R Data Structures

Training Session with Health Systems Consult Ltd, Abuja
Tuesday 15 June 2021

Presenter: Dr Victor A. Ordu

"To understand computations in R, two slogans are helpful:

- Everything that exists is an object.
- Everything that happens is a function call."

— John Chambers

Source: H Wickham (2014). *Advanced R*. Chapman & Hall, Boca Raton. http://adv-r.had.co.nz/

Types of Data Structures

1. Atomic Vectors
2. Data frames
3. Matrices
4. Lists

Dimensionality vs Sameness

	Homogenous	Heterogeneous
1-dimension	*Atomic vectors*	*Lists*
2-dimensions	*Matrices*	*Data frames*

6 Kinds of Atomic Vectors

1. Character
2. Integer
3. Double (or numeric)
4. Logical
5. Raw
6. Complex

Memory model

x				
2.6	3.2	3.8	4.4	5

y				
2.6	3.2	3.8	4.4	5

Characteristics of Atomic Vectors

- All elements are of a particular data **type** (in lay language, “type” would be numbers, words, etc.)
- One-dimensional
- Scalars are actually vectors of length 1

Making atomic vectors

- By assignment
  - The concatenate function, `c()`
    - Latin: *con* – ***caten*** – *atus* (chain)
    - Some call it “combine” function
    - Indispensible in creating vectors

- You can ‘grow’ a vector
  - You may ask how, much, much, much, much later

Character vectors

- Strings are always placed in quotation marks when coding i.e. “boy”, “Health”, “R is easy to learn”, “A string can be a whole sentence!”, “9”.
- Some character vectors are inbuilt into R e.g. `letters`, `LETTERS`, `month.abb`, `month.name`
- Remember use quotation marks: “ ” or ‘ ’.
- We can create empty vectors with specific lengths e.g. `character(length = 10)` or `character(10)`
- Limit: Approx. 2^{31} (about 2 billion) characters!

Exercise

- Start a clean slate with `rm(list = ls())`
- Make a character vector `Name` containing full names (both Surname and Given Name in each element) of 10 adults
- Make a second vector `Facility` of names of 10 facilities (imaginary, please!)
- Use `typeof()` to check what type of vector `Name` is
- Confirm the type of `Facility` using `is.character()`
- Note: We can use `as.character()` to convert another vector to a character vector.

- Integer vectors
  - 1L, 2L, 3L
    - Why the ‘L’?
  - Not numerical per se
  - Wide range – max up to 2,147,483,647

  - Exercise
    - Make an integer vector `Age` of 10 adult subjects
    - Make an integer vector `StaffStrength` for 10 facilities

Integer vectors

- 1L, 2L, 3L
  - Why the ‘L’?
- Not numerical per se
- Wide range – max up to 2,147,483,647

- Exercise
  - Make an integer vector `Age` of 10 adult subjects
  - Make an integer vector `StaffStrength` for 10 facilities

Numeric (double) vectors

- These are real numbers
- Existence of inbuilt numeric vectors i.e. mathematical constants e.g. `pi`

Logical vectors

- TRUE/FALSE (not true/false); T/F
- Zero is FALSE; **any** non-zero is TRUE

- Exercise
  - Make a logical vector `PermitSighted` for 10 facilities.
  - Make another one `usingPPE` for 10 individuals.
  - Use `str()`, `typeof()`, `is.logical()`, to explore them.

Named vectors

- Each element of an atomic vector can be named e.g.

```
> cardinal <- c(E = "East", W = "West", N = "North", S = "South")
> cardinal
      E       W       N       S
 "East"  "West" "North" "South"
> names(cardinal)
[1] "E" "W" "N" "S"
```

Coercion

- Remember: All elements of atomic vectors are of the same time
- This is why they are called atomic.
- When we mix elements that are usually interpretable as different type, R will attempt coercion.
- Where this is not feasible, an error is signalled.

Implicit coercion

- Governed by precedence rules
- Character >> numeric >> integer >> logical

```
> c("Me", "you", 10)
[1] "Me"  "you" "10"
> c("Me", pi, 5)
[1] "Me"                "3.14159265358979" "5"
> val <- c(pi, 6L)
> is.integer(val)
[1] FALSE
> is.numeric(val)
[1] TRUE
> c(TRUE, "is")
[1] "TRUE" "is"
> c(FALSE, 0)
[1] 0 0
```

Explicit coercion

- Use functions to force elements to a given type.
- Can fail when it doesn't make sense

```
> num <- c(pi, exp(1))
> num
[1] 3.141593 2.718282
> as.integer(num)
[1] 3 2
> as.character(num)
[1] "3.14159265358979" "2.71828182845905"
> as.logical(num)
[1] TRUE TRUE
> numstring <- c("5", "444")
> wordstring <- c("five", "four hundred and forty-four")
> as.numeric(numstring)
[1]   5 444
> as.numeric(wordstring)
[1] NA NA
Warning message:
NAs introduced by coercion
```

Factors

- Integer values that are mapped to “strings”
- Used to represent categorical data
- Each category is called a level
- One of the most powerful uses of R, especially for modelling

- Exercise
  - Make a vector `industryType` using 3 categories – small, medium, large – for 10 facilities only.
  - Make a factor `industryCategory` by calling the function `factor()` on `industryType`.
  - Now use `typeof()`, `is.factor`, `is.character`, `is.integer()` to review these 2 objects.

To be continued…