Solución ejercicios - Sesión 3

Capacitación en R y herramientas de productividad

PE Servicios Compartidos para la Producción Estadística

Solución ejercicios - Sesión 3

En primer lugar, deben descomprimir el archivo "nacimientos.rar" que se encuentra disponible en la carpeta 'data/tarea' en la sesion 3 de nuestro canal en Teams (Generación I). Encontrarán los siguientes archivos Excel:

- **nac2017_j1.xlsx**
- **nac2017_j2.xlsx**
- **nac2017_j3.xlsx**

Estas bases de datos **fueron creadas a partir de la base de datos oficial de nacimientos del 2017**, pero no contienen todos los registros, para evitar que tengan problemas con el ejercicio quienes no cuenten con una buena memoria RAM en sus computadores.

1- En primer lugar, carguen estos 3 objetos en su entorno y explórelos. ¿Qué son y cómo se relacionan estos objetos?

Debemos cargas las 3 bases de datos en nuestro entorno de trabajo. Dado que las bases están en formato Excel, debemos usar la librería `readxl`. Podemos setear una ruta con `setwd()` para ahorrar código y evitar errores al escribir las rutas.

```
setwd("C:/Users/Trabajo/PE SSCC/Capacitacion R/sesion_3/tarea")
```

```
library(readxl)

nac2017_j1 <- read_excel("data/nac2017_j1.xlsx")
nac2017_j2 <- read_excel("data/nac2017_j2.xlsx")
nac2017_j3 <- read_excel("data/nac2017_j3.xlsx")
```

Una forma sencilla de explorar los objetos es utilizar la función `glimpse()` de `tidyverse`. Si bien la primera intuición es abrir la base para verla como si fuera una hoja de cálculo, cuando se trata de bases voluminosas, esta acción puede tardar mucho tiempo o incluso hacer que tengamos que cerrar nuestra sesión de RStudio.

```r
library(tidyverse)

glimpse(nac2017_j1)
```

```
## Rows: 21,919
## Columns: 5
## $ id     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1...
## $ nacion <chr> "C", "C", "C", "C", "C", "C", "C", "C", "C", "C", "C", "C", ...
## $ sexo   <dbl> 1, 1, 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 2, 2, 1, ...
## $ peso   <dbl> 3295, 3815, 2485, 3060, 3445, 2360, 3530, 4100, 2680, 3445, ...
## $ talla  <dbl> 49, 51, 47, 47, 51, 45, 51, 51, 47, 50, 49, 48, 49, 51, 51, ...
```

```r
glimpse(nac2017_j2)
```

```
## Rows: 15,344
## Columns: 7
## $ id      <dbl> 1, 2, 4, 5, 6, 9, 10, 11, 12, 13, 15, 17, 18, 19, 20, 21, 2...
## $ dia_nac <dbl> 27, 27, 28, 10, 14, 3, 9, 11, 24, 4, 18, 26, 20, 24, 1, 28,...
## $ mes_nac <dbl> 11, 1, 6, 4, 10, 8, 2, 8, 4, 4, 10, 9, 2, 1, 1, 6, 11, 4, 1...
## $ ano_nac <dbl> 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017,...
## $ dia_ins <dbl> 30, 2, 3, 13, 16, 7, 10, 14, 26, 6, 23, 28, 20, 25, 3, 29, ...
## $ mes_ins <dbl> 11, 2, 7, 4, 10, 8, 2, 8, 4, 4, 10, 9, 2, 1, 1, 6, 11, 5, 1...
## $ ano_ins <dbl> 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017,...
```

```r
glimpse(nac2017_j3)
```

```
## Rows: 6,575
## Columns: 7
## $ id      <dbl> 3, 7, 8, 14, 16, 22, 28, 29, 30, 31, 36, 38, 45, 48, 51, 54...
## $ dia_nac <dbl> 21, 18, 6, 4, 10, 19, 18, 2, 6, 16, 18, 9, 21, 25, 10, 6, 2...
## $ mes_nac <dbl> 3, 10, 1, 10, 5, 8, 6, 2, 7, 8, 4, 12, 1, 7, 6, 1, 12, 5, 2...
## $ ano_nac <dbl> 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017,...
## $ dia_ins <dbl> 23, 19, 17, 5, 18, 21, 29, 7, 10, 17, 20, 11, 6, 26, 16, 9,...
## $ mes_ins <dbl> 3, 10, 1, 10, 5, 8, 6, 2, 7, 8, 4, 12, 2, 7, 6, 1, 1, 5, 2,...
## $ ano_ins <dbl> 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017,...
```

Observamos que todos los objetos tienen distinta cantidad de filas, y que **nac2017_j1** tiene variables distintas, pero **nac2017_j2** y **nac2017_j3** sí tienen las mismas variables. Si somos perspicaces notaremos, además, que la suma de las filas de **nac2017_j2** y **nac2017_j3** es exactamente el número de filas de **nac2017_j1**. Las tres bases de datos comparten una variable: "id".

En relación al contenido, notamos que **nac2017_j1** contiene variables como "nacion", "sexo", "peso" y "talla", mientras que las otras dos bases de datos, además del "id", solo contienen varibles que permiten construir la fecha de nacimiento e inscripción de los nacidos.

2- Genera en nac2017_j2 y nac2017_j3 las variables "fecha_nac", "fecha_ins" y "dif_days", tal como lo hiciste en el ejercio durante la clase.

Podemos crear las 3 variables de una vez utilizando *pipes*, en cada una de las bases de datos por separado. Esto es posible aún cuando "dif_days" requiere de "fecha_ins" y "fecha_nac" para su creación, porque **R** evalúa la sentencia de izquierda a derecha, tal como leemos en el idioma español.

```
library(lubridate)
nac2017_j2 <- nac2017_j2 %>%
  mutate(fecha_nac = make_date(ano_nac, mes_nac, dia_nac),
         fecha_ins = make_date(ano_ins, mes_ins, dia_ins),
         dif_days = (fecha_nac %--% fecha_ins) / days(1))

nac2017_j3 <- nac2017_j3 %>%
  mutate(fecha_nac = make_date(ano_nac, mes_nac, dia_nac),
         fecha_ins = make_date(ano_ins, mes_ins, dia_ins),
         dif_days = (fecha_nac %--% fecha_ins) / days(1))
```

3- Ahora, une nac2017_j1 con nac2017_j2, conservando todos los registros de nac2017_j1 y solo las variables "fecha_nac", "fecha_ins" y "dif_days" de nac2017_j2, que acabas de crear.

La forma de traer variables de una base a otra, manteniendo todas las observaciones de la primera base es utilizando `left_join()`.

Crearemos otro objeto con las bases de datos unidas. Es posible hacerlo de esta manera:

```
nac2017_j4 <- nac2017_j1 %>% left_join(nac2017_j2[,c(1,8,9,10)], by = "id")
```

Pero también puede utilizarse la función `select()` para escoger las variables que queremos conservar. Recuerden que siempre hay que conservar la llave, sino las bases no se unirán, y el *software* arrojará un error.

```
nac2017_j4 <- nac2017_j1 %>%
  left_join(nac2017_j2 %>%
              select(id, fecha_nac, fecha_ins, dif_days), by = "id")
```

4- ¿Qué sucedió al unir nac2017_j1 con nac2017_j2? ¿Se unieron todos los registros? Si no lo hicieron, ¿por qué pasó eso?

Podrán notar que solo se unieron los registros de la base **nac2017_j2** que estaban presentes (a través de su "id") en la base **nac2017_j1**. En los casos donde no hubo *match*, los registros de la base **nac2017_j1** se mantuvieron, pero sin información de las variables "fecha_nac", "fecha_ins" y "dif_days".

5- Ahora ensambla nac2017_j2 y nac2017_j3 y este nuevo objeto únelo con nac2017_j1, conservando solo "fecha_nac", "fecha_ins" y "dif_days" del objeto ensamblado.

Utilizamos `bind_rows()` para ensamblar las dos bases de datos. Como saben, deben conserva la llave ("id") junto con las otras variables si luego quieren hacer un `join()` para unir los objetos.

```
nac2017_j5 <- nac2017_j2 %>% bind_rows(nac2017_j3) %>%
  select(id, fecha_nac, fecha_ins, dif_days)

con_left <- nac2017_j1 %>% left_join(nac2017_j5, by = "id")
con_inner <- nac2017_j1 %>% inner_join(nac2017_j5, by = "id")
```

6- ¿Qué pasó ahora?

Al ensamblar **nac2017_j2** y **nac2017_j3**, ahora pega perfectamente el objeto ensamblado con **nac2017_j1**. Entonces, se unirán todos los registros, sin valores `NA`. Y no importa si uso `left_join()` o `inner_join()`.

```
dim(con_left)
```

```
## [1] 21919     8
```

```
dim(con_inner)
```

```
## [1] 21919     8
```