Introduction to R

Rakhi Saxena

(Deshbandhu College, University of Delhi)

What is R?

- R refers to two things:
  - R, the programming language, and
  - R, the software that you use to run programs written in R
- R is an **interpreted language** (also called a scripting language)
  - (code doesn't need to be compiled before you run it)
- Supports **object oriented programming**
  - (data and functions are combined inside classes)
- Also supports **functional programming**
  - (functions are *first-class objects*; you treat them like any other variable, and you can call them recursively)
- R is OPEN SOURCE

Installing R

- On a Linux machine
  - it is likely that your package manager will have R available (maybe not the latest version)
- For everyone else
  - go to *http://www.r-project.org*
  - Click the link that says “download R”
  - Choose a mirror close to you, choose a link in the “Download and Install R” pane at the top of the page that’s appropriate to your operating system.
- Shortcut for Windows user
  - setup file at

    http://<CRAN MIRROR>/bin/windows/base/release.htm.

Using R

- R comes with a GUI
  - Comes with a command-line interpreter
  - facilities for displaying plots and help pages
  - and basic text editor

Choosing an IDE

- Better experience of R by using an IDE
  - **Emacs + ESS** (Emacs Speaks Statistics)
    - *http://www.gnu.org/software/emacs/*
    - *http://ess.rproject.org/*
  - **Eclipse/Architect**
    - *http://eclipse.org ; http://www.walware.de/goto/statet*
    - *http://www.openanalytics.eu/downloads/architect*
  - **Rstudio**
    - *http://www.rstudio.org*
  - **Revolution-R**
    - *http://www.revolutionanalytics.com/products/revolution-r.php*
  - **Live-R**
    - *http://live-analytics.com/*

Your First Program

- Open up R GUI, or IDE you've decided to use
- Find the command prompt (in the code editor window)
- Type
  - mean(1:5)
- Hit Enter to run the line of code.

Your First Program

How to Get Help in R

- Help on a function or a dataset that you know the name of, type ? followed by the name of the function.
- To find functions, type two question marks (??) followed by a keyword related to the search.
- Functions **help** and **help.search** do the same things as ? and ??
- **apropos** function finds variables (including functions) that match its input
  - can do fancier matching using regular expressions
- Most functions have examples that you can run to get a better idea of how they work. Use **example** function to run these
- Packages - other oftware that R can use to extend its functionality

Help Demo Program

VECTORS, MATRICES, AND ARRAYS

Vectors

- Colon operator :
  - for creating sequences from one number to another
  - **8.5:4.5 #sequence of numbers from 8.5 down to 4.5**
- c function
  - for concatenating values ,vectors to create longer vectors
  - **c(1, 1:3, c(5, 8), 13) #values concatenated into single vector**
- Vector function
  - creates a vector of a specified type and length
  - Each of the values in the result is zero, FALSE, or an empty string, or whatever the equivalent of “nothing” is
  - **vector("numeric", 5)**
  - wrapper functions exist for each type to save you typing
  - **numeric(5)**

Vector Demo program

Lengths

- All vectors have a ***length***
  - length function returns how many elements
  - nonnegative integer; zero-length vectors are allowed
  - Missing values still count toward the length

- Character vectors
  - length is the number of strings
  - not the number of characters in each string
  - For that, use **nchar**

Length Demo program

Names

- Each element of vector can be given a name.
- Labelling the elements makes code readable.
- Can specify names when a vector is created
  - *name = value*
  - If the name of an element is a valid variable name, it doesn't need to be enclosed in quotes.
  - Can name some elements of a vector and leave others blank
- names function can also be used to retrieve the names of a vector
- If a vector has no element names, then the names function returns NULL

Names Demo program

Indexing Vectors

- Also called *subsetting* or *subscripting* or *slicing*
- Passing a vector of positive numbers
  - returns slice of vector containing elements at those locations.
  - The first position is 1 (not 0, as in some other languages).
- Passing a vector of negative numbers
  - returns the slice of the vector containing the elements everywhere *except* at those locations.
- Passing a logical vector
  - returns the slice of the vector containing the elements where the index is TRUE.
- For named vectors, passing a character vector of names
  - returns the slice of the vector containing the elements with those names.

Indexing Demo program

Vector Recycling and Repetition

- What happens in case of arithmetic on vectors of different lengths?
  - When adding two vectors, R will recycle elements in the shorter vector to match the longer one
- rep function lets us create vector with repeated elements
- rep.int - Like the seq function, rep has a simpler and faster variant, for the most common case
- rep_len - paralleling seq_len, lets us specify the length of the output vector

Recycling Demo program

Matrices and Arrays

Creating Arrays and Matrices

- array function: pass in a vector of values and a vector of dimensions
  - Can also provide names for each dimension
- matrix function: specify the number of rows or the number of columns
  - matrix can also be created using the array function

Matrix Demo program

Rows, Columns, and Dimensions

- dim function
  - returns a vector of integers of the dimensions of the variable
- nrow and ncol
  - return the number of rows and columns
- length function
  - returns the product of each of the dimensions
- reshape a matrix or array by assigning a new dimension with dim
  - Use with caution since it strips dimension names
- nrow, ncol, and dim return NULL when applied to vectors.
- NROW and NCOL functions
  - are counterparts to nrow and ncol that pretend vectors are matrices with a single column (that is, column vectors)

Matrix Demo program

Row, Column, and Dimension Names

- rownames and colnames functions
  - Like vectors have names for the elements

Matrix Demo program

Indexing Arrays

- Four choices for specifying the index
  - positive integers, negative integers, logical values, and element names
- Permissible to specify the indices for different dimensions in different ways
- Indices for each dimension separated by commas

Matrix Demo program

Combining Matrices

- c function converts matrices to vectors before concatenating them
- cbind and rbind functions
  - *bind* matrices together by columns and rows

Matrix Demo program

Array Arithmetic

- Standard arithmetic operators (+, -, *, /) work **element-wise** on matrices and arrays, just they like they do on vectors
- t function transposes matrices
  - but not higher-dimensional arrays, where the concept isn't well defined
- %*% and %o% Operators
  - For inner and outer matrix multiplication
  - Dimension names are taken from the first input, if they exist
- solve function
  - power operator, ^, also works element-wise on matrices
  - so to invert a matrix you cannot simply raise it to the power of minus one

Matrix Demo program

LISTS AND DATA FRAMES

Lists and Data Frames

- Vectors, matrices, and arrays
  - contain elements that are all of the same type.
- Lists and data frames
  - let us combine different types of data in a single variable.

Creating Lists

- list function
  - simply list the contents, with each argument separated by a comma
  - elements can be any variable type - vectors, matrices, even functions
  - can name elements during construction, or afterward using the names function
  - possible for elements of lists to be lists themselves

    List Demo program

Atomic and Recursive Variables

- Lists are considered to be ***recursive*** variables
  - Due to ability to contain other lists within them
- Vectors, matrices, and arrays are ***atomic***
- functions is.recursive and is.atomic let us test variables to see what type they are

  List Demo program

List Dimensions and Arithmetic

- List's length is the number of top-level elements that it contains
- Lists don't have dimensions
  - dim function returns NULL
- nrow, NROW, ncol, NCOL functions
  - work on lists in the same way as on vectors

List Demo program

Indexing Lists

- Can access elements of the list using
  - square brackets []
  - Positive or negative numeric indices
  - element names
  - logical index
- Result of indexing operations is *another list*
- Instead, to access the *contents* of the list elements
  - Double square brackets ([[]]) can be given a single positive integer denoting the index to return
  - single string naming that element
  - For named elements of lists, use the dollar sign operator, $
    - IDEs will autocomplete the name for you (use Tab in Rstudio)

List Demo program

Converting Between Vectors and Lists

- as.list function to convert Vectors to lists
- Possible to convert list to vector if each element of list contains a scalar value
- Lists are very useful for storing data of the same type, but with a nonrectangular shape
  - This sort of list can be converted to a vector using the function unlist

List Demo program

Combining Lists

- c function for concatenating vectors also works for concatenating lists
- If c is used to concatenate lists and vectors
  - the vectors are converted to lists before the concatenation occurs

List Demo program

NULL

- NULL is a special value that represents an empty variable
- Used when you want to specify that an element should exist, but should have no contents
- NULL can be used to remove elements of a list
  - Setting an element to NULL (even if it already contains NULL) will remove it.
- To set an existing element to be NULL
  - cannot simply assign the value of NULL, since that will remove the element
  - Instead, it must be set to list(NULL)
- Important to understand difference between **NULL** and the special missing value **NA**
  - Biggest difference is that NA is a scalar value, whereas NULL takes up no space at all—it has length zero

List Demo program

Data Frames

- Data frames are used to store spreadsheet-like data
  - They can either be thought of as matrices where each column can store a different type of data
  - or non-nested lists where each element is of the same length

Dataframe Demo program

Creating Data Frames

- data.frame function
- Each column can have a different type than the other columns
  - but all the elements within a column are the same type
- possible to create a data frame by passing different lengths of vectors
  - as long as the lengths allow the shorter ones to be recycled an exact number of times
  - technically, lowest common multiple of all the lengths must be equal to the longest vector

Dataframe Demo program

Indexing Data Frames

- Pairs of the four different vector indices (positive integers, negative integers, logical values, and characters) can be used in exactly the same way as with matrices

Dataframe Demo program

FUNCTIONS

Functions

- Typing the name of a function shows you the code that runs when you call it
- To create our own functions, we just assign them as we would any other variable
- Functions can be used just like other variable types, so we can pass them as arguments to other functions, and return them from functions

Function Demo program

THANK YOU !