CLASE 81 EST DESCRIPTIVA

VICTOR MIGUEL TERRON MACIAS

22/5/2021

MOSTRANDO DATOS DE DATAFRAMES EN R

Para colocar la instrucción se requiere:

DATAFRAME: Un dataframe es una tabla de doble entrada, formada pro variables en las coilumnas y observaciones de estas variables en als 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 sesion 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 en la sesión actual.

DATAFRAME DE IRIS

```
df=iris

head(df,10)#se utiliza para mostrar cierta cantidad (10) de filas del dataframe
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
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
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

1-10 of 10 rows

```
tail(df,10)#se utiliza para mostrar cierta cantidad (10) de filas del df pero del final
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
141	6.7	3.1	5.6	2.4	virginica
142	6.9	3.1	5.1	2.3	virginica
143	5.8	2.7	5.1	1.9	virginica
144	6.8	3.2	5.9	2.3	virginica
145	6.7	3.3	5.7	2.5	virginica
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica

1-10 of 10 rows

```
str(df)#para conocer la estructura global de un dataframe
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
names(df)#para producir un vector con los nombres de las columnas
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
rownames(df)#obtener un vector con los identificadores de las filas, R entiende
```

```
  [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"
 [13] "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24"
 [25] "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36"
 [37] "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"
 [49] "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"
 [61] "61"  "62"  "63"  "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"  "72"
 [73] "73"  "74"  "75"  "76"  "77"  "78"  "79"  "80"  "81"  "82"  "83"  "84"
 [85] "85"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"
 [97] "97"  "98"  "99"  "100" "101" "102" "103" "104" "105" "106" "107" "108"
[109] "109" "110" "111" "112" "113" "114" "115" "116" "117" "118" "119" "120"
[121] "121" "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
[133] "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143" "144"
[145] "145" "146" "147" "148" "149" "150"
```

```
#siempre que estos identificadores son palabras, aunque sean números, de ahí que
#los imprima entre comillas


colnames(df)#obtener un vector con los identificadores de als columnas
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
dimnames(df)#producir una lista formada por dos vectores (el de los
```

```
[[1]]
  [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"
 [13] "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24"
 [25] "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36"
 [37] "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"
 [49] "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"
 [61] "61"  "62"  "63"  "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"  "72"
 [73] "73"  "74"  "75"  "76"  "77"  "78"  "79"  "80"  "81"  "82"  "83"  "84"
 [85] "85"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"
 [97] "97"  "98"  "99"  "100" "101" "102" "103" "104" "105" "106" "107" "108"
[109] "109" "110" "111" "112" "113" "114" "115" "116" "117" "118" "119" "120"
[121] "121" "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
[133] "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143" "144"
[145] "145" "146" "147" "148" "149" "150"

[[2]]
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
#identificadores de las filas y el de los nombres de las columnas)


nrow(df)#consultar el numero de filas de un dataframe
```

```
[1] 150
```

```
ncol(df)#consultar el numero de columnas de un dataframe
```

```
[1] 5
```

```
dim(df)#obtener el vector con numero de filas y el numero de columnas
```

```
[1] 150   5
```

```
df$Species# obtener una columna concreta de un dataframe, el resultado será un
```

```
  [1] setosa     setosa     setosa     setosa     setosa     setosa
  [7] setosa     setosa     setosa     setosa     setosa     setosa
 [13] setosa     setosa     setosa     setosa     setosa     setosa
 [19] setosa     setosa     setosa     setosa     setosa     setosa
 [25] setosa     setosa     setosa     setosa     setosa     setosa
 [31] setosa     setosa     setosa     setosa     setosa     setosa
 [37] setosa     setosa     setosa     setosa     setosa     setosa
 [43] setosa     setosa     setosa     setosa     setosa     setosa
 [49] setosa     setosa     versicolor versicolor versicolor versicolor
 [55] versicolor versicolor versicolor versicolor versicolor versicolor
 [61] versicolor versicolor versicolor versicolor versicolor versicolor
 [67] versicolor versicolor versicolor versicolor versicolor versicolor
 [73] versicolor versicolor versicolor versicolor versicolor versicolor
 [79] versicolor versicolor versicolor versicolor versicolor versicolor
 [85] versicolor versicolor versicolor versicolor versicolor versicolor
 [91] versicolor versicolor versicolor versicolor versicolor versicolor
 [97] versicolor versicolor versicolor versicolor virginica  virginica
[103] virginica  virginica  virginica  virginica  virginica  virginica
[109] virginica  virginica  virginica  virginica  virginica  virginica
[115] virginica  virginica  virginica  virginica  virginica  virginica
[121] virginica  virginica  virginica  virginica  virginica  virginica
[127] virginica  virginica  virginica  virginica  virginica  virginica
[133] virginica  virginica  virginica  virginica  virginica  virginica
[139] virginica  virginica  virginica  virginica  virginica  virginica
[145] virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

```
#vector o un factor, según como esté definida la columna dentro del dataframe
#las var de un DF son internas, no están definidas en el entorno global
#de trabajo de R

df$Petal.Length[1:10]#obtener las primeras 10 posiciones
```

```
 [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5
```

```
#species obtiene un factor

df[1:10,1:5]#para extraer trozos del dataframe por filas y columnas donde se
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
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
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

1-10 of 10 rows

```
#Le tiene que pasar dos parametros que pueden ser intervalos, condicionales
#num naturales, nada

#significa numero de filas, numero de columnas
```

df=iris

head(df,10)#se utiliza para mostrar cierta cantidad (10) de filas del dataframe

tail(df,10)#se utiliza para mostrar cierta cantidad (10) de filas del df pero del final

str(df)#para conocer la estructura global de un dataframe

names(df)#para producir un vector con los nombres de las columnas

rownames(df)#obtener 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(df)#obtener un vector con los identificadores de als columnas

dimnames(df)#producir una lista formada por dos vectores (el de los #identificadores de las filas y el de los nombres de las columnas)

nrow(df)#consultar el numero de filas de un dataframe

ncol(df)#consultar el numero de columnas de un dataframe

dim(df)#obtener el vector con numero de filas y el numero de columnas

df$Species# obtener una columna concreta de un dataframe, el resultado será un #vector o un factor, según como esté definida la columna dentro del dataframe #las var de un DF son internas, no están definidas en el entorno global #de trabajo de R

df$Petal.Length[1:10]#obtener las primeras 10 posiciones #species obtiene un factor

df[1:10,10:20]#para extraer trozos del dataframe por filas y columnas donde se le tiene que pasar dos parametros que pueden ser intervalos, condicionales

ACCESOS A DATAFRAME

```
df[1:10,]
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
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
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

1-10 of 10 rows

```
df[1:15,2:5]
```

	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	3.5	1.4	0.2	setosa
2	3.0	1.4	0.2	setosa
3	3.2	1.3	0.2	setosa
4	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
6	3.9	1.7	0.4	setosa
7	3.4	1.4	0.3	setosa

	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
8	3.4	1.5	0.2	setosa
9	2.9	1.4	0.2	setosa
10	3.1	1.5	0.1	setosa

1-10 of 15 rows Previous **1** 2 Next

```
#PARA FILTRAR POR CONDICIONALES
df[df$Species=="setosa" & df$Sepal.Width>4,]#la condicion solo se aplica a
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
16	5.7	4.4	1.5	0.4	setosa
33	5.2	4.1	1.5	0.1	setosa
34	5.5	4.2	1.4	0.2	setosa

3 rows

```
#filas no a columnas, por ello las columnas no llevan un filtro
#se crea una sub tabla

#una vez tenemos una condicion booleana el resultado es un df

df[df$Species=="setosa" & df$Sepal.Width>4,][c(1,3),c(2,5)]
```

	Sepal.Width <dbl>	Species <fct>
16	4.4	setosa
34	4.2	setosa

2 rows

LEYENDO TABLAS DE DATOS

read.table() Para definir un dataframe a partir de una tabla de datos contenida en un fichero, el 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.

CARGA DE FICHEROS LOCALES

```
df=read.table("../Python R/cities.csv",header=TRUE,sep = ",")
head(df,10)
str(df)
```

_dec__ sirve para colocar el separador para numeros decimales, por ejemplo hay ocasiones donde se separa por comas, si queremos cambiar el nombre de las columnas tenemos lo siguiente:

col.names=c("Nombre_col1","Nombre_col_2","Nombre_col_3") esto dentro del parentesis principal del read table.

para separar por tabulador se coloca sep="

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. Se requiere cuando se coloca un dataframe que tiene un vector de palabras, es en este caso donde R puede interpretarlo como que se va a hacer la conversión automáticamente a los caracteres deben ser convertidos automáticamente a factores.

importacion de ficheros csv desde https, no se puede utilizar directamente la funcion read.table()

Para entrar a links de repositorios de https es necesario instalar el paquete de RCurl, install.packages("RCurl",dependencies=TRUE)

```
library(RCurl)
df3=read.table(textConnection(getURL("https://maitra.public.iastate.edu/stat501/datasets/olive.dat")), stringsAsFactors = FALSE,header = TRUE)
str(df3)
```

read.csv() es para importar ficheros en formato CSV

read.xls() o **read.slsx()** para importar hojas de calculo tipo excel u open office en formato XLS o XLSX, respectivamente. Se necesita el paquete xlsx

read.mtb() para importar tablas de datos minitab. Se encesita el paquete foreign

read.spss() para importar tablas de datos SPSS. Se necesita el paquete foreign

EXPORTAR DATAFRAME A UN FICHERO ARCHIVO

write.table(dataframe,file="nombrearchivo") se puede usar el sep para indicar el simbolo de separacion, y también se puede usar dec para indicar la separación entre la parte entera y decimal de los datos

```
write.table(df,file="Nombrearchivo.txt",dec=".")
students2=read.table("Nombrearchivo.txt",header=TRUE)
str(students2)
```

CONSTRUYENDO DATAFRAMES

Para construir un dataframe es necesatio utilizar la siguiente instrucción:

data.frame(vector1,vector2,vectorN) R considera del mismo tipo de datos todas las entradas de una columna de un dataframe Las variables tomarán los nombres de los vectores. Estos nombres se pueden especificar en el argumento de data.frame entrando a una construcción de la forma nombre_variable=vector

rownames para especificar los identificadores de las filas

Está también disponible el uso del parámetro stringsAsFactors para evitar la transformación de las columnas de tipo palabra en factores

```
NALG=c(1,2,0,5,4,6,7,5,5,8)
ANAL=c(3,3,2,7,9,5,6,8,5,6)
STAT=c(4,5,4,8,8,9,6,7,9,10)
GRAD=data.frame(ALGEBRA=NALG,ANALITICA=ANAL,ESTADISTICAS=STAT)
str(GRAD)
```

```
'data.frame':   10 obs. of  3 variables:
 $ ALGEBRA     : num  1 2 0 5 4 6 7 5 5 8
 $ ANALITICA   : num  3 3 2 7 9 5 6 8 5 6
 $ ESTADISTICAS: num  4 5 4 8 8 9 6 7 9 10
```

EJERCICIO DE CREACIÓN DE DATAFRAMES

```
gender=c("H","M","H","M","H")
age=c(23,45,20,30,18)
family=c(2,3,4,5,6)
df5=data.frame(Genero=gender,Edad=age,Familia=family,stringsAsFactors = TRUE)
row.names(df5)=c("C1","C2","C3","C4","C5")

str(df5)
```

```
'data.frame':   5 obs. of  3 variables:
 $ Genero : Factor w/ 2 levels "H","M": 1 2 1 2 1
 $ Edad   : num  23 45 20 30 18
 $ Familia: num  2 3 4 5 6
```

```
df5
```

	Genero <fct>	**Edad** <dbl>	**Familia** <dbl>
C1	H	23	2
C2	M	45	3
C3	H	20	4
C4	M	30	5
C5	H	18	6

5 rows

Construyendo data frames:

fix(DataFrame) para crear/editar un dataframe con el editor de datos

names(DataFrame) para cambiar los nombres de las variables

rownames(DatFrame) para modificar los identificadores de las filas, **TODOS DEBEN SER DIFERENTES**

dimnames(DataFrame)=list(vector_nombre_fill,vec_nom_col) para modificar el nommbre de las filas y de las columnas simultaneamente

```
dimnames(df5)=list(c("Antonio","Ricardo","Juan Gabriel","Victor","TERRON"),
                   c("Sexo","Años","MiembrosFam"))
df5
```

	Sexo <fct>	**Años** <dbl>	**MiembrosFam** <dbl>
Antonio	H	23	2
Ricardo	M	45	3
Juan Gabriel	H	20	4
Victor	M	30	5
TERRON	H	18	6

5 rows

Tambien es posivble acceder a una nueva fila pero para crearla:

dataframe[num_fila]=c()

```
df5=rbind(df5,c("H",30,1))
df5
```

	Sexo <fct>	**Años** <chr>	**MiembrosFam** <chr>
Antonio	H	23	2
Ricardo	M	45	3
Juan Gabriel	H	20	4
Victor	M	30	5
TERRON	H	18	6
6	H	30	1

6 rows

Para agregar un elementoa a un dataframe es correcto utilizar un rbind() para evitar la generacion de NA en caso de insertar en una columna que no corresponde a la siguiente sino a otra posición

Para agregar columnas se utiliza cbind(dataframeoriginal,lista a agregar)

```
GRAD
```

ALGEBRA <dbl>	ANALITICA <dbl>	ESTADISTICAS <dbl>
1	3	4
2	3	5
0	2	4
5	7	8
4	9	8
6	5	9
7	6	6
5	8	7
5	5	9
8	6	10

1-10 of 10 rows

```
calculus=c(1:10)
GRAD=cbind(GRAD,calculus)
head(GRAD)
```

	ALGEBRA <dbl>	ANALITICA <dbl>	ESTADISTICAS <dbl>	calculus <int>
1	1	3	4	1
2	2	3	5	2
3	0	2	4	3
4	5	7	8	4
5	4	9	8	5
6	6	5	9	6

6 rows

Se puede agregar otra columna de la siguiente forma:

```
df5$Ingresos=c(10000,15000,20000,30000,40000,50000)
df5
```

	Sexo <fct>	Años <chr>	MiembrosFam <chr>	Ingresos <dbl>
Antonio	H	23	2	10000
Ricardo	M	45	3	15000
Juan Gabriel	H	20	4	20000
Victor	M	30	5	30000
TERRON	H	18	6	40000
6	H	30	1	50000

6 rows

CAMBIANDO TIPOS DE DATOS EN UN DATAFRAME

Se utilizan los diferentes comandos:

as.character para transformar todos los datos de un objeto en palabras

as.integer para transformar todos los datos de un objeto a numeros enteros

as.numeric para transformar todos los datos de un objeto a numeros reales

as.factor para transformar los datos de un objeto a factores

MÁS SOBRE SUB-DATAFRAMES

droplevels(DataFrame) para borrar los niveles sobrantes de todos los factores, ya que las columnas que son factores heredan en los sub-dataframes todos los niveles del factor original, aunque no aparezcan en el trozo que hemos extraido

select(DataFrame,parámetros) para específicar que queremos extraer de un dataframe

starts_with("x") extrae del dataframe varias variables cuyo nombre empieza con la palabra "x"

ends_with("x") extrae del dataframe las variables cuyo nombre termina con la palabra "x"

contains("x") extrae del dataframe las variables cuyo nombre contenga la palabra "x"

NOTA: Se necesita el paquete **dplyr** o mejor aún **tidyverse**

```
df5[df5$Sexo=="M",]->df_mujeres
str(df_mujeres)
```

```
'data.frame':   2 obs. of  4 variables:
 $ Sexo       : Factor w/ 2 levels "H","M": 2 2
 $ Años       : chr  "45" "30"
 $ MiembrosFam: chr  "3" "5"
 $ Ingresos   : num  15000 30000
```

```
#SE ELIMINA EL NIVEL DADO QUE NO SE UTILIZA Y ES IUN SOBRANTE
df_mujeres=droplevels(df_mujeres)
str(df_mujeres)
```

```
'data.frame':   2 obs. of  4 variables:
 $ Sexo       : Factor w/ 1 level "M": 1 1
 $ Años       : chr  "45" "30"
 $ MiembrosFam: chr  "3" "5"
 $ Ingresos   : num  15000 30000
```

TIDIYVERSE

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
v ggplot2 3.3.3     v purrr   0.3.4
v tibble  3.1.2     v dplyr   1.0.6
v tidyr   1.1.3     v stringr 1.4.0
v readr   1.4.0     v forcats 0.5.1
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
iris_petal=select(iris,starts_with("s"))
head(iris_petal)
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Species <fct>
1	5.1	3.5	setosa
2	4.9	3.0	setosa
3	4.7	3.2	setosa
4	4.6	3.1	setosa
5	5.0	3.6	setosa
6	5.4	3.9	setosa

6 rows

```
iris_length=select(iris,ends_with("h"))
iris_length
```

Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4
4.6	3.4	1.4	0.3
5.0	3.4	1.5	0.2
4.4	2.9	1.4	0.2
4.9	3.1	1.5	0.1

1-10 of 150 rows Previous **1** 2 3 4 5 6 … 15 Next

Sintaxis de sub-set Sirve para extraer una determinada subtabla a partir de una determinada tabla

subset(DataFrame,condicion,select=columnas) para extraer del dataframe las filas que cumplen con la condición y las columnas específicadas.

Si queremos todas las filas no hay que especificar ninguna condicion.

Si queremos todas las columnas no hace falta especificar parámetro select

Las variables en la condición se especifican con su nombre, sin añadir antes el nombre del dataframe

```
subset(iris,Species=="versicolor",select=c(1:3))->versicolor#solo selecciona columna 1 a 3 del d
ataframe
rownames(versicolor)=1:nrow(versicolor)

head(versicolor,5)
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>
1	7.0	3.2	4.7
2	6.4	3.2	4.5
3	6.9	3.1	4.9
4	5.5	2.3	4.0

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>
5	6.5	2.8	4.6

5 rows

```
str(versicolor)
```

```
'data.frame':   50 obs. of  3 variables:
 $ Sepal.Length: num  7 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 ...
 $ Sepal.Width : num  3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7 ...
 $ Petal.Length: num  4.7 4.5 4.9 4 4.6 4.5 4.7 3.3 4.6 3.9 ...
```

```
setosa=droplevels(versicolor)
str(versicolor)
```

```
'data.frame':   50 obs. of  3 variables:
 $ Sepal.Length: num  7 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 ...
 $ Sepal.Width : num  3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7 ...
 $ Petal.Length: num  4.7 4.5 4.9 4 4.6 4.5 4.7 3.3 4.6 3.9 ...
```

APLICANDO OTRAS FUNCIONES A LOS VECTORES

sapply(DataFrame,funcion) para aplicar una funcion a todas las columnas de un dataframe en un solo paso, es sumamente importante el uso de na.rm dado que elimina datos tipo NA

aggregate(variables~factores,data=DataFrame,FUN=funcion) para aplicar una funcion a variables de un dataframe clasificadas por niveles de un o más de un factor. Si queremos aplicar la función a más de una variable tendremos que agruparlas por cbind(), Si queremos separar las variables mediante más de un factor tendremos que agruparlas con signos +

```
str(iris)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
sapply(subset(iris,select=1:4),mean)#sacar promedio
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
```

```
sapply(subset(iris,select = 1:4),sum)#sumar
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       876.5        458.6        563.7        179.9
```

```
funcion=function(x){sqrt(sum(x^2))}
sapply(iris[1:4],funcion)
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    72.27621     37.82063     50.82037     17.38764
```

```
df=data.frame(C1=c(1:5,NA),C2=c(5:9,NA))
sapply(df, mean)
```

```
C1 C2
NA NA
```

```
sapply(df, mean,na.rm=TRUE)
```

```
C1 C2
 3  7
```

```
str(iris)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
aggregate(cbind(Sepal.Length,Petal.Length)~Species,data=iris,FUN=mean,na.rm=TRUE)
```

Species <fct>	Sepal.Length <dbl>	Petal.Length <dbl>
setosa	5.006	1.462
versicolor	5.936	4.260
virginica	6.588	5.552

3 rows

CONVIRTIENDOD E COLUMNA DE NUMEROS A FACTORES

```
head(mtcars)
```

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <dbl>	am <dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0

6 rows | 1-10 of 12 columns

```
str(mtcars)
```

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
#Si tienes sospecha de que hay varibales que deberian ser factores
mtcars$cyl=as.factor(mtcars$cyl)
mtcars$gear=as.factor(mtcars$gear)
mtcars$carb=as.factor(mtcars$carb)
str(mtcars)
```

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
 $ carb: Factor w/ 6 levels "1","2","3","4",..: 4 4 1 1 2 1 4 2 2 4 ...
```

```
aggregate(mpg~cyl+gear+carb,data=mtcars,FUN=mean,na.rm=TRUE)
```

cyl <fct>	gear <fct>	carb <fct>	mpg <dbl>
4	3	1	21.50
6	3	1	19.75
4	4	1	29.10
8	3	2	17.15
4	4	2	24.75
4	5	2	28.20
8	3	3	16.30
8	3	4	12.62
6	4	4	19.75
8	5	4	15.80

1-10 of 12 rows Previous **1** 2 Next

```
#variable~factores siempre separados +
```

VARIABLES GLOBALES

attach(DataFrame) 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 dataframe y el simbolo $. Si ya hubiera existido una variable definida con el mismo nombre que una variable del dataframe al que aplicamos attach, hubieramos obtenido un mensaje de error al ejecutar attach y no se hubiera reescrito la variable global original

detach(DataFrame) para devolver la situación original, elkiminando del entorno global las variables del DataFrame

```
attach(mtcars)
#directamente puedo acceder a todas las variabels del DataFrame
mpg
```

EJERCICIO

importar el datasaet que se puede encobntrar en la siguiente url, debes responder a diferentes preguntas:

1. ¿Cuántos deportistas aparecen en el data frame?
2. ¿Cuántos han ganado medallas de oro, cuantas de plata?
3. ¿Cuántos de bronce?
4. ¿En cuantos lugares se han hecho olimpiadas de invierno?
5. ¿Cuántos hombres y cuantas mujeres hay?

```
dataej=read.csv("http://winterolympicsmedals.com/medals.csv")
str(dataej)
dataej=data.frame(dataej,stringsAsFactors = TRUE)
dataej$Medal=as.factor(dataej$Medal)
dataej$Year=as.factor(dataej$Year)
medoro=subset(dataej$Sport,dataej$Medal=="Gold")
medplata=subset(dataej$Sport,dataej$Medal=="Silver")
medbronze=subset(dataej$Sport,dataej$Medal=="Bronze")

table(dataej$NOC)
table(dataej$Year)

#medallas_oro=select(dataej,contains("Medal")
dataej[dataej$Medal=="Silver"]->medallas_silver
dataej[dataej$Medal=="Bronze"]->medallas_bronze
dataej=data.frame(Ano=Year,Ciudad=City,Deporte=Sport,Disciplina=Discipline,NOC=NOC,Event=Evento,
Genero_Evento=Event.gender,Medalla=Medal,stringsAsFactors = TRUE)
```

Solución:

```
DFEjercicio=read.csv("http://winterolympicsmedals.com/medals.csv")
#1. ¿Cuantos deposrtistas aparecen en el dataframe?
dim(DFEjercicio)
```

```
[1] 2311    8
```

```
#2. ¿Cuántos han ganado medallas de oro, cuantos de plata y cuantos de bronce?
count(DFEjercicio,Medal)
```

Medal	n
<chr>	<int>
Bronze	764

Medal <chr>	n <int>
Gold	774
Silver	773

3 rows

```
#3. ¿En cuantos lugares han hecho olimpiadas de invierno?
length(unique(as.factor(DFEjercicio$City)))
```

```
[1] 17
```

```
#4. ¿Cuántos hombres y cuantas mujeres hay?
count(DFEjercicio,Event.gender)
```

Event.gender <chr>	n <int>
M	1386
W	802
X	123

3 rows

```
#5. ¿En qué año participaron más deposrtistas?
max(DFEjercicio$Year)
```

```
[1] 2006
```

```
#6. El campo NOC indica el país ganador de la medalla, ¿Qué país puede presumir de haber ganado
 más medallas de oro en los juegos de invierno entre 1960 y 1996?
which.max(table(DFEjercicio$NOC))
```

```
NOR
 30
```