PROBABILIDAD Y ESTADÍSTICA

Curso de R

Gustavo A. Colmenares
gcolmenares@yachaytech.edu.ec
gcolmena@gmail.com

4- Estructuras de datos

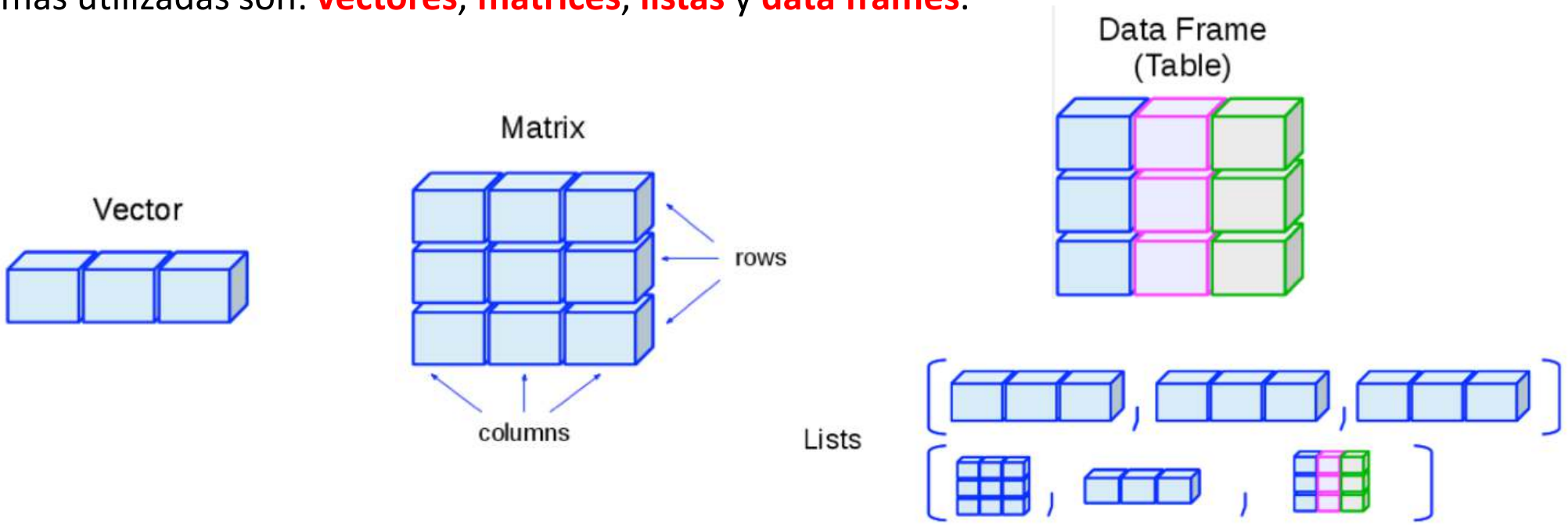
UNIVERSIDAD YACHAY TECH

SCHOOL OF MATHEMATICAL AND COMPUTATIONAL SCIENCES

Objetos en R

La información que manipulamos en **R** se estructura en forma de objetos. Un **objeto** es una estructura de datos que viene definido por una serie de **atributos**. Las **funciones** genéricas (como por ejemplo summary o plot) reconocen estos atributos y llevan a cabo distintos tipos de acciones en función del tipo de objeto.

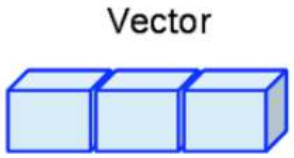
Las **estructuras de datos** que proporciona **R** permiten almacenar en un mismo objeto varios valores. Las más utilizadas son: **vectores**, **matrices**, **listas** y **data frames**.



Vectores

Un **vector** es la estructura de datos más sencilla en **R**. Un vector es una colección de uno o más datos del mismo tipo. Todos los vectores tienen:

- **Tipo**: Un vector tiene el mismo tipo que los datos que contiene. Si tenemos un vector que contiene datos de tipo numérico, el vector será también de tipo numérico. Los vectores son atómicos, pues sólo pueden contener datos de un sólo tipo, no es posible mezclar datos de tipos diferentes dentro de ellos.
- **Largo**: Es el número de elementos que contiene un vector. El largo es la única dimensión que tiene esta estructura de datos.



Como los vectores son la estructura de datos más sencilla de **R**, datos simples como el número 3, son en realidad vectores. En este caso, un vector de tipo numérico y largo igual a 1.

Vectores

Creamos vectores usando la función **c()** (combinar). Llamamos esta función y le damos como argumento los elementos que deseamos combinar en un vector, separados por comas.

```
# Vector numérico
c(1, 2, 3, 5, 8, 13)
```

```
# Vector de cadena de texto
c("arbol", "casa", "persona")
```

```
# Vector lógico
c(TRUE, TRUE, FALSE, FALSE, TRUE)
```

Una vez construido el vector se acostumbra a etiquetarlo con un nombre corto y representativo de la información que almacena.

Vectores

Existen algunas operaciones al aplicarlas a un vector, se aplican a cada uno de sus elementos. A este proceso le llamamos **vectorización**.

Las operaciones aritméticas y relacionales pueden vectorizarse. Si las aplicamos a un vector, la operación se realizará para cada uno de los elementos que contiene.

Por ejemplo, creamos un vector numérico.

```
mi_vector <- c(2, 3, 6, 7, 8, 10, 11)
```

Si aplicamos operaciones aritméticas, obtenemos un vector con un resultado por cada elemento.

```
# Operaciones aritméticas
mi_vector + 2

## [1]  4  5  8  9 10 12 13
```

Vectores

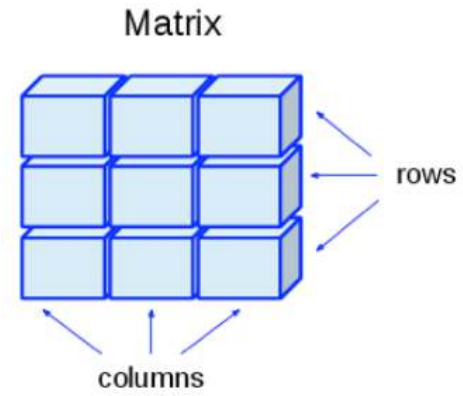
```
mi_vector * 2
```

```
## [1]  4  6 12 14 16 20 22
```

Esta manera de aplicar una operación es muy eficiente. Comparada con otros procedimientos, requiere de menos tiempo de cómputo, lo cual a veces es considerable, en particular cuando trabajamos con un número grande de datos.

Matrices

Las **matrices** son la extensión natural de los vectores a dos dimensiones: “largo”" y “alto”. Al igual que un vector, únicamente pueden contener datos de un sólo tipo. Como las matrices son usadas de manera regular en matemáticas y estadística, es una estructura de datos de uso común en **R**.



Las matrices se pueden crear concatenando vectores con las funciones **cbind()** o **rbind():**

```
x <- c(3, 7, 1, 8, 4)
y <- c(7, 5, 2, 1, 0)
cbind(x, y)  # por columnas

rbind(x, y)  # por filas
```

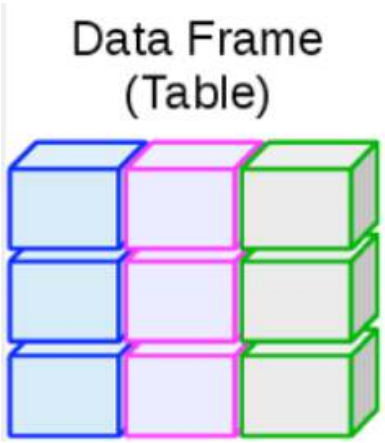
Creamos matrices en **R** con la función **matrix()**. La función **matrix()** acepta dos argumentos, **nrow** y **ncol**. Con ellos especificamos el número de filas y columnas que tendrá la matriz. Por defecto, los valores se colocan por columnas.

Data Frames

Los **data frames** son estructuras de datos de dos dimensiones (rectangulares) que pueden contener datos de diferentes tipos, por lo tanto, son heterogéneas. Esta estructura de datos es la más usada para realizar análisis de datos.

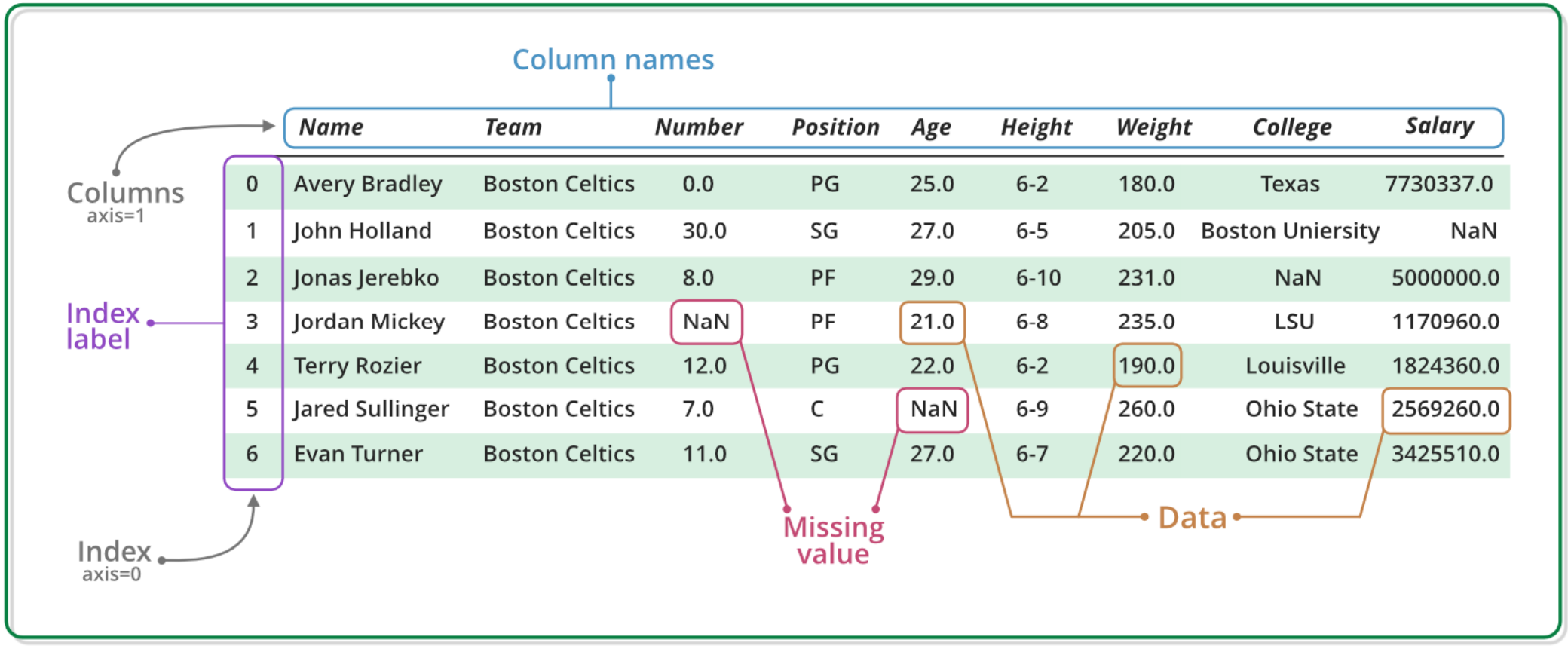
Podemos entender a los data frames como una versión más flexible de una matriz. Mientras que en una matriz todas las celdas deben contener datos del mismo tipo, los filas de un data frame admiten datos de distintos tipos, pero sus columnas conservan la restricción de contener datos de un sólo tipo.

En términos generales, las filas en un data frame representan casos, individuos u observaciones, mientras que las columnas representan atributos, rasgos o variables.



Data Frames

En la práctica son estructuras para trabajar con datos de diferentes tipos:



	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

Data Frames

Por ejemplo, así lucen los primeros cinco renglones del objeto iris, el famoso conjunto de datos Iris de Ronald Fisher, que está incluido en todas las instalaciones de **R**.

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

Los primeras cinco filas corresponden a cinco casos, en este caso flores. Las columnas son variables con los rasgos de cada flor: largo y ancho de sépalo, largo y ancho de pétalo, y especie.

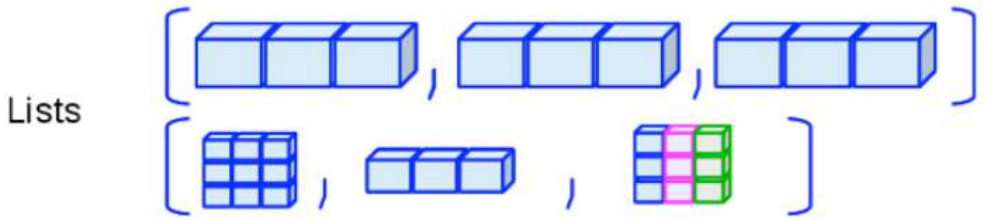
Para crear un data frame usamos la función **data.frame()**. Esta función nos pedirá un número de vectores igual al número de columnas que deseemos. Todos los vectores que proporcionemos deben tener el mismo largo.

Listas

Las **listas**, al igual que los vectores, son estructuras de datos unidimensionales, sólo tienen largo, pero a diferencia de los vectores cada uno de sus elementos puede ser de diferente tipo o incluso de diferente clase, por lo que son estructuras heterogéneas.

Podemos tener listas que contengan datos atómicos, vectores, matrices, data frames u otras listas. Esta última característica es la razón por la que una lista puede ser considerada un **vector recursivo**, pues es un objeto que puede contener objetos de su misma clase.

Para crear una lista usamos la función list(), que nos pedirá los elementos que deseamos incluir en nuestra lista. Para esta estructura, no importan las dimensiones o largo de los elementos que queramos incluir en ella.



UNIVERSIDAD YACHAY TECH

SCHOOL OF MATHEMATICAL AND COMPUTATIONAL SCIENCES

FIN

Gustavo A. Colmenares gcolmenares@yachaytech.edu.ec / gcolmena@gmail.com