

# R Introduction

Young W. Lim

2024-09-16 Mon

- 1 Based on
- 2 Introcuton (B)
  - Introduction

## "An Introduction to R" Notes on R: A Programming Environment for Data Analysis and Graphics

W. N. Venables, D. M. Smith, and the R Core Team

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

# Data permanency and removing objects (1)

- the entities that **R** creates and manipulates are known as **objects**
  - variables
  - arrays of numbers
  - character strings
  - functions
  - more general structures built from such components
- during an R session, objects are created and stored by name

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

## Data permanency and removing objects (2)

- The R command  
`> objects()`  
(alternatively, `ls()`) can be used to display the names of (most of) the objects which are currently stored within R.
- the collection of **objects** currently stored is called the **workspace**

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

# Types of R objects (1-1)

- a **vector** is an ordered collection of numerical, character, complex or logical objects.  
vectors are collection of atomic component or **modes** the same data type
- a **matrix** is a multidimensional collection of data entries of the same type.  
matrices have two dimensions.  
rownames and colnames

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

# Types of R objects (1-2)

- a **list** is an ordered collection of objects that can be of different **modes**  
different data types
- though a **data.frame** is a restricted **list** with class **data.frame**, it may be regarded as a **matrix** with columns that can be of different **modes**.

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

# Types of R objects (1-3)

- It is displayed in matrix form, rows by columns.  
(Its like an excel spreadsheet)
- A `data.frame` is a list of variables of the same number of rows with unique row names, given class `data.frame` if no variables are included, the row names determine the number of rows.

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)



## Types of R objects (2)

- A **factor** is a vector of **categorical variables**, it can be ordered or unordered.
- **array** an array in R can have one, two or more dimensions. useful to store multiple related **data.frame** (for example when I jack-knife or permute data). Note if there are insufficient objects to fill the array, R recycles (see below)

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

# Dataframe and class objects in Python

- By definition, a class is a code template for creating objects.
- This means that you can define a class that will create a certain object for you when this class has been instantiated.
- Then, the **DataFrame** is a type of **pandas object**.
- Therefore, you can say there's the **pandas DataFrame class**, that is code template that can create a DataFrame for you.
  - **pandas** is a fast, powerful, flexible and easy to use open source **data analysis and manipulation tool**, built on top of the Python programming language.

<https://365datascience.com/question/difference-between-dataframe-and-class-object>

# Classes in R language (1)

- **Classes** and **Objects** are basic concepts of Object-Oriented Programming that revolve around the real-life entities.
- Everything in R is an **object**.
- An **object** is simply a data structure that has some **methods** and **attributes**.
- A **class** is just a blueprint or a sketch of these **objects**.
  - represents the set of properties or **methods** that are common to all **objects** of one type.

<https://www.geeksforgeeks.org/classes-in-r-programming/>

## Classes in R language (2)

- Unlike most other programming languages, R has a three-class system.
  - S3 class
  - S4 class
  - Reference class

<https://www.geeksforgeeks.org/classes-in-r-programming/>

# S3 Class (1)

- **S3** is the simplest yet the most popular OOP system
- lacks formal definition and structure
- an object of this type can be created by just adding an **attribute** to it.

<https://www.geeksforgeeks.org/classes-in-r-programming/>

## S3 Class (2)

- in S3 systems, **methods** don't belong to the **class**.
- they belong to **generic functions**
- means that we can't create our own **methods** here, as we do in other programming languages like C++ or Java.
- but we can define what a **generic method** (for example print) does when applied to our objects.

<https://www.geeksforgeeks.org/classes-in-r-programming/>

# S4 Class (1)

- Programmers of other languages like C++, Java might find S3 to be very much different than their normal idea of classes
  - as it lacks the structure that classes are supposed to provide.
- **S4** is a slight improvement over **S3**
  - its objects have a proper definition
  - gives a proper structure to its objects.

<https://www.geeksforgeeks.org/classes-in-r-programming/>

## S4 Class (2)

- As shown in the above example,
  - `setClass()` is used to define a class and
  - `new()` is used to create the objects.
- The concept of methods in **S4** is similar to **S3**, i.e., they belong to\* generic functions\*.

<https://www.geeksforgeeks.org/classes-in-r-programming/>



# Reference Class

- **Reference Class** is an improvement over **S4 Class**.
- Here the **methods** belong to the **classes**.
- These are much similar to object-oriented classes of other languages.
- Defining a Reference class is similar to defining S4 classes.
- we use `setRefClass()` instead of `setClass()` and `"*fields*"` instead of `"*slots*"`.

<https://www.geeksforgeeks.org/classes-in-r-programming/>

# R Objects (1)

- Every programming language has its own **data types** to store values or any information so that the user can assign these **data types** to the variables and perform **operations** respectively.
- **Operations** are performed accordingly to the **data types**

<https://www.geeksforgeeks.org/r-objects/?ref=lbp>

## R Objects (2)

- These **data types** can be
  - character
  - integer
  - float
  - long
  - etc.
- Based on the **data type**,  
memory/storage is allocated to the variable.
  - for example, in C language
    - character variables are assigned with 1 byte of memory
    - integer variable with 2 or 4 bytes of memory
    - other data types have different memory allocation for them.

<https://www.geeksforgeeks.org/r-objects/?ref=lbp>

- Unlike other programming languages, variables are assigned to **objects** rather than **data types** in R programming.

<https://www.geeksforgeeks.org/r-objects/?ref=lbp>

# R Object Classes (1)

- R possesses a simple generic function mechanism which can be used for an object-oriented style of programming.
- Method dispatch takes place based on the class of the first argument to the generic function.

- Usage

```
class(x)
```

```
class(x) <- names
```

```
unclass(x)
```

```
inherits(x, name)
```

<https://www.math.ucla.edu/~anderson/rw1001/library/base/html/class.html>

# R Object Classes (2-1)

- an R **object** is a data object which has a **class attribute**
- a **class attribute** is a vector of character strings giving the names of the classes from which the object inherits

<https://www.math.ucla.edu/~anderson/rw1001/library/base/html/class.html>

## R Object Classes (2-2)

- when a generic function `fun` is applied to an **object** with **class attribute** `c("first", "second")`, the system searches for a function called `fun.first` and, if it finds it, applies it to the **object**.
- If no such function is found, a function called `fun.second` is tried.
- If no class name produces a suitable function, the function `fun.default` is used.

<https://www.math.ucla.edu/~anderson/rw1001/library/base/html/class.html>

# R Object Classes (3-1)

- The **function** `class` prints the vector of names of classes an object inherits from.
- correspondingly, `class <- names` sets the **classes** an object inherits from.

<https://www.math.ucla.edu/~anderson/rw1001/library/base/html/class.html>



## R Object Classes (3-2)

- `unclass(x)` returns (a copy of) its argument with its class information removed.
- `inherits(x, name)` indicates whether its first argument inherits from a class with name equal to its second argument

<https://www.math.ucla.edu/~anderson/rw1001/library/base/html/class.html>

# Attributes of R objects (1)

## ① Basic Attributes

- The most basic and fundamental properties of every objects is its **mode** and **length**
- these are intrinsic attributes of every object.  
Examples of **mode** are "logical", "numeric", "character", "list", "expression", "name/symbol" and "function".

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

# Attributes of R objects (2)

## ① Basic Attributes (continued)

---

character	a character string
numeric	a real number, which can be an integer or a double
integer	an integer
logical	a logical (true/false) value

---

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

# Attributes of R objects (3-1)

## 2 Other Attributes, dimension

Object	Modes
vector	numeric, character, complex or logical
matrix	numeric, character, complex or logical
list	numeric, character, complex, logical, function, expression, ...
data frame	numeric, character, complex or logical
factor	numeric or character
array	numeric, character, complex or logical

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

## Attributes of R objects (3-2)

- Whether object allows elements of different modes.
- For example all elements in a vector or array have to be of the same mode.
- Whereas a list can contain any type of object including a list.

[https://www.w3schools.com/statistics/statistics\\_statistical\\_inference.php](https://www.w3schools.com/statistics/statistics_statistical_inference.php)

# The S3 System (1)

- S3 refers to a class system built into R.
- The system governs how R handles objects of different classes.
- Certain R functions will look up an object's S3 class, and then behave differently in response.

<https://rstudio-education.github.io/hopr/s3.html#summary-7>

## The S3 System (2)

- Certain R functions will look up an object's S3 class, and then behave differently in response.
- The `print` function is like this.

When you print a numeric vector, `print` will display a number:

```
num <- 1000000000  
print(num)  
## 1000000000
```

- But if you give that number the S3 class `POSIXct` followed by `POSIXt`, `print` will display a time:

```
class(num) <- c("POSIXct", "POSIXt")  
print(num)  
## "2001-09-08 19:46:40 CST"
```

<https://rstudio-education.github.io/hopr/s3.html#summary-7>

# The S3 System (3)

- If you use objects with classes - and you do - you will run into R's S3 system.
- S3 behavior can seem odd at first, but is easy to predict once you are familiar with it.
- R's S3 system is built around three components:
  - attributes (especially the class attribute)
  - generic functions
  - methods

<https://rstudio-education.github.io/hopr/s3.html#summary-7>



# Attributes (1)

- In Attributes, you learned that many R objects come with attributes, pieces of extra information that are given a name and appended to the object.
- Attributes do not affect the values of the object, but stick to the object as a type of metadata that R can use to handle the object.
- For example, a data frame stores its row and column names as attributes.
- Data frames also store their class, "data.frame", as an attribute.

<https://rstudio-education.github.io/hopr/s3.html#summary-7>