DataFrames 1

Dereck Amesquita

Marzo 2021

ZZZZ # Los primeros pasos Con "data()" Podremos ver los distintos datasets cargados en R. Con "data(package=.packages(all.available = TRUE))" podremos ver todos los datasets a nuestra dispocision, los cuales podemos descargar.

```
data()

data(package=.packages(all.available = TRUE))
```

##Trabajando con iris

```
df= iris
#df nos mostraria toda la tabla completa, le ponemos el numeral para anularlo
```

Podemos remover la variable creada con "remove"

```
remove(df)
```

Pero para seguir trabajando volveremos a añadir el dataframe. Head nos proporciona los 5 primeros elementos y tail nos muestra los 5 ultimos, Esto solo si los dejamos predeterminados, ya que si queremos podemos hacer que solo muestren el numero de elementos que queramos. Names nos da el nombre de las variables dentro del dataframe. Str nos brinda la informacion mas importante.

```
df= iris
head(df,3)
```

```
##   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
```

```
tail(df)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 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
```

```
names(df)
```

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

```
str(df)
```

```
## '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 ...
```

Extraer informacion del dataframe

- `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

```
rownames(df)
```

```
##   [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"
```

```
colnames(df)
```

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

```
dimnames(df)
```

```
## [[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"
```

```
nrow(df)
```

```
## [1] 150
```

```
ncol(df)
```

```
## [1] 5
```

```
dim(df)
```

```
## [1] 150   5
```

Con el simbolo del dolar "$" podemos especificar una variable del dataframe

Podemos restringir columnas de lo que un dataframe nos mostrara. En el ejemplo vemos que pedimos las filas del 1 al 10 y las columnas de la 2 a la 4. Tambien podemos ser mas especificos con los que mostrara. En la segunda funcion le ponemos coma despues del 4 para especificar que queremos todas las columnas. Los que nos arroja es todos los elementos que sean de la especie setosa mayores a 4 en sepal..

```
df[1:10,2:4]
```

```
##    Sepal.Width Petal.Length Petal.Width
## 1          3.5          1.4         0.2
## 2          3.0          1.4         0.2
## 3          3.2          1.3         0.2
## 4          3.1          1.5         0.2
## 5          3.6          1.4         0.2
## 6          3.9          1.7         0.4
## 7          3.4          1.4         0.3
## 8          3.4          1.5         0.2
## 9          2.9          1.4         0.2
## 10         3.1          1.5         0.1
```

```
df[df$Species=="setosa" & df$Sepal.Length >5,]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 11          5.4         3.7          1.5         0.2  setosa
## 15          5.8         4.0          1.2         0.2  setosa
## 16          5.7         4.4          1.5         0.4  setosa
## 17          5.4         3.9          1.3         0.4  setosa
## 18          5.1         3.5          1.4         0.3  setosa
## 19          5.7         3.8          1.7         0.3  setosa
## 20          5.1         3.8          1.5         0.3  setosa
## 21          5.4         3.4          1.7         0.2  setosa
## 22          5.1         3.7          1.5         0.4  setosa
## 24          5.1         3.3          1.7         0.5  setosa
## 28          5.2         3.5          1.5         0.2  setosa
## 29          5.2         3.4          1.4         0.2  setosa
## 32          5.4         3.4          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
## 37          5.5         3.5          1.3         0.2  setosa
## 40          5.1         3.4          1.5         0.2  setosa
## 45          5.1         3.8          1.9         0.4  setosa
## 47          5.1         3.8          1.6         0.2  setosa
## 49          5.3         3.7          1.5         0.2  setosa
```

Importar datos desde una nube

Debemos procurar que los datos se encuentren en su estado baso. En el caso de github debemos ir a la parte de raw, que es donde nos muestra solo los datos. Con str podremos consultar como se encuentra nuestra data. Con sep podremos decirle a R como se encuentra separado nuestro dataframe, con header le indicamos si tiene cabecera o no.

```
de=read.csv("https://raw.githubusercontent.com/dereckamesquita/Introduccion-a-R/main/proyecto%20real/ag
head(de,3)
```

```
##      ï..Date once tres_diez uno_tres total Organic.traffic dominios pages
## 1 23/07/2020    6         0        0     6        0.081262        1     1
## 2 24/07/2020    8         0        0     8        0.081266        1     1
## 3 25/07/2020   12         0        0    12        0.214237        1     1
```

```
str(de)
```

```
## 'data.frame':    165 obs. of  8 variables:
##  $ ï..Date        : chr  "23/07/2020" "24/07/2020" "25/07/2020" "26/07/2020" ...
##  $ once           : int  6 8 12 13 19 25 61 217 282 300 ...
##  $ tres_diez      : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ uno_tres       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total          : int  6 8 12 13 19 25 61 217 283 301 ...
##  $ Organic.traffic: num  0.0813 0.0813 0.2142 0.2142 0.2144 ...
##  $ dominios       : int  1 1 1 2 2 2 2 2 2 2 ...
##  $ pages          : int  1 1 1 2 2 2 2 2 2 2 ...
```

Guardar un dataframe

Supongamos que haremos un cambio al nombre de una columna. En la funcion de colnames le estamos diciendo a R que se meta al dataframe y que nos busque el lugar donde colnames es igual a total. Es decir: [colnames(de)=="total] nos arrojara un numero.

```
colnames(de)[colnames(de)=="total"]="Keywords"
colnames(de)
```

```
## [1] "ï..Date"         "once"            "tres_diez"       "uno_tres"
## [5] "Keywords"        "Organic.traffic" "dominios"        "pages"
```

```
write.csv(de, file = "../datasets/nuevadata.csv", sep=",")
```

```
## Warning in write.csv(de, file = "../datasets/nuevadata.csv", sep = ","): attempt
## to set 'sep' ignored
```

```
#Importaremos la nueva data
nueva=read.csv("../datasets/nuevadata.csv", sep = ",")
head(nueva,3)
```

```
##   X    ï..Date once tres_diez uno_tres Keywords Organic.traffic dominios pages
## 1 1 23/07/2020    6         0        0        6        0.081262        1     1
## 2 2 24/07/2020    8         0        0        8        0.081266        1     1
## 3 3 25/07/2020   12         0        0       12        0.214237        1     1
```

```
str(nueva)
```

```
## 'data.frame':    165 obs. of  9 variables:
##  $ X              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ï..Date        : chr  "23/07/2020" "24/07/2020" "25/07/2020" "26/07/2020" ...
##  $ once           : int  6 8 12 13 19 25 61 217 282 300 ...
##  $ tres_diez      : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ uno_tres       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Keywords       : int  6 8 12 13 19 25 61 217 283 301 ...
##  $ Organic.traffic: num  0.0813 0.0813 0.2142 0.2142 0.2144 ...
##  $ dominios       : int  1 1 1 2 2 2 2 2 2 2 ...
##  $ pages          : int  1 1 1 2 2 2 2 2 2 2 ...
```

Armar tus propios dataframes

Podemos armar un dataframe a partir de la union de vectores.

```
Peru=c(10,15,20)
Argentina=c(4,48,20)
Chile=c(5,15,17)
paises=data.frame(per=Peru, arg=Argentina, chi=Chile,  stringsAsFactors = default.stringsAsFactors())
head(paises)
```

```
##   per arg chi
## 1  10   4   5
## 2  15  48  15
## 3  20  20  17
```

```
#modificacion parcial de dataframe
npais=paises$chi/2
npais
```

```
## [1] 2.5 7.5 8.5
```

Modificaremos los names del dataframe, podemos cambiar el nombre de las columnas he incluso el de las filas, que inicialmente tiene un valor numerico creciente el cual se puede consultar a traves de rownames

```
#cambiando nombre de columbas
nombres=c("peruanos","argentinos","chilenos")
colnames(paises)=nombres
paises
```

```
##   peruanos argentinos chilenos
## 1       10          4        5
## 2       15         48       15
## 3       20         20       17
```

```
# Le daremos nombre a las filas
sector=c("salud","economia","justicia")
rownames(paises)=sector
paises
```

```
##          peruanos argentinos chilenos
## salud          10          4        5
## economia       15         48       15
## justicia       20         20       17
```

```
# con fix podemos cambiar segun nosotros queramos
#fix(paises)
paises[2:3,]
```

```
##          peruanos argentinos chilenos
## economia       15         48       15
## justicia       20         20       17
```

Mostrar columnas o mostrar filas

Para mostrar columnas es mucho mas sencillo, solo debemos pedirle la ubicacion [a:b] desde a hasta b o simplmente [a] Para mostrar las filas deberemos usar [a:b,] solo debemos añadir una coma para hacerselo saber a R.

```r
#Mostrar filas
paises[1:3]
```

```
##          peruanos argentinos chilenos
## salud          10          4        5
## economia       15         48       15
## justicia       20         20       17
```

```r
#Mostrar columnas
paises[2:3,]
```

```
##          peruanos argentinos chilenos
## economia       15         48       15
## justicia       20         20       17
```

Agregar columnas y filas

Columnas Agregar nueva columna, simplemente lo hacemos con $ y lo renombramos

```r
paises$new=c(2,5,3)
colnames(paises)[colnames(paises)=="new"]="bolivianos"
paises
```

```
##          peruanos argentinos chilenos bolivianos
## salud          10          4        5          2
## economia       15         48       15          5
## justicia       20         20       17          3
```

Agregar un nueva fila Comenzamos creando un nuevo dataframe

```r
nue=data.frame(4,5,6)
rownames(nue)="tecnologia"
colnames(nue)=nombres
#Hasta aqui bastaria, pero recordemos que hemos agregado una columna
nue$bolivianos=5

#Ahora usamos rbin para añadir nuestro dataframe. Podemos ver que el df es similar
paises <- rbind(paises,nue)
paises
```

```
##            peruanos argentinos chilenos bolivianos
## salud            10          4        5          2
## economia         15         48       15          5
## justicia         20         20       17          3
## tecnologia        4          5        6          5
```

Otra forma de añadir filas

```r
nuev=data.frame(peruanos=4,argentinos=6,chilenos=5,bolivianos=4)
paises <- rbind(paises,nuev)
paises
```

```
##            peruanos argentinos chilenos bolivianos
## salud            10          4        5          2
## economia         15         48       15          5
## justicia         20         20       17          3
## tecnologia        4          5        6          5
## 1                 4          6        5          4
```

```r
#Solo tenemos que modificar el nombre
```

Eliminar filas y columnas

Bastara con delimitar el dataframe y guardarlo en si mismo. df[a:b,c:d] Nos indica que tomara las filas desde a hasta b, considerando las columnas c hasta d. al hacer df=df[a:b,c:d] estaremos sobreescribiendo nuestro dataframe. Ademas podemos transponer el dataframe con t

```r
#Eliminaremos la ultima fila que agregamos
paises=paises[1:4,]
t(paises)
```

```
##            salud economia justicia tecnologia
## peruanos      10       15       20          4
## argentinos     4       48       20          5
## chilenos       5       15       17          6
## bolivianos     2        5        3          5
```

Filtrado de datos

Resulta que nos han pedido identificar aquellos paises con una economia mayor a 10. Comenzamos transponiendo nuestro dataframe Creamos una variable para las economias grandes (ecogra)

```r
paises=t(paises)
paises
```

```
##            salud economia justicia tecnologia
## peruanos      10       15       20          4
## argentinos     4       48       20          5
## chilenos       5       15       17          6
## bolivianos     2        5        3          5
```

```r
#corregiremos nuestro dataframe
paises=data.frame(paises,  stringsAsFactors = default.stringsAsFactors())
str(paises)
```

```
## 'data.frame':    4 obs. of  4 variables:
##  $ salud     : num  10 4 5 2
##  $ economia  : num  15 48 15 5
##  $ justicia  : num  20 20 17 3
##  $ tecnologia: num  4 5 6 5
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## 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()
```

```
ecogra=filter(paises, economia>5)
ecogra
```

```
##            salud economia justicia tecnologia
## peruanos      10       15       20          4
## argentinos     4       48       20          5
## chilenos       5       15       17          6
```

Pipe se demuestra como %>%, hare que todo lo de la izquierda se meta dentro de la funcion de su derecha.

```
x=2:5
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.75    3.50    3.50    4.25    5.00
```

```
#Es lo mismo que:
x %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.75    3.50    3.50    4.25    5.00
```

Los subests

Tener un subsets nos ayudara a trabajar con datos especificos. con select = 0 le estamos indicando que no queremos ninguna columna. Esto tambien lo podemos hacer con filter

```
subset(paises, economia>5, select = c(0) )
```

```
## data frame with 0 columns and 3 rows
```

```
subset(paises, economia>5, select = c(economia) )
```

```
##            economia
## peruanos         15
## argentinos       48
## chilenos         15
```

```
subset(paises, economia>5, select = c(1:4) )
```

```
##            salud economia justicia tecnologia
## peruanos      10       15       20          4
## argentinos     4       48       20          5
## chilenos       5       15       17          6
```

Funciones en dataframe

Aplicar funciones a un dataframe es muy util en Data Science, pero en R no podemos usar bucles. Por este inconveniente usaremos la funcion sapply Supongamos que queremos calcular cierta suma dentro de un subset de paises. Comenzamos definiendo la funcion. ### La funcion sapply Le diremos a sapply que tome todas las filas y columnas

```
g = function(x){x*2/3}
sapply(paises[,],g)
```

```
##         salud  economia justicia tecnologia
## [1,] 6.666667 10.000000 13.33333   2.666667
## [2,] 2.666667 32.000000 13.33333   3.333333
## [3,] 3.333333 10.000000 11.33333   4.000000
## [4,] 1.333333  3.333333  2.00000   3.333333
```

```
#Tambien podemos obtener la media

sapply(paises[,], mean)
```

```
##      salud   economia   justicia tecnologia
##       5.25      20.75      15.00       5.00
```

La funcion mutate

Mutate nos permitira crear nuevas columnas con ciertas funciones. Crearemos el promedio de salud y economia en una variable llamada promse.

```
paises %>%
  mutate(promse=(salud+economia)/2)
```

```
##            salud economia justicia tecnologia promse
## peruanos      10       15       20          4   12.5
## argentinos     4       48       20          5   26.0
## chilenos       5       15       17          6   10.0
## bolivianos     2        5        3          5    3.5
```

```
# tambien se puede por:
mutate(paises, dar=salud+economia)
```

```
##            salud economia justicia tecnologia dar
## peruanos      10       15       20          4  25
## argentinos     4       48       20          5  52
## chilenos       5       15       17          6  20
## bolivianos     2        5        3          5   7
```

```
paises= paises[,1:4]
paises
```

```
##            salud economia justicia tecnologia
## peruanos      10       15       20          4
## argentinos     4       48       20          5
## chilenos       5       15       17          6
## bolivianos     2        5        3          5
```

La funcion attach y detach

Podemos hacer que R entienda cuales son las variables globales. Es poco eficiente escribiri "paises$economia" en su lugar con attach podemos hacerlo mas simple.

```
#economia
#ese codigo no sirve
attach(paises)
#economia
detach(paises)
#economia
```