

# Programación para el Análisis de Datos

Estructuras de datos en 

Daniel Fraiman

Maestría en Ciencia de Datos, Universidad de San Andrés

1 / 27

## Estructuras de datos

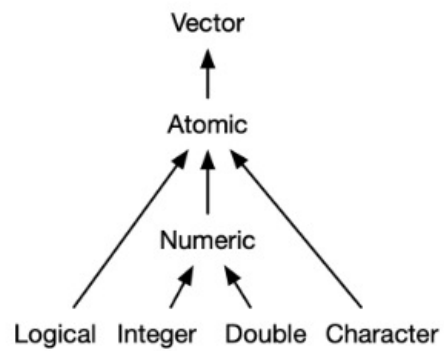
Homogeneas	Heterogeneas
Vector	Lista
Matriz	Data frame
Array	

2 / 27

# VECTORES

3 / 27

## Vectores



4 / 27

# Vectores

## Creación de vectores numéricos

- > x <-c(1,5,2,1,8) # c viene de concatenar
- > x <-c(1:9) # 1,2,3,4,5,6,7,8,9
- > x <-(1:9) # 1,2,3,4,5,6,7,8,9
- > x <-seq(1:9) # 1,2,3,4,5,6,7,8,9
- > x <-seq(3,10,2) # inicio, fin, paso, 3,5,7,9
- > x <-seq(1,10,length=3) # inicio, fin, cantidad, 1, 5.5, 10
- > x <- c(8L, 5L, 6L) # lo guarda como entero y no como doble precisión

5/27

# Vectores

## Coordenadas/posiciones del vector []

- > x[3] # valor que se encuentra en la posición 3 del vector
- > x[2:4] # valores que están en las posiciones del 2 al 4.
- > x[seq(3,10,2)] # valores que están en las posiciones 3,5,7,9.
- > x[-3] # vector sin la tercera coordenada

6/27

# Vectores

## Creación vectores de cadena de caracteres

- > x <-c("a","b","c") # no se usan esas comillas, son las rectas
- > x <-c("auto","casa","leon") # evitar tildes
- > x <-letters[2:6] # b,c,d,e,f,g (letters es un vector existente en R base)
- > x <-LETTERS[1:3] # A, B, C
- > x <-letters[seq(1,5,2)] # a,c,e

7/27

# Vectores

## Creación de vectores aleatorios numéricos y categóricos

- > x <-runif(100,0,1) # 100 variables uniformes [0,1] independientes
- > x <-rnorm(10,0,1) # 10 variables normales 0, 1
- > x <-sample(c(1:10),3) # obtenemos 3 valores distintos elegidos al azar de entre los números del 1 al 10.  
# si escribimos x <-sample(c(1:10),11) dará un error
- > x <- sample(c(1:5),12,replace=T) # obtenemos 12 valores elegidos al azar de entre los números del 1 al 10.  
# replace=T permite la repetición
- > x <- sample(letters[1:3],3,prob=c(0.1,0.2,0.7),replace=T) # podemos fijar la probabilidad de ser elegida cada coordenada.

8/27

## Vectores

### Creación vectores lógicos

- > `x <-c(TRUE,TRUE,FALSE)`
- > `x <-c("TRUE","TRUE","FALSE")` # esto no es un vector lógico

9/27

## Vectores

### Algunas funciones sobre vectores ()

- > `length(x)` # longitud del vector x
- > `sum(x)` # suma (solo para vectores numéricos)
- > `mean(x)` # promedio (solo para vectores numéricos)
- > `unique(x)` # dice cuales son los valores diferentes
- > `intersect(x,y)` # dice cuales son los elementos comunes en x e y
- > `union(x,y)` # cuales son todos los elementos (distintos) en x e y
- > `setdiff(x,z)` # todos los elementos que están solamente en x (y no en z)

10/27

# MATRICES

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

11/27

## Matrices

### Creación de matrices

- > `x <- matrix(nrow=2, ncol=2) ; x[1,1] <- -2 ; x[1,2] <- -3 ; x[2,1] <- -14 ; x[2,2] <- -6` # el primer índice es la fila el segundo la columna
- > `x <- matrix(nrow=2, ncol=2) ; x[,] <- c(2, 14, 3, 6)` # va complementando las columnas de a una
- > `x <- matrix(nrow=10, ncol=10); x[,] <- runif(100, 0, 1)`

### Coordenadas/posiciones de la matriz `[]`

- > `x[3,2]` # valor que se encuentra en la fila 3 y columna 2
- > `x[1,3:5]` # valores que están en la fila 1 y columna 3, 4 y 5.
- > `x[,3]` # valores que están en la columna 3
- > `x[1, ]` # valores que están en la fila 1

12/27

## Algunas operaciones/funciones sobre matrices

- > `dim(x)` # dimension (numero de filas y columnas)
- > `t(x)` # matriz transpuesta
- > `apply(x,1,funcion)` # aplica la función definida sobre las filas
- > `apply(x,2,funcion)` # aplica la función definida sobre las columnas
- > `x <- matrix(nrow=2,ncol=2) ; x[1,1]<-2 ; x[1,2]<-3; x[2,1]<-14; x[2,2]<-6 ; ¿apply(x,1,sum)?`
- > `x%*%y` # producto matricial

13 / 27

## ARRAYS

14 / 27

# Arrays

## Creación de arrays

- > `x <-array(dim=c(2,3,10))` # objeto de 2x3x10, son 10 matrices de 2 filas x 3 columnas.
- > `x <-array(dim=c(2,3,10,2))` # objeto de 2x3x10x2, son 2 conjuntos de 10 matrices de 2 filas x 3 columnas.
- > `x <-array(dim=c(2,3,10)) ; x[,,]<-runif(2*3*10,0,1)` # ¿cómo completa?

## Algunas funciones sobre arrays

- > `dim(x)` # dimension
- > `apply(x,1,funcion)` # aplica la función definida sobre [`*`, ,]
- > `apply(x,2,funcion)` # aplica la función definida sobre [, `*`, ]
- > `apply(x,3,funcion)` # aplica la función definida sobre [, ,`*` ]

15/27


# LISTAS

16/27



# Listas

## Listas

- Son las estructuras más flexibles en .
- En una lista podemos tener (simultáneamente) vectores, matrices, dataframe, y más listas.

## Creación de Listas

- > `x <-list(); x[[1]]<-c("a","b"); x[[2]]<-c(1:10)` # dentro de este objeto tenemos dos vectores de distinta longitud y tipo.
- > `x <-list(); x[[1]]<-c("a","b"); x[[2]]<-matriz`
- > `x <-list(); x[[1]]<-c("a","b"); x[[2]]<-matriz ;  
x[[3]]<-dataframe`
- > `x <- list(nombre=c("Juan","Maria"),  
casado=c(T,F),no.hijos=c(3,1), edad.hijos=list(c(4,7,9),2))`

17/27

# Listas

## Coordenadas/posiciones de la lista

- > `x[[1]]` # objeto que esta en 1.
- > `x$nombre; x$casado ; x$no.hijos ; x$edad.hijos[[1]]`

## Algunas funciones sobre listas

- > `length(x)` # cuantos objetos tiene la lista
- > `lapply(x,funcion)` # aplica la funcion sobre cada uno de los objetos de la lista
- > `lapply(x,length)`
- > `lapply(x,sum)` # para que funcione todos los elementos tienen que ser numerico

18/27

# DATA FRAME

19/27

## Data Frames

### Data Frames

- Son las planillas o tablas de datos.
- La cantidad de filas es la misma para cada columna (como en una matriz).
- La diferencia respecto de una matriz es que el data frame no es atómico. Podemos tener simultáneamente columnas numéricas y columnas de caracteres. Además puede haber más de un elemento en un lugar del data.frame.

### Creación de Data Frames

- > x<-data.frame(edad=c(22,21,34), corona=c("-", "-", "+"),  
sexo=c("H", "M", "H"))
- > x<-data.frame(edad=c(22,21,34), corona=c("-", "-", "+"),  
sexo=c("H", "M", "H"), hijos=I(list(1, c(2, 1), 3)))

20/27

# Data Frames

## Creación de Data Frames

```
> x<-data.frame(edad=c(22,21,34), corona=c("-", "-", "+"),  
sexo=c("H", "M", "H"), hijos=c(0,2,1),  
edad.hijos=I(list(NULL,c(1,3),3)))
```

21 / 27

## Algunos vectores con contenidos “raros”

- Factor
- Fecha

22 / 27

# Factor

## Factor

Una variable categórica (que sus valores son categorías) se puede representar en un vector con sus categorías. Como hicimos con vectores de caracteres. Muchas veces queremos explicitar que es un variable categórica y por lo tanto la convertimos a factor. Algunos métodos gráficos y de análisis necesitan que la variable esté cargada como factor.

## Un factor tiene:

- los valores que puede tomar la variable (independiente si fueron observados). **levels**
- los nombres reales de las categorías. **labels**

23 / 27

# Factores

## Creación de factores

- ```
> x<-  
  factor(c(1,1,1,2),levels=c(1,2),labels=c("Masculino","Femenino"))  
  
> x<-factor(c("B","A","A"),levels=c("A","B","C"),  
  labels=c("Gato","Perro","Hamster")) # notar que los levels  
  pueden ser más de los que observamos en los datos
```

24 / 27

## Fechas

### Fechas con librería *lubridate*

- > x <- ydm\_hms("2021-15-12 10:32:20") #  
year\_day\_month\_hour\_min\_sec
- > x <- ydm\_hms("21-15-12 10:32:20") #  
year\_day\_month\_hour\_min\_sec
- > x <- ydm\_hm("2021-22-12 10:45") # year\_day\_month\_hour\_min
- > x <- dmy\_h("22-11-2021 10") # day\_month\_year\_hour
- > x <- dmy\_h("22/11/2021 10")
- > x <- dmy\_h("22/11/2021 10PM")

### Algunas funciones sobre fechas

- > day(x) # 29 día del mes , hour(x), minute(x), year(x), date(x)
- > wday(x,label=T,abbr=F) # lunes o martes ... o domingo

25 / 27

## Resumen

### Comando class

- > class(x)
  - character: "a", "hola"
  - numeric: 2, 15.5
  - integer: 2L (the L tells R to store this as an integer)
  - logical: TRUE, FALSE
  - complex: 1+4i (número complejo, parte real e imaginaria)
  - factor: variable categórica
  - Date, POSIXct, POSIXt: Solo fecha, o fecha con hora y zona horaria

26 / 27

### ¿Cómo agregamos datos sobre estructuras existentes?

- > `x<-rbind(x,nueva_fila)` # tiene que ser consistente con lo anterior
- > `x<-cbind(x,nueva_columna)` # tiene que ser consistente con lo anterior