWMPr package will continue to be developed - eventually submitted to CRAN

Check the SWMPr GitHub page for ongoing development

Feedback, suggestions, bugs, complaints - email Marcus at beck.marcus@epa.gov

R Resources

The SWMPr help manual:

Package 'SWMPr'

November 10, 2014

Type Package

Title SWMPr package for estuarine monitoring data

Version 0.4.0

Date 2014-11-04

Author Marcus Beck

Maintainer Marcus Beck <mbafs2012@gmail.com>

Description This packages provides functions for retrieving, organizing, and analyzing monitoring data from the National Estuarine Research Reserve System. Data that can be used with this package are collected as part of the System Wide Monitoring Program and maintained online at the Centralized Data Management Office.

BugReports https://github.com/fawda123/SWMPr/issues

License CC0

Imports data.table,plyr,reshape2,XML

Suggests SSOAP

Depends R (>= 3.0.0),ggmap,ggplot2,zoo

R Resources

The R reference card:

R Reference Card

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07
Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

Getting help

Most R functions have online documentation.
`help(topic)` documentation on `topic`
`?topic` id.
`help.search("topic")` search the help system
`apropos("topic")` the names of all objects in the search list matching the regular expression "topic"
`help.start()` start the HTML version of help
`str(a)` display the internal *str*ucture of an R object
`summary(a)` gives a "summary" of `a`, usually a statistical summary but it is *generic* meaning it has different operations for different classes of `a`
`ls()` show objects in the search path; specify `pat="pat"` to search on a pattern
`ls.str()` str() for each variable in the search path
`dir()` show files in the current directory
`methods(a)` shows S3 methods of `a`
`methods(class=class(a))` lists all the methods to handle objects of class `a`

Input and output

`load()` load the datasets written with `save`
`data(x)` loads specified data sets
`library(x)` load add-on packages
`read.table(file)` reads a file in table format and creates a data frame from it; the default separator `sep=""` is any whitespace; use `header=TRUE` to read the first line as a header of column names; use `as.is=TRUE` to prevent character vectors from being converted to factors; use `comment.char=""` to prevent "#" from being interpreted as a comment; use `skip=n` to skip `n` lines before reading data; see the help for options on row naming, NA treatment, and others
`read.csv("filename",header=TRUE)` id. but with defaults set for reading comma-delimited files
`read.delim("filename",header=TRUE)` id. but with defaults set for reading tab-delimited files
`read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE)` read a table of *f*ixed width *f*ormatted data into a 'data.frame'; `widths` is an integer vector, giving the widths of the fixed-width fields
`save(file,...)` saves the specified objects (...) in the XDR platform-independent binary format
`save.image(file)` saves all objects
`cat(..., file="", sep=" ")` prints the arguments after coercing to character; `sep` is the character separator between arguments
`print(a, ...)` prints its arguments; generic, meaning it can have different methods for different objects
`format(x,...)` format an R object for pretty printing
`write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ")` prints `x` after converting to a data frame; if `quote` is `TRUE`, character or factor columns are surrounded by quotes ("); `sep` is the field separator; `eol` is the end-of-line separator; `na` is the string for missing values; use `col.names=NA` to add a blank column header to get the column headers aligned correctly for spreadsheet input
`sink(file)` output to `file`, until `sink()`

Most of the I/O functions have a `file` argument. This can often be a character string naming a file or a connection. `file=""` means the standard input or output. Connections can include files, pipes, zipped files, and R variables.

On windows, the file connection can also be used with `description = "clipboard"`. To read a table copied from Excel, use
`x <- read.delim("clipboard")`
To write a table to the clipboard for Excel, use
`write.table(x,"clipboard",sep="\t",col.names=NA)`
For database interaction, see packages `RODBC`, `DBI`, `RMySQL`, `RPgSQL`, and `ROracle`. See packages `XML`, `hdf5`, `netCDF` for reading other file formats.

Data creation

`c(...)` generic function to combine arguments with the default forming a vector; with `recursive=TRUE` descends through lists combining all elements into one vector
`from:to` generates a sequence; ":" has operator priority; 1:4 + 1 is "2,3,4,5"
`seq(from,to)` generates a sequence `by=` specifies increment; `length=` specifies desired length
`seq(along=x)` generates `1, 2, ..., length(along)`; useful for `for` loops
`rep(x,times)` replicate `x` `times`; use `each=` to repeat "each" element of `x` `each` times; `rep(c(1,2,3),2)` is 1 2 3 1 2 3; `rep(c(1,2,3),each=2)` is 1 1 2 2 3 3
`data.frame(...)` create a data frame of the named or unnamed arguments; `data.frame(v=1:4,ch=c("a","B","c","d"),n=10);` shorter vectors are recycled to the length of the longest
`list(...)` create a list of the named or unnamed arguments; `list(a=c(1,2),b="hi",c=3i);`
`array(x,dim=)` array with data `x`; specify dimensions like `dim=c(3,4,2)`; elements of `x` recycle if `x` is not long enough
`matrix(x,nrow=,ncol=)` matrix; elements of `x` recycle
`factor(x,levels=)` encodes a vector `x` as a factor
`gl(n,k,length=n*k,labels=1:n)` generate levels (factors) by specifying the pattern of their levels; `k` is the number of levels, and `n` is the number of replications
`expand.grid()` a data frame from all combinations of the supplied vectors or factors
`rbind(...)` combine arguments by rows for matrices, data frames, and others
`cbind(...)` id. by columns

Slicing and extracting data

Indexing vectors

`x[n]`	n^{th} element
`x[-n]`	all *but* the n^{th} element
`x[1:n]`	first `n` elements
`x[-(1:n)]`	elements from `n+1` to the end
`x[c(1,4,2)]`	specific elements
`x["name"]`	element named `"name"`
`x[x > 3]`	all elements greater than 3
`x[x > 3 & x < 5]`	all elements between 3 and 5
`x[x %in% c("a","and","the")]`	elements in the given set

Indexing lists

`x[n]`	list with elements `n`
`x[[n]]`	n^{th} element of the list
`x[["name"]]`	element of the list named `"name"`
`x$name`	id.

Indexing matrices

`x[i,j]`	element at row `i`, column `j`
`x[i,]`	row `i`
`x[,j]`	column `j`
`x[,c(1,3)]`	columns 1 and 3
`x["name",]`	row named `"name"`

Indexing data frames (matrix indexing plus the following)

`x[["name"]]`	column named `"name"`
`x$name`	id.

Variable conversion

`as.array(x)`, `as.data.frame(x)`, `as.numeric(x)`, `as.logical(x)`, `as.complex(x)`, `as.character(x)`, `...` convert type; for a complete list, use `methods(as)`

Variable information

`is.na(x)`, `is.null(x)`, `is.array(x)`, `is.data.frame(x)`, `is.numeric(x)`, `is.complex(x)`, `is.character(x)`, `...` test for type; for a complete list, use `methods(is)`
`length(x)` number of elements in `x`
`dim(x)` Retrieve or set the dimension of an object; `dim(x) <- c(3,2)`
`dimnames(x)` Retrieve or set the dimension names of an object
`nrow(x)` number of rows; `NROW(x)` is the same but treats a vector as a one-row matrix
`ncol(x)` and `NCOL(x)` id. for columns
`class(x)` get or set the class of `x`; `class(x) <- "myclass"`
`unclass(x)` remove the class attribute of `x`
`attr(x,which)` get or set the attribute `which` of `x`
`attributes(obj)` get or set the list of attributes of `obj`

Data selection and manipulation

`which.max(x)` returns the index of the greatest element of `x`
`which.min(x)` returns the index of the smallest element of `x`
`rev(x)` reverses the elements of `x`
`sort(x)` sorts the elements of `x` in increasing order; to sort in decreasing order: `rev(sort(x))`
`cut(x,breaks)` divides `x` into intervals (factors); `breaks` is the number of cut intervals or a vector of cut points
`match(x, y)` returns a vector of the same length than `x` with the elements of `x` which are in `y` (`NA` otherwise)
`which(x == a)` returns a vector of the indices of `x` if the comparison operation is true (`TRUE`), in this example the values of `i` for which `x[i] == a` (the argument of this function must be a variable of mode logical)
`choose(n, k)` computes the combinations of k events among n repetitions $= n!/[(n-k)!k!]$
`na.omit(x)` suppresses the observations with missing data (`NA`) (suppresses the corresponding line if `x` is a matrix or a data frame)
`na.fail(x)` returns an error message if `x` contains at least one `NA`

R Resources

The Short R introduction:

A (very) short introduction to R

Paul Torfs & Claudia Brauer

Hydrology and Quantitative Water Management Group

Wageningen University, The Netherlands

3 March 2014

1 Introduction

R is a powerful language and environment for statistical computing and graphics. It is a public domain (a so called "GNU") project which is similar to the commercial S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S, and is much used in as an educational language and research tool.

The main advantages of R are the fact that R is freeware and that there is a lot of help available

`http://www.r-project.org/`

and do the following (assuming you work on a windows computer):

- click `download CRAN` in the left bar
- choose a download site
- choose `Windows` as target operation system
- click `base`
- choose `Download R 3.0.3 for Windows` [1] and choose default answers for all questions

It is also possible to run R and RStudio from a USB stick instead of installing them. This could be useful when you don't have administrator rights on your computer. See our separate note "How to use portable versions of R and RStudio" for help on this topic.

2.2 Install RStudio

After finishing this setup, you should see an "R" icon on you desktop. Clicking on this would start up the standard interface. We recommend, however, to use the RStudio interface. [4] To install RStudio, go to:

`http://www.rstudio.org/`

R Resources

The Short R introduction:

12.3 Error messages

- `No such file or directory` or `Cannot change working directory`

Make sure the working directory and file names are correct.

- `Object 'x' not found`

The variable `x` has not been defined yet. Define `x` or write apostrophes if `x` should be a character string.

- `Argument 'x' is missing without default`

You didn't specify the compulsory argument `x`.

- `+`

R is still busy with something or you forgot closing brackets. Wait, type `}` or `)` or press `ESC`.

- `Unexpected ')' in ")"` or `Unexpected '}' in "}"`

The opposite of the previous. You try to close something which hasn't been opened yet. Add opening brackets.

- `Unexpected 'else' in "else"`

Put the `else` of an if-statement on the same line as the last bracket of the "then"-part: `}else{`.

- `Missing value where TRUE/FALSE needed`

Something goes wrong in the condition-part (`if(x==1)`) of an if-statement. Is `x` `NA`?

- `The condition has length > 1 and only the first element will be used`

In the condition-part (`if(x==1)`) of an if-statement, a vector is compared with a scalar. Is `x` a vector? Did you mean `x[i]`?

- `Non-numeric argument to binary operator`

You are trying to do computations with something which is not a number. Use `class(...)` to find out what went wrong or use `as.numeric(...)` to transform the variable to a number.

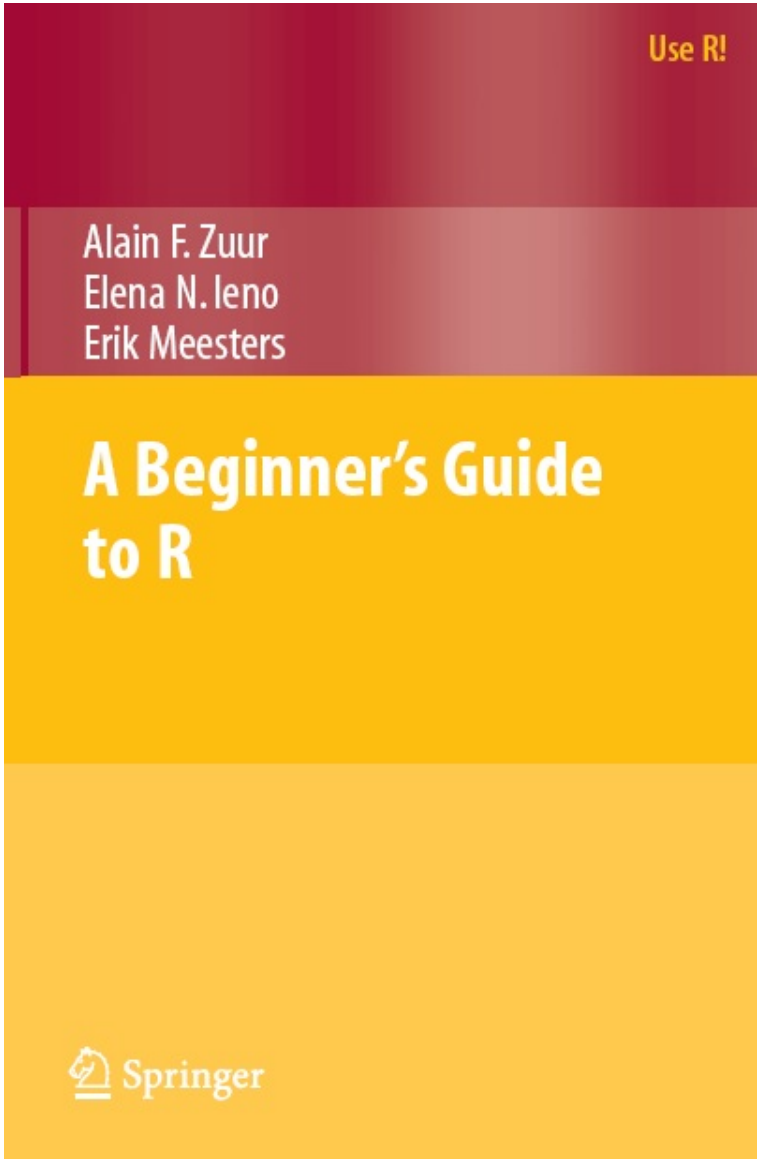
- `Argument is of length zero` or `Replacement is of length zero`

The variable in question is `NULL`, which means that it is empty, for example created by `c()`. Check the definition of the variable.

R Resources

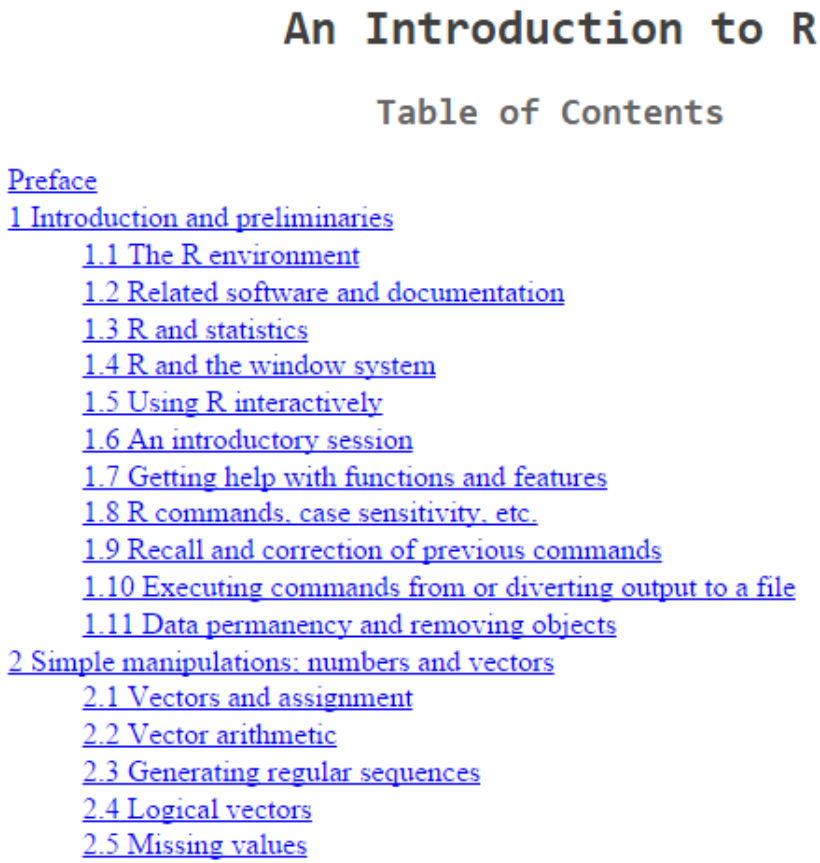
One of several usesful R texts:

- Getting data into R
- Accessing variables and subsets
- Simple functions
- Loops
- Graphing
- Introduction to the Lattice package
- Common mistakes



R Resources

The CRAN R tutorial (detailed):



An Introduction to R

Table of Contents

Preface
1 Introduction and preliminaries
- 1.1 The R environment
- 1.2 Related software and documentation
- 1.3 R and statistics
- 1.4 R and the window system
- 1.5 Using R interactively
- 1.6 An introductory session
- 1.7 Getting help with functions and features
- 1.8 R commands, case sensitivity, etc.
- 1.9 Recall and correction of previous commands
- 1.10 Executing commands from or diverting output to a file
- 1.11 Data permanency and removing objects

2 Simple manipulations; numbers and vectors
- 2.1 Vectors and assignment
- 2.2 Vector arithmetic
- 2.3 Generating regular sequences
- 2.4 Logical vectors
- 2.5 Missing values

R Resources

My suggested help workflow:

1. Check the help file for a function - usually the syntax is incorrect.
2. Check online - A Google search of the problem will usually return an answer. Best to use the actual error message as a search term.
3. Ask a real person that knows about R - I try to be as responsive as possible to emails.
4. Post online, e.g., Stack Overflow or R-help, usually only after all other options are exhausted. Make sure you follow posting guidelines.
5. Do not give up!

R Resources

Final comments about learning R:

- R is becoming the de facto statistical analysis program
- R will fundamentally change how you work with data
- You determine the flow of the analysis, not the other way around
- Time spent banging your head on the wall is time spent learning
- Initial time investments will have huge returns - you will become more efficient
- Do not give up!