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, Assignment Operator : <-
  - Variables: nouns
  - Functions: verbs
  - Naming conventions
- Packages: ready made functions and datasets from others
  - Install once
  - Load every time you need it
- Help : ?
- 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 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...

- Data types (character, double, integer, logical)
- Data structures
  - Scalars
  - Vectors
  - more next time
- What are the different data types and data structures in R?
- What are differences in each of these
- How do I create these, access, update data within the various data structures?

Disclaimer! Many New Terms coming!

Don't worry about memorizing and remembering everything right now.

Instead, focus on recognizing the way `R` has things broken down

1. Data types

Types of Objects aka Data types

Objects(data) in `R`: numbers, letters, words and more.

A data type describes the type, or category, of the data (not the data itself).

- **integer**: integer (1, 2, 3) (`I <- 1L`)
- **double**: floating decimal (10.5, 55.0, 78.6) (`N <- 100`)
- **numeric**: integer or double (`f <- 10.5`)
- **character**: takes string values (e.g. a person's name, address) and must be surrounded by quotes.(`course <- 'PSTAT 194'`, `room = "SH 5500"`).
- **logical**: TRUE (1), FALSE (0) (`t <- TRUE`)
- **factor**: categorical variable with different levels(non-numeric data in categories like eye color can be black, brown, blue, etc)
- Use `typeof()` or `class()` function to check the object type

Logical data type from Comparision operator

`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)*

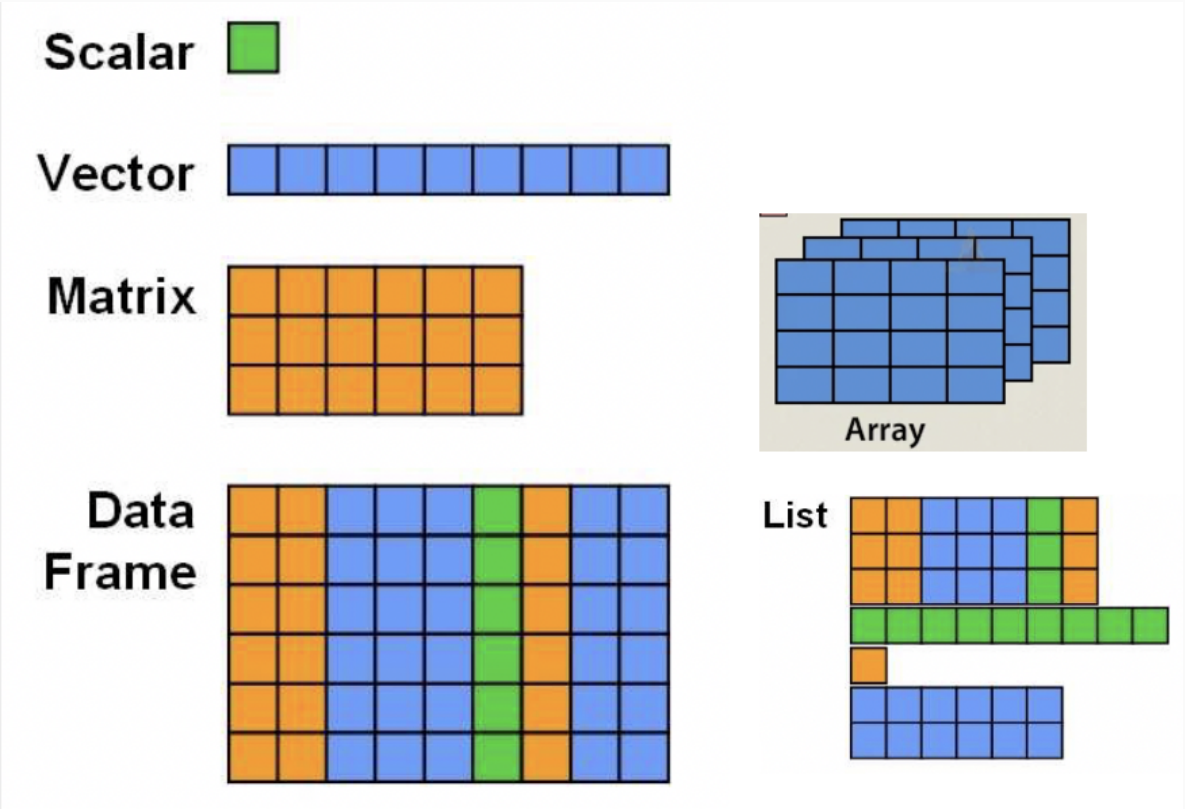
Work Along

Go L02-Examples.Rmd Section 1.

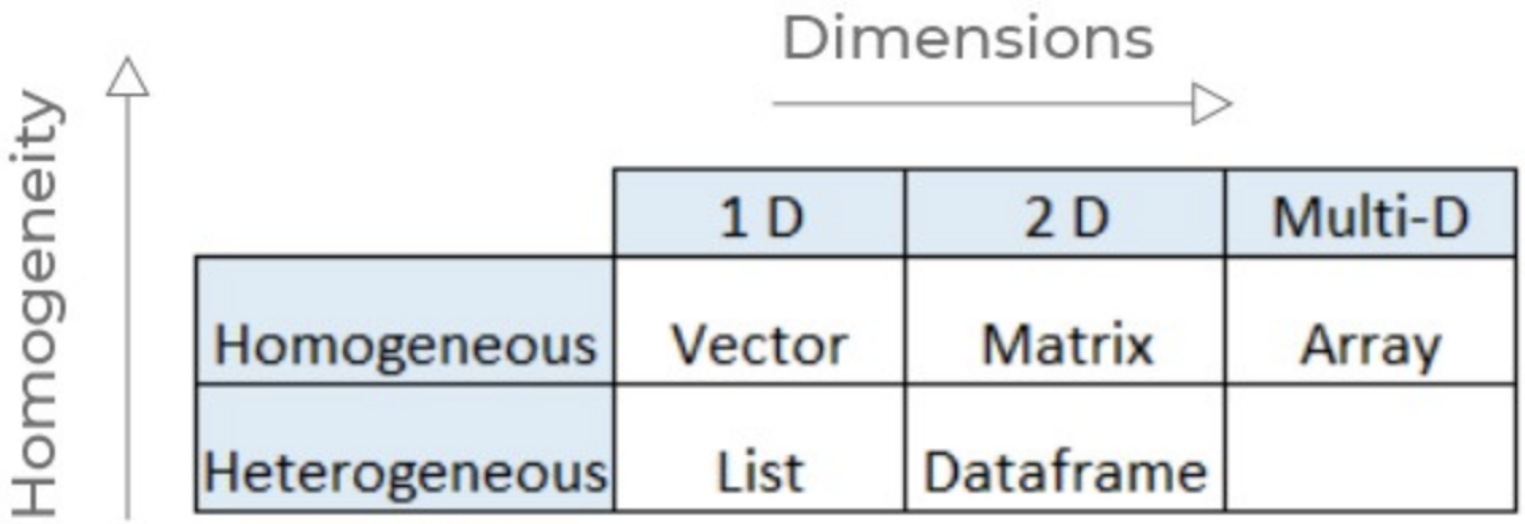
2. Data structures

Data structures

Data structures are tools(objects) for holding multiple data types in one object.



Data structures : dimensionality and data type



	1 D	2 D	Multi-D
Homogeneous	Vector	Matrix	Array
Heterogeneous	List	Dataframe	

Things to know about each of these data structures

- differences between different data structures
- How to create and store 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

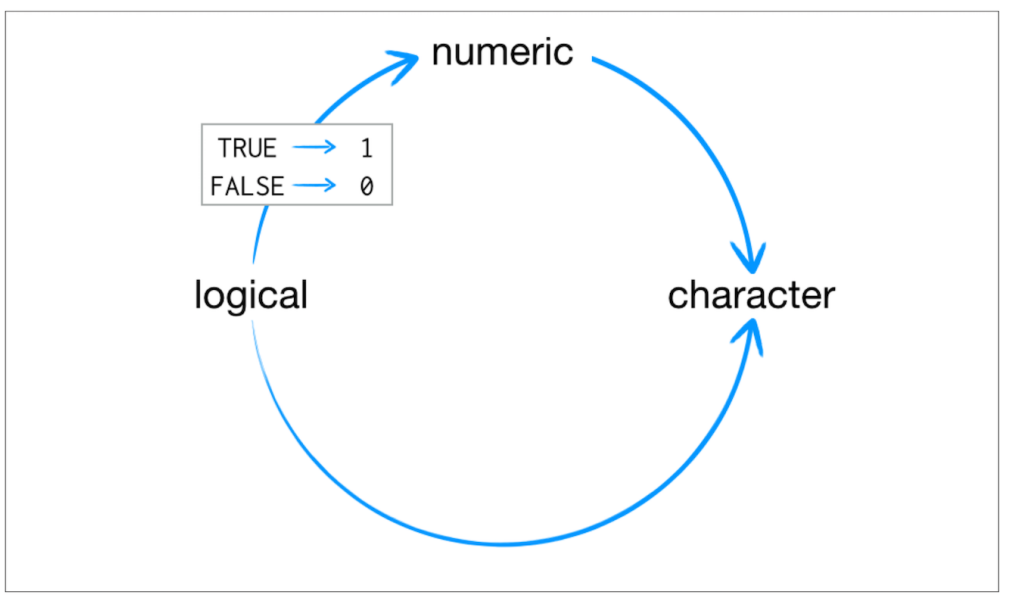
Go L02-Examples.Rmd Section 3. Creating Vectors with combine function

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))
```

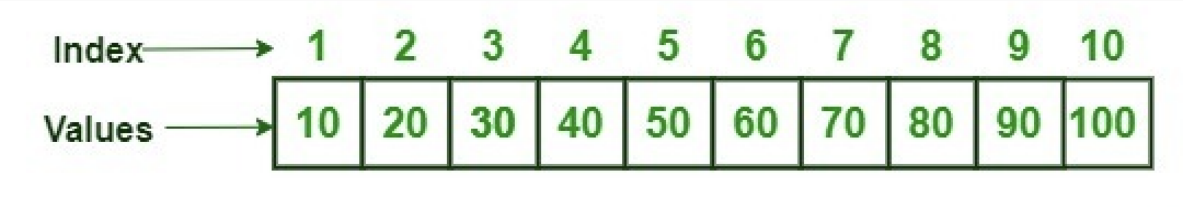
```
## [1] "Type of `new_auto_coerced` is : character"
```

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

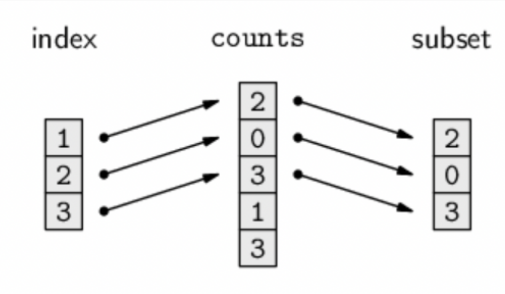
How do we access, update elements of a vector?



- 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

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.

Summary

- Data types (character, double, integer, logical)
- Data structures
  - Scalars
  - Vectors
  - more next time

Post-Lecture 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.

Questions you should be able to answer

- What are the different data types and data structures in R?
- What are differences in each of these
- How do I create these, access, update data within the various data structures?

Next we will see...

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

Learning Programming is HARD!

E. Kale Edmiston PhD
@EKaleEdmiston

Follow

A friend/colleague who is an excellent programmer offhandedly told me the other day that coding is 90% googling error messages & 10% writing code. Until this point, I thought that all the time I spent googling error messages meant I was bad at coding. What a perspective change!

8:12 AM - 4 Jan 2019

151 Retweets **1,069** Likes

27 151 1.1K