Bio 204: Biological Data Analysis

Introduction to R

Instructor: Paul M. Magwene

Fall 2016

Overview of Lecture

- Introduction to R
  - R resources
  - Important programming concepts
  - Introduction to data types and data structures in R
- Literate programming
- Hands-On Session

What is R?

- 'A language and environment for statistical computing and graphics'
- First developed in the mid-90s
- Derives from the S language
  - S was developed at Bell Labs in the mid-80s
- Advantages
  - Free and open-source
  - Much of the academic statistical community has adopted it
  - Active developer and user community
  - Wealth of built-in and user contributed libraries available for all types of analyses
- Disadvantages
  - GUI not as well developed as commercial statistical packages
    - S-Plus; site licensed by Duke - see OIT website
  - Has higher learning curve than some other simpler statistical software
  - Command-line can be intimidating

R Resources on the Web

- Home Page
  - `http://www.r-project.org`
- Comprehensive R Archive Network (CRAN)
  - `http://cran.r-project.org/mirrors.html`
  - See especially the 'Task Views'
    - Statistical and population genetics
    - Environmental and ecological analysis
    - Spatial statistics
- Introductions and Tutorials
  - see `http://cran.r-project.org/other-docs.html`

Programming Concepts & Vocabulary

Some Important Programming Concepts

- Data Types
  - refer to the types of values that can be represented in a computer program
  - determine the representation of values in memory
  - determine the operations you can perform on those values
  - Examples: integers, strings, floating point values
- Data Structures
  - a way of storing and accessing collections of data
  - different structures are more efficient for particular types of operations
  - Examples: lists, hash tables, stacks, queues, trees
- Variables
  - Variables are references to objects/values in memory
  - Think of them as labels that point to particular places in a computer's memory

More Important Programming Concepts

- Statement
  - an instruction that a computer program can execute
  - Example: `print("Hello, World!")`
- Operators
  - Symbols representing specific computations
  - Example: `+`, `-`, `*` (addition, subtraction, multiplication)
- Expression
  - a combination of values, variables, and operators
  - Example: `1 + 1`
- Functions (subroutines, procedures, methods)
  - A piece of code that carries out a specific task, set of instructions, calculations, etc.
  - Typically used to encapsulate algorithms

Basic Data Types, Data Structures and Operators in R

Arithmetic Operators and Mathematical Functions in R

```
> 10 + 2 # addition
[1] 12
> 10 - 2 # subtraction
[1] 8
> 10 * 2 # multiplication
[1] 20
> 10 / 2 # division
[1] 5
> 10 ^ 2 # exponentiation
[1] 100
> 10 ** 2 # alternate exponentiation
[1] 100
> sqrt(10) # square root
[1] 3.162278
> 10 ^ 0.5 # same as square root
[1] 3.162278
> pi*(3)**2  # R knows some useful constants
[1] 28.27433
> exp(1) # exponential function
[1] 2.718282
```

Numeric Data Types in R

- Floating point values ('doubles')

```
> typeof(10.0)
[1] "double"
```

- Integers
  - Default numeric type is double, must explicitly ask for integers if single values

```
> typeof(as.integer(10))
[1] "integer"
> typeof(10L) # appending L is alternate way to get an integer
[1] "integer"
```

- Complex numbers

```
> typeof(1 + 1i)
[1] "complex"
```

Boolean Values and Operators and Comparison Operators

Boolean ('logical') values represent binary True/False states.

```
> x <- TRUE
> typeof(x)
[1] "logical"
> y <- FALSE
> x <- T # shorthand for TRUE
> y <- F # shorthand for FALSE

> TRUE & FALSE   # logical and
[1] FALSE
> TRUE | FALSE   # logical or
[1] TRUE
> !True          # logical negation
[1] FALSE
> !FALSE
[1] TRUE
> y <- FALSE
> isTRUE(y)
[1] FALSE
```

Comparison Operators

The common comparison operators when applied to numerical values return Boolean reults.

```
> 5 < 10  # less than
[1] TRUE
> 5 > 10 # greater than
[1] FALSE
> 10 <= (5 + 5) # less than or equal to
[1] TRUE
> 10 >= (5 + 5.1) # greater than or equal to
[1] FALSE
> 10 == (2 * 5)  # equality (not be be confused with =)
[1] TRUE
> 10 != (3 + 3 + 3 + 3) # not equal to
[1] TRUE
```

Characters (strings) in R

Character strings are written between single or double quotes.

```
> x <- 'Hello' # or x <- "Hello"
> typeof(x)
[1] "character"
> y < - "World"
> paste(x,y)  # join strings
[1] "Hello World"
> z <- paste(x,y)
> substr(z, 1, 3)
[1] "Hel"
> substr(z, 4, 8)
[1] "lo Wo"
```

Simple Data Structures in R: Vectors

Vectors are the simplest data structure in R

- vectors represent an ordered list of items

```
> x <- c(2,4,6,8)
> y <- c('joe','bob','fred')
```

- vectors have length (possibly zero) and type

```
> typeof(x)
[1] "double"
> length(x)
[1] 4
> typeof(y)
[1] "character"
```

Simple Data Structures in R: Vectors

Accesing the objects in a vector is accomplished by 'indexing':

- The elements of the vector are assigned indices `1`...n where n is the length of the vector

```
> x <- c(2,4,6,8)
> length(x)
[1] 4
> x[1]
[1] 2
> x[2]
[1] 4
> x[3]
[1] 6
> x[4]
[1] 8
```

Simple Data Structures in R: Vectors

- Single objects are usually represented by vectors as well

```
> x <- 10.0
> length(x)
[1] 1
> x[1]
[1] 10
```

- Every element in a vector is of the same type
  - If this is not the case the the values are coerced to enforce this rule

```
> x <- c(1+1i, 2+1i, 'Fred', 10)
> x
[1] "1+1i" "2+1i" "Fred" "10"
```

Arithmetic Operators Work on Vectors in R

Most arithmetic operators and mathematical functions work element-by-element on vectors in R

```
> x <- c(2, 4, 6, 8)
> y <- c(0, 1, 2, 3)
> x + y
[1]  2  5  8 11
> x - y
[1] 2 3 4 5
> x * y
[1]  0  4 12 24
> x^2
[1]  4 16 36 64
> sqrt(x)
[1] 1.414214 2.000000 2.449490 2.828427
```

Things to Remember

- Try it out - programming involves experimentation
- Practice - learning to program, like learning a foreign language, requires lots of practice.
- Persist - many new tools/concepts can be hard to grasp at first. Keep plugging away until you get that ‘Aha!’ moment