2. Data types and Data structures in R

Principles of Data Science with R

Dr. Uma Ravat

PSTAT 10

R essentials: summary

- Console and Environment Panes, Command Prompt
- Objects
  - Variables: nouns
  - Functions: verbs
  - Naming conventions
- Packages: ready made functions and datasets from others
  - Install once
  - Load every time you need it
- Help : ?
- Assignment Operator : <-
  - printing objects
- Comments: #
  - **use them!** for yourself, the grader
- Coding style : **have one** and be consistent
  - See chapters 1-3 of the tidyverse style guide
- Environment

Post-Lecture 2 To DO

1. Review the lecture again
2. Write down a summary of today's lecture. Include all functions we went over and a short description of what each function does.

You will be asked to do this to your homework.

Data types and Data structures

Data types in R

- **character** (also known as string, ' or "): `"a"`, `'PSTAT'`
- **double** (also known as **numeric**): `2`, `15.5`
  - Any number with **(or without)** a decimal point
- **integer** (whole numbers): `2L`
  - the `L` tells R to store the number 2 as an integer
- **logical**: `TRUE`, `FALSE` (same as 1 or 0)

Check the object's data type : `typeof()` function

Go L02-Examples.Rmd Section 1.

Data Type - Logical

`TRUE`, `FALSE` values

- $<$ *(less than)*
- $<=$ *(less than or equal to)*
- $>$ *(greater than)*
- $>=$ *(greater than or equal to)*
- $==$ *(exactly equal to)*
- **!=** *(not equal to)*
- **!x** *(Not x)*
- **x | y** *(x OR y)*
- **x & y** *(x AND y)*
- **isTRUE(x)** *(test if x is TRUE)*

Data structures

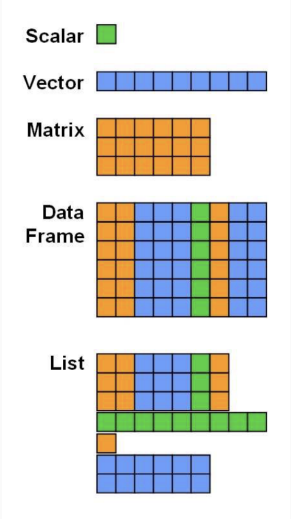
Data structures

A *data structure* is a mechanism to group related data values into an **object.**

Everything that exists in R is an object.

R has many data structures :

- **scalar:** stores one value at a time
- **(atomic) vector :** stores a sequence of values, all the same type
- **matrix :** data is stored in rows and columns, all of same type
- **data frame :** data is stored in rows and columns, each column can be a different data type.
- **list :** elements can have different types
- **factors :** for non-numeric or categorical data eg color of eyes



Things to know about each of these data structures

- differences between different data structures
- Creating and storing data in each of them in R
- What functionality (functions) does each come with
- Selecting and updating the data stored in each one of them

SCALAR data structure

Scalars can hold only one value at a time.

EXAMPLE

```
(x <- 4) # no need to write print, use () instead!
```

```
## [1] 4
```

```
(y <- "Hello! Have you fallen asleep?")
```

```
## [1] "Hello! Have you fallen asleep?"
```

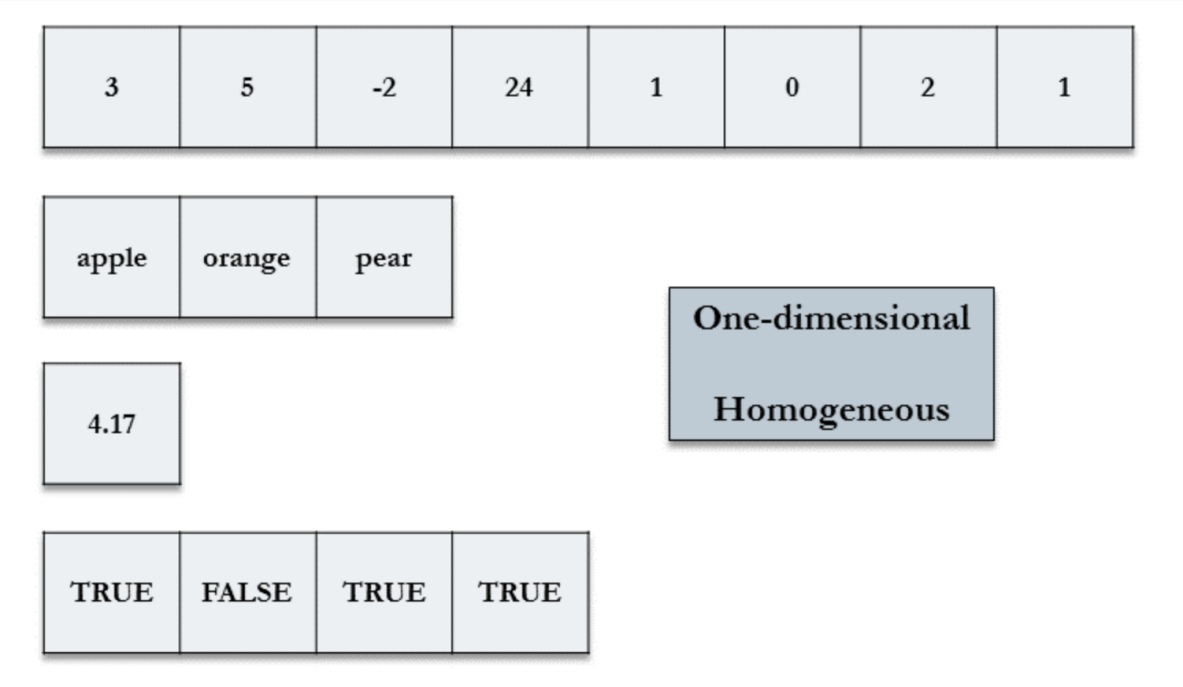
```
(asleep <- FALSE)
```

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

Vectors

Vectors store a sequence of values, all of the same type

- most common and basic data structure in `R`
- workhorse of `R`
- also refered to as **atomic vectors**
- one-dimensional and homogeneous data structure
- A scalar data structure is just a vector of length 1.



Let's look at creating vectors and some functions

- c() : the `combine` function

Functions:

- typeof() : What type of data is stored in the vector
- length() : the number of elements contained in the vector.
- sort() : the `sort` or `ordering` function

Go L02-Examples.Rmd Section 3. Creating Vectors with combine function

R allows you to coerce to any data type

Know what you are doing with `as.datatype()` functions

```
a <- 23; b <- '23'; c <- as.character(a)
a; b; c
```

```
## [1] 23
```

```
## [1] "23"
```

```
## [1] "23"
```

```
typeof(a); typeof(b); typeof(c)
```

```
## [1] "double"
```

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

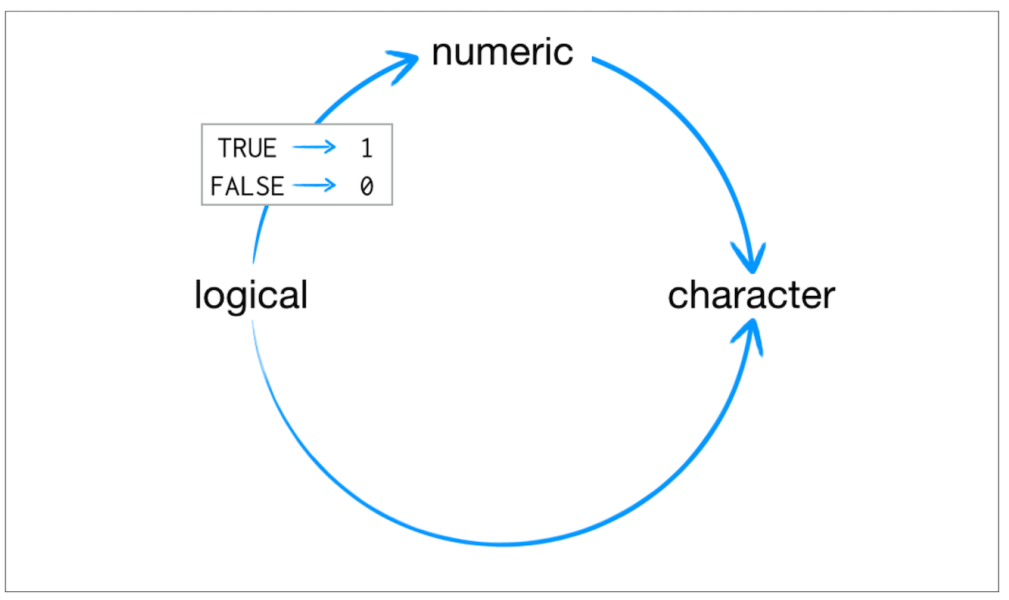
```
## [1] "character"
```

R will also automatically coerce for you too!

WARNING! Be careful, R doesn't complain while coercing

This is a source of frustration for beginning programmers!

- Vectors can only contain one data type.
- Vectors are coerced to the simplest type required to represent all information.



Automatic Coercion in R

```r
auto_coerced <- c(1, "8", 5)
auto_coerced
```

```
## [1] "1" "8" "5"
```

```r
typeof(auto_coerced)
```

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

```r
l <- TRUE
typeof(l)
```

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

```r
new_auto_coerced <- c(auto_coerced, l)
new_auto_coerced
```

```
## [1] "1"    "8"    "5"    "TRUE"
```

```r
typeof(new_auto_coerced)
```

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

```r
# Better formatting using paste and strings along with the variable
paste("Type of `new_auto_coerced` is :",   typeof(new_auto_coerced))
```

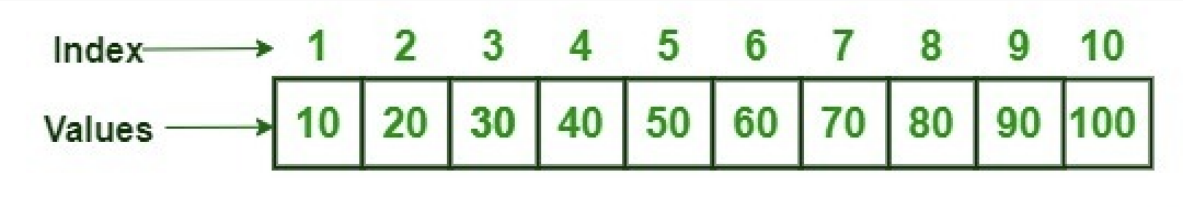
```
## [1] "Type of `new_auto_coerced` is : character"
```

Creating vectors: More (Faster) ways to create (long) vectors

- `:` - the colon operator
- `seq()` - the `sequence generation` function
- `rep()` - the `replicate` function

Go L02-Examples.Rmd Section 6. Creating a vector faster using :, seq , rep

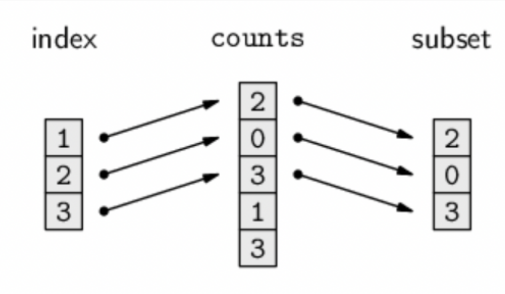
How do we access, update elements of a vector?



- Using their **index** and the square bracket operator **[ ]**

Go L02-Examples.Rmd Section 7.Accessing and updating an element of a vector using the square bracket operator [ ]

Subseting a vector



- Selecting only certain elements
  - Using `[ ]` operator
  - Using `:` operator for extracting successive elements
  - Using `c()` function for extracting non-successive elements
  - that match a selection criteria by comparision
    - $<$ for less than
    - $>$ for greater than
    - $\leq$ for greater than or equal to
    - $\geq$ for less than or equal to
    - $==$ for equal to each other
    - $!=$ for not equal to each other

go L02-Examples.Rmd Section 8. Subsetting a vector

Working with vectors: Vectorized operations

Many operations in R are already vectorized.

```
(a <- 1:5); (b <- 6:10)
```

```
## [1] 1 2 3 4 5
```

```
## [1]  6  7  8  9 10
```

```
a + b # try other math operations
```

```
## [1]  7  9 11 13 15
```

```
(x <- (5:10)^2)
```

```
## [1]  25  36  49  64  81 100
```

```
log(x)
```

```
## [1] 3.218876 3.583519 3.891820 4.158883 4.394449 4.60517
```

Vector math: Adding a scalar to a vector!

```
x <- 1:10
x + 6
```

```
##  [1]  7  8  9 10 11 12 13 14 15 16
```

Some more functions for vectors

```
(quiz_score <- c(10, 0, 5))
```

```
## [1] 10  0  5
```

```
diff(quiz_score)
```

```
## [1] -10   5
```

Even more functions for vectors (more generally for objects)

```r
# Assign names to entries in our score vector
names(quiz_score) <- c("Quiz1", "Quiz2", "Quiz3")
quiz_score
```

```
## Quiz1 Quiz2 Quiz3
##    10     0     5
```

```r
# view the names that we assigned to our score vector.
names(quiz_score)
```

```
## [1] "Quiz1" "Quiz2" "Quiz3"
```

```r
attributes(quiz_score) # metadata about the object
```

```
## $names
## [1] "Quiz1" "Quiz2" "Quiz3"
```

1. What's wrong with this code? (`esc` will rescue you!_)

```
hello <- "Hello world!
```

Suppose we have test scores for 5 students: Bob, Alice, Alex, Juan and Amy.

Their scores are 8, 7, 8, 10, and 5 respectively.

1. Create a vector of these scores.
2. Find the mean score in two ways (using `mean` and using `sum`).
3. Find the median score.
4. Assign the name of each student to their test score.
5. Retrieve Alice's score in two ways.
6. Retrieve Amy's and Alice's score, in that order.
7. Retrieve all except Amy's score.

questions you should be able to answer

- "What are the different data types in R?"
- "What are the different data structures in R?"
- "How do I access data within the various data structures?"

Post-Lecture 2 To DO

1. Review the lecture again
2. Write down a summary of today's lecture. Include all functions we went over and a short description of what each function does.

You will be asked to do this to your homework.

Next we will see...

- More data structures
  - vectors
  - matrix
  - array
  - factor
  - logical operators