Introduction to Data Science

Daniel Gutierrez, Data Scientist

Los Angeles, Calif.

Course Outcomes

- Introduction to R Programming – Part 3

Lesson Objectives

- Vectorized operations
- if control structure
- Logical expressions
- for control structure, nested loops
- while control structure
- repeat control structure
- Defining and using functions in R, recursion, argument passing
- `lapply()` loop function
- `sapply()` loop function

R Fundamentals – Part 3

- As a statistical tool, R has the ability to perform vectorized operations like in linear algebra
- Compare elements of a vector
- Perform calculations on vectors: element wise addition, subtraction, multiplication and division
- Perform calculations on matrices: element wise addition, subtraction, multiplication and division
- Perform matrix multiplication
- Compute inverse of a matrix
- Compute transpose of a matrix

R Fundamentals – Part 3

- Flow of control structure: `if` statement
- Explore a variety of logical expressions and operators
- Flow of control structure: `for` statement
- Nesting `for` statements, e.g. traversing rows and columns of a matrix
- Flow of control structure: `while` statement
- Flow of control structure: `repeat` statement
- Infinite loops, something to avoid!
- User defined functions in R: structure, argument passing, recursion

R Fundamentals – Part 3

- Writing `for` and `while` loops is useful when programming but not particularly easy when working interactively on the command line. There are some functions which implement looping to make life easier.
  - `lapply()`: Loop over a list and evaluate a function on each element
  - `sapply()`: Same as `lapply()` but try to simplify the result
  - `apply()`: Apply a function over the margins of an array
  - `tapply()`: Apply a function over subsets of a vector
  - `mapply()`: Multivariate version of `lapply()`
- An auxiliary function `split()` is also useful, particularly in conjunction with `lapply()`.

R Fundamentals – Part 3

- `lapply()` takes three arguments: a list X, a function (or the name of a function) FUN, and other arguments via its… argument. If X is not a list, it will be coerced to a list using `as.list()`.

```
> lapply
function (X, FUN, ...)
{
    FUN <- match.fun(FUN)
    if (!is.vector(X) || is.object(X))
        X <- as.list(X)
    .Internal(lapply(X, FUN))
}
```

- The actual looping is done intentionally in C code.

R Fundamentals – Part 3

- `sapply()` will try to simplify the result of `lapply()` if possible.
  - If the result is a list where every element is length 1, then a vector is returned
  - If the result is a list where every element is a vector of the same length (>1), a matrix is returned
  - If it can't figure things out, a list is returned

Code module

- WEEK 3-1 Code module – vectorized operations
- WEEK 3-2 Code module – matrix inverse and transpose
- WEEK 3-3 Code module – `if` control structure
- WEEK 3-4 Code module – logical expressions and operators
- WEEK 3-5 Code module – `for` control structure
- WEEK 3-6 Code module – `while` control structure
- WEEK 3-7 Code module – `repeat` control structure
- WEEK 3-8 Code module – user defined functions
- WEEK 3-9 Code module – `lapply()` loop function
- WEEK 3-10 Code module – `lapply()` and `sapply()`

Summary

- In WEEK 3 we continued building our data science toolbox by added a number of tools used for data science projects
- Vectorized operations
- Flow-of-control structures
- Logical expressions
- User defined functions