# # Agrupar datos

{#grouping-data}

Esta página cubre cómo agrupar y agregar datos para el análisis descriptivo. Hace uso de la familia de paquetes **tidyverse** para funciones comunes y fáciles de usar.

La agrupación de datos es un componente esencial de la gestión y el análisis de datos. Los datos agrupados se resumen estadísticamente por grupos y pueden representarse gráficamente por grupos. Las funciones del paquete **dplyr** (parte del **tidyverse**) facilitan la agrupación y las operaciones posteriores.

En esta página se tratarán los siguientes temas:

* Agrupar datos con la función group\_by()
* Datos no agrupados
* resumir() datos agrupados con estadísticas
* La diferencia entre count() y tally()
* arrange() aplicado a los datos agrupados
* filter() aplicado a los datos agrupados
* mutate() aplicada a los datos agrupados
* select() aplicado a datos agrupados
* El comando **básico** R aggregate() como alternativa

## 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

Importamos los datos de casos de una epidemia de ébola simulada. Si quieres seguirlo, [clica para descargar linelist "limpia"](https://github.com/epirhandbook/Epi_R_handbook/raw/master/data/case_linelists/linelist_cleaned.rds) (como archivo .rds). Los datos se importan mediante la función import() del paquete **rio**. Consulta la página sobre [importación y exportación](#import-and-export) para conocer las distintas formas de importar datos.

Las primeras 50 filas del listado:

## Agrupación

La función group\_by() de **dplyr** agrupa las filas por los valores únicos de la columna que se le especifica. Si se especifican varias columnas, las filas se agrupan por las combinaciones únicas de valores entre las columnas. Cada valor único (o combinación de valores) constituye un grupo. Los cambios posteriores en set de datos o los cálculos pueden realizarse en el contexto de cada grupo.

Por ejemplo, el siguiente comando toma linelist y agrupa las filas por valores únicos en la columna resultado, guardando la salida como un nuevo dataframe ll\_por\_resultado. La(s) columna(s) de agrupación se colocan dentro de los paréntesis de la función group\_by().

**Ten en cuenta que no hay ningún cambio perceptible en los datos** después de ejecutar group\_by(), hasta que se aplique otro verbo **de dplyr como** mutate(), summarise(), o arrange() en el dataframe "agrupado".

Sin embargo, puede "ver" las agrupaciones imprimiendo el dataframe. Al imprimir un dataframe agrupado, verá que se ha transformado en un [objeto de clasetibble](https://tibble.tidyverse.org/) que, al imprimirse, muestra qué agrupaciones se han aplicado y cuántos grupos hay, escrito justo encima de la fila de cabecera.

### Grupos únicos

**Los grupos creados reflejan cada combinación única de valores en las columnas de agrupación.**

Para ver los grupos y el número de filas en cada grupo, pase los datos agrupados a tally(). Para ver sólo los grupos únicos sin recuento puede pasar a group\_keys().

Vea a continuación que hay **tres** valores únicos en el resultado de la columna de agrupación: "Muerte", "Recuperación" y NA. Véase que hubo nrow(linelist %>% filter(outcome == "Death")) muertes, nrow(linelist %>% filter(outcome == "Recover")) recuperaciones, y nrow(linelist %>% filter(is.na(outcome)) sin resultado registrado.

Se puede agrupar por más de una columna. A continuación, el dataframe se agrupa por resultado y género, y luego se cuenta. Observe cómo cada combinación única de resultado y género se registra como su propio grupo, incluyendo los valores faltantes para cualquier columna.

### Nuevas columnas

También puede crear una nueva columna de agrupación dentro de la sentencia group\_by(). Esto equivale a llamar a mutate() antes de group\_by(). Para una tabulación rápida este estilo puede ser útil, pero para una mayor claridad en su código considere crear esta columna en su propio paso mutate() y luego canalizarla a group\_by().

### Añadir/soltar columnas de agrupación

Por defecto, si ejecuta group\_by() sobre datos que ya están agrupados, se eliminarán los grupos antiguos y se aplicarán los nuevos. Si desea añadir nuevos grupos a los existentes, incluya el argumento .add = TRUE.

\*\* Mantener todos los grupos\*\*

Si se agrupa en una columna de de tipo factor, puede haber niveles del factor que no estén presentes en los datos. Si agrupa en esta columna, por defecto esos niveles no presentes se descartan y no se incluyen como grupos. Para cambiar esto de manera que todos los niveles aparezcan como grupos (incluso si no están presentes en los datos), establezca .drop = FALSE en su comando group\_by().

## Desagregue

Los datos que han sido agrupados permanecerán agrupados hasta que sean específicamente desagrupados mediante ungroup(). Si se olvida de desagrupar, puede dar lugar a cálculos incorrectos. A continuación se muestra un ejemplo de eliminación de todas las agrupaciones:

También puede eliminar la agrupación sólo para columnas específicas, colocando el nombre de la columna dentro de ungroup().

**NOTA:** El verbo count() desagrupa automáticamente los datos después del recuento.

## Resumir

Véase la sección **dplyr** de la página [Tablas descriptivas](#descriptive-tables) para una descripción detallada de cómo producir tablas de resumen con summarise(). Aquí abordamos brevemente cómo cambia su comportamiento cuando se aplica a datos agrupados.

La función **dplyr** summarise() (o summarize()) toma un dataframe y lo convierte en un nuevo dataframe de resumen, con columnas que contienen los estadísticos de resumen que definas. En un dataframe sin agrupar, las estadísticas de resumen se calcularán a partir de todas las filas. La aplicación de summarise() a los datos agrupados produce esas estadísticas de resumen para cada grupo.

La sintaxis de summarise() es tal que se proporciona el nombre de la(s) **nueva(s)** columna(s) de resumen, un signo de igualdad y, a continuación, una función estadística para aplicar a los datos, como se muestra a continuación. Por ejemplo, min(), max(), median() o sd(). Dentro de la función estadística, indique la columna con la que se va a operar y cualquier argumento relevante (por ejemplo, na.rm = TRUE). Puedes utilizar sum() para contar el número de filas que cumplen un criterio lógico (con doble igual ==).

A continuación se muestra un ejemplo de summarise() aplicado sin datos agrupados. Las estadísticas devueltas se producen a partir del set de datos completo.

Por el contrario, a continuación se muestra la misma sentencia summarise() aplicada a los datos agrupados. Las estadísticas se calculan para cada grupo de resultados. Observe cómo las columnas de agrupación se trasladan al nuevo dataframe.

**SUGERENCIA:** La función summarise funciona tanto con la ortografía del Reino Unido como con la de EE.UU. - summarise() y summarize() llaman a la misma función.

## Recuentos y cuentas

count() y tally() proporcionan una funcionalidad similar pero son diferentes. Lea más sobre la distinción entre tally() y count() [aquí](https://dplyr.tidyverse.org/reference/tally.html)

### tally()

tally() es la abreviatura de summarise(n = n()), y no agrupa los datos. Por lo tanto, para lograr recuentos agrupados debe seguir un comando group\_by(). Puedes añadir sort = TRUE para ver primero los grupos más grandes.

### contar()

En cambio, count() hace lo siguiente:

1. aplica group\_by() a la(s) columna(s) especificada(s)
2. aplica summarise() y devuelve la columna n con el número de filas por grupo
3. aplica ungroup()

Al igual que con group\_by() puede crear una nueva columna dentro del comando count():

count() puede llamarse varias veces, con la funcionalidad "combinada". Por ejemplo, para resumir el número de hospitales presentes para cada género, ejecuta lo siguiente. Ten en cuenta que el nombre de la columna final se ha cambiado de "n" por defecto para mayor claridad (con nombre =).

### Añadir recuentos

A diferencia de count() y summarise(), puede utilizar add\_count() para añadir una nueva columna n con los recuentos de filas por grupo, conservando todas las demás columnas del dataframe.

Esto significa que el número de recuento de un grupo, en la nueva columna n, se imprimirá en cada fila del grupo. Para fines de demostración, añadimos esta columna y luego reordenamos las columnas para facilitar la visualización. Consulta la sección siguiente sobre el [filtro del tamaño del grupo](#group_filter_grp_size) para ver otro ejemplo.

### Añadir los totales

Para añadir fácilmente filas o columnas de suma total después de utilizar tally() o count(), consulta la sección del **conserje de** la página [Tablas descriptivas](#tbl_janitor). Este paquete ofrece funciones como adorn\_totals() y adorn\_percentages() para añadir totales y convertirlos para mostrar porcentajes. A continuación se muestra un breve ejemplo:

Para añadir filas de totales más complejas que incluyan estadísticas de resumen distintas de las sumas, consulta [esta sección de la página Tablas descriptivas](#tbl_dplyr_totals).

## Agrupación por fecha

Al agrupar datos por fecha, debe tener (o crear) una columna para la unidad de fecha de interés - por ejemplo "día", "epiweek", "mes", etc. Puedes crear esta columna utilizando floor\_date() de **lubridate**, como se explica en la [sección Semanas epidémicas de](#dates_epi_wks) la página [Trabajar con fechas](#working-with-dates-1). Una vez que tenga esta columna, puede utilizar count() de **dplyr** para agrupar las filas por esos valores de fecha únicos y lograr recuentos agregados.

Un paso adicional común para las situaciones de fechas, es "rellenar" cualquier fecha en la secuencia que no esté presente en los datos. Utiliza complete() de **tidyr** para que la serie de fechas agregadas esté completa, incluyendo todas las unidades de fecha posibles dentro del rango. Sin este paso, una semana sin casos reportados podría no aparecer en tus datos.

Dentro de complete() redefine su columna de fecha como una secuencia de fechas seq.Date() desde el mínimo hasta el máximo - así las fechas se expanden. Por defecto, los valores del recuento de casos en cualquier nueva fila "expandida" serán NA. Puedes establecerlos a 0 utilizando el argumento fill = de complete(), que espera una lista con nombre (si su columna de recuentos se llama n, proporcione fill = list(n = 0). Consulta ?complete para obtener más detalles y la página [Trabajar con fechas](#dates_epi_wks) para ver un ejemplo.

### Lista de casos en días

Aquí hay un ejemplo de agrupación de casos en días sin usar complete(). Obsérvese que las primeras filas omiten las fechas sin casos.

A continuación añadimos el comando complete() para asegurarnos de que todos los días del rango están representados.

### Lista de casos en semanas

Se puede aplicar el mismo principio para las semanas. Primero cree una nueva columna que sea la semana del caso utilizando floor\_date() con unidad = "semana". A continuación, utiliza count() como en el caso anterior para obtener los recuentos de casos semanales. Termine con complete() para asegurarse de que todas las semanas están representadas, incluso si no contienen casos.

Aquí están las primeras 50 filas del dataframe resultante:

### Lista de casos en meses

Para agregar casos en meses, vuelva a utilizar floor\_date() del paquete **lubridate**, pero con el argumento unidad = "meses". Esto redondea cada fecha hacia abajo al día 1 de su mes. La salida será el tipo Date. Ten en cuenta que en el paso complete() también utilizamos by = "months".

### Recuentos diarios en semanas

Para agregar los recuentos diarios en recuentos semanales, utiliza floor\_date() como arriba. Sin embargo, utiliza group\_by() y summarize() en lugar de count() porque necesita sumar() los recuentos de casos diarios en lugar de limitarse a contar el número de filas por semana.

#### Recuentos diarios en meses

Para agregar los recuentos diarios en recuentos de meses, utiliza floor\_date() con unidad = "mes" como en el caso anterior. Sin embargo, utiliza group\_by() y summarize() en lugar de count() porque necesita sumar() los recuentos de casos diarios en lugar de limitarse a contar el número de filas por mes.

## Ordenar los datos agrupados

El uso del verbo **dplyr** arrange() para ordenar las filas de un dataframe se comporta igual cuando los datos están agrupados, a menos que se establezca el argumento .by\_group =TRUE. En este caso, las filas se ordenan primero por las columnas de agrupación y luego por cualquier otra columna que se especifique en arrange().

## Filtro de datos agrupados

### filtrar()

Cuando se aplica junto con funciones que evalúan el dataframe (como max(), min(), mean()), estas funciones se aplicarán ahora a los grupos. Por ejemplo, si desea filtrar y mantener las filas en las que los pacientes están por encima de la edad media, esto se aplicará ahora por grupo, filtrando para mantener las filas por encima de la edad media del grupo.

### Filas de corte por grupo

La función **dplyr** slice(), que [filtra las filas según su posición](https://dplyr.tidyverse.org/reference/slice.html) en los datos, también puede aplicarse por grupo. Recuerde que debe tener en cuenta la ordenación de los datos dentro de cada grupo para obtener la "rebanada" deseada.

Por ejemplo, para recuperar sólo los últimos 5 ingresos de cada hospital:

1. Agrupar linelist por columna hospitalaria
2. Ordenar los registros de la fecha de hospitalización más reciente a la más antigua dentro de cada grupo de hospitales
3. Rebanada para recuperar las 5 primeras filas de cada hospital

slice\_head() - selecciona n filas de la parte superior  
slice\_tail() - selecciona n filas del final  
slice\_sample() - selecciona aleatoriamente n filas  
slice\_min() - selecciona n filas con los valores más altos en order\_by = columna, usa with\_ties = TRUE para mantener los empates  
slice\_max() - selecciona n filas con los valores más bajos en order\_by = columna, utiliza with\_ties = TRUE para mantener los empates

Consulta la página de [desduplicación](#de-duplication) para ver más ejemplos y detalles sobre slice().

### Filtro por tamaño de grupo

La función add\_count() añade una columna n a los datos originales dando el número de filas en el grupo de esa fila.

A continuación, add\_count() se aplica a la columna hospital, por lo que los valores de la nueva columna n reflejan el número de filas del grupo de hospitales de esa fila. Observe cómo se repiten los valores de la columna n. En el ejemplo siguiente, el nombre de la columna n podría cambiarse utilizando name = dentro de add\_count(). Para fines de demostración reordenamos las columnas con select().

De este modo, resulta fácil filtrar los casos que fueron hospitalizados en un hospital "pequeño", por ejemplo, un hospital que admitió a menos de 500 pacientes:

## Mutar en datos agrupados

Para conservar todas las columnas y filas (no resumir) y añadir una nueva columna que contenga estadísticas de grupo, utiliza mutate() después de group\_by() en lugar de summarise().

Esto es útil si se desea obtener estadísticas de grupo en los datos originales con todas las demás columnas presentes, por ejemplo, para los cálculos que comparan una fila con su grupo.

Por ejemplo, este código calcula la diferencia entre la demora en el ingreso de una fila y la demora media de su hospital. Los pasos son:

1. Agrupar los datos por hospital
2. Utiliza la columna days\_onset\_hosp (retraso hasta la hospitalización) para crear una nueva columna que contenga el retraso medio en el hospital de esa fila
3. Calcular la diferencia entre las dos columnas

Seleccionamos() sólo ciertas columnas para mostrarlas, con fines de demostración.

## Seleccionar en datos agrupados

El verbo select() funciona con datos agrupados, pero las columnas de agrupación siempre se incluyen (aunque no se mencionen en select()). Si no desea estas columnas de agrupación, utiliza primero ungroup().

## Recursos

A continuación, algunos recursos útiles para obtener más información:

Puedes realizar cualquier función de resumen sobre datos agrupados; consulta la [hoja de trucos de transformación de datos de RStudio](https://github.com/rstudio/cheatsheets/blob/master/data-transformation.pdf)

La página de Data Carpentry sobre [**dplyrLas**](https://datacarpentry.org/R-genomics/04-dplyr.html) páginas de referencia de **tidyverse** sobre [group\_by()](https://dplyr.tidyverse.org/reference/group_by.html) y [agrupación](https://dplyr.tidyverse.org/articles/grouping.html)

Esta página sobre [Manipulación de datos](https://itsalocke.com/files/DataManipulationinR.pdf)

[Resumir con condiciones en dplyr](https://stackoverflow.com/questions/23528862/summarize-with-conditions-in-dplyr)