R 2019

01/04/2019

REPASO DE LA CLASE ANTERIOR

LECTURA/ESCRITURA

"read_csv()" lee un archivo delimitado por comas.

```
read_csv("/mi/path/al/directorio/mtcars.csv")
read_csv("https://github.com/tidyverse/readr/raw/master/inst/extdata/mtcars.csv")
```

Se pueden controlar cosas como col_types, col_names, delim, skip, n_max, y otros.

Existe una "write_csv" equivalente.

PIPING CON *MAGRITTR* Y OPERADOR %>%

```
x %>% f en vez de f(x) # el valor de x se direcciona a f()
```

```
x %>% f(., arg2, arg3) # por default pasa al 1er argumento
x %>% f(arg2, arg3)    # por lo tanto, puede obviarse el punto
x %>% f(arg1, ., arg3) # puede usarse el punto para direccionar a otro lado

mean_by_state %>% # mi data.frame
mutate(frac = mean/overall_mean) %>% # creo la columna frac
filter(frac >= 1) %>% # filtro la columna frac
arrange(desc(frac)) # ordeno de mayor a menor
```

COMBINANDO DATA.FRAMES (SIN LLAVES)

x1	x2
A	1
B	2
C	3
B	2
C	3
D	4

x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4

dplyr::**bind_rows(y, z)**

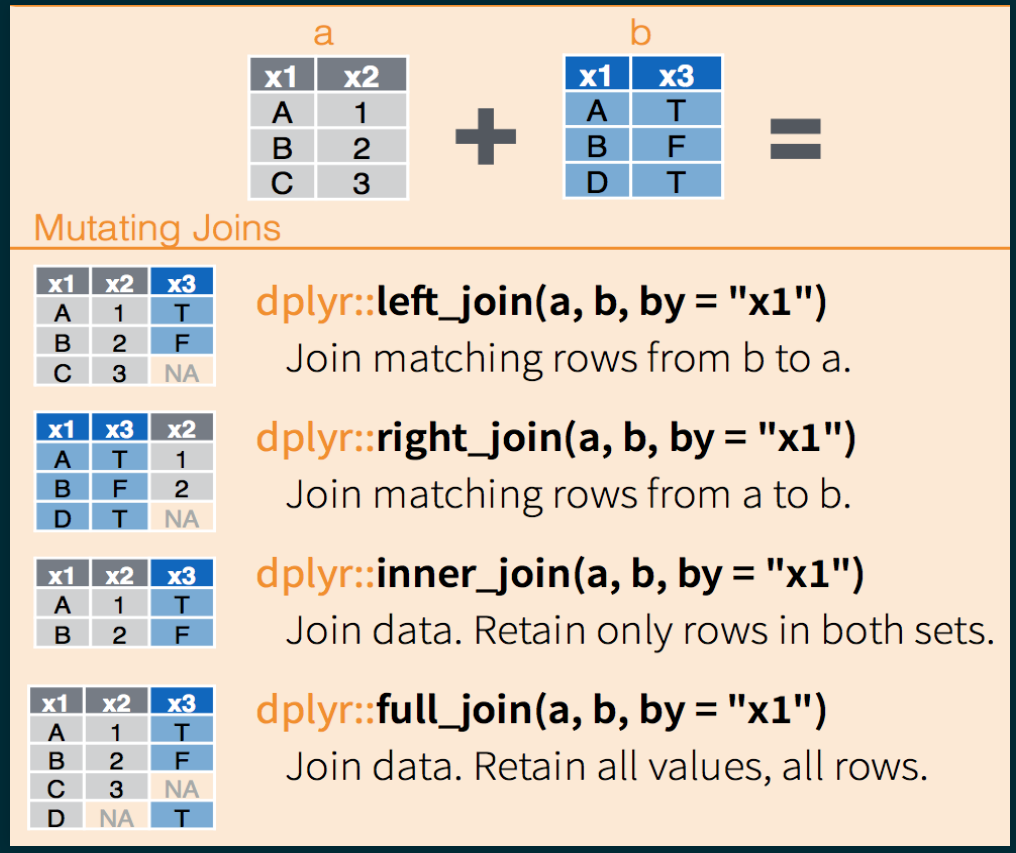
Append z to y as new rows.

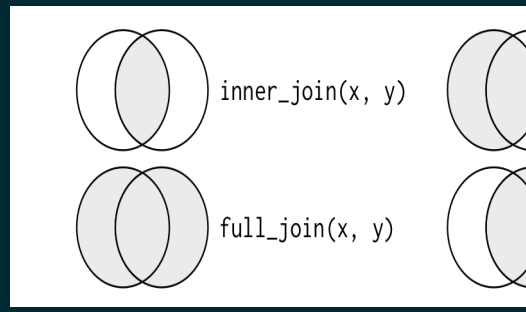
dplyr::**bind_cols(y, z)**

Append z to y as new columns.

Caution: matches rows by position.

COMBINANDO DATA.FRAMES (CON LLAVES)





- recuerden que lo tienen el *cheatsheet* de `dplyr`

RSTUDIO

- Working directory / Workspace

```
getwd() # para saber a qué directorio apunta mi sesión
setwd("path/to/dir") # para definir a qué directorio quiero apuntar
```

- Proyectos: Conjunto de archivos que van juntos: miproyecto.Rproj, .RData, .Rsession, etc.
- ¿Cómo grabo lo que hago?
    1. Abrir un proyecto nuevo y grabar después
    2. Grabar en algún directorio y después decirle que haga un proyecto ahí
    3. Sin proyectos, gestionar todo uno mismo (no muy recomendado)

'TYPES': TIPOS DE DATOS

- 'logical': TRUE o FALSE (o NA)
- 'integer': enteros: ..., -1, 0, 1, ...
- 'double': irracionales: 3.1415926
- 'character': alfanuméricos: "pi"
- 'complex': complejos: 1+i10
- 'raw': charToRaw("buenas") 62 75 65 6e 61 73

`CLASS()`, `MODE()` Y `TYPEOF()`

Para preguntar qué tipo es.

`mode` y `typeof` devuelven los tipos más básicos.

`class` suele devolver (si existe la clase) la estructura del dato.

LOGICAL

```r
v <- TRUE
print(class(v))
#> [1] "logical"
is.logical(v)
#> [1] TRUE
c(TRUE, TRUE, FALSE, NA)
#> [1]  TRUE  TRUE FALSE    NA
20/5 == 4
#> [1] TRUE
1:10 %% 3 == 0 #: para generar una secuencia, %% es el operador módulo (hagan ?: y ?%%)
#>  [1] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
```

INTEGER

```
v <- 2L
print(class(v))
is.integer(v)
is.numeric(v)
#> [1] TRUE
typeof(1)
#> [1] "double"
typeof(1L)
#> [1] "integer"
1.5L
#> [1] 1.5
```

Valor especial: NA

DOUBLE

```
v <- 23.5
print(class(v))
is.double(v)
is.numeric(v)

x <- sqrt(2) ^ 2
x
#> [1] 2
x - 2
#> [1] 4.44e-16
c(-1, 0, 1) / 0
#> [1] -Inf  NaN  Inf
```

Los 'double' son siempre aproximaciones!

Valores especiales: NA, NaN, Inf y -Inf

Usar: is.finite(), is.infinite(), is.na(), is.nan()

CHARACTER

```
v <- "TRUE" # v <- "Maldad pura"
print(class(v))
is.character(v)

x <- "Las cadenas de caracteres pueden tener una longitud arbitrariamente larga mal que
```

COMPLEX Y RAW

Para números complejos, complex

```
v <- 2+5i
print(class(v))
is.complex()
```

Para trabajar en bytes, se puede usar raw

```
v <- charToRaw("Hello")
print(class(v))
is.raw()
```

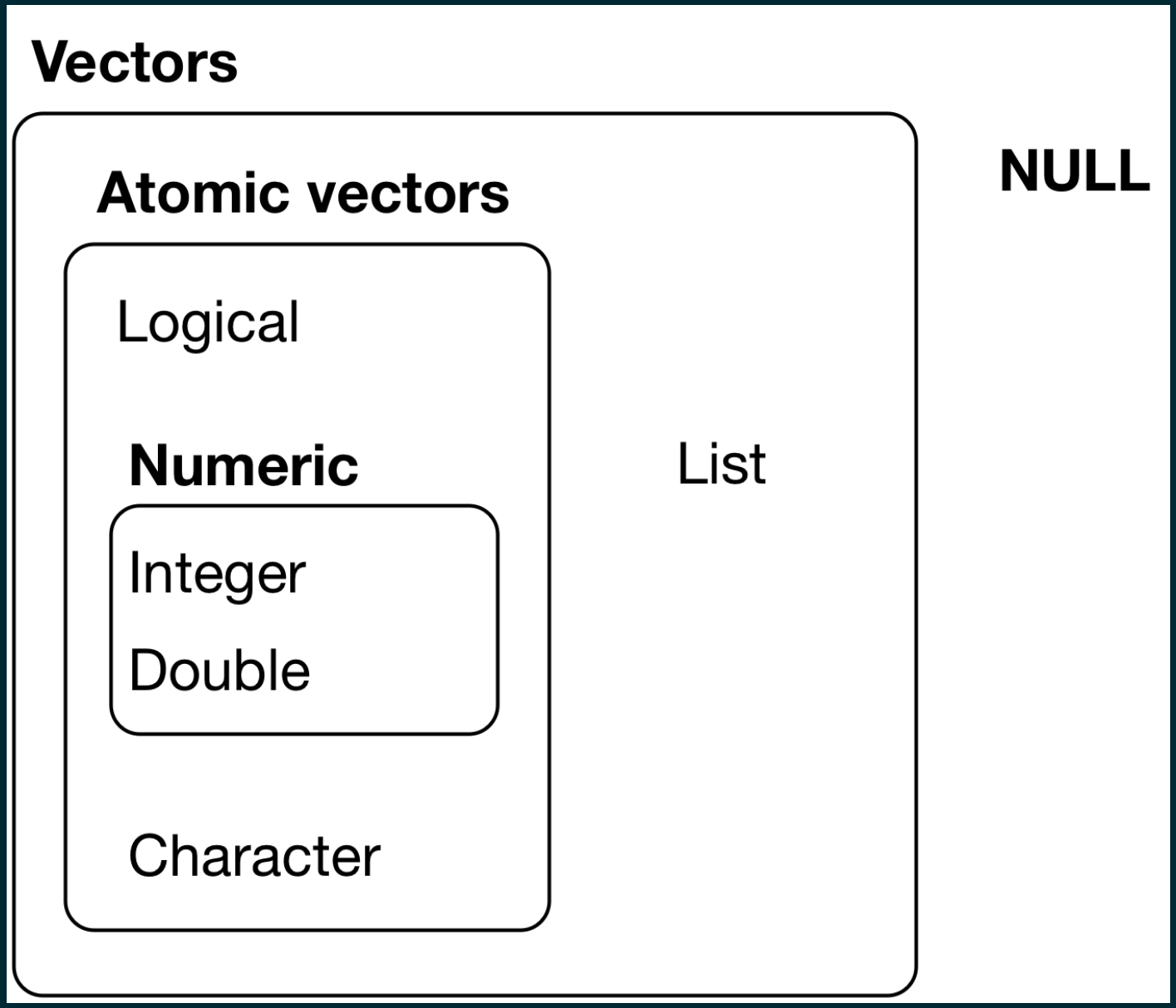
OBJETOS R: VECTORES

Existen dos clases de vectores:

1. Vectores 'atómicos' (*atomic vectors*), todos los elementos del mismo tipo
    - puede haber de los 6 tipos: 'logical', 'integer', 'double', 'character', 'complex' y 'raw'
    - Integer y double son tratados como 'numeric'
    - No hay escalares en R, si no vectores de longitud 1
1. Listas
    - Pueden a su vez contener listas (vectores recursivos)
    - *data.frames* son caso especial cuando los vectores que la componen son de igual longitud

Las propiedades más importantes de los vectores son que tipo de vector es -*typeof()*-, que longitud tiene -*length()*- y cuales atributos tiene asociados. Los atributos son metadatos arbitrarios que se pueden asociar a cualquier objeto R. Se determinan y consultan con *attr()* y *attributes()*.

VECTORES R


Vectors
Atomic vectors
Logical
Numeric
Integer
Double
Character
List
NULL

PROPIEDADES DE VECTORES

Tipo, que se determina con *typeof()*

```
typeof(letters)
#> [1] "character"
typeof(1:10)
#> [1] "integer"
```

Longitud, que se determina con *length()*

```
x <- list("a", "b", 1:10)
length(x)
#> [1] 3
```

Atributos, que se determinan con *attributes()* y *attr()*

```
y <- 1:10
attr(y, "mi_atributo") <- "Esto es un vector"
```

Los tres atributos más importantes se obtienen con *names()*, *class()* y *dim()*. Este último atributo permite expandir los vectores atómicos a *arrays* multidimensionales (del estilo de los tensores), cuyo caso especial 2d son las matrices. *dim()* generaliza también el papel de *length()* al caso de arrays, mientras que *nrow()* y *ncol()* lo hacen para el caso especial de matrices ().

LISTAS

Sus elementos pueden tener cualquier tipo, longitud (dimensión!) o atributos, incluyendo otras listas o funciones ¯_(ツ)_/¯

```
# una lista simple
x <- list(1, 2, 3)
str(x)

# elementos con nombre
x_named <- list(a = 1, b = 2, c = 3)
str(x_named)

y <- list("a", 1L, 1.5, TRUE)
str(y)

# mezcla de tipos en el mismo pbjeto
y <- list("a", 1L, 1.5, TRUE)
str(y)

# listas de listas
z <- list(list(1, 2), list(3, 4))
str(z)

# ya conociemos las listas
is.list(mtcars)
```

PRÁCTICA 5

Descargar práctica 5.