

## Tema 5 - Data frames

Juan Gabriel Gomila & María Santos

# Data frames

Data frame. Un data frame es una tabla de doble entrada, formada por variables en las columnas y observaciones de estas variables en las filas, de manera que cada fila contiene los valores de las variables para un mismo caso o un mismo individuo.

- ▶ `data()`: para abrir una ventana con la lista de los objetos de datos a los que tenemos acceso en la sesión actual de R (los que lleva la instalación básica de R y los que aportan los paquetes que tengamos cargados).
  - ▶ Si entramos `data(package=.packages(all.available = TRUE))` obtendremos la lista de todos los objetos de datos a los que tenemos acceso, incluyendo los de los paquetes que tengamos instalados, pero que no estén cargados en la sesión actual.

## Obteniendo información del data frame

- ▶ `head(d.f,n)`: para mostrar las  $n$  primeras filas del data frame. Por defecto se muestran las 6 primeras filas
- ▶ `tail(d.f,n)`: para mostrar las  $n$  últimas filas del data frame. Por defecto se muestran las 6 últimas
- ▶ `str(d.f)`: para conocer la estructura global de un data frame
- ▶ `names(d.f)`: para producir un vector con los nombres de las columnas

## Obteniendo información del data frame

```
str(Orange)
```

```
Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame'
 $ Tree          : Ord.factor w/ 5 levels "3"<"1"<"5"<"2"<"4"
 $ age           : num  118 484 664 1004 1231 ...
 $ circumference: num  30 58 87 115 120 142 145 33 69 111 ...
 - attr(*, "formula")=Class 'formula' language circumference ~ age
 .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "labels")=List of 2
 ..$ x: chr "Time since December 31, 1968"
 ..$ y: chr "Trunk circumference"
 - attr(*, "units")=List of 2
 ..$ x: chr "(days)"
 ..$ y: chr "(mm)"
```

## Obteniendo información del data frame

```
head(Orange,4)
```

	Tree	age	circumference
1	1	118	30
2	1	484	58
3	1	664	87
4	1	1004	115

```
tail(Orange,4)
```

	Tree	age	circumference
32	5	1004	125
33	5	1231	142
34	5	1372	174
35	5	1582	177

# Obteniendo información del data frame

- ▶ `rownames(d.f)`: para producir un vector con los identificadores de las filas
  - ▶ R entiende siempre que estos identificadores son palabras, aunque sean números, de ahí que los imprima entre comillas
- ▶ `colnames(d.f)`: para producir un vector con los identificadores de las columnas
- ▶ `dimnames(d.f)`: para producir una list formada por dos vectores (el de los identificadores de las filas y el de los nombres de las columnas)
- ▶ `nrow(d.f)`: para consultar el número de filas de un data frame
- ▶ `ncol(d.f)`: para consultar el número de columnas de un data frame
- ▶ `dim(d.f)`: para producir un vector con el número de filas y el de columnas

## Obteniendo información del data frame

- ▶ `d.f$nombre_variable`: para obtener una columna concreta de un dataframe
  - ▶ El resultado será un vector o un factor, según cómo esté definida la columna dentro del data frame
  - ▶ Las variables de un data frame son internas, no están definidas en el entorno global de trabajo de R

## Sub-data frames

- ▶  $d.f[n,m]$ : para extraer “trozos” del data frame por filas y columnas (funciona exactamente igual que en matrices) donde  $n$  y  $m$  pueden definirse como:
  - ▶ intervalos
  - ▶ condiciones
  - ▶ números naturales
  - ▶ no poner nada
  - ▶ Si sólo queremos definir la subtabla quedándonos con algunas variables, basta aplicar el nombre del data frame al vector de variables
  - ▶ Estas construcciones se pueden usar también para reordenar las filas o columnas



## Sub-data frames

```
data0range = 0range  
data0range[c(10:12),]
```

	Tree	age	circumference
10	2	664	111
11	2	1004	156
12	2	1231	172

```
data0range[c(2,17),c(1,3)]
```

	Tree	circumference
2	1	58
17	3	75

## Sub-data frames

```
data0range[2,3]
```

```
[1] 58
```

```
data0range[data0range$circumference<=50,]
```

	Tree	age	circumference
1	1	118	30
8	2	118	33
15	3	118	30
22	4	118	32
29	5	118	30
30	5	484	49

# Leyendo tablas de datos

- ▶ `read.table()`: para definir un data frame a partir de una tabla de datos contenida en un fichero
  - ▶ Este fichero puede estar guardado en nuestro ordenador o bien podemos conocer su url. Sea cual sea el caso, se aplica la función al nombre del fichero o a la dirección entre comillas

Aquí tenéis una lista de data frames para practicar

## Parámetros de read.table()

- ▶ `header = TRUE`: para indicar si la tabla que importamos tiene una primera fila con los nombres de las columnas. El valor por defecto es `FALSE`
- ▶ `col.names = c(...)`: para especificar el nombre de las columnas. No olvidéis que cada nombre debe ir entre comillas
- ▶ `sep`: para especificar las separaciones entre columnas en el fichero (si no es un espacio en blanco). Si es así, hay que introducir el parámetro pertinente entre comillas
- ▶ `dec`: para especificar el signo que separa la parte entera de la decimal (si no es un punto. Si es así, hay que introducir el parámetro pertinente entre comillas

## Parámetros de read.table()

```
path = "https://raw.githubusercontent.com/joanby/r-basic/main/data/students.csv"
students = read.table(path,
  col.names = c("technicalDisciplines", "aptitude", "maths",
    "generalKnowledge"))
head(students, 8)
```

	technicalDisciplines	aptitude	maths	language	generalKnowledge
1	1	23	50	59	
2	1	70	30	79	
3	1	25	50	86	
4	1	0	40	67	
5	1	0	25	70	
6	1	20	60	78	
7	1	97	65	75	
8	1	1	45	61	

## Más parámetros de `read.table()`

- ▶ `stringsAsFactors`: para prohibir la transformación de las columnas de palabras en factores debemos usar `stringsAsFactors=FALSE` (ya que por defecto, R realiza dicha transformación)
- ▶ Para importar un fichero de una página web segura (cuyo url empiece con `https`), no podemos entrar directamente la dirección en `read.table()`; una solución es instalar y cargar el paquete `RCurl` y entonces usar la instrucción `read.table(textConnection(getURL("url ")),...)`.

## Leyendo diferentes tipos de fichero

- ▶ `read.csv()`: para importar ficheros en formato CSV
- ▶ `read.xls()` o `read.xlsx()`: para importar hojas de cálculo tipo Excel u OpenOffice en formato XLS o XLSX, respectivamente. Se necesita el paquete `xlsx`
- ▶ `read.mtb()`: para importar tablas de datos Minitab. Se necesita el paquete `foreign`
- ▶ `read.spss()`: para importar tablas de datos SPSS. Se necesita el paquete `foreign`

# Exportando datos a ficheros

- ▶ `write.table(df, file = "")`: para exportar un data frame a un fichero
  - ▶ `file = ""`: es donde indicaremos el nombre que queremos darle al fichero
  - ▶ Podemos usar el parámetro `sep` para indicar el símbolo de separación de columnas. Siempre entre comillas
  - ▶ También podemos utilizar el parámetro `dec` para indicar la separación entre la parte entera y decimal de los datos



## Exportando datos a ficheros

```
write.table(students, file = "../data/StudentsData", dec =  
students2 = read.table("../data/StudentsData", header = TRUE,  
str(students2)
```

```
'data.frame':   82 obs. of  5 variables:  
 $ technicalDisciplines: int  1 1 1 1 1 1 1 1 1 1 ...  
 $ aptitude             : int  23 70 25 0 0 20 97 1 30 23 ...  
 $ maths                : int  50 30 50 40 25 60 65 45 45 40 ...  
 $ language             : int  59 79 86 67 70 78 75 61 65 64 ...  
 $ generalKnowledge     : int  10 18 22 16 20 22 22 22 16 18 ...
```

## Construyendo data frames

- ▶ `data.frame(vector_1, ..., vector_n)`: para construir un data frame a partir de vectores introducidos en el orden en el que queremos disponer las columnas de la tabla
  - ▶ R considera del mismo tipo de datos todas las entradas de una columna de un data frame
  - ▶ Las variables tomarán los nombres de los vectores. Estos nombres se pueden especificar en el argumento de `data.frame` entrando una construcción de la forma `nombre_variable = vector`
  - ▶ `rownames`: para especificar los identificadores de las filas
  - ▶ También en esta función podemos hacer uso del parámetro `stringsAsFactors` para evitar la transformación de las columnas de tipo palabra en factores

## Construyendo data frames

```
Algebra = c(1,2,0,5,4,6,7,5,5,8)
Analysis = c(3,3,2,7,9,5,6,8,5,6)
Statistics = c(4,5,4,8,8,9,6,7,9,10)
grades = data.frame(Alg = Algebra, An = Analysis, Stat = Statistics)
str(grades)
```

```
'data.frame':    10 obs. of  3 variables:
 $ Alg : num  1 2 0 5 4 6 7 5 5 8
 $ An  : num  3 3 2 7 9 5 6 8 5 6
 $ Stat: num  4 5 4 8 8 9 6 7 9 10
```

# Construyendo data frames

- ▶ `fix(d.f)`: para crear / editar un data frame con el editor de datos
- ▶ `names(d.f)`: para cambiar los nombres de las variables
- ▶ `rownames(d.f)`: para modificar los identificadores de las filas. Han de ser todos diferentes
- ▶ `dimnames(d.f)=list(vec_nom_fil, vec_nom_col)`: para modificar el nombre de las filas y de las columnas simultáneamente

# Construyendo data frames

- ▶ `d.f[núm_fila,] = c(...)`: para añadir una fila a un data frame
  - ▶ Las filas que añadimos de esta manera son vectores, y por tanto sus entradas han de ser todas del mismo tipo
  - ▶ Si no añadimos las filas inmediatamente siguientes a la última fila del data frame, los valores entre su última fila y las que añadimos quedarán no definidos y aparecerán como NA
  - ▶ Para evitar el problema anterior, vale más usar la función `rbind()` para concatenar el data frame con la nueva fila

## Construyendo data frames

```
Calculus = c(5,4,6,2,1,0,7,8,9,6)
grades2 = cbind(grades, Calculus)
head(grades2)
```

	Alg	An	Stat	Calculus
1	1	3	4	5
2	2	3	5	4
3	0	2	4	6
4	5	7	8	2
5	4	9	8	1
6	6	5	9	0

# Construyendo data frames

- ▶ `d.f$new_var`: para añadir una nueva variable al data frame
  - ▶ Podemos concatenar columnas con un data frame existente mediante la función `cbind()`. De este modo se puede añadir la columna directamente sin necesidad de convertirla antes a data frame
  - ▶ Esta nueva variable ha de tener la misma longitud que el resto de columnas del data frame original. Si no, se añadirán valores NA a las variables del data frame original o a la nueva variable hasta completar la misma longitud

# Cambiando los tipos de datos

- ▶ `as.character`: para transformar todos los datos de un objeto en palabras
- ▶ `as.integer`: para transformar todos los datos de un objeto a números enteros
- ▶ `as.numeric`: para transformar todos los datos de un objeto a números reales



## Más sobre sub-data frames

- ▶ `droplevels(d.f)`: para borrar los niveles sobrantes de todos los factores, ya que las columnas que son factores heredan en los sub-data frames todos los niveles del factor original, aunque no aparezcan en el trozo que hemos extraído
- ▶ `select(d.f, parámetros)`: para especificar que queremos extraer de un data frame
  - ▶ `starts_with("x")`: extrae del data frame las variables cuyo nombre empieza con la palabra "x"
  - ▶ `ends_with("x")`: extrae del data frame las variables cuyo nombre termina con la palabra "x"
  - ▶ `contains("x")`: extrae del data frame las variables cuyo nombre contiene la palabra "x"
  - ▶ Se necesita el paquete `dplyr` o mejor aún `tidyverse`

## Más sobre sub-data frames

- ▶ `subset(d.f, condición, select = columnas)`: para extraer del data frame las filas que cumplen la condición y las columnas especificadas
  - ▶ Si queremos todas las filas, no hay que especificar ninguna condición
  - ▶ Si queremos todas las columnas, no hace especificar el parámetro `select`
  - ▶ Las variables en la condición se especifican con su nombre, sin añadir antes el nombre del data frame

## Aplicando funciones a data frames

- ▶ `sapply(d.f, función)`: para aplicar una función a todas las columnas de un data frame en un solo paso
  - ▶ `na.rm=TRUE`: para evitar que el valor que devuelva la función para las columnas que contengan algún NA sea NA
- ▶ `aggregate(variables~factors,data=d.f,FUN=función)`: para aplicar una función a variables de un data frame clasificadas por los niveles de un, o más de un, factor
  - ▶ Si queremos aplicar la función a más de una variable, tenemos que agruparlas con un `cbind`
  - ▶ Si queremos separar las variables mediante más de un factor, tenemos que agruparlos con signos `+`

# Variables globales

- ▶ `attach(d.f)`: para hacer que R entienda sus variables como globales y que las podamos usar por su nombre, sin necesidad de añadir delante el nombre del data frame y el símbolo `$`
  - ▶ Si ya hubiera existido una variable definida con el mismo nombre que una variable del data frame al que aplicamos `attach`, hubiéramos obtenido un mensaje de error al ejecutar esta función y no se hubiera reescrito la variable global original
- ▶ `detach(d.f)`: para devolver la situación original, eliminando del entorno global las variables del data frame