Taller R – Rifa Valentina

Elecciones, loops y funciones

Nicolás Ratto

2022/02/02

Contenidos curso

- Manipulación avanzada de data frames (funciones `pivot` y combinación de data frames)
- Manipulación de fechas: paquete `lubridate`
- Visualización de datos con `ggplot2`
- Estimar desde diseños muestrales complejos (paquete `survey` y `srvyr`)
- **Procesos iterativos y funciones**

Interacción con pc y envaironment

`ls(),list.fles(),getwd()`

Introducción

Siempre R está trabajando desde una ruta del computador

Desde esta ruta comienza la lectura y carga de archivos

Para conocer esta ruta:

```
getwd()
```

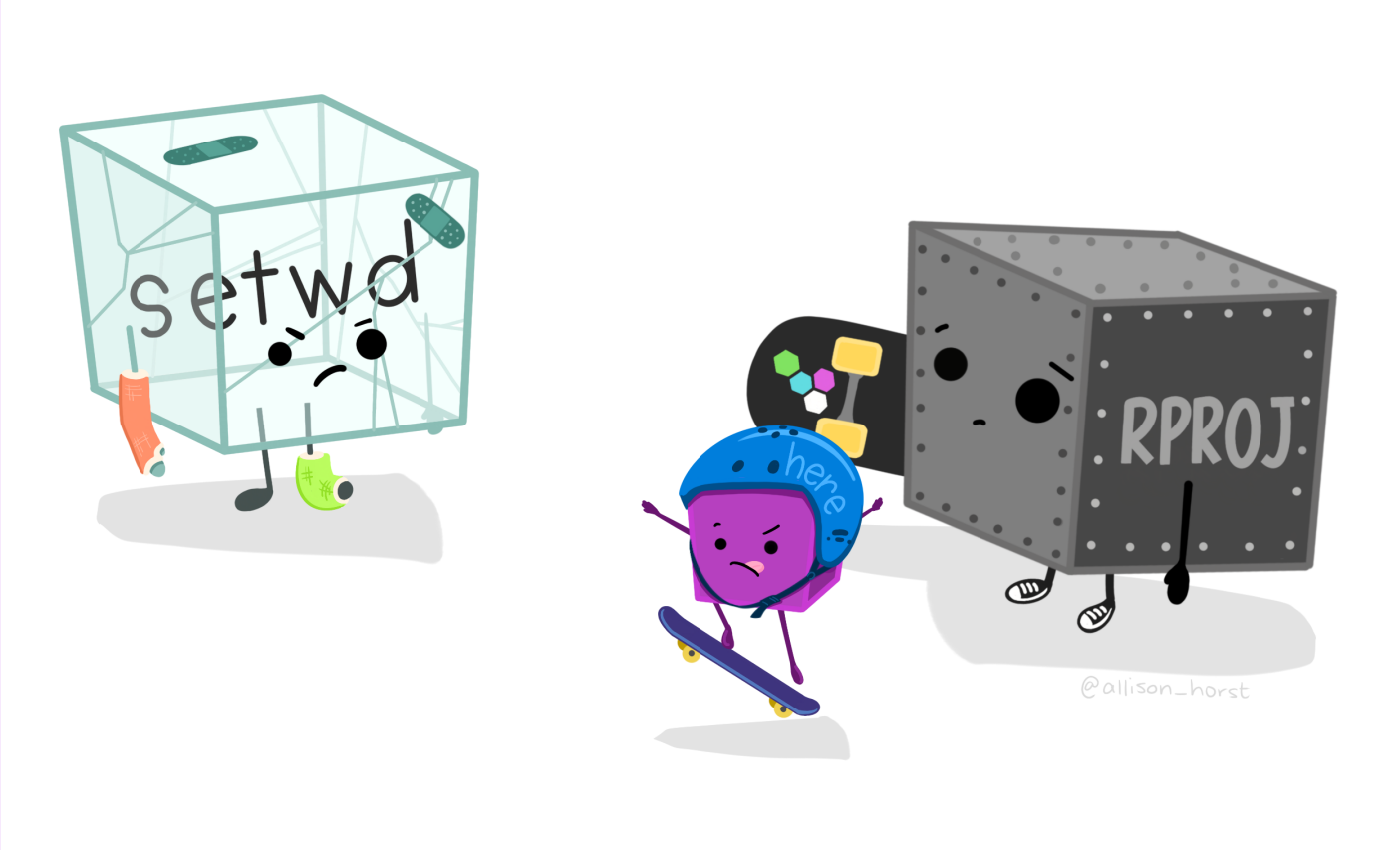
```
## [1] "C:/Users/Nratto/Documents/Github/tallerR_rifa"
```

Para modificarla solo dos formas recomendadas.

- Crear un nuevo RProject
- Abrir un nuevo documento RMarkdown

Es decir, **nunca utilizar setwd()**

¡Muerte a direct absolutos!



Introducción

Ya sabemos cargar desde el pc archivos en diferentes formatos.

Pero, ¿como saber todo lo que existe en una ruta? (una vez que definimos la que nos interesa)

```
list.files()
```

```
##  [1] "1.-trasponer.html"   "1.-trasponer_files"  "1. trasponer.Rmd"
##  [4] "2.-tiempo.html"      "2.-tiempo_files"     "2. tiempo.Rmd"
##  [7] "3.-ggplot2.html"     "3.-ggplot2_files"    "3. ggplot2.Rmd"
## [10] "4.-survey.html"      "4. survey.Rmd"       "5.-iteraciones.html"
## [13] "5.-iteraciones.Rmd"  "5. iteraciones.Rmd"  "6.-openxlsx.html"
## [16] "6. openxlsx.Rmd"     "data"                "datos-encuesta"
## [19] "datos.xlsx"          "datos_ene"           "datos_pt"
## [22] "imagenes"            "libs"                "output"
## [25] "pdf"                 "README.md"           "tallerR_rifa.Rproj"
## [28] "xaringan-themer.css"
```

Para retroceder e ir a otra carpeta

```
list.files(path = "../vignettes")
```

Introducción

En nuestra ruta de interés, podemos preguntarnos si existe una carpeta o archivo

```
file.exists("1.-trasponer.html")
```

```
## [1] TRUE
```

```
file.exists("6.-hacer-paquetes.html")
```

```
## [1] FALSE
```

```
file.exists("datos_ene")
```

```
## [1] TRUE
```

Introducción

Si la carpeta no existe, podemos crearla

```
ifelse(
    dir.exists("datos_ene"),
    "Directorio existe",
    dir.create("datos_ene")
)
```

```
## [1] TRUE
```

¿Funcionó?

```
ifelse(
    dir.exists("datos_ene"),
    "Directorio existe",
    dir.create("datos_ene")
)
```

```
## [1] "Directorio existe"
```

Introducción

Si es un archivo también se puede

Crear dato basura

```
basura <- data.frame(x=c(1:10),y=c(100:109))
```

Exportar dato si no existe

```
ifelse(
    file.exists("datos_ene/basura.xlsx"),
    "dato existe",
    writexl::write_xlsx(basura,"datos_ene/basura.xlsx")
)
```

```
## [1] "C:\\Users\\Nratto\\Documents\\Github\\tallerR_rifa\\datos_ene\\basura.
```

Introducción

Creemos más basura

```
basura2 <- data.frame(x=c(1:10),y=c(100:109))
data3 <- data.frame(x=c(1:10),y=c(100:109))
data4 <- data.frame(x=c(1:10),y=c(100:109))
```

¿Para saber que hay en nuestro ambiente?

```
ls()
```

```
## [1] "basura"       "basura2"      "blue"         "dark_yellow"  "data3"
## [6] "data4"        "gray"         "light_yellow"
```

Según patrón

```
ls(pattern = "basura")
```

```
## [1] "basura"  "basura2"
```

```
rm(list=ls(pattern = "basura")) # borrar
```

Descargar datos desde R

```
download.file()
```

Descargar datos desde R

Para automatizar todo o hacerlo lo más reproducible posible, nos evitaremos incluso descargar los datos mediante *clicks*.

Salvo que queramos descargar .rar, `download.file` sirve:

```
download.file(url="link a pagina",
              destfile = "ruta del pc en donde guardar archivo",
              method = "curl",
              mode="wb")
```

```
url <- "https://www.ine.cl/docs/default-source/ocupacion-y-desocupacion/

destfile <- "datos_ene/ene_2021_10.csv"

download.file(url, destfile, mode = "wb")
```

```
if(!file.exists(destfile)){
download.file(url, destfile, mode = "wb")
        }
```

Elecciones o condiciones

`if()` e `ifelse()`

if()

Ya lo vimos un poco, la lógica es:

```
if (condition) true_action
if (condition) true_action else false_action
```

```
if((1+1)==2){
  "suma correcta"
}
```

```
## [1] "suma correcta"
```

```
if((1+2)==2){
  "suma correcta"
}
```

Condición false, no pasa nada...

if()

Para tener respuesta con condición incorrecta, agregar `else`

```
n <- 1
if((n+2)==2){
  "suma correcta"
} else{
  "suma incorrecta"
}
```

```
## [1] "suma incorrecta"
```

```
n <- 0
if(n+n > 0){
  "n es positivo"
} else if(n+n < 0) {
  "n es negativo"
} else{
  "n es cero"
}
```

```
## [1] "n es cero"
```

if()

```
x <- 70
```

```
if (x > 90) {
"A"
} else if (x > 80) {
"B"
} else if (x > 50) {
"C"
} else {
"F"
}
```

```
## [1] "C"
```

ifelse()

Es un if vectorizado

En if x solo puede tener una longitud de 1

```
x <- 70:90

if (x > 90) {
"A"
} else {
"F"
}
```

```
## Warning in if (x > 90) {: la condición tiene longitud > 1 y sólo el primer
## elemento será usado

## [1] "F"
```

ifelse()

```
x <- 1:10
ifelse(x %% 5 == 0,
       "XXX",
       as.character(x))
```

```
##  [1] "1"   "2"   "3"   "4"   "XXX" "6"   "7"   "8"   "9"   "XXX"
```

Otro if vectorizado y más general es `dplyr::case_when()`

```
dplyr::case_when(
x %% 5 == 0 ~ "fizz",
x %% 2 == 0 ~ "buzz",
is.na(x) ~ "???",
TRUE ~ as.character(x)
)
```

```
##  [1] "1"    "buzz" "3"    "buzz" "fizz" "buzz" "7"    "buzz" "9"    "fizz"
```

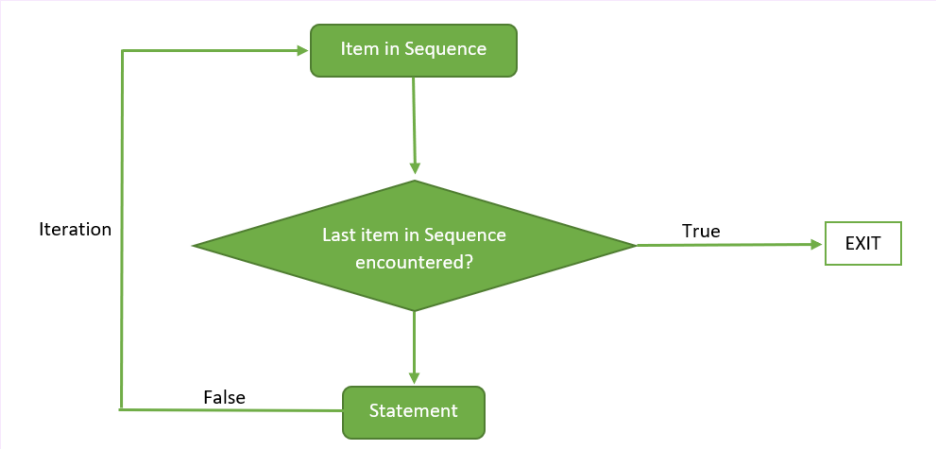
Procesos iterativos o loops

`for()`

for()

Para cada valor X en una secuencia que va de Y a Z, ejecuta una acción.

```
for (item in vector) perform_action
```



`while()` y `repeat()` pueden ser útiles. En los personal no he tenido que usarlos.

- `while` hace una acción mientras la condición es TRUE
- `repeat` repite una acción por siempre (hasta que encuentre un quiebre)

for()

Ejemplo simple:

```
for(i in 1:5){
  print(i + 1)
}
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
```

Imaginen la potencia de esto para exportar o importar cientos de archivos...

Veamos un ejemplo con las bases de puestos de trabajo de la EMR

for()

Crear secuencia de fechas

```
library(dplyr)
library(lubridate)

inicio <- "2016-01-01"
fin <- "2017-02-01"

meses <- ((ymd(inicio) %--% ymd(fin)) / months(1) )

ano_mes_dia <- ymd(inicio) %m+% months(0:meses) %>% as.character()
ano_mes_dia
```

```
##  [1] "2016-01-01" "2016-02-01" "2016-03-01" "2016-04-01" "2016-05-01"
##  [6] "2016-06-01" "2016-07-01" "2016-08-01" "2016-09-01" "2016-10-01"
## [11] "2016-11-01" "2016-12-01" "2017-01-01" "2017-02-01"
```

```
meses <- length(ano_mes_dia)
```

for()

Crear secuencia de meses en español

```
meses_esp <- rep(c("enero","febrero","marzo","abril","mayo","junio",
      "julio","agosto","septiembre","octubre","noviembre","diciembre"),
      trunc(meses/12))

meses_esp <- c(meses_esp,meses_esp[0:(meses%%12)])

meses_esp
```

```
##  [1] "enero"      "febrero"    "marzo"      "abril"      "mayo"
##  [6] "junio"      "julio"      "agosto"     "septiembre" "octubre"
## [11] "noviembre"  "diciembre"  "enero"      "febrero"
```

```
length(meses_esp)==length(ano_mes_dia)
```

```
## [1] TRUE
```

for()

Descargar bases

```
for(i in 1:meses){

download.file(paste0("https://www.ine.cl/docs/default-source/sueldos-y-s
     year(ano_mes_dia[i]),"/",
     meses_esp[i],"-",
     year(ano_mes_dia[i]),".csv?sfvrsn=54c360d8_6&download=true"),
     destfile = paste0("datos_pt/",
                       year(ano_mes_dia[i]),"_",
                       sprintf("%02d", month(ano_mes_dia[i])),".csv"),
     method = "libcurl")
}
```

```
list.files("datos_pt/")
```

```
##  [1] "2016_01.csv" "2016_02.csv" "2016_03.csv" "2016_04.csv" "2016_05.csv"
##  [6] "2016_06.csv" "2016_07.csv" "2016_08.csv" "2016_09.csv" "2016_10.csv"
## [11] "2016_11.csv" "2016_12.csv" "2017_01.csv" "2017_02.csv"
```

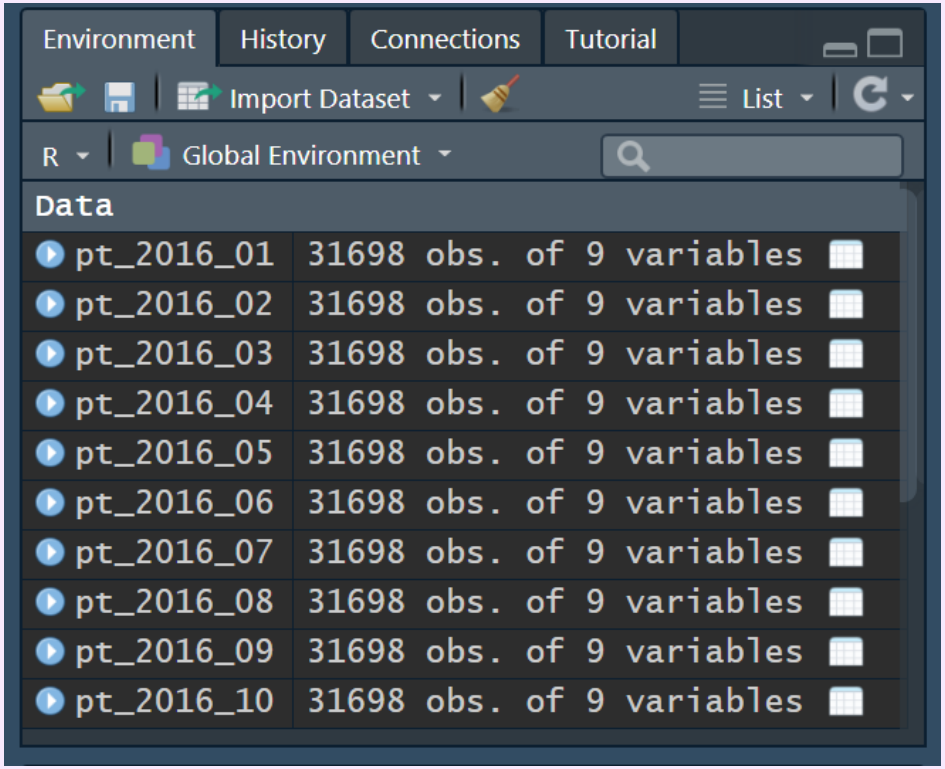
for()

Ahora cargar bases en R

```
archivos <- list.files("datos_pt/",full.names = TRUE)
nombres <- paste0(
  "pt_",
  stringr::str_remove_all(list.files("datos_pt/"),
                                     ".csv")
)
```

```
for(i in 1:length(archivos)){
  assign(nombres[i],
read.csv2(archivos[i])
)
}
```

for()



Las bases se pueden tomar desde el *environment* para nuevos procesos o nunca guardarlas en el ambiente, para no gastar memoria.

for()

Por ejemplo, saber el número de observaciones que tiene cada base

```
base_vacia <- data.frame(periodo=ano_mes_dia,
                         nfilas =NA)
```

```
head(base_vacia,1)
```

```
##      periodo nfilas
## 1 2016-01-01     NA
```

```
for(i in 1:length(archivos)){
  base_vacia[i,2] <- nrow(read.csv2(archivos[i]))
}
```

```
head(base_vacia,3)
```

```
##      periodo nfilas
## 1 2016-01-01  31698
## 2 2016-02-01  31698
```

Crear funciones

`function()`

Funciones

Las funciones que regularmente creamos son para evitar duplicaciones de códigos.

Si estas funciones resuelven problemas para otros y/o tenemos interesantes datos para compartir con otros, podría ser pertinente crear una `paquete`.

Las funciones son objetos, tal como los vectores.

Los tres componentes de una función: argumentos, cuerpo y ambiente

El ambiente es la estructura de datos que determina como la función encuentra los valores asociados a los nombres

```
f02 <- function(x, y) {
# Comentario
x + y
}
```

```
f02(3,5)
```

```
## [1] 8
```

funciones de utilidad

(no empaquetadas)

Primera letra en mayúscula

```
firstup <- function(x) {
  substr(x, 1, 1) <- toupper(substr(x, 1, 1))
  x
}
```

```
texto <- c("manufactura","comercio")
firstup(texto)
```

```
## [1] "Manufactura" "Comercio"
```

funciones de utilidad

Visualizar como excel

```
tabladeprueba <- data.frame(a=1:10,b=21:30)

aexcel <- function (datos)
{
  writexl::write_xlsx(datos, paste0("datos.xlsx"))
  writeLines(paste0("El documento fue guardado en tu actual working dire
                    getwd()))
  shell("datos.xlsx")
}
```

```
aexcel(tabladeprueba)
```

Código escrito por David Jorquera, compañero de INE.

funciones de utilidad

```
n_meses <- function(x){
  x <- tolower(x)
  x <- dplyr::case_when(
              x %in% c("ene","enero") ~ 1,
              x %in% c("feb","febrero") ~ 2,
              x %in% c("mar","marzo") ~ 3,
              x %in% c("abr","abril") ~ 4,
              x %in% c("may","mayo") ~ 5,
              x %in% c("jun","junio") ~ 6,
              x %in% c("jul","julio") ~ 7,
              x %in% c("ago","agosto") ~ 8,
              x %in% c("sep","sept","septiembre") ~ 9,
              x %in% c("oct","octubre") ~ 10,
              x %in% c("nov","noviembre") ~ 11,
              x %in% c("dic","diciembre") ~ 12)
  x
}
```

```
meses_significativos <- c("May","sePt","febrero")
n_meses(meses_significativos)
```

Recursos web utilizados

Xaringan: Presentation Ninja, de Yihui Xie. Para generar esta presentación.

Ilustraciones de Allison Horst

Advanced R, by Hadley Wickham

Loops in R