functions λ in R

A function is a group of instructions that take inputs, performs computations, and returns a result.

```
1   functionName <- function(x) {
2   logical computations
3   }
```

The above illustrates the structure of a function with one argument, x. in formal terms, x function(x) refers to the formal argument (parameter), with the actual value input of x referred to as the actual argument.

Variables existing within functions are local variables, they exist only within the computations of the function and disappear after the result is computed. Variable assignments outside of functions are global variables and will retain their values until reassigned or removed from the environment.

Variables within functions can also be assigned default arguments that will use the defined function values if not specified by the user within the function parameters.

data structures in R

Vectors drive the core functionality of R programming. Vector elements in R must be consistent of the same data types (mode) throughout.

Scalars (individual numbers) technically do not exist in R. Scalars are treated as one-element vectors.

Character Strings are also one-element vectors with a character data type (or mode character).

Matrices in R correspond to the mathematical concept of the same nature. A Matrix is a rectangular array of numbers. Technically, a matrix is a vector with two additional attributes; 1) the number of rows; and 2) the number of columns.

Like a vector in R, a **list** in R is a container for values. The difference between vectors and lists are that lists can contain many different data types. List elements are accessed in R with the dollar sign $ syntax.

In application, a typical data set consists of multiple types of data that cannot be stored in a single matrix in R alone. Instead, **data frames** are created to store actual data sets. A data frame in R is a list, with each component of the list being a vector related to each column in the data set.

In R, objects are instances of classes. **Class instances** are R lists with the class name as an extra attribute. Class objects in R are often utilized as R's S3 classes. The S3 classes are from the S language, version 3.

vectors [ ] in R

Vectors are the fundamental data type in R; the following topics are discussed in detail:

- … Recycling → automatic lengthening of vectors in certain settings
- … Filtering → extracting subsets of vectors
- … Vectorization → functions are applied element-wise to vectors

recycling in R

When performing an operation with two vectors that requires an equal vector length, R automatically recycles (repeats) the shorter vector until it reaches the same length as the longer vector.

common vector operations = in R

With R being a functional programming language, all mathematical operators are actually functions. R executes operations element-wise on vectors and matrices. Importantly in relation to Linear Algebra, R performs the multiplicative, and other numeric operations, element-by-element.

Indexing vectors occurs when a subvector is formed by selecting elements of the given vector for specific indices. The format is **`vector1[vector2]`**; the result is a selection of elements from **`vector1`** whose indices are provided by **`vector2`**. As stated previously, R indexes vectors by 1, and not 0, like other languages.

The colon **`:`** operator creates a range of numbers that is useful for indexing vectors and functions. It is important to understand the operator precedence constraints in R (order of operations). For example, the colon **`:`** operator has a higher precedence than the subtraction **`–`** operator, but less precedence than the parenthesis **`()`** operator.

The sequence **`seq()`** function is a generalization of the colon **`:`** operator, which generates a sequence in arithmetic progression. Unlike the colon **`:`** operator, the sequence **`seq()`** function can specify units apart.

The sequence **`seq()`** function is additionally useful in loops, where the **`length()`** function falls short in loops where the argument is null, and thus should have zero iterations of the loop.

The repeat **`rep()`** function allows the placement of the same constant in vectors; creates a vector of size **`x`** times the copies of **`x`**. Note the use of the **`each =`** argument having nonexpected behavior in **`rep()`**. The **`any()`** and **`all()`** functions are convenient to examine the logical outputs of any given argument.

vectorized operations in R

Operations in R are vectorized when a function is applied individually to each element in a vector. Simple to complex operators can be vectorized in functions; inclusive of transcendental functions—square roots, logs, trig functions, etc.—are all vectorized. Formally speaking, scalars are one-element vectors in R and arithmetic operators (+) are functions within themselves.

```
> y <- c(3, 6, 9)
> y + 4               #the numeric 4 will be added element-wise to vector 'y'
[1]  7 10 13

> '+'(y, 4)      #'+' as a function of vector 'y' with input 4 yields the same results
[1]  7 10 13
```

Note the recycling took part in the above function as ‘4’ was recycled into matching vector length (4,4,4). Vectorization should be accounted for when writing functions and code in R for ‘code ethics and safety’.

NA and NULL values Y in R

"No such Animal" values are accounted for specifically in R as either NA or NULL values. Missing data in statistical samples is denoted as NA; implying that the data is simply missing from the dataset. Conversely, the NULL assignment of a value in a dataset implies that the example value simply *does not exist*; rather than being existent but unknown.

```
> x <- c(3, NA, 9)        #3-element vector with the middle value set = NA
> x
[1]  3 NA  9
> mean(x)                 #R accounts for the NA value in the function mean()
[1] NA
> mean(x, na.rm = T)      #argument to remove NA values from the vector input
[1] 6
> x <- c(3, NULL, 9)      #3-element vector with the middle value set = NULL
> mean(x)                 #R does not account for NULL values in the function mean()
[1] 6
```

One use of NULL is to build vectors within loops; each iteration adding a new element(s) to the vector. Concretely, NULL values in R are treated as nonexistent; NULL is a special R object without a class (mode).

```
> x <- NULL             #one-element vector of value = NULL
> length(x)
[1] 0
> y <- NA               #one-element vector of value = NA
> length(y)
[1] 1
```

vector names and classes in R

Vector elements can be names with the names( ) function; vectors can also be referenced by their names as "strings". The concatenate c( ) function has unique properties in R:

- … Arguments passed to c( ) are reduced to the type of lowest common denominator (mode/class)
- … c( ) has a flattening effect for vectors