ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

EDUCACIÓN PROFESIONAL

Programación en R para Ciencia de Datos

Miguel Jorquera

DBDC-202010

Educación Profesional

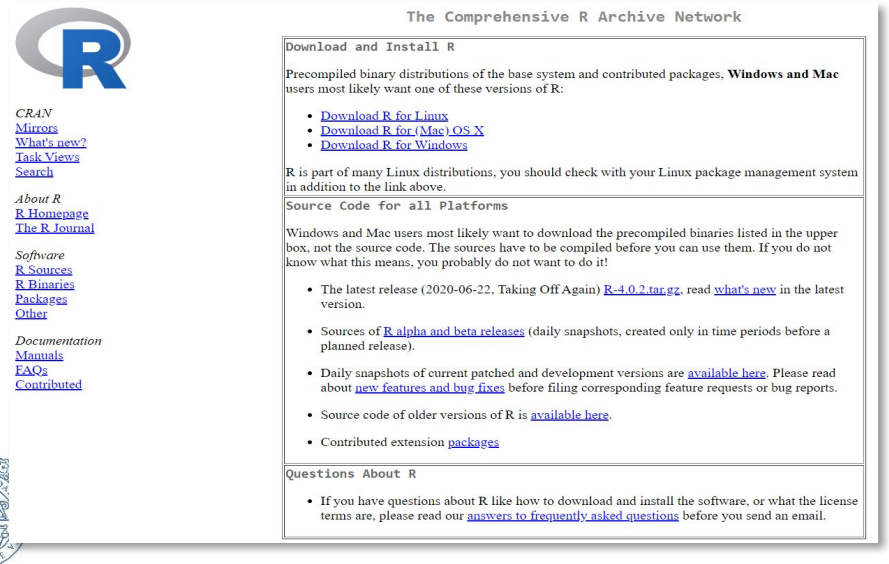
Escuela de Ingeniería

RESUMEN

RESUMEN

Entorno de desarrollo

- **Instalación del R en local**
  Ultima versión disponible desde CRAN (https://cran.r-project.org/).
- **Interfaz de desarrollo local**
  Rstudio (https://www.rstudio.com/)
- **Interfaz cloud**
  Google Colab (R) (colab.fan/r) (crear script desde cero con kernel R)
  Nuestro repositorio(https://github.com/majorquev/Introduccion_a_ciencia_de_datos_R.git)



RESUMEN

Sintaxis básica

- Asignación de variables

```r
letra  <- "M"

vector <- c(10,20,30)

matriz <- matrix(c(10,20,30,40), byrow  = FALSE, ncol = 2)

lista  <- list(num = c(10,20,30),
               char = c("a","b"),
               lista2 = list(nombre = "juan", edad = 31))

df <- data.frame(id = 11:14,
                 nombre = c("bastián","miguel","jorge","felipe")
                 )

names(vector) <- c("n1","n2","n3")
colnames(matriz) <- c("c1","c2")
rownames(matriz) <- c("f1","f2")
```

```
[1] "letra"
'M'
```

```
[1] "vector"
n1:  10 n2:  20 n3:  30
```

```
[1] "matriz"
A matrix:
 2 × 2 of
 type dbl
   c1 c2
f1 10 30
f2 20 40
```

```
[1] "lista"

$num
    10 · 20 · 30
$char
    'a' · 'b'
$lista2

    $nombre
        'juan'
    $edad
        31
```

```
[1] "df"
 A data.frame:
     4 × 2
  id   nombre
<int>  <chr>
11     bastián
12     miguel
13     jorge
14     felipe
```

RESUMEN

Sintaxis básica

```
[1] "vector"
n1:      10 n2:      20 n3:      30
```

```
[1] "matriz"
A matrix:
 2 × 2 of
 type dbl
    c1 c2
f1 10 30
f2 20 40
```

```
1 # Acceso al 2do elemento de vector (por posición, nombre e indicatrices lógicas)
2 vector[2]
3 vector["n2"]
4 vector[c(F,T,F,F)]
```

```
n2: 20
n2: 20
n2: 20
```

```
[36] 1 # Acceso al elemento (2,1) -segunda fila, primera columna-
     2 # por posición,nombre de fila y  columna, y por indicatrices lógicas
     3 matriz[2,1]
     4 matriz["f2","c1"]
```

```
20
20
```

```
[16] 1 # fila completa
     2 matriz[2,]
     3 matriz["f2",]
```

```
c1:      20 c2:      40
c1:      20 c2:      40
```

```
[17] 1 # columna completa
     2 matriz[,1]
     3 matriz[,"c1"]
```

```
f1:      10 f2:      20
f1:      10 f2:      20
```

RESUMEN

Sintaxis básica

```
[1] "matriz"
A matrix:
2 × 2 of
type dbl
   c1 c2
f1 10 30
f2 20 40
```

```
[19]  1 # podemos acceder mediante inticatrices lógicas a uno omás elementos de la matriz
      2 indices <- matrix(c(F,T,T,F),byrow = T, ncol  =   2)
      3 indices
```

```
A matrix: 2 × 2
   of type lgl
FALSE TRUE
TRUE  FALSE
```

```
[23]  1 matriz
```

```
A matrix:
2 × 2 of
type dbl
   c1 c2
f1 10 30
f2 20 40
```

```
      1 matriz[indices] # accedemos a los elementos de posición (1,2)  y (2,1)
      2
```

```
20 · 30
```

RESUMEN

EDUCACIÓN PROFESIONAL

Sintaxis básica

```
[1] "lista"

$num
        10 · 20 · 30
$char
        'a' · 'b'
$lista2

        $nombre
                'juan'
        $edad
                31
```

```
1 # Acceso por posición, nombre y '$'
2 # Recordar que el doble corchete nos da acceso al elemento dentro del slot
3 # lista[1] ylista["num"] nos retorna una lista con el obeto del slot
4
5 lista[[1]]
```

```
10 · 20 · 30
```

```
[30]  1 lista[["num"]]
```

```
10 · 20 · 30
```

```
[31]  1 lista$num
```

```
10 · 20 · 30
```

RESUMEN

EDUCACIÓN
PROFESIONAL

Sintaxis básica

```
[1] "lista"

$num
    10 · 20 · 30
$char
    'a' · 'b'
$lista2

    $nombre
        'juan'
    $edad
        31
```

```
1 # Así, por ejemplo si queremos acceder al segundo elemento del vector 'num'
2 lista[[1]][2]
```

```
20
```

```
[34]  1 lista[["num"]][2]
```

```
20
```

```
[35]  1 lista$num[2]
```

```
20
```

RESUMEN

Sintaxis básica

```
[1] "df"
A data.frame:
    4 × 2
  id   nombre
<int> <chr>
11    bastián
12    miguel
13    jorge
14    felipe
```

Acceso a los **vectores** columnas por posición, nombre y '$'

```
1 df[,2] # hereda acceso como matriz
2 df[,"nombre"] # hereda acceso como matriz
3 df$nombre # hereda acceso como lista
```

```
'bastián' · 'miguel' · 'jorge' · 'felipe'
'bastián' · 'miguel' · 'jorge' · 'felipe'
'bastián' · 'miguel' · 'jorge' · 'felipe'
```

```
[42]  1 # se hereda la estructura de lista
      2 str(df)
```

```
'data.frame':   4 obs. of  2 variables:
 $ id    : int  11 12 13 14
 $ nombre: chr  "bastián" "miguel" "jorge" "felipe"
```

```
[44]  1 #accedemos a sus elementos como en una lista
      2 df[[2]]
      3 df[["nombre"]]
      4
```

```
'bastián' · 'miguel' · 'jorge' · 'felipe'
'bastián' · 'miguel' · 'jorge' · 'felipe'
```

RESUMEN

EDUCACIÓN PROFESIONAL

Sintaxis básica

```
[1] "df"
A data.frame:
    4 × 2
  id   nombre
<int> <chr>
11    bastián
12    miguel
13    jorge
14    felipe
```

```
1 # NOTA:
2 # Al igual que en una lista, la sintaxis df[2] df["nombre"] retorna un data.frame (lista) con el vector requerido
3 df[2]
4 df["nombre"]
```

```
A data.frame:
  4 × 1
 nombre
  <chr>
bastián
miguel
jorge
felipe
A data.frame:
  4 × 1
 nombre
  <chr>
bastián
miguel
jorge
felipe
```

RESUMEN

Sintaxis básica

```
[1] "df"
```

A data.frame: 4 × 2

id	nombre
<int>	<chr>
11	bastián
12	miguel
13	jorge
14	felipe

```
[ ]  1 # Por ejemplo, podemos buscar el registro con id igual a 13
```

```
1 df[df$id == 13,]
```

A data.frame: 1 × 2

	id	nombre
	<int>	<chr>
3	13	jorge

```
[52]  1 # Por ejemplo, podemos buscar el registros cuyo nombre sea "miguel"
      2 df[df$nombre == "miguel",]
```

A data.frame: 1 × 2

	id	nombre
	<int>	<chr>
2	12	miguel

```
[53]  1 # Por ejemplo, podemos buscar los registros con id mayor o igual a 12
      2 df[df$id >= 12,]
```

A data.frame: 3 × 2

	id	nombre
	<int>	<chr>
2	12	miguel
3	13	jorge
4	14	felipe

TEMAS PARA HOY

TEMAS PARA HOY

EDUCACIÓN PROFESIONAL

Camino a la manipulación de tablas

- Adquisición de datos desde archivos csv/web
- Instalación de packages
- Preparación de datos: limpieza y transformación
  - dplyr verbs (filter, arrange, mutate, group_by, select)
  - Joins
  - Imputación de datos

IMPORTACIÓN DE DATOS

TEMAS PARA HOY

Camino a la manipulación de tablas

- En R es posible cargar archivos con valores delimitados con algún separador, en particular fuentes como csv son bien soportadas y se encuentran dentro de las de uso más extendido.
- Algunas de las funciones más utilizadas para cargar archivos con extensión csv o delimitados por otro carácter son:

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
read.csv(file, header = TRUE, sep = ",", quote = "\"",
      dec = ".", fill = TRUE, comment.char = "", ...)
read.csv2(file, header = TRUE, sep = ";", quote = "\"",
           dec = ",", fill = TRUE, comment.char = "", ...)
read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)
read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ",", fill = TRUE, comment.char = "", ...)
```

{base}

TEMAS PARA HOY

Camino a la manipulación de tablas

- El package `data.table` ofrece una opción optimizada para realizar la carga a mayor velocidad a través de la función `fread()`

```
fread(input, file, text, cmd, sep="auto", sep2="auto", dec=".", quote="\"",
nrows = Inf, header="auto",
na.strings=getOption("datatable.na.strings","NA"),  # due to change to ""; see NEWS
stringsAsFactors=FALSE, verbose=getOption("datatable.verbose", FALSE),
skip="__auto__", select=NULL, drop=NULL, colClasses=NULL,
integer64=getOption("datatable.integer64", "integer64"),
col.names,
check.names=FALSE, encoding="unknown",
strip.white=TRUE, fill=FALSE, blank.lines.skip=FALSE,
key=NULL, index=NULL,
showProgress=getOption("datatable.showProgress", interactive()),
data.table=getOption("datatable.fread.datatable", TRUE),
nThread=getDTthreads(verbose),
logical01=getOption("datatable.logical01", FALSE),  # due to change to TRUE; see NEWS
keepLeadingZeros = getOption("datatable.keepLeadingZeros", FALSE),
yaml=FALSE, autostart=NA, tmpdir=tempdir()
```

`{data.table}`

TEMAS PARA HOY

Camino a la manipulación de tablas

- También existen packages para cagar tablas desde hojas de archivos Excel, siendo `readxl` uno de los más utilizados.

```
read_excel(
  path,
  sheet = NULL,
  range = NULL,
  col_names = TRUE,
  col_types = NULL,
  na = "",
  trim_ws = TRUE,
  skip = 0,
  n_max = Inf,
  guess_max = min(1000, n_max),
  progress = readxl_progress(),
  .name_repair = "unique"
)
```

```
read_xls(
  path,
  sheet = NULL,
  range = NULL,
  col_names = TRUE,
  col_types = NULL,
  na = "",
  trim_ws = TRUE,
  skip = 0,
  n_max = Inf,
  guess_max = min(1000, n_max),
  progress = readxl_progress(),
  .name_repair = "unique"
)
```

```
read_xlsx(
  path,
  sheet = NULL,
  range = NULL,
  col_names = TRUE,
  col_types = NULL,
  na = "",
  trim_ws = TRUE,
  skip = 0,
  n_max = Inf,
  guess_max = min(1000, n_max),
  progress = readxl_progress(),
  .name_repair = "unique"
)
```

`{readxl}`

ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

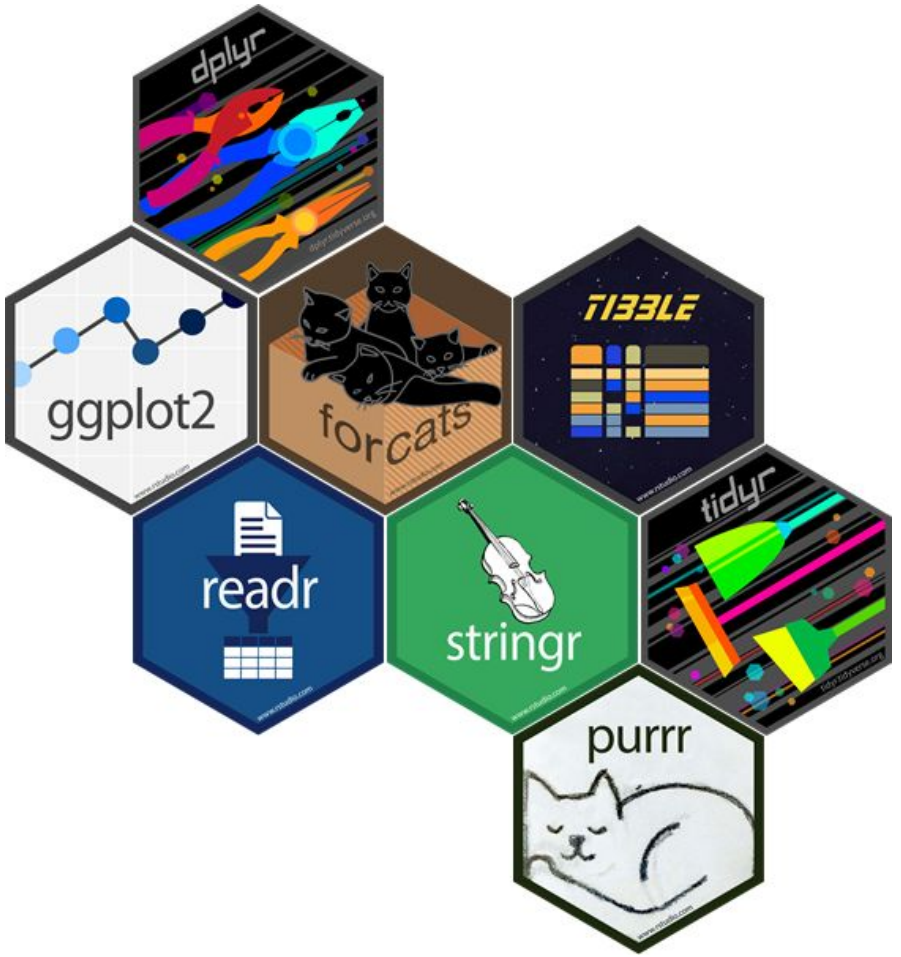
EDUCACIÓN
PROFESIONAL

MANIPULACIÓN DE TABLAS I

TEMAS PARA HOY

Camino a la manipulación de tablas

- Si bien hay variedad de herramientas para llevar a cabo la fase exploratoria, nosotros nos centraremos en la utilización de dos packages principalmente
  - `dplyr` para consultas
    - Generación de información agregada.
    - Tablas de frecuencia.
    - Facilita el cálculo de estadísticos descriptivos en general



TEMAS PARA HOY

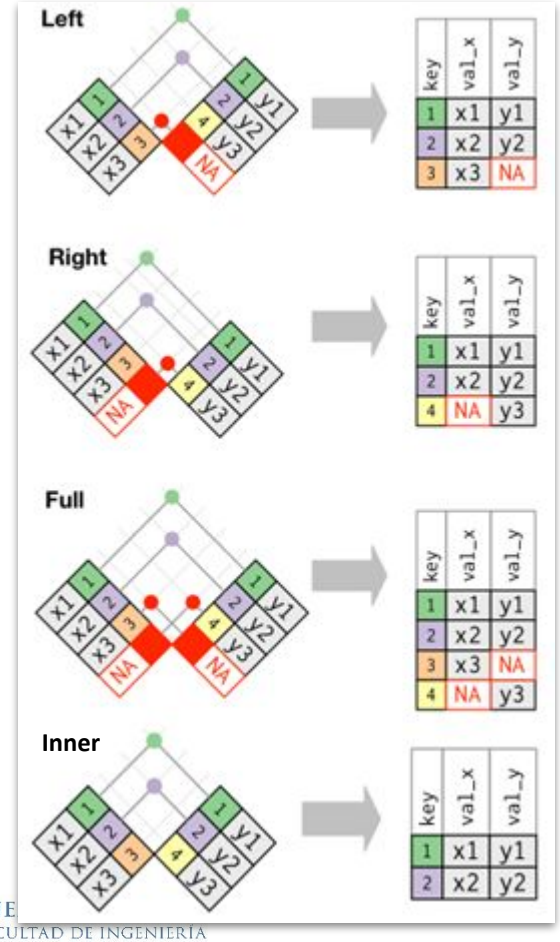
Camino a la manipulación de tablas

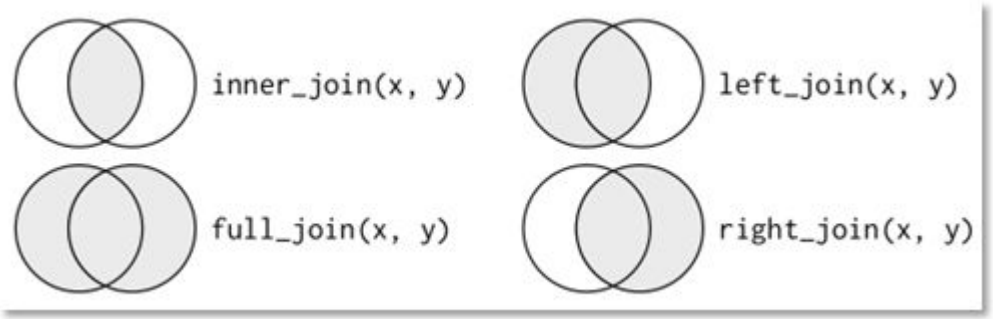
- `filter()` : Para filtrar los datos (por filas).
- `arrange()` : Para ordenar un dataset.
- `select()` : Paa seleccionar y renombrar columnas.
- `mutate()` : Para crear (o modificar) columnas.
- `group_by()` : Para agrupar tablas
- `summarise()` : Para generar medidas agregadas.
- `sample_n()` y `sample_frac()` : Para generar muestras aleatorias.

TEMAS PARA HOY

Camino a la manipulación de tablas

- Datos relacionales y cruce entre tablas





dplyr	merge
inner_join(x, y)	merge(x, y)
left_join(x, y)	merge(x, y, all.x = TRUE)
right_join(x, y)	merge(x, y, all.y = TRUE)
full_join(x, y)	merge(x, y, all.x = TRUE, all.y = TRUE)

dplyr	SQL
inner_join(x, y, by = "z")	SELECT * FROM x INNER JOIN y USING (z)
left_join(x, y, by = "z")	SELECT * FROM x LEFT OUTER JOIN y USING (z)
right_join(x, y, by = "z")	SELECT * FROM x RIGHT OUTER JOIN y USING (z)
full_join(x, y, by = "z")	SELECT * FROM x FULL OUTER JOIN y USING (z)