

Fastbook 06

Programación en R

Trabajando con dataframes (I)



06. Trabajando con dataframes (I)

Tal y como hemos comentado en otras ocasiones, la presente asignatura está dividida en dos partes principales.

En la primera (del fastbook 01 al 04) se han sentado las bases del lenguaje de programación R. Con este sexto fastbook estrenamos la fase 2, cuyo foco se sitúa en la práctica de analytics.

¡Empezamos!

Autor: Juan Jiménez

Análisis exploratorio de datos

Orden del dataframe

Transformaciones estructurales

Renombrado de columnas

Introducción a dplyr

Modificación y creación de columnas

Selección de columnas

Conclusiones

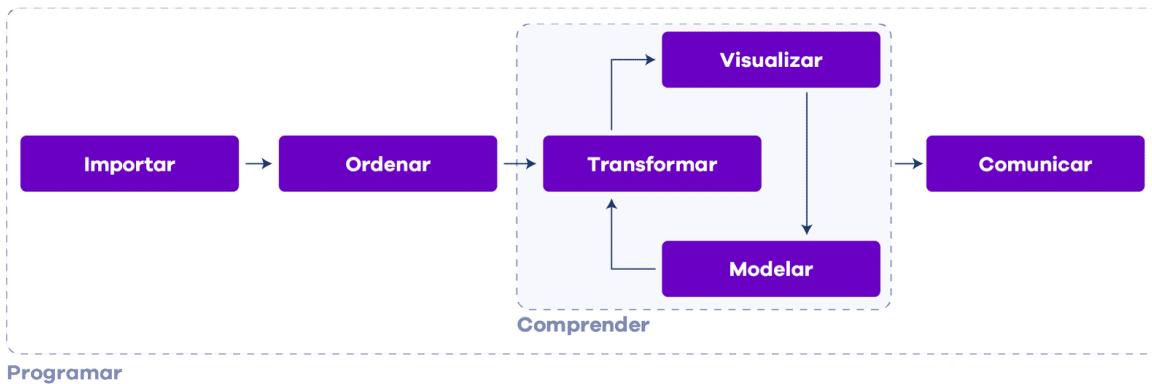
Filtrado de muestras

Análisis exploratorio de datos

X Edix Educación

Antes de nada, conviene realizar una parada que nos sirva de introducción a lo que vamos a ver en esta última parte de la asignatura.

La siguiente imagen, tal y como se mostró en el fastbook 01, contiene el flujo de **tareas** a las que se enfrenta un **data scientist** en su trabajo diario.



1

La importación de datos ya ha sido abordada, por lo que ya contamos con ese conocimiento.

2

Aunque ya hemos trabajado con algunos conceptos de orden y transformación de dato, tanto este como el siguiente fastbook vuelven a poner el foco en esta tarea, pero desde una perspectiva más avanzada.

3

Debido a su magnitud, la modelización y visualización de datos está cubierta por otras asignaturas del programa.

4

Por último, los dos últimos fastbooks nos enseñarán a construir informes desde R que nos sirvan para comunicar nuestros resultados de una forma eficiente y elegante.

Tanto en este como en el siguiente fastbook vamos a centrarnos en la **manipulación y transformación avanzada de dataframes**. Para ello, haremos uso de dos paquetes que están diseñados para trabajar con este tipo de estructuras: **tidyverse**.

Transformaciones estructurales

X Edix Educación

La primera transformación que vamos a aprender necesita del paquete `tidyverse`, y nos permite cambiar el formato de nuestros datos de long a wide y viceversa. Pero ¿qué son estos formatos?

Cuando trabajamos con datos estructurados (tablas), podemos diferenciar entre **dos tipos de información**: aquella que nos sirve para **identificar y definir a nuestra muestra**, y la que nos aporta información sobre sus **características y atributos**.

Identificación		Atributos		
Nombre		Edad	Peso	Altura
Juan		23	72	175
Andrea		34	64	171
Laura		67	59	162
Carlos		14	58	169
David		82	87	182
María		41	70	174
⋮		⋮	⋮	⋮

En muchas ocasiones, la información que identifica las muestras está constituida por más de una columna.

Identificación			Atributos		
Día	Tienda	Producto	Unidades	Precio unitario	Venta
2020-10-02	1514	Manzana a granel	262	1.49	390.38
2020-10-02	2352	Leche 1L	653	0.72	470.16
2020-10-02	2352	Galletas 250g	355	2.39	848.45
2020-10-02	7263	Leche 1L	789	0.72	568.08
2020-10-03	2352	Manzana a granel	124	1.49	184.76
2020-10-03	1514	Galletas 250g	41	2.39	97.99
⋮	⋮	⋮	⋮	⋮	⋮

Los dos ejemplos que acabamos de ver usan **formato wide**: cada atributo tiene su columna.

Identificación			Atributos		
Día	Tienda	Producto	Unidades	Precio unitario	Venta
2020-10-02	1514	Manzana a granel	262	1.49	390.38
2020-10-02	2352	Leche 1L	653	0.72	470.16
2020-10-02	2352	Galletas 250g	355	2.39	848.45
2020-10-02	7263	Leche 1L	789	0.72	568.08
2020-10-03	2352	Manzana a granel	124	1.49	184.76
2020-10-03	1514	Galletas 250g	41	2.39	97.99
⋮	⋮	⋮	⋮	⋮	⋮

← →

Formato Wide

El formato **long** condensa todos los atributos en dos columnas: nombre de atributo y valor. Debido a esto, una misma identificación (muestra) aparece en tantas filas como parámetros tenga. De ahí viene el término *long*.

Identificación			Atributos	
Día	Tienda	Producto	Atributo	Valor
2020-10-02	1514	Manzana a granel	Unidades	262
2020-10-02	1514	Manzana a granel	Precio unitario	1.49
2020-10-02	1514	Manzana a granel	Venta	390.38
2020-10-02	2352	Leche 1L	Unidades	653
2020-10-02	2352	Leche 1L	Precio unitario	0.72
2020-10-02	2352	Leche 1L	Venta	470.16
⋮	⋮	⋮	⋮	⋮

Formato Long

Para movernos entre estos formatos vamos a usar las funciones *pivot_longer()* y *pivot_wider()* que, como ya hemos comentado, forman parte del paquete tidyverse.

Función *pivot_longer()*

Parámetro	Clase	Definición
data	dataframe	Dataframe en formato wide que queremos transformar a formato long.
cols	vector	Columnas que queremos pasar a formato long y que por lo tanto pasarán a formar parte de la estructura atributo–valor.
names_to	vector	Nombre de la columna que va a albergar el nombre de los atributos.
values_to	vector	Nombre de la columna que va a albergar el valor de los atributos.

Función *pivot_wider()*

Parámetro	Clase	Definición
data	dataframe	Dataframe en formato long que queremos transformar a formato wide.
id_cols	vector	Nombre de las columnas que identifican nuestra muestra.
names_from	vector	Nombre de la columna a partir de la cual vamos a extraer el nombre de cada atributo.
values_from	vector	Nombre de la columna a partir de la cual vamos a extraer el valor de cada atributo.

Introducción a dplyr

X Edix Educación

El resto de las transformaciones que vamos a ver hacen uso del paquete dplyr. Antes de pasar a ellas, es necesario realizar una pequeña introducción al uso y las peculiaridades de este paquete.

En primer lugar, debemos hablar sobre el **operador pipe**.

El operador pipe sirve para pasar un determinado parámetro a la función que queremos ejecutar. En la práctica se utiliza para concatenar la aplicación de múltiples funciones. Su símbolo es **%>%**.

```
#Definimos un dataframe
df <- data.frame( nombre = c(Juan, Andrea, Laura, Carlos, David, NA),
                  edad = c(23,34,67,14,82,41),
                  peso = c(72, 64, 59, 58, 87, 70),
                  altura = c(175, 171, 162, NA, NA, 174) )

#Para calcular el número de NAs en cada una de las columnas, podemos ejecutar la
#siguiente linea
lapply(as.data.frame(is.na(df)),sum)

#o usar el operador pipe (visualmente es mucho más intuitivo)
df %>% is.na() %>% as.data.frame() %>% lapply(sum)
```

Ejemplo de uso del operador *pipe*.

En segundo lugar, tenemos que mencionar los **objetos tibble**.

Son una estructura de datos muy similar a un dataframe (rectangular y organizada en filas y columnas) que se integra dentro del paquete **tibble**. Dplyr importa este paquete de forma automática porque, aunque puede trabajar con dataframes ordinarios, está diseñado para hacerlo con tibbles.

¿Cuáles son las **diferencias entre un dataframe** (forma parte del R base) y un **tibble** (del paquete tibble)?

- El método `print` es distinto. Los tibbles incluyen ciertas mejoras que facilitan el trabajo: muestran el número de filas y columnas, el tipado de cada variable y los NA resaltados en color rojo.
- Si bien los tibble pueden utilizar rownames, por defecto los eliminan y se desaconseja usar ese atributo.
- Los tibble son más estrictos a la hora de acceder a una columna. Si en un dataframe queremos acceder a la columna edad y escribimos `$ed`, va a funcionar. Sin embargo, si se trata de un tibble, mostrará un error.

El uso de las funciones `data.frame()` y `tibble()` nos permiten movernos de un tipo de objeto a otro.

Tras conocer estos dos aspectos, es hora de pasar a la acción a través de las funciones más importantes que se integran dentro de este paquete.

Selección de columnas

X Edix Educación

La función `select()` de dplyr sirve para quedarnos con un determinado conjunto de columnas de nuestro dataframe o tibble.

Recibe el dataframe o tibble y el nombre de las distintas variables que queremos seleccionar.

```
#Definimos un dataframe
df <- data.frame( nombre = c(Juan, Andrea, Laura, Carlos, David, NA),
                  edad = c(23,34,67,14,82,41),
                  peso = c(72, 64, 59, 58, 87, 70),
                  altura = c(175, 171, 162, NA, NA, 174) )
df <- as_tibble(df)

#Seleccionamos las columnas nombre y edad y volvemos a guardar
df <- df %>% select(nombre, edad)

#También podemos usar select seguir manteniendo todas las columnas, pero en otro orden
df <- df %>% select(nombre, altura, peso, edad)
```

Filtrado de muestras

X Edix Educación

La función *filter()* de dplyr sirve para quedarnos con las muestras (filas) que cumplan con unas determinadas condiciones de filtrado que se construyen en base al valor de las variables.

```
#Definimos un dataframe

df <- data.frame( nombre = c(Juan, Andrea, Laura, Carlos, David, NA),
                  edad = c(23,34,67,14,82,41),
                  peso = c(72, 64, 59, 58, 87, 70),
                  altura = c(175, 171, 162, NA, NA, 174) )

df <- as_tibble(df)

#Nos quedamos con aquellas muestras cuyo nombre es Juan o Andrea

df %>% filter(nombre=="Juan" | nombre=="Andrea")

#Nos quedamos con aquellas muestras que tienen mayoría de edad y altura superior o igual a 175

df %>% filter(edad>=18 & altura>=175)

#Nos quedamos con aquellas muestras cuyo peso es inferior a los 65 kilos y
#posteriormente seleccionamos las columnas nombre y edad

df %>% filter(peso < 65) %>% select(nombre, edad)
```

Orden del dataframe

 Edix Educación

La función *arrange()* de dplyr sirve para ordenar las muestras en base a los valores de una o más columnas.

Recibe el dataframe o tibble y el nombre de las distintas variables por las que lo queremos ordenar. Las columnas que situemos primero tendrán mayor prioridad. Podemos usar la función *desc()* para indicar orden descendente.

```
#Definimos un dataframe  
  
df <- data.frame( nombre = c("Juan", "Andrea", "Laura", "Carlos", "David", NA),  
                  edad = c(23,34,67,14,82,41),  
                  peso = c(72, 64, 59, 58, 87, 70),  
                  altura = c(175, 171, 162, NA, NA, 174) )  
  
df <- as_tibble(df)  
  
#Ordenamos la tabla por la variable nombre (orden alfabético)  
df %>% arrange(nombre)  
  
#Ordenamos la tabla de menor a mayor edad  
df %>% arrange(edad)  
#Ordenamos la tabla de mayor a menor edad  
df %>% arrange(desc(edad))  
  
#Ordenamos la tabla de menor a mayor edad y en segundo lugar de menor a mayor altura (en  
el caso de que dos o más muestras tengan la misma edad vendrán ordenadas por altura)  
df %>% arrange(edad, altura)
```

Renombrado de columnas

X Edix Educación

La función `rename()` de dplyr sirve para modificar el nombre de las columnas de nuestra tabla.

Para ello, tendremos que introducir el dataframe o tibble, así como los cambios de nombre que queremos llevar a cabo. El formato a usar debe ser `nombre_nuevo = nombre_antiguo`.

```
#Definimos un dataframe

df <- data.frame( nombre = c("Juan", "Andrea", "Laura", "Carlos", "David", NA)
                  edad = c(23,34,67,14,82,41),
                  peso = c(72, 64, 59, 58, 87, 70),
                  altura = c(175, 171, 162, NA, NA, 174) )

df <- as_tibble(df)

#Modificamos el nombre de la variable "nombre"

df <- df %>% rename(name = nombre)

#Modificamos el nombre de las variables "edad", "peso" y "altura"

df <- df %>% rename(age = edad, weight = peso, height = altura)
```

Modificación y creación de columnas

X Edix Educación

La función `mutate()` es una de las más potentes de dplyr. Nos permite modificar y construir nuevas variables en nuestro dataframe.

Su funcionamiento se basa en recibir el dataframe y la definición de las variables que queremos construir o modificar. Dicha definición puede ser un valor constante, un vector o, lo que resulta más interesante, construirse a partir de otras columnas.

```
#Definimos un dataframe  
df <- data.frame( nombre = c("Juan", "Andrea", "Laura", "Carlos", "David", NA),  
                  edad = c(23,34,67,14,82,41),  
                  peso = c(72, 64, 59, 58, 87, 70),  
                  altura = c(175, 171, 162, NA, NA, 174) )  
  
df <- as_tibble(df)  
  
#Construimos la variable "numero_de_letras_en_el_nombre"  
df <- df %>% mutate(numero_de_letras_en_el_nombre = nchar(nombre))  
  
#Calculamos el Índice de Masa Corporal de cada persona (peso/(altura^2))  
df <- df %>% mutate(IMC = peso/(altura^2))  
  
#Introducimos un nuevo vector de información (apellido)  
df <- df %>% mutate(apellido = c("Fernandez", "Jimenez", "Cruz", "Rodriguez", "Ruiz",  
"Hernandez"))
```

Conclusiones

 Edix Educación

En este fastbook hemos aprendido a utilizar dos nuevos paquetes que nos ayudan a desarrollar tareas relacionadas con la transformación del dato. En concreto, estos paquetes se centran en **dataframes** y **tibbles**, que son el tipo de estructura R con el que cualquier analista consume el 99% de su tiempo.

Las transformaciones que hemos abordado son:

- Transformaciones estructurales (formatos wide y long).
- Selección de columnas.
- Filtrado de muestras.
- Orden.
- Renombrado.
- Modificación.

Como ya os he adelantado, el siguiente fastbook es una continuación del presente. En él abordaremos nuevos conceptos de manipulación y transformación de dataframes para que seáis capaces de resolver retos analíticos desde la programación, que es, al fin y al cabo, el objetivo de esta asignatura.

En concreto, trabajaremos la agregación de datos y el cruce de tablas.

¡Enhорabuena! Fastbook superado

edix

Creamos Digital Workers