# # Pivotar datos

{#pivoting-data}

En la gestión de datos, se puede entender que el pivoteo se refiere a uno de los dos procesos:

1. La creación de tablas dinámicas, que son tablas de estadísticas que resumen los datos de una tabla más extensa
2. La conversión de una tabla de formato **largo** a formato **ancho**, o viceversa.

**En esta página, nos centraremos en la última definición.** La primera es un paso crucial en el análisis de datos, y se trata en las páginas [Agrupar datos](#grouping-data) y [Tablas descriptivas](#descriptive-tables).

En esta página se tratan los formatos de los datos. Es útil conocer la idea de "datos ordenados", en la que cada variable tiene su propia columna, cada observación tiene su propia fila y cada valor tiene su propia celda. Se puede encontrar más información sobre este tema [en este capítulo en línea de R for Data Science](https://r4ds.had.co.nz/tidy-data.html).

## Preparación

### Cargar paquetes

Este trozo de código muestra la carga de los paquetes necesarios para los análisis. En este manual destacamos p\_load() de **pacman**, que instala el paquete si es necesario y lo carga para su uso. También puede cargar los paquetes instalados con library() de **.** Consulta la página sobre [los fundamentos de R](#r-basics) para obtener más información sobre los paquetes de R.

### Importar datos

### Datos del recuento de la malaria

En esta página, utilizaremos unos datos ficticio de casos diarios de malaria, por centro y grupo de edad. Si quieres seguirlo, [clica aquí para descargarlo (como archivo .rds)](https://github.com/epirhandbook/Epi_R_handbook/raw/master/data/malaria_facility_count_data.rds). Importe los datos con la función import() del paquete **rio** (maneja muchos tipos de archivos como .xlsx, .csv, .rds - vea la página de [importación y exportación](#import-and-export) para más detalles).

A continuación se muestran las primeras 50 filas.

### Datos del caso Linelist

En la parte posterior de esta página, también utilizaremos los datos de casos de una epidemia de ébola simulada. Si quieres seguir el proceso, [clica aqui para descargar el listado "limpio"](https://github.com/epirhandbook/Epi_R_handbook/raw/master/data/case_linelists/linelist_cleaned.rds) (como archivo .rds). Importa tus datos con la función import() del paquete **rio** (acepta muchos tipos de archivos como .xlsx, .rds, .csv - vea la página de [importación y exportación](#import-and-export) para más detalles).

## De ancho a largo

### "Formato "ancho

Los datos suelen introducirse y almacenarse en un formato "amplio", en el que las características o respuestas de un sujeto se almacenan en una sola fila. Aunque esto puede ser útil para la presentación, no es ideal para algunos tipos de análisis.

Tomemos como ejemplo el set de datos count\_data importado en la sección Preparación. Puedes ver que cada fila representa un "centro-día". Los recuentos de casos reales (las columnas más a la derecha) se almacenan en un formato "amplio", de modo que la información de cada grupo de edad en un día determinado del centro se almacena en una sola fila.

Cada observación de este conjunto de datos se refiere a los recuentos de paludismo en una de las 65 instalaciones en una fecha determinada, que va desde count\_data$data\_date %>% min() hasta count\_data$data\_date %>% max(). Estas instalaciones están situadas en una provincia (Norte) y cuatro distritos (Spring, Bolo, Dingo y Barnard). El set de datos proporciona los recuentos globales de malaria, así como los recuentos específicos por edad en cada uno de los tres grupos de edad: <4 años, 5-14 años y 15 años o más.

Los datos "amplios" como éste no se ajustan a las normas de "datos ordenados", porque los encabezados de las columnas no representan realmente "variables", sino que representan valores de una hipotética variable "grupo de edad".

Este formato puede ser útil para presentar la información en una tabla, o para introducir datos (por ejemplo, en Excel) a partir de formularios de informes de casos. Sin embargo, en la etapa de análisis, estos datos normalmente deben ser transformados a un formato "más largo" más alineado con los estándares de "datos ordenados". El paquete R **ggplot2,** en particular, funciona mejor cuando los datos están en un formato "largo".

La visualización de los recuentos totales de malaria a lo largo del tiempo no plantea ninguna dificultad con los datos en su formato actual:

Sin embargo, ¿qué pasaría si quisiéramos mostrar las contribuciones relativas de cada grupo de edad a este recuento total? En este caso, necesitamos asegurarnos de que la variable de interés (grupo de edad), aparezca en set de datos en una sola columna que pueda pasarse al argumento aes() de {ggplot2} "mapping aesthetics".

### pivot\_longer()

La función **tidyr** pivot\_longer() hace que los datos sean "más largos". **tidyr** forma parte del **tidyverse** de los paquetes de R.

Acepta un rango de columnas para transformar (especificado a cols =). Por lo tanto, puede operar sólo en una parte de unos datos. Esto es útil para los datos de la malaria, ya que sólo queremos pivotar las columnas de recuento de casos.

En este proceso, terminará con dos "nuevas" columnas - una con las categorías (los antiguos nombres de las columnas), y otra con los valores correspondientes (por ejemplo, recuento de casos). Puedes aceptar los nombres por defecto para estas nuevas columnas, o puede especificar los suyos propios a names\_to = y values\_to = respectivamente.

Veamos pivot\_longer() en acción...

### Pivote estándar

Queremos utilizar la función pivot\_longer() de **tidyr** para convertir los datos "anchos" en un formato "largo". Concretamente, para convertir las cuatro columnas numéricas con datos sobre los recuentos de malaria en dos nuevas columnas: una que contenga los grupos de edad y otra que contenga los valores correspondientes.

Observe que el dataframe recién creado (df\_long) tiene más filas (12.152 frente a 3.038); se ha hecho más largo. De hecho, es precisamente cuatro veces más largo, porque cada fila de los datos originales representa ahora cuatro filas en df\_long, una para cada una de las observaciones de recuento de malaria (<4 años, 5-14 años, 15 años+ y total).

Además de ser más largo, el nuevo conjunto de datos tiene menos columnas (8 frente a 10), ya que los datos que antes se almacenaban en cuatro columnas (las que empiezan por el prefijo malaria\_) se almacenan ahora en dos.

Dado que los nombres de estas cuatro columnas comienzan con el prefijo malaria\_, podríamos haber hecho uso de la práctica función "tidyselect" starts\_with() para conseguir el mismo resultado (véase la página [Limpieza de datos y funciones básicas](#cleaning-data-and-core-functions) para conocer más sobre estas funciones de ayuda).

o por posición:

o por rango de nombres:

Estas dos nuevas columnas reciben los nombres por defecto de nombre y valor, pero podemos anular estos valores por defecto para proporcionar nombres más significativos, que pueden ayudar a recordar lo que se almacena dentro, utilizando los argumentos names\_to y values\_to. Utilicemos los nombres age\_group y counts:

Ahora podemos pasar este nuevo conjunto de datos a {ggplot2}, y asignar la nueva columna recuento al eje y y la nueva columna grupo\_de\_edad al argumento fill = (el color interno de la columna). Esto mostrará los recuentos de malaria en un gráfico de barras apilado, por grupo de edad:

Examina esta nueva trama y compárala con la que hemos creado antes: ¿qué ha fallado?

Nos hemos encontrado con un problema común al manejar los datos de vigilancia: hemos incluido también los recuentos totales de la columna malaria\_tot, por lo que la magnitud de cada barra en el gráfico es el doble de lo que debería ser.

Podemos manejar esto de varias maneras. Podríamos simplemente filtrar estos totales dset de datos antes de pasarlo a ggplot():

Como alternativa, podríamos haber excluido esta variable al ejecutar pivot\_longer(), manteniéndola así en set de datos como una variable independiente. Observa cómo se "expanden" sus valores para llenar las nuevas filas.

### Pivotar datos de múltiples tipos

El ejemplo anterior funciona bien en situaciones en las que todas las columnas que se quieren "pivotar más" son del mismo tipo (carácter, numérico, lógico...).

Sin embargo, habrá muchos casos en los que, en el trabajo de campo, se trabaje con datos preparados por personas no especializadas y que sigan su propia lógica no estándar - como señaló Hadley Wickham (haciendo referencia a Tolstoi) en su [artículo seminal](https://vita.had.co.nz/papers/tidy-data.pdf) sobre los principios de **Tidy Data**: "Como las familias, los conjuntos de datos ordenados son todos iguales, pero cada conjunto de datos desordenado es desordenado a su manera".

Un problema particularmente común que encontrará será la necesidad de pivotar columnas que contienen diferentes tipos de datos. Este pivote resultará en el almacenamiento de estos diferentes tipos de datos en una sola columna, lo cual no es una buena situación. Hay varios enfoques que uno puede tomar para separar el desorden que esto crea, pero hay un paso importante que puedes seguir usando pivot\_longer() para evitar crear tal situación tu mismo.

Tomemos una situación en la que ha habido una serie de observaciones en diferentes pasos de tiempo para cada uno de los tres elementos A, B y C. Ejemplos de estos elementos podrían ser individuos (por ejemplo, contactos de un caso de ébola que se rastrean cada día durante 21 días) o puestos de salud de aldeas remotas que se supervisan una vez al año para garantizar que siguen funcionando. Utilicemos el ejemplo del rastreo de contactos. Imaginemos que los datos se almacenan de la siguiente manera:

Como puede verse, los datos son un poco complicados. Cada fila almacena información sobre un elemento, pero con la serie temporal cada vez más alejada hacia la derecha a medida que avanza el tiempo. Además, los tipos de columnas alternan entre valores de fecha y caracteres.

Un ejemplo particularmente malo que encontró este autor fue el de los datos de vigilancia del cólera, en el que se añadieron 8 nuevas columnas de observaciones cada día en el transcurso de **4 años**. El simple hecho de abrir el archivo de Excel en el que se almacenaban estos datos me llevó más de 10 minutos en mi ordenador portátil.

Para trabajar con estos datos, necesitamos transformar el dataframe a formato largo, pero manteniendo la separación entre una columna de fecha y una columna de caracteres (estado), para cada observación de cada elemento. Si no lo hacemos, podríamos terminar con una mezcla de tipos de variables en una sola columna (un gran "no-no" cuando se trata de gestión de datos y de datos ordenados):

Arriba, nuestro pivote ha fusionado fechas y caracteres en una sola columna de valor. R reaccionará convirtiendo toda la columna en tipo carácter, y se pierde la utilidad de las fechas.

Para evitar esta situación, podemos aprovechar la estructura sintáctica de los nombres de las columnas originales. Hay una estructura de nombres común, con el número de observación, un guión bajo, y luego "estado" o "fecha". Podemos aprovechar esta sintaxis para mantener estos dos tipos de datos en columnas separadas después del pivote.

Para ello:

* Proporcionar un vector de caracteres al argumento names\_to =, siendo el segundo elemento (".value" ). Este término especial indica que las columnas pivotadas se dividirán basándose en un carácter de su nombre...
* También debe proporcionar el carácter de "división" al argumento names\_sep =. En este caso, es el guión bajo "\_".

Así, la denominación y división de las nuevas columnas se basa en el guión bajo de los nombres de las variables existentes.

**Toques finales**:

Ten en cuenta que la columna de fecha está actualmente en el tipo carácter - podemos convertirla fácilmente en tipo fecha utilizando las funciones mutate() y as\_date() descritas en la página [Trabajar con fechas](#working-with-dates-1).

También podemos querer convertir la columna de observación a un formato numérico eliminando el prefijo "obs" y convirtiendo a numérico. Podemos hacer esto con str\_remove\_all() del paquete **stringr** (véase la página [Caracteres y cadenas](#characters-and-strings)).

Y ahora, podemos empezar a trabajar con los datos en este formato, por ejemplo, trazando un mosaico de calor descriptivo:

## De largo a ancho

En algunos casos, es posible que queramos convertir unos datos a un formato más amplio. Para ello, podemos utilizar la función pivot\_wider().

Un caso de uso típico es cuando queremos transformar los resultados de un análisis en un formato que sea más digerible para el lector (como una [tabla para su presentación](#tables-for-presentation)). Por lo general, se trata de transformar unos datos en el que la información de un sujeto está repartida en varias filas en un formato en el que esa información se almacena en una sola fila.

### Datos

Para esta sección de la página, utilizaremos la lista de casos (véase la sección [Preparación](#pivot_prep)), que contiene una fila por caso.

Aquí están las primeras 50 filas:

Supongamos que queremos conocer los recuentos de individuos en los diferentes grupos de edad, por género:

Esto nos da un largo conjunto de datos que es genial para producir visualizaciones en **ggplot2**, pero no es ideal para la presentación en una tabla:

### Pivote más amplio

Por lo tanto, podemos utilizar pivot\_wider() para transformar los datos en un formato mejor para incluirlos como tablas en nuestros informes.

El argumento nombres\_desde especifica la columna de la que generar los nuevos nombres de columna, mientras que el argumento valores\_desde especifica la columna de la que tomar los valores para rellenar las celdas. El argumento id\_cols = es opcional, pero se puede proporcionar un vector de nombres de columnas que no deben ser pivotadas, y que por tanto identificarán cada fila.

Esta tabla es mucho más fácil de leer y, por tanto, mejor para incluirla en nuestros informes. Se puede convertir en una tabla bonita con varios paquetes, como **flextable** y **knitr**. Este proceso se elabora en la página [Tablas para la presentación](#tables-for-presentation).

## Llenar

En algunas situaciones después de un pivote, y más comúnmente después de un bind, nos quedan huecos en algunas celdas que nos gustaría rellenar.

### Datos

Por ejemplo, tome dos conjuntos de datos, cada uno con observaciones para el número de medición, el nombre del centro y el recuento de casos en ese momento. Sin embargo, el segundo conjunto de datos también tiene la variable Año.

Cuando realizamos un bind\_rows() para unir los dos conjuntos de datos, la variable Year se rellena con NA para aquellas filas en las que no había información previa (es decir, el primer conjunto de datos):

### llenar()

En este caso, Año es una variable útil para incluir, especialmente si queremos explorar las tendencias a lo largo del tiempo. Por lo tanto, utilizamos fill() para rellenar esas celdas vacías, especificando la columna a rellenar y la dirección (en este caso **hacia arriba**):

Alternativamente, podemos reordenar los datos para que tengamos que rellenar en sentido descendente:

Ahora tenemos unos datos útil para trazar:

Pero es menos útil para presentarlo en una tabla, así que practiquemos la conversión de este largo y desordenado dataframe en un dataframe más amplio y ordenado:

N.B. En este caso, tuvimos que especificar que sólo se incluyeran las tres variables Instalación, Año y Casos, ya que la variable adicional Medida interferiría en la creación de la tabla:

## Recursos

Aquí hay un [tutorial](https://datacarpentry.org/r-socialsci/03-dplyr-tidyr/index.html) útil