R Lab 2: Working with Objects and Functions

Ikuma Ogura

August 21, 2019

Today

- Using R as a calculator
- Object
  - Object class/type
- Functions

Using R as Calculator

```
5 + 3 # summation
```

```
## [1] 8
```

```
4 - 7 # subtration
```

```
## [1] -3
```

```
5 * 6 # product
```

```
## [1] 30
```

```
7 / 3 # division
```

```
## [1] 2.333333
```

Using R as Calculator (cont.)

```
7 %% 3 # modular (residual)
```

```
## [1] 1
```

```
2 ^ 6 # power
```

```
## [1] 64
```

```
(2 + 5) * 4 + 2 ^ 3 # note the order of calculation
```

```
## [1] 36
```

Object

- In R, we store information as an **object**. Once we create an object, we can refer it to its name.
- We assign values to an object using the assignment operator <-.
  - We can also use = for assignment (although not recommended)
- Object class/type
  - what kind of information is stored in the object and how it is stored
  - `typeof()` or `class()` command to see the object type

Object Class/Type

- Data types
  - **Character**: character strings
  - **Numeric**: numbers
  - **Logical**: boolean data (`TRUE`/`FALSE`)
  - *Factor*
- Data structures
  - **Vector**: a single-dimension sequence of data of the same type
  - **Matrix**: a two-dimension sequence of data of the same type
  - **Data Frame**: a two-dimension structure of data of varying data types
  - *List*

Object: Example

```
# Numeric vector
num <- c(4, 2, 6, 8, 5)
# Character vector
program.lang <- c("R", "Python", "C", "Java")
# Logical vector
comparison <- (num >= 5)
comparison
```

```
## [1] FALSE FALSE  TRUE  TRUE  TRUE
```

Object: Example (cont.)

```
# Object class/type
class(num)
```

```
## [1] "numeric"
```

```
class(program.lang)
```

```
## [1] "character"
```

```
class(comparison)
```

```
## [1] "logical"
```

Logical Operators

Operator	Meaning
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`==`	equal to
`!=`	not equal to
`|`	or
`&`	and
`is.na()`	`TRUE` if missing
`!is.na()`	`FALSE` if missing

Logical Operators: Example

```
7 < 5
```

```
## [1] FALSE
```

```
(6 > 4) | (8 < 5)
```

```
## [1] TRUE
```

```
(7 > 3) & (9 <= 11)
```

```
## [1] TRUE
```

Command/Function

- We use a command/function to perform some tasks on an object/objects
- Argument: the definitions, directions, or objects that are passed to a command/function
- If we specify multiple arguments,
  - separate the arguments by commas
  - it is desirable to specify them along with their names unless they are obvious
    - ★ The code looks like `funcname(arg1 = input1, arg2 = input2)`
- We can access to function help files either by `?funcname` or `help("funcname")`
  - However, it is often difficult to understand what the help files are saying..
  - Google search, ask others (including me!)...

Command/Function: Example

```
log(num)
```

```
## [1] 1.3862944 0.6931472 1.7917595 2.0794415 1.6094379
```

```
sqrt(num)
```

```
## [1] 2.000000 1.414214 2.449490 2.828427 2.236068
```

```
length(num)
```

```
## [1] 5
```

```
sum(num)
```

```
## [1] 25
```

Command/Function: Example (cont.)

```
sort(num)
```

```
## [1] 2 4 5 6 8
```

```
sort(num, decreasing = TRUE)
```

```
## [1] 8 6 5 4 2
```

Command/Function: Example (cont.)

- Output of `help("sort")`

sort {base} R Documentation

Sorting or Ordering Vectors

Description

Sort (or *order*) a vector or factor (partially) into ascending or descending order. For ordering along more than one variable, e.g., for sorting data frames, see `order`.

Usage

```
sort(x, decreasing = FALSE, ...)

## Default S3 method:
sort(x, decreasing = FALSE, na.last = NA, ...)

sort.int(x, partial = NULL, na.last = NA, decreasing = FALSE,
         method = c("auto", "shell", "quick", "radix"), index.return = FALSE)
```

Arguments

`x`	for `sort` an `R` object with a class or a numeric, complex, character or logical vector. For `sort.int`, a numeric, complex, character or logical vector, or a factor.
`decreasing`	logical. Should the sort be increasing or decreasing? For the `"radix"` method, this can be a vector of length equal to the number of arguments in `...`. For the other methods, it must be length one. Not available for partial sorting.
`...`	arguments to be passed to or from methods or (for the default methods and objects without a class) to `sort.int`.
`na.last`	for controlling the treatment of `NA`s. If `TRUE`, missing values in the data are put last; if `FALSE`, they are put first; if `NA`, they are removed.
`partial`	`NULL` or a vector of indices for partial sorting

Tomorrow

- Working with vectors and matrices with R