6. Preparación y limpieza de datos

Índice

1. Preparación de datos

En la colección de `tidyverse` se pueden encontrar varios paquetes pensados para cada una de las fases del preprocesamiento de los datos, una vez obtenidos de sus fuentes.

La primera de estas fases se corresponde con la preparación, es decir, la reestructuración de dichos datos (en caso de que sea necesario) para que cumplan los conocidos como los tres criterios de los "datos ordenados" (en inglés *tidy data*):

1. Cada variable tiene su columna propia.
2. Cada observación tiene su fila propia.
3. Cada valor tiene su celda propia.

Esto ofrece como principales ventajas:

- Normalización de los datos a un estándar que está ampliamente aceptado y consolidado.
- Uso de una manera consistente de trabajar que se alinea con el tidyverse.
- Facilidad para trabajar con la lógica vectorizada.

Lo más frecuente es encontrar datos no ordenados según estos criterios, presentándose las siguientes situaciones:

- Encabezados de columnas que son valores en lugar de variables.
- Una misma columna contiene varias variables.
- Variables que están almacenadas tanto en filas como en columnas.

tibble

Uno de los paquetes de `tidyverse` y con el que trabajan los demás de la colección es `tibble`, que define una nueva estructura de datos con dicho nombre muy similar al data frame pero con algunas diferencias:

- Está más optimizada (realiza evaluación perezosa).
- Proporcionan una mayor información sobre el contenido y la estructura.
- No cambian tipos de datos, nombres de variables o filas, etc.

Cuando se utiliza el paquete `readr`, este devuelve siempre un `tibble`. Otra opción es convertir un data frame a `tibble`, por ejemplo:

```
library(tibble)

notasDf <- data.frame(asignatura = c("Matemáticas", "Física", "Economía"),
nota = c(8.5, 7, 4.5))
```

```
notasTibble <- as_tibble(notas)
```

En caso de querer convertir un `tibble` en un data frame:

```
notasDf2 <- as.data.frame(notasTibble)
```

tidyr

De cara a realizar la ordenación, el paquete `tidyr` ofrece dos funciones para pivotar un data frame, es decir, alternar entre las disposiciones principales posibles para los datos, conocidas como formato ancho y formato largo:

Formato ancho

Nombre	Economía	Matemáticas	Programación
Carmen	5.0	3.5	9.0
Luis	6.5	7.0	4.0
María	8.0	8.5	6.5

Formato largo

Nombre	Asignatura	Nota
Carmen	Economía	5.0
Luis	Economía	6.5
María	Economía	8.0
Carmen	Matemáticas	3.5
Luis	Matemáticas	7.0
María	Matemáticas	8.5
Carmen	Programación	9.0
Luis	Programación	4.0
María	Programación	6.5

Donde el formato largo es aquel en el que las columnas del data frame representan variables y las filas observaciones, de manera que cada dato pertenece a una variable y una observación única.

Las funciones mencionadas son `pivot_longer` y `pivot_wider`, que convierten a formato largo y ancho, respectivamente:

```
library(tidyr)

notasDf <- data.frame(
  nombre = c('María', 'Luis', 'Carmen'),
  edad = c(18, 22, 20),
  Matemáticas = c(8.5, 7, 3.5),
  Economía = c(8, 6.5, 5),
  Programación = c(6.5, 4, 9)
)

notasDf_largo <- pivot_longer(df, Matemáticas:Programación, names_to =
"Asignatura", values_to = "Nota")
```

```
notasDf_ancho <- pivot_wider(df_largo, names_from = Asignatura, values_from
= Nota)
```

Además del pivoteo, existen otras funciones dentro del paquete para realizar la preparación de los datos, agrupadas en categorías:

- Rectangular
  - unnest_longer
  - unnest_wider
  - hoist
- Anidar
  - nest
  - unnest
- Dividir y combinar
  - separate
  - extract
  - unite
- Manejar valores perdidos
  - complete
  - drop_na
  - fill
  - replace_na

2. Limpieza de datos

Una vez ordenados adecuadamente, es conveniente eliminar posibles duplicados o valores perdidos (también llamados valores faltantes, representados en R por el tipo de datos `NA`) que puedan estropear las fases de trabajo futuras.

Es posible obtener una estructura de datos con los valores perdidos presentes en un data frame:

```
notasDf <- data.frame(
  nombre = c('María', 'Luis', 'Carmen'),
  edad = c(18, NA, 20),
  Matemáticas = c(NA, 7, 3.5),
  Economía = c(8, 6.5, NA),
  Programación = c(NA, NA, 9)
)

notasDfNA <- is.na(notasDf)
```

Para después comprobar cuántos hay en total, de manera que si hay alguno, tratar de localizarlos a posteriori:

```
sum(notasDfNA)
```

dplyr

Este es otro paquete muy útil de `tidyverse` que ofrece una gramática más declarativa para realizar ciertas acciones con los datos más sencilla y cómodamente, facilitando la codificación.

En el caso de la limpieza y siguiendo con el ejemplo anterior de búsqueda y tratamiento de valores perdidos, se puede obtener la cantidad de valores perdidos por cada columna:

```
library(dplyr)

countNA <- function(x) {
  return (sum(is.na(x)))
}

summarise_all(.tbl = notasDf, .funs = countNA)
```

O alternativamente usando el operador *pipe* del paquete con un enfoque más funcional, y obteniendo solo aquellas columnas que tengan algún valor perdido:

```
notasDf %>%
  summarise_all(.funs = function(x) sum(is.na(x))) %>%
  select_if(function(x) x > 0)
```

Para reemplazar los valores perdidos por otro valor arbitrario, que puede ser por ejemplo uno incoherente, muy alto o la media:

```
notasDf %>%
  select(edad, Matemáticas, Economía, Programación) %>%
  arrange(edad) %>%
  mutate_all(~ifelse(is.na(.x), -1, .x)) %>%
  # mutate_all(~ifelse(is.na(.x), mean(.x, na.rm = TRUE), .x)) %>%
  tail()
```

Otras operaciones de limpieza que se podrían hacer son:

- Eliminar filas con valores faltantes:

  ```
  notasDf %>% na.omit()
  ```

- Eliminar filas duplicadas:

  ```
  notasDf %>% distinct(.keep_all=TRUE)
  ```

Referencias

Preprocesamiento de datos
Limpieza de datos (I)
Limpieza de datos (II)

tibble documentation
tidyr documentation
dplyr documentation