



R Programming For Ecologists: Meeting 4



R Language Programming Techniques

NCEAS Scientific Computing Team



Four options: Executing R Programs

- **From the R Graphical Interface**

- Use *setwd()* and *source()* commands to load script files

- **From *R Commander* (Rcmdr) or *SciViews* GUIs**

- Use the menu options to select, edit, and run scripts
- SciViews: MS Windows environment only

- **From the DOS Command Line**

- R CMD BATCH *R script file R output file*
- Be sure that the R installation /bin directory is in the DOS path

- **From the UNIX / LINUX Command Line**

- R CMD BATCH as above
- UNIX background processes allow multiple batch job submission....
- ...*Special Topic: using R on Nick Brand's Linux Cluster*
 - *Great for submitting multiple replicants of an R script*
 - *See WIKI entry at help.nceas.ucsb.edu*



A quick review: R Data Objects

- **R operates on entities known as objects**
 - Intrinsic object attributes *mode* and *length*
 - **Mode:** type of data stored in the object
 - **Length:** number of elements stored in the object
 - The foundation object is the **vector**
 - **Collection of elements, all of same *mode*:**
 - numeric, complex, logical, character, raw
- **R object hierarchy**
 - vector -> array -> matrix -> factor -> list -> data frame -> function
 - **Objects inherit attributes from ‘parent’, and add new ones**
 - Example: array is vector with ‘dim’ attribute (row/column order)
 - **Assign using vector’s dim() attribute or use the array method()**
 - **Operate on specific parts of array using an *index array***
 - R Data Frame: A container for data tables
 - **Contain spreadsheets or RDBMS tables**



R: a function-oriented language

- Discussion from: R Core Team, *An Introduction to R*
- R commands consist of *expressions* and *assignments*
 - *expressions* return a single value
 - *assignments* contain expressions.....
 - ..they pass the expression value to a *variable* (data object)
 - Many R expressions consist of *functions*
- ***functions* are containers for groups of expressions**
 - functions in expressions return a single value
 - Other ways functions communicate with calling environment:
 - *argument list* elements
 - *global (scope) variables*
 - *Example R programs.....*
- R processing features implemented as functions..
- ...Users can *extend* R by writing their own functions



Writing your own R functions

■ R programs: Four stages of development

- R *script*: a collection of R commands
 - For prototyping and quick-turnaround calculations
 - Typically calls one or more existing R functions
- R *function*: self-contained programming object
 - Scripts often comprised of several functions
- R *program*: A collection of functions, called from a 'driver' function
 - for solutions you intend to reuse and/or share with others
- R *package*: 'professional grade' R program
 - Rigorously tested and documented to CRAN standards
 - Downloadable as a self-installing and configuring module
 - Contains information on dependencies (on other R modules)

■ Deciding to write a new R program

- The **first** question to ask: Do I *really need* to write this function?
 - Needed functionality may *already* be included in existing R package



Key issues when writing R functions

- **Communicating with the calling function**
 - via the *argument list*
 - via *global variables*
- **Function argument list issues**
 - Correct syntax
 - Arguments in function argument list, calling programs must agree
 - Default argument values
 - The ‘...’ argument: for passing values through to another function
- **Goals for computing resource efficiency**
 - minimize execution time
 - **Utilize intrinsic R features whenever possible**
 - minimize memory ‘footprint’
 - **Avoid replication of large data objects**
 - Global variables



Flow-of-control in R functions

- **Looping**

- *while, repeat, and for* statements

```
while (i < 10)
  <R expression>
for (i in 1:10)
  <R expression>
```

- **break and next statements: skip execution cycle(s)**

- **Branching**

- *if* statement

```
if (expr_1) expr_2 else expr_3
```

- *ifelse()* function

```
ifelse (condition, a ,b)
```

- **Warning: Explicit loops on vectors are inefficient**

- Code that takes 'whole object' view likely to be both clearer and faster in R
- Most R functions and operators efficiently process vectors as an object
 - **avoid element-by-element processing of objects whenever possible**



Sample R functions: basic concepts

- **Key issue in writing functions: information flow between functions**
- **Inter-function communication and object scope**
- **Simple examples to demonstrate concepts**
 - **getSimpleAssignment():** value of a function expression
 - **defaultArgumentDemo():** default function arguments
 - **simpleScope():** local and global objects, function arg passing
 - **demoDotDotArgs():** '...' arguments 'passthru' to an second function
 - **newBinaryOperator():** define a new binary data op with R function
 - **simpleMatrix():** explicit, and intrinsic matrix manipulation
 - **matrixBenchmark():** measure time difference between explicit and intrinsic 'filtering' techniques



Intrinsic R methods: re-organizing data frames

- **R has many intrinsic (built-in) data management tools**
 - Use these **instead** of writing your own functions
 - ***inherited*** methods operate on *any* object of type ***data.frame***
- **subset(): Extract subsets from data frame**
 - Use the *help(subset)* examples
 - Note the chance to demonstrate ‘index vectors’
- **merge(): Database *join* of two data frames**
 - Give brief definition of ‘joins’
- **stack() / unstack(): (un-)concatenate multiple vectors**
- **reshape(): transforms data frame between *long* and *wide* data format**
 - Examples from ***stack()*** and ***reshape()*** help files
 - Jim R’s sample program



Connecting to external data sets

- **Data of interest often stored on disk in arbitrary format**
 - ASCII flat file (e.g., CSV format)
 - Spreadsheet (e.g., MS Excel)
 - Other mathematical and statistical software packages
 - e.g., **SAS and MATLAB**
 - Relational Data Base Management System (RDBMS)
 - e.g., MS Access, Oracle, PostgreSQL
- **Impractical to export data to flat file format..**
- **R extensions read many external formats *directly***
 - ***read.table()*: ASCII flat files e.g., .csv files into R data frames**
 - ***foreign* package: SAS, SPSS, others**
 - **R.Matlab package: connect to MATLAB server, read/write MAT files**
 - ***RpgSQL, RODBC, RmySQL* packages: RDBMS files**
 - ***Rstreams* package: arbitrary-format binary data files**
 - ***XML* package: XML documents**
 - ***Concise working examples to be posted soon on SciComp Web site***



Brief Summary: creating an R Package

- **For R programs to share with the greater R Community**
 - Package: Downloadable, self-installing, includes documentation
 - **Packages must trigger 'dependant' package installation**
 - Must operate on all current R-supported hardware platforms
 - Packages subjected to standard QA and testing checklist
 - **Submitted to CRAN for acceptance**
- **Package creation is significant effort**
 - **See 'Writing R Extensions' PDF document for details**
- **Summary of package creation steps:**
 1. Develop and test the R (and other) code modules
 2. Create the required package structure (files and folders)
 3. Prepare the 'Configure and Cleanup' installation script
 4. Test package using the **R CMD check** and **R CMD build** commands
 5. Submit package to CRAN
 6. Be prepared to support the package **after** distribution



Conclusions

- **R easy to run in both interactive and batch modes**
 - Several options on major computer platforms
- **R has powerful function-based programming framework**
 - Enables development of reusable, extensible code modules
 - **Which require only moderate effort to assemble**
 - ...Including comprehensive distribution PACKAGE capability
 - Requires more effort, but allows all R users to use your methods
 - Select any of the four options (script / function / program / package)
 - **Based on your project scope and future plans for the code**
 - **Be sure you MUST write functions before doing so**
 - The capability that you need may already exist
 - ...as an intrinsic R method
 - ...in an existing R package