Tidyverse

Manipulación y transformación de datos en R

Lucía Coudet

Septiembre de 2018

Dplyr basics

Existen **6 funciones** que nos permiten resolver una amplia gama de desafíos en lo que concierne a manipulación de datos:

- Levantar observaciones por sus valores: `filter()`
- Reordenar filas: `arrange()`
- Levantar variables por sus nombres: `select()`
- Crear nuevas variables a partir de existentes `mutate()` (y `transmute()`)
- Agrupar observarciones con `group_by()`
- Generar medidas de resúmen a partir de distintos valores: `summarise()`

Funciones útiles - Select helpers

Tidyverse también nos provee de varias funciones que nos permiten seleccionar variables en función de sus nombres. Alguna de ellas son:

- `starts_with()`: empieza con un prefijo.
- `ends_with()`: termina con un prefijo.
- `contains()`: contiene una determinada "string".

Tidy data

Lamentablemente los datos suelen venir desordenados, entre otras cosas debido a que por lo general no se levantan desde una lógica adecuada para nosotros, los que los analizamos. El paquete `tidyr` nos ofrece varias funciones útiles para ordenarlos.

- Cuando las columnas son niveles de una variable las reunimos en una sola con `gather()`.
- `spread()`: es el opuesto de `gather()`.
- Si queremos separar los valores de una variable lo hacemos con `separate()`.

Otras funciones de dplyr: Joins

Y si queremos `matchear` (unir) los datos de la base en la cuál estamos trabajando con datos de otra, solo precisamos una variable de matcheo y una de la siguientes funciones que más se adecue a nuestro deseo, las cuales las podemos pensar como operaciones entre conjuntos:

- `full_join()`: unión de conjuntos.
- `inner_join()`: intersección de conjuntos.
- `left_join()`: Se queda con todas las filas de la primera base que se le pasa.
- `right_join()`: Se queda con todas las filas de la segunda base que se le pasa.

Observación: Se generan`Na's` para el caso en que la variable de matcheo presenta valores dispares entre las bases que queremos joinear, según la función que seleccionemos.

Base de datos utilizada

Utilizamos una base de datos extraída del **Estudio Longitudinal de Bienestar en el Uruguay** llevado a cabo por el Instituto de Economía (iecon), el cual consiste en un relevamiento longitudinal representativo de los niños que concurren al sistema de educación primaria pública.

Se recoge información referente a múltiples dimensiones del bienestar de los niños en la muestra así como también del resto de los integrantes de sus hogares, entre los que se destacan:

- situación nutricional.
- habilidades cognitivas y no cognitivas.
- imaginación; razonamiento y sentimientos.
- actividades de ocio einteracción social.
- logros educativos.
- ingresos.
- calidad de la vivienda y bienestar subjetivo, entre otros.

Relevamiento de información

Tres "Olas"

- La primera realizada en el año 2004 a 3000 niños del primer grado de escuela.
- La segunda realizada en el año 2005 a los mismos niños analizados en la primera ola.
- La tercera realizada en el año 2012 a los mismo niños analizados en la primera ola.

Las olas contienen bases

- De **personas**, con información referente al niño y personas del hogar donde reside.
- De **hogares**, con información más enfocada al niño, sus capacidades de relacionamiento, emociones, amigos, etc.

Utilizamos la base de personas correspondiente a la Tercer Ola de relevamiento

Paso 1: Leemos los datos

Los datos están en un archivo .csv por lo que utilizamos la función `read_csv` del paquete `readr`.

```
personas <- readr::read_csv('Personas_Ola_3.csv')
dim(personas)
```

```
## [1] 10447    46
```

Queremos quedarnos solo con algunas variables:

- Sexo.
- Edad.
- Situación conyugal.
- Nivel educativo que cursa o cursó.
- Ingresos.
- Ocupación principal / Tareas.
- Acceso a planes de ayuda económica o alimenticia

Seleccionamos columnas con select()

- Si queremos seleccionar una variable ponemos su nombre.
- Si queremos sacar una variable ponemos su nombre con un signo de - adelante.
- Si queremos seleccionar (o sacar) un un rango seguido de columnas, por ej de la A a la F, lo hacemos con `A:F`, o bien `-(A:F)`.

```
personas <- personas %>% select(-h13, -h11, -(h9b1:h9b8),
                                -(h4:h5), -h2, -h1, -g8b,
                                -g8a, -g7, -g3, -g2b, -f1,
                                -g2a, -g6, -h12)
```

Seleccionamos columnas con select()

Si por ejemplo queremos quitar de la base todas las variables que comienzan con la letra h, la función `starts_with()` nos resuelve el problema

```
personas <- personas %>% select(-starts_with('h'))

names(personas)
```

```
##  [1] "nform" "nper"  "f1"    "f2"    "f3"    "f4"    "f5
##  [9] "f7"    "f8"    "f9"    "f10"   "f11"   "f12"   "g1
## [17] "g2b"   "g3"    "g4"    "g5"    "g6"    "g7"    "g8
## [25] "ola"
```

- `ends_with()` y `contains()` operan de forma análoga.

Renombramos columnas con rename

Los nombres de las variables nos confunden, se los cambiamos con la función `rename()` del paquete `dplyr`.

```
personas <- personas %>%
      dplyr::rename( sit.conyugal = f2,
                     parent.jefe = f10,
                     edad = f11,
                     sexo = f12,
                     dedicacion.p = g1,
                     tipo.p = g4,
                     tareas.p = g5,
                     jub.pen = h3,
                     transf.pais = h7,
                     transf.ext = h8,
                     afam = h9a,
                     benef.esp = h10)
```

Variable nper y niño de la muestra

La variable `nper` etiqueta con un número a cada miembro del hogar. No sabemos a priori qué valor toma para los niños seleccionados en la muestra.

Sospechamos que cuando toma valor 1 se trata del niño de la muestra.

Estos niños deberían tener todos menos de 20 años, por lo que filtramos la base por edad, agrupamos según `nper` y contamos la cantidad de casos para cada grupo.

Filtramos observaciones con filter() y hacemos resúmenes por grupos

```
npersona <- personas %>% filter(edad < 20) %>%
      group_by(nper) %>%
      summarise(conteo = n())
```

Operadores lógicos

Algunos operadores lógicos útiles a la hora de filtrar:

- Mayor a: >
- Menor a: <
- Mayor igual a: >=
- Menor igual a: <=
- Igual a: ==
- Distinto de: !=

Filtramos observaciones con filter() y hacemos resúmenes por grupos

nper	conteo
1	2137
2	48
3	220
4	908
5	613
6	384

`nper == 1` contiene 2137 observaciones, lo cuál coincide con la cantidad de niños en la muestra.

Pero nos gustaría ordenar esa tabla según `conteo` en forma decreciente. Lo hacemos con la función `arrange()`.

Ordenamos filas con arrange()

```
npersona <- arrange(npersona, desc(conteo))
```

nper	conteo
1	2137
4	908
5	613
13	541
6	384
7	226
14	223
3	220

Jefes de hogar por sexo

Tenemos información sobre los jefes de hogar para cada hogar correspondiente a los niños de la muestra.

Nos interesa descubrir si hay una diferencia por sexo, es decir si hay predominancia de hogares con jefe de hogar hombre o mujer.

Para ello

- Filtramos la variable `parent.jefe` para los casos que toma valor 1 que corresponde a los jefes de hogar.
- Agrupamos por sexo.
- Contamos cuantos casos hay en cada categoría (9=NS/NC).
- Creamos una variable que indique el porcentaje.

Jefes de hogar por sexo - mutate()

```
sexo <- personas %>% filter(parent.jefe == 1) %>%
       group_by(sexo) %>%
       summarise(conteo = n()) %>%
       mutate(porcentaje = round(conteo / sum(conteo), 2)*10
```

sexo	conteo	porcentaje
1	1183	56
2	934	44
9	3	0

Jefes de hogar por sexo - transmute()

La diferencia con `mutate()` es que me deja sólo la variable que genero a partir de la función `transmute()`, mientras que `mutate()` incorpora la nueva variable al final del data set.

```
sexo <- personas %>% filter(parent.jefe == 1) %>%
       group_by(sexo) %>%
       summarise(conteo = n()) %>%
       transmute(porcentaje = round(conteo / sum(conteo), 2)
                 sexo = sexo)
```

porcentaje	sexo
56	1
44	2
0	9

Análisis territorial

Nos gustaría tener la variable **departamento** que indica el departamento al cuál pertenece el hogar de la muestra, a los efectos de observar si el resultado anterior presenta **disparidades a nivel territorial**.

Tenemos está información en la **base de Hogares** y una variable de matcheo que es `nform`, la cuál indica el **número de formulario** y vale lo mismo para todos los miembros del hogar.

Nos interesa mantener todas las observaciones de la base de personas, y "pegarles" el departamento al cuál corresponden, por lo cual usamos la función `left_join`.

left_join()

```
hogares <- readr::read_csv('Hogares_Ola_3.csv')
```

Podemos hacerlo en 2 pasos o en 1:

Opción 1: Seleccionamos la columna que nos interesa y luego unimos las bases

```
hogares <- hogares %>% select(nform, dpto_cod)
personas <- personas %>% left_join(hogares,
                                   by = 'nform')
```

Opción 2: Anidamos funciones

```
personas <- personas %>%
            left_join(select(hogares, nform, dpto_cod),
                      by = 'nform')
```

Renombramos la variable departamento

```
personas <- personas %>%
            dplyr::rename(depto = dpto_cod)
```

Observación: Se igula el nombre nuevo al nombre viejo.

Jefes de hogar por sexo y departamento

```
jefes <- personas %>%
      filter(parent.jefe == 1, sexo != 9) %>%
      group_by(depto, sexo) %>%
      summarise(conteo = n()) %>%
      mutate(porcentaje = round((conteo / sum(conteo))*100
```

Jefes de hogar por sexo y departamento

depto	sexo	conteo	porcentaje
1	1	474	54.80
1	2	391	45.20
2	1	151	75.50
2	2	49	24.50
3	1	122	58.10
3	2	88	41.90
5	1	65	68.42
5	2	30	31.58
8	1	50	40.65
8	2	73	59.35
11	1	159	59.33
11	2	109	40.67
13	1	107	39.93
13	2	161	60.07
17	1	55	62.50

Jefes de hogar por sexo y departamento

La tabla se despliega de manera poco amigable para la hora de la interpretación. Nos gustaría tener como filas los departamentos, el sexo en columnas y en las celdas los valores de la variable `porcentaje`.

Esto lo podemos hacer fácilmente con la función `spread()`

Función spread()

```
jefes <- jefes %>% select(-conteo) #sacamos a conteo

jefes <- jefes %>% spread(key = sexo, value = porcentaje)
```

Argumentos de la función `spread()`

- La base de datos (se la pasamos afuera encadenandola con %>%)
- **key**: Se le indica la columna que contiene la información que va a ir a columnas
- **value**: Se le indica el valor que va a ir en las celdas.

Función spead()

depto	1	2
1	54.80	45.20
2	75.50	24.50
3	58.10	41.90
5	68.42	31.58
8	40.65	59.35
11	59.33	40.67
13	39.93	60.07
17	62.50	37.50

Jugamos con la tabla

Si quisieramos volver a la tabla anterior, lo hacemos con la función opuesta a `spread()`, que es `gather()`.

`gather()`, al igual que spread, precisa los argumentos **key** y **value**.

Argumentos de la función `gather()`

- **key**: El nombre de la variable cuyos valores se encuentran como nombres de columnas.
- **value**: El nombre de la variable cuyos valores están "dispersos" por todas las celdas.

Función gather()

```
jefes <- jefes %>%
      gather(key = 'sexo', value = 'porcentaje', -depto)
```

Observación: Dentro de gather pasamos `-depto`, ya que sino se cree que la variable departamento es otra de las categorías de la variable sexo y la gatherea también.

Función gather()

depto	sexo	porcentaje
1	1	54.80
2	1	75.50
3	1	58.10
5	1	68.42
8	1	40.65
11	1	59.33
13	1	39.93
17	1	62.50
1	2	45.20
2	2	24.50
3	2	41.90
5	2	31.58
8	2	59.35
11	2	40.67
13	2	60.07

Taller

Pasos previos:

- Desde la consola de R, instalamos el paquete `tidyverse`: `install.packages('tidyverse')`.
- Cargamos la librería: `library(tidyverse)`.
- Bajamos la base de datos desde la página de la fcea: `http://fcea.edu.uy/datos/bases-de-datos.html`
- Le indicamos a R el directorio de trabajo: setwd('ruta/al/directorio')
- Cargamos los datos:

```
personas <- readr::read_csv('Personas_Ola_3.csv')`
```

Taller

Anteriormente habíamos encontrado que el 56% de los jefes de hogar son hombres mientras que el 44% mujeres.

Nos interesa sebr cuál es la *situación conyugal* de esas jefas de hogar a los efectos de determinar si son jefas de hogar porque no tienen pareja o si teniendo pareja declararon ser jefas de hogar, lo cual sería evidencia de **empoderamiento femenino**.

Actividad 1: Situación conyugal de las jefas de hogar

- Calcular, para las jefas de hogar, cuántas observaciones hay en cada nivel de las variable situación conyugal (`f2`).
- Crear una variable que se llame *porcentaje* que indique el porcentaje de mujeres jefas de hogar para cada categoría de la variable situación conyugal (`f2`).

Solución actividad 1: Situación conyugal de las jefas de hogar

```
sit.con <- personas %>%
      filter( f12 == 2 & f10 == 1) %>%
      group_by(f2) %>%
      summarise(conteo = n())
```

f2	conteo
1	180
2	171
3	220
4	287
5	75
9	1

Solución actividad 1: Situación conyugal de las jefas de hogar

```
sit.con <- personas %>%
      filter( f12 == 2 & f10 == 1) %>%
      group_by(f2) %>%
      summarise(conteo = n()) %>%
      mutate(porcentaje = round((conteo / sum(conteo))*100
      arrange(desc(porcentaje))
```

f2	conteo	porcentaje
4	287	30.73
3	220	23.55
1	180	19.27
2	171	18.31
5	75	8.03
9	1	0.11

Tareas en la ocupación principal

En lo que respecta al ámbito laboral, queremos saber si existe una diferencia de calificación en las tareas que realizan las jefas de hogar en sus trabajos respecto a las que realizan los jefes.

Actividad 2: Tareas en la ocupación principal de los jefes de hogar por sexo

- Calcular los porcentajes de hombres y de mujeres jefes de hogar para cada nivel de la variables *tareas que realiza en la ocupación principal* (g5)
- **observación**: previo a hacer los calculos sacar los NAs, como se realiza en el código a continuación.

```
tareas <- personas %>%
      filter(!is.na(f12), !is.na(g5), g5 != 99, f12 != 9)
```

Solución actividad 2: Tareas en la ocupación principal

```
tareas <- personas %>%
      filter(!is.na(f12), !is.na(g5), f12 != 9, g5 != 99) %
      filter(f10 == 1) %>%
      group_by(g5, f12) %>%
      summarise(conteo = n())
```

```
tareas <- personas %>%
      filter(!is.na(f12), !is.na(g5), g5 != 99, f12 != 9) %
      filter(f10 == 1) %>%
      group_by(g5, f12) %>%
      summarise(conteo = n()) %>%
      mutate(porcentaje = round((conteo / sum(conteo))*100
```

Solución actividad 2: Tareas en la ocupación principal

```
knitr::kable(tareas)
```

g5	f12	conteo	porcentaje
1	1	55	68.75
1	2	25	31.25
2	1	40	44.94
2	2	49	55.06
3	1	88	70.40
3	2	37	29.60
4	1	50	48.08
4	2	54	51.92
5	1	252	52.83
5	2	225	47.17
6	1	26	92.86
6	2	2	7.14

Tareas en la ocupación principal

La tabla anterior no nos gusta porque no podemos visualizar de una manera entendible los resultados.

Actividad 3: tidy data

- Ordenar la tabla anterior de forma tal que en las columnas aparezca el sexo y en las celdas los valores de la variable porcentaje.

Solución actividad 3: tidy data

```
tareas <- tareas %>% select(-conteo)

tareas <- tareas %>% spread(key = f12, value = porcentaje)

tareas <- tareas %>% arrange(desc(`2`))
```

Solución actividad 3: tidy data

```
knitr::kable(tareas)
```

g5	1	2
2	44.94	55.06
4	48.08	51.92
5	52.83	47.17
9	59.20	40.80
1	68.75	31.25
3	70.40	29.60
8	85.71	14.29
7	88.64	11.36
6	92.86	7.14
10	96.15	3.85

TERMINAMOS!

Figure 1

Referencias

Hadley Wickham, Romain Francois, Lionel Henry and Kirill Müller (2017). dplyr: A Grammar of Data Manipulation. R package version 0.7.4. https://CRAN.R-project.org/package=dplyr

Hadley Wickham and Lionel Henry (2017). tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions. R package version 0.7.1.https://CRAN.R-project.org/package=tidyr

Hadley Wickham, Jim Hester and Romain Francois (2017). readr: Read Rectangular Text Data. R package version 1.1.1. https://CRAN.R-project.org/package=readr

Hadley Wickam and Garett Grolemund, R for Data Science. Import, tidy, transform, visualize, and model data (2016).