# #Conceptos básicos de ggplot

**{#ggplot-basics}**

**ggplot2** es el paquete de R más popular para la visualización de datos. Su función ggplot() es el núcleo de este paquete, y todo este enfoque se conoce coloquialmente como "ggplot", con las figuras resultantes a veces llamadas afectuosamente "ggplots". El "gg" en estos nombres se refiere a la "**g**ramática de los **g**ráficos" utilizada para construir las figuras. **ggplot2** se beneficia de una amplia variedad de paquetes R complementarios que mejoran aún más su funcionalidad.

La sintaxis es significativamente diferente de **la de** R base, y tiene una curva de aprendizaje asociada. El uso de **ggplot2** generalmente requiere que el usuario formatee sus datos de una manera que sea altamente compatible con **tidyverse**, lo que en última instancia hace que el uso conjunto de estos paquetes sea muy eficaz.

En esta página cubriremos los fundamentos de la creación de graficos con **ggplot2**. Consulta la página de [consejos de ggplot](#ggplot-tips) para sugerencias y técnicas avanzadas para lograr que sus gráficos se vean realmente bien.

Hay varios tutoriales extensos de **ggplot2** enlazados en la sección de recursos. También puede descargar esta [hoja de trucos de visualización de datos con ggplot](https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf) desde el sitio web de RStudio. Si quieres inspirarte en formas de visualizar tus datos de forma creativa, te sugerimos que revises sitios web como la [galería de gráficos de R](https://www.r-graph-gallery.com/) y [Data-to-viz](https://www.data-to-viz.com/caveats.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 podes cargar los paquetes instalados con library() de R **base.** Consulta la página sobre [los fundamentos de R](#r-basics) para obtener más información sobre los paquetes de R.

### Importar datos

Importamos el conjunto de datos de casos de una epidemia de Ebola simulada. Si quieres seguir el proceso, [cliqua para descargar linelist "limpia"](https://github.com/epirhandbook/Epi_R_handbook/raw/master/data/case_linelists/linelist_cleaned.rds) (como archivo .rds). Para importar sus datos utilizando la función import() del paquete **rio** (acepta muchos tipos de archivos como .xlsx, .rds, .csv - consulta la página de [importación y exportación](#import-and-export) para más detalles).

A continuación se muestran las primeras 50 filas del listado. Nos centraremos en las variables continuas edad, wt\_kg (peso en kilos), ct\_blood (valores de CT) y days\_onset\_hosp (diferencia entre la fecha de inicio de síntomas y la hospitalización).

### Limpieza general

Cuando se preparan los datos para trazarlos (graficarlos), lo mejor es hacer que los datos se adhieran a [los estándares de datos "ordenados" tanto como](https://r4ds.had.co.nz/tidy-data.html) sea posible. En las páginas de gestión de datos de este manual, como [Limpieza de datos y funciones básicas, se explica](#cleaning-data-and-core-functions) cómo conseguirlo.

Algunas formas sencillas de preparar nuestros datos para que sea mas fáciles trazarlos pueden incluir la mejora del contenido de los datos para su visualización, lo que no equivale necesariamente a una manipulación de los datos mas sencilla. Por ejemplo:

* Sustituye los valores NA de una columna de caracteres por la cadena de caracteres "Desconocido"
* Considera la posibilidad de convertir la columna en de tipo factor para que sus valores tengan niveles ordinales prescritos
* Limpia los valores de algunas columnas para cambiar texto "amigable con los datos" con barra baja, etc. a texto normal o a mayúsculas y minúsculas (ver [Caracteres y cadenas](#characters-and-strings))

He aquí algunos ejemplos de esto en acción:

### Pivotar a lo largo

Como una cuestión de estructura de datos, para **ggplot2** a menudo queremos pivotar nuestros datos en formatos largos. Lea más sobre esto en la página de [Pivoteo de datos](#pivoting-data).

Por ejemplo, digamos que queremos trazar datos que están en un formato "a lo ancho", como por ejemplo para cada caso en linelist y sus síntomas. A continuación creamos una minilista llamada symptoms\_data que contiene sólo las columnas case\_id y symptoms.

Así es como se ven las primeras 50 filas de esta minilista - ¿ves cómo están formateadas "a lo ancho" con cada síntoma como una columna?

Si quisiéramos trazar el número de casos con síntomas específicos, estamos limitados por el hecho de que cada síntoma es una columna específica. Sin embargo, podemos hacer pivotar las columnas de síntomas a un formato más largo como este:

Aquí están las primeras 50 filas. Observe que cada caso tiene 5 filas - una para cada síntoma posible. Las nuevas columnas symptom\_name y symptom\_is\_present son el resultado del pivote. Ten en cuenta que este formato puede no ser muy útil para otras operaciones, pero es útil para trazar.

## Fundamentos de ggplot

**"Gramática de los gráficos" - ggplot2**

El trazado con **ggplot2** se basa en "añadir" capas de trazado y elementos de diseño unos sobre otros, añadiendo cada comando a los anteriores con un símbolo de suma (+). El resultado es un objeto de trazado multicapa que se puede guardar, modificar, imprimir, exportar, etc.

Los objetos ggplot pueden ser muy complejos, pero el orden básico de las capas suele ser el siguiente:

1. Comenza con el comando ggplot() de la línea de base - esto "abre" el ggplot y permite agregar las funciones subsecuentes con +. Normalmente, el conjunto de datos también se especifica en este comando
2. Añadí capas "geom" - estas funciones visualizan los datos como geometrías (formas), por ejemplo, como un gráfico de barras, un gráfico de líneas, un gráfico de dispersión, un histograma (¡o una combinación!). Todas estas funciones comienzan con geom\_ como prefijo.
3. Añadí elementos de diseño al gráfico, como etiquetas de ejes, título, fuentes, tamaños, esquemas de color, leyendas o rotación de ejes.

Un ejemplo sencillo de código esqueleto es el siguiente. Explicaremos cada componente en las secciones siguientes.

## ggplot()

El comando de apertura de cualquier gráfico ggplot2 es ggplot(). Este comando simplemente crea un lienzo en blanco sobre el que añadir capas. Se "abre" el camino para añadir más capas con un símbolo +.

Normalmente, el comando ggplot() incluye el argumento data = para el gráfico. Esto establece el conjunto de datos que se utilizará de manera predeterminada para las capas posteriores del gráfico.

Este comando terminará con un + después de su paréntesis de cierre. Esto deja el comando "abierto". El ggplot sólo se ejecutará/aparecerá cuando el comando completo incluya una capa final sin un + al final.

## Geoms

Un lienzo en blanco no es suficiente: necesitamos crear geometrías (formas o tipos de grafico) a partir de nuestros datos (por ejemplo, gráficos de barras, histogramas, gráficos de dispersión, gráficos de caja).

Esto se hace añadiendo capas "geoms" al comando inicial ggplot(). Hay muchas funciones de **ggplot2** que crean "geoms". Cada una de estas funciones comienza con "geom\_", por lo que nos referiremos a ellas genéricamente como geom\_XXXX(). Hay más de 40 geoms disponibles en **ggplot2** y muchos otros creados por aficionados. Véalos en la [galería de ggplot2](https://exts.ggplot2.tidyverse.org/gallery/). Algunos geoms de uso comun se enumeran a continuación:

* Histogramas - geom\_histogram()
* Gráficos de barras - geom\_bar() o geom\_col() (véase [la sección "Gráfico de barras"](#ggplot_basics_bars))
* Gráficos de caja - geom\_boxplot()
* Puntos (por ejemplo, gráficos de dispersión) - geom\_point()
* Gráficos de líneas - geom\_line() o geom\_path()
* Líneas de tendencia - geom\_smooth()

En un gráfico se pueden exponer uno o varios geoms. Cada uno se añade a los comandos anteriores **de ggplot2** con un +, y se agregan secuencialmente de manera que los geoms posteriores se trazan encima de los anteriores.

## Asignación de datos al gráfico

A la mayoría de las funciones geom hay que darle instrucciones sobre qué elementos utilizar para crear sus formas, por lo que hay que indicarles cómo se deben asignar las columnas de los datos a los distintos componentes del gráfico, como los ejes, los colores de las formas, los tamaños de las formas, etc. Para la mayoría de las funciones geom, los componentes esenciales que deben asignarse a las columnas de los datos son el eje x y (si es necesario) el eje y.

Este "mapeo" (o asignación) se produce con el argumento mapping =. Los mapeos que proporcione a mapping deben estar envueltos en la función aes(), por lo que hay que escribir algo como mapping = aes(x = col1, y = col2), como se muestra a continuación.

A continuación, en el comando ggplot() los datos se identifican utilizando el termino linelist . En el argumento mapping = aes() la columna age se asigna al eje x, y la columna wt\_kg se asigna al eje y.

Después de agregar un +, los comandos de trazado pueden continar. Se crea una forma o tipo de grafico con la función de "geom" denominada geom\_point(). Este geom hereda los mapeos del comando ggplot() anterior - conoce las asignaciones eje-columna y procede a visualizar esas relaciones como puntos en el lienzo.

Otro ejemplo que presentamos a continuación demuestra el uso de los mismos datos pero conun mapeo ligeramente diferente y utilizando un geom diferente. Ahora utilizamos la función geom\_histograma() que sólo requiere una columna mapeada en el eje x, ya que el eje y de conteo de casos (‘count’) se genera automáticamente.

### Estética del grafico

En la terminología de ggplot, la "estética" de un gráfico tiene un significado específico. Se refiere a una propiedad visual de los datos trazados. Ten en cuenta que "estética" aquí se refiere a los datos que se trazan en geoms / formas - no a lo que aparece en la periferia, tales como títulos, etiquetas de los ejes, el color de fondo, como podría comúnmente asociarse con la palabra "estética". En ggplot esos detalles se llaman "temas" y se ajustan dentro de un comando denominado theme() (ver [esta sección](#ggplot_basics_themes)).

Por lo tanto, la estética de los objetos de ploteo puede ser colores, tamaños, transparencias, colocación, etc. de los datos ploteados. No todos los geoms tendrán las mismas opciones estéticas, pero muchas pueden ser utilizadas por la mayoría de los geoms. He aquí algunos ejemplos:

* shape = Representar un punto con geom\_point() con forma de punto, estrella, triángulo o cuadrado...
* fill = El color interior (por ejemplo, de una barra o boxplot)
* color = El color de la línea exterior o borde de una barra, boxplot, etc., o el color del punto si se utiliza geom\_point()
* size = El tamaño (por ejemplo, grosor de línea, tamaño de punto)
* alpha = Transparencia (1 = opaco, 0 = invisible)
* binwidth = Ancho de los bins (o cubos) del histograma
* width = Ancho de las columnas del "diagrama de barras"
* linetype = Tipo de línea (por ejemplo, sólida, discontinua, punteada)

A esta estética de los objetos del graficose le pueden asignar valores de dos maneras:

1. Se asigna un valor estático (por ejemplo, color = "blue") que se aplica a todas las observaciones trazadas
2. Se asigna a una columna de los datos (por ejemplo, color = hospital) de manera que la visualización de cada observación depende de su valor en esa columna

### Asignar un valor estático

Si se desea que la estética del objeto de trazado sea estática, es decir, que sea la misma para cada observación de los datos, se escribe su asignación dentro del geom pero fuera del comando mapping = aes(). Estas asignaciones podrían escribirse como size = 1 o color = "blue". Aquí hay dos ejemplos:

* En el primer ejemplo, el mapping = aes() está en el comando ggplot() y los ejes se asignan a las columnas de edad (age) y peso (wt\_kg) en los datos. La estética del gráfico color =, size =, y alpha = (transparencia) se asignan a valores estáticos. Aclaramos que la asignación de valores estéticos de naturaleza estática se hace en la función geom\_point(), ya que se pueden añadir otros geoms después que tomarían valores estéticos diferentes
* En el segundo ejemplo, el histograma requiere sólo el eje x mapeado a una columna. El binwidth = (el ancho de los cubos), el color = (el color del borde de los cubos), el fill = (color interno o color de relleno de los cubos), y el alpha = (la transparencia del color de los cubos) se establecen dentro del geom como valores estáticos.

### Escalado a los valores de la columna

Como alternativa al uso de estéticas de naturaleza estática, se pueden graficar objetos con tamaños proporcionales a sus valores como aparecen en su respectiva columna. Con este enfoque, la visualización de esta estética dependerá del valor de esa observación en la columna de datos correspondiente. Si los valores de la columna son continuos, la escala de visualización (en la leyenda) para esa estética será continua. Si los valores de la columna son discretos, la leyenda mostrará cada valor y los datos trazados aparecerán claramente "agrupados" (lea más en la sección de [agrupación](#ggplotgroups) de esta página).

Para conseguir esto, se asigna esa estética de gráfico a un nombre de columna o variable (sin utilizar comillas). Esto debe hacerse dentro del comando *mapping = aes()* (nota: hay varios lugares en el código donde puedes hacer estas asignaciones de mapeo, como se discute [a continuación](##ggplot_basics_map_loc)).

Presentamos dos ejemplos a continuación.

* En el primer ejemplo, la estética d color = (de cada punto) está mapeada a la columna age - ¡y ha aparecido una escala en una leyenda! Por ahora sólo hay que tener en cuenta que la escala existe - mostraremos cómo modificarla en secciones posteriores.
* En el segundo ejemplo, dos nuevas estéticas de trazado se asignan a columnas (color = y size =), mientras que las estéticas de trazado shape = y alpha = se asignan a valores estáticos fuera de cualquier función de mapping = aes().

Nota: Las asignaciones de los ejes siempre se asignan a las columnas de los datos o variables (no a los valores estáticos), y esto se hace siempre dentro de mapping = aes().

Es importante mantener un seguimiento de las capas y las estéticasse hacen gráficos más complejos, por ejemplo, gráficos con múltiples geom. En el ejemplo siguiente, la estetica size = se asigna dos veces - una para geom\_point() y otra para geom\_smooth() - ambas veces como un valor estático.

### Dónde hacer las asignaciones

La asignación de estéticas dentro de mapping = aes() puede hacerse en varios lugares en sus comandos e incluso puede escribirse más de una vez. Esto puede ser escrito en el comando ggplot() inicial, y/o en cada geom individual debajo. Los matices incluyen:

* Las asignaciones de estéticas realizadas en el comando ggplot() inicial se heredarán por defecto en cualquier geom a continuación, al igual que se heredan x = e y =
* Las asignaciones realizadas dentro de un geom se aplican sólo a ese geom

Del mismo modo, el comando data = especificado en el ggplot() inicial se aplicará por defecto a cualquier geom que se agregue a continuacion, pero también se podrían especificar datos para cada geom (pero esto es más difícil).

Así, cada uno de los siguientes comandos creará el mismo gráfico:

### Grupos

Puedes agrupar fácilmente los datos y "graficar por grupo". De hecho, ¡ya lo has hecho!

Asigna la columna que quieres agrupar a la estética adecuada, dentro del comando mapping = aes(). Arriba, hemos demostrado esto usando valores continuos cuando asignamos el tamaño del punto usando size = a la columna age. Sin embargo, esto funciona de la misma manera con columnas o variables discretas/categóricas.

Por ejemplo, si quieres agrupar los puntos por género asignándole un color distinto a cada genero, deberás establecer mapping = aes(color = gender). Automáticamente aparecerá una leyenda. Esta asignación puede hacerse dentro de mapping = aes() en el comando ggplot() inicial (y ser heredado por el geom), o podría asignarse dentro de mapping = aes() escrito dentro del comando de geom. Ambos enfoques se muestran a continuación:

Tené en cuenta que dependiendo del tipo de geom, tendras que utilizar diferentes argumentos para agrupar los datos. Para geom\_point() lo más probable es que tengas que utilizar color =, shape = o size =. Mientras que para geom\_bar() es más probable que utilices fill =. Esto dependera del tipo de geom y de la estética del gráfico que deses usar para reflejar las agrupaciones.

Para tu información - la forma más básica de agrupar los datos es utilizando sólo el argumento group = dentro de mapping = aes(). Sin embargo, esto por sí mismo no cambiará los colores, el relleno o las formas. Tampoco creará una leyenda. Sin embargo, los datos están agrupados, por lo que las visualizaciones estadísticas puedran verse afectadas.

Para ajustar el orden de los grupos en un gráfico, consulte la página de [consejos de ggplot](#ggplot-tips) o la página sobre [Factores](#factors). Hay muchos ejemplos de gráficos agrupados en las secciones siguientes sobre el trazado de datos continuos y categóricos.

## Facetas / Múltiplos pequeños

Las facetas, o "pequeños gráficos múltiples", se utilizan para dividir un gráfico en una figura de varios paneles, con un panel ("faceta") representando un grupo de datos. El mismo tipo de gráfico se crea varias veces, cada vez utilizando un subgrupo del mismo conjunto de datos.

El facetado es una funcionalidad que viene con **ggplot2**, por lo que las leyendas y los ejes de los "paneles" facetados se alinean automáticamente. Hay otros paquetes que abordamos en la página de [consejos de ggplot](#ggplot-tips) que se utilizan para combinar gráficos representando conjuntos de datos completamente diferentes (**cowplot** y **patchwork**) en una figura.

El facetado se realiza con una de las siguientes funciones de **ggplot2**:

1. facet\_wrap() Para mostrar un panel diferente para cada nivel de una unica variable. Un ejemplo de esto podría ser mostrar una curva de epidemia diferente para cada hospital de una región. Las facetas se ordenan alfabéticamente, a menos que la variable sea un factor con otro orden definido.

* Puedes invocar ciertas opciones para determinar la disposición de las facetas, por ejemplo, nrow = 1 o ncol = 1 para controlar el número de filas o columnas en las que se organizan los gráficos con facetas.

1. facet\_grid() Se utiliza cuando se quiere introducir una segunda variable en la disposición de las facetas. Aquí cada panel de una cuadrícula muestra la intersección entre los valores de dos columnas. Por ejemplo, las curvas epidémicas para cada combinación hospital-grupo de edad con los hospitales en la parte superior (columnas) y los grupos de edad en los lados (filas).

* nrow y ncol no son relevantes, ya que los subgrupos se presentan en una cuadrícula

Cada una de estas funciones acepta una sintaxis de fórmula para especificar la(s) columna(s) para el facetado. Ambas aceptan hasta dos columnas, una a cada lado de la tilde ~.

* Para facet\_wrap() lo más frecuente es escribir una sola columna precedida de una tilde ~ como facet\_wrap(~hospital). Sin embargo, puedes escribir dos columnas facet\_wrap(outcome~hospital) - cada combinación única se mostrará en un panel separado, pero no se organizarán en una cuadrícula. Los encabezados mostrarán los términos combinados y éstos no tendrán una lógica específica para las columnas frente a las filas. Si quieres proporcionar una sóla variable de facetado, debes utilizar un punto . como marcador de posición en el otro lado de la fórmula - mira los ejemplos de código.
* Para facet\_grid() también puedes especificar una o dos columnas en la fórmula (rows~ columns). Si sólo quieres especificar una, puedes colocar un punto . al otro lado de la tilde como facet\_grid(. ~ hospital) o facet\_grid(hospital ~ .).

Las facetas pueden contener rápidamente una cantidad abrumadora de información, por lo que conviene asegurarse de no tener demasiados niveles de cada variable por la que se elija hacer la faceta. He aquí algunos ejemplos rápidos con el conjunto de datos sobre la malaria (véase [Descargar el manual y los datos](#download-handbook-and-data)), que consiste en el recuento diario de casos de malaria en los centros, por grupos de edad.

A continuación importamos y hacemos algunas modificaciones rápidas para simplificar la tarea:

A continuación se muestran las primeras 50 filas de los datos sobre la malaria. Observa que hay una columna malaria\_tot, pero también columnas para los recuentos por grupo de edad (que se utilizarán en el segundo ejemplo de facet\_grid()).

### facet\_wrap()

Por el momento, vamos a centrarnos en las columnas malaria\_tot y Distrito. Ignoremos por ahora las columnas de recuento por edad. Trazaremos las curvas epidémicas con geom\_col(), que produce una columna para cada día a la altura del eje y especificada en la columna malaria\_tot (los datos ya son recuentos diarios, por lo que utilizamos geom\_col() - véase [la sección "Diagrama de barras" más adelante](#ggplot_basics_bars)).

Cuando añadimos el comando facet\_wrap(), especificamos una tilde y a continuación la columna sobre la que hacer la faceta (District en este caso). Podes colocar otra columna a la izquierda de la tilde, - esto creará una faceta para cada combinación - pero le recomendamos que lo haga con facet\_grid() en su lugar. En este caso, se crea una faceta para cada valor único de District.

### facet\_grid()

Podemos utilizar un enfoque de facet\_grid() para cruzar dos variables. Digamos que queremos cruzar District y edad. Bien, necesitamos hacer algunas transformaciones de datos en las columnas de edad para poner estos datos en el formato "largo" preferido por ggplot. Los grupos de edad tienen sus propias columnas - los queremos en una sola columna llamada age\_group y otra llamada num\_cases. Consulte la página sobre [Pivoteo de datos](#pivoting-data) para obtener más información sobre este proceso.

Ahora las primeras 50 filas de datos tienen este aspecto:

Cuando se asignan las dos variables a facet\_grid(), lo más fácil es utilizar la notación de fórmula (por ejemplo, x ~ y) donde x son filas e y son columnas. Aquí está el gráfico, utilizando facet\_grid() que muestra los gráficos para cada combinación de las columnas age\_group y District.

### Ejes libres o fijos

Las escalas de los ejes que se muestran en gráficos facetados son por defecto las mismas (fijas) en todas las facetas. Esto es útil para las comparaciones cruzadas, pero no siempre es apropiado.

Al utilizar facet\_wrap() o facet\_grid(), podemos añadir scales = "free\_y" para "liberar" los ejes y de los paneles para que se ajuste la escala adecuadamente en relación a su subconjunto de datos. Esto es particularmente útil si los recuentos reales son pequeños para una de las subcategorías y las tendencias son difíciles de ver. En lugar de "free\_y" también podemos escribir "free\_x" para hacer lo mismo con el eje x (por ejemplo, para las fechas) o "free" para liberar ambos ejes. Ten en cuenta que en facet\_grid, las escalas de y serán las mismas para las facetas en la misma fila, y las escalas de x serán las mismas para las facetas en la misma columna.

Cuando se utiliza facet\_grid solamente, podemos añadir space = "free\_y" o space = "free\_x" para que la altura o el ancho de la faceta sea ponderada en relación a los valores de la figura en su interior. Esto sólo funciona si ya se ha asignado scale = "free" (y o x).

### Orden del nivel de los factores en las facetas

Consulte esta [entrada](https://juliasilge.com/blog/reorder-within/) sobre cómo reordenar los niveles de los factores dentro de las facetas.

## Almacenamiento de gráficos

### Guardar los gráficos

Cuando se ejecuta un comando ggplot(), el gráfico se imprimirá en el panel de Plots RStudio de manera predeterminada. Sin embargo, también podes guardar el gráfico como un objeto utilizando el operador de asignación <- y asignandole un nombre. Entonces el grafico no se imprimirá a menos que se ejecute el nombre del objeto mismo. También podes imprimirlo envolviendo el nombre del grafico con print(), pero esto sólo es necesario en ciertas circunstancias, como cuando el grafico se crea dentro de un for loop o bucle utilizado para imprimir múltiples gráficos a la vez (véase la página [Iteración, bucles y listas](#iteration-loops-and-lists)).

### Modificación de gráficos guardados

Una gran ventaje de **ggplot2** es que podes definir un gráfico (como se ve arriba), y luego añadirle capas empezando por su nombre sin necesidad de que repetir todos los comandos que crearon el gráfico original.

Por ejemplo, si se desea modificar el gráfico age\_by\_wt que se definió anteriormente, para incluir una línea vertical a la edad de 50 años, sólo tendríamos que añadir un + y empezar a añadir capas adicionales al gráfico.

### Exportación de gráficos

La exportación de ggplots es fácil con la función ggsave() de **ggplot2**. Puede funcionar de dos maneras, ya sea:

* Especifica el nombre del objeto del gráfico, a continuación, la ruta del archivo y el nombre del archivo incluyendo la extensión
  + Por ejemplo: ggsave(my\_plot, here("plots", "my\_plot.png"))
* Ejecuta el comando con sólo una ruta de archivo, para guardar el último gráfico que se imprimió en pantalla
  + Por ejemplo: ggsave(here("plots", "my\_plot.png"))

Puedes exportar como png, pdf, jpeg, tiff, bmp, svg, o varios otros tipos de archivos, especificando la extensión del archivo en la ruta del mismo.

También puede especificar los argumentos width =, height = y units = (ya sea "in", "cm" o "mm"). También podes especificar dpi = asignando un número para la resolución del trazado (por ejemplo, 300). Consulta los detalles de la función ejecutando ?ggsave o leyendo la [documentación en línea](https://ggplot2.tidyverse.org/reference/ggsave.html).

Recuerda que podes utilizar la sintaxis here() para proporcionar la ruta de archivo deseada. Consulta la página de [importación y exportación](#import-and-export) para obtener más información.

## Etiquetas

Seguramente querrás añadir o ajustar las etiquetas del gráfico. Esto se hace más fácilmente dentro de la función labs() que se añade al gráfico con + al igual que los geoms.

Dentro de labs() podes proporcionar cadenas de caracteres a estos argumentos:

* x = e y = El título del eje x y del eje y (etiquetas)
* title = El título de la grafico principal
* subtitle = El subtítulo del gráfico, en texto más pequeño debajo del título
* caption = El pie del gráfico, que aparecerá en la parte inferior derecha de manera predeterminada

Aquí está el mismo gráfico que hicimos antes, pero con etiquetas más bonitas:

Observa cómo en la asignación del pie del grafico hemos utilizado str\_glue() del paquete **stringr** para integrar código R dinámico dentro del texto de la cadena. El pie del grafico mostrará la fecha "Datos a partir de:" que refleja la fecha máxima de hospitalización en el listado de datos. Lea más sobre esto en la página sobre [Caracteres y cadenas](#characters-and-strings).

Una nota sobre la especificación del título de la leyenda: No hay un argumento "título de la leyenda", ya que podría tener múltiples escalas en su leyenda. Dentro de labs(), podes escribir el argumento de la estética del gráfico utilizado para crear la leyenda, y proporcionar el título de esta manera. Por ejemplo, arriba asignamos color = age para crear la leyenda. Por lo tanto, proporcionamos color = a labs() y asignamos el título de la leyenda deseado ("Age" con A mayúscula). Si se crea la leyenda con aes(fill = COLUMN), entonces en labs() se escribiría fill = para ajustar el título de esa leyenda. La sección sobre escalas de color en la página [de consejos de ggplot](#ggplot-tips) proporciona más detalles sobre la edición de leyendas, y un enfoque alternativo utilizando las funciones scales\_().

## Temas

Una de las mejores partes de **ggplot2** es el nivel de control que tienes sobre el gráfico - ¡puedes definir lo que quieras! Como se mencionó anteriormente, los aspectos de diseño del gráfico que no está relacionados con las formas/geometrías de los datos se ajustan dentro de la función theme(). Por ejemplo, el color de fondo del gráfico, la presencia/ausencia de líneas de cuadrícula, y la fuente/tamaño/color/alineación del texto (títulos, subtítulos, pie de grafico, texto de los ejes...). Estos ajustes pueden realizarse de dos maneras:

* Añadiendo una función theme\_() [completa](https://ggplot2.tidyverse.org/reference/ggtheme.html) para realizar ajustes de barrido – estas funciones de tema completo incluyen theme\_classic(), theme\_minimal(), theme\_dark(), theme\_light() theme\_grey(), theme\_bw() entre otras
* Ajustando cada pequeño aspecto del grafico individualmente dentro de theme()

### Temas completos

Como son bastante sencillas, demostraremos las funciones del tema completo a continuación y no las describiremos más aquí. Ten en cuenta que cualquier microajuste con theme() debe hacerse después de utilizar un tema completo.

Escríbilos con paréntesis vacíos.

### Modificar el tema

La función theme() puede tomar un gran número de argumentos, cada uno de los cuales edita un aspecto específico del gráfico. No hay manera de que podamos cubrir todos los argumentos, pero describiremos el patrón general para ellos y te mostraremos cómo encontrar el nombre del argumento que necesitas. La sintaxis básica es esta:

1. Dentro de theme() escribe el nombre del argumento del elemento de l grafico que queres editar, como plot.title =
2. Proporciona una función element\_() al argumento

* Lo más habitual es utilizar element\_text(), pero también element\_rect() para los colores de fondo del lienzo, o element\_blank() para eliminar los elementos del grafico

1. Dentro de la función element\_(), escribi las asignaciones de argumentos para realizar los ajustes finos que desees

Esa descripción es bastante abstracta, así que aquí hay algunos ejemplos.

El siguiente gráfico parece tonto, pero sirve para mostrarte una variedad de formas en las que podes ajustar su gráfico.

* Comenzamos con el gráfico age\_by\_wt definido anteriormente y añadimos theme\_classic()
* Para realizar ajustes más finos, añadimos theme() e incluimos un argumento por cada elemento del grafico que queremos ajustar

Puedes ser util organizar los argumentos en secciones lógicas. A continuación se describen algunos argumentos utilizados:

* legend.position = es el único argumento que acepta valores simples como "abajo", "arriba", "izquierda" y "derecha".Por lo general, los argumentos relacionados con el texto requieren que se coloquen los detalles dentro de element\_text().
* Tamaño del título se ajusta con element\_text(size = 30)
* La alineación horizontal del pie del grafico se logran con el argumento element\_text(hjust = 0) (de derecha a izquierda)
* El subtítulo aparece en cursiva gracias al argumento element\_text(face = "italic")

Aquí hay algunos argumentos de theme() especialmente comunes. Reconocerás algunos patrones, como añadir .x o .y para aplicar el cambio sólo a un eje.

| **argumento de** theme() | **Lo que ajusta** |
| --- | --- |
| plot.title = element\_text() | El título |
| plot.subtitle = element\_text() | El subtítulo |
| plot.caption = element\_text() | La leyenda (familia, cara, color, tamaño, ángulo, vjust (justificación vertical), hjust (justificación horizontal)...) |
| axis.title = element\_text() | Títulos de los ejes (tanto x como y) (tamaño, cara, ángulo, color...) |
| axis.title.x = element\_text() | Título del eje x solamente (usar .y para el eje y solamente) |
| axis.text = element\_text() | Texto de los ejes (x e y) |
| axis.text.x = element\_text() | Texto del eje x solamente (usar .y para el eje y solamente) |
| axis.ticks = element\_blank() | Eliminar las marcas del eje |
| axis.line = element\_line() | Líneas del eje (color, tamaño, tipo de línea: sólida, punteada, etc.) |
| strip.text = element\_text() | Texto de la tira de facetas (color, cara, tamaño, ángulo...) |
| strip.background = element\_rect() | tira de facetas (relleno, color, tamaño...) |

Pero ¡hay tantos argumentos de tema! ¿Cómo podría recordarlos todos? No te preocupes, es imposible recordarlos todos. Por suerte, hay algunas herramientas que te ayudarán:

La documentación de **tidyverse** sobre la [modificación del tema](https://ggplot2.tidyverse.org/reference/theme.html), tiene una lista completa.

**CONSEJO:** Ejecuta theme\_get() de **ggplot2** para imprimir en pantalla una lista de los más de 90 argumentos de theme() en la consola.

**CONSEJO:** Si alguna vez quieres eliminar un elemento de un gráfico, también puedes hacerlo a través de theme(). Basta con pasar element\_blank() a un argumento para que desaparezca por completo. Para eliminar leyendas, puedes asignar legend.position = "none".

## Colores

Consulte esta [sección sobre las escalas de color de la página de consejos de ggplot](#ggplot_tips_colors).

## Utilización de pipping en ****ggplot2****

Cuando se utilizan tuberías (“pipping’) para limpiar y transformar los datos, es fácil pasar los datos transformados a ggplot().

Las tuberías que pasan el conjunto de datos de función a función se convertiran a + una vez que se invoque a la función ggplot(). Ten en cuenta que, en este caso, no es necesario especificar el argumento data =, ya que éste se define automáticamente como el conjunto de datos canalizado.

Así es como podría verse:

## Trazado de datos continuos

A lo largo de esta página, ya has visto muchos ejemplos de trazado de datos continuos. Aquí los consolidamos brevemente y presentamos algunas variaciones.   
Las visualizaciones que aquí se cubren incluyen:

* Gráficos para una variable continua:
  + **Histograma**, un gráfico clásico para presentar la distribución de una variable continua.
  + **Gráfico de caja** (también llamado de caja y bigotes), para mostrar los percentiles 25, 50 y 75, los extremos de la cola de la distribución y los valores atípicos ([limitaciones importantes](https://www.data-to-viz.com/caveat/boxplot.html)).
  + **Gráfico de fluctuación**, para mostrar todos los valores como puntos que se "fluctúan" para que se puedan ver (casi) todos, incluso cuando dos tienen el mismo valor.
  + **Gráfico del violín**, muestra la distribución de una variable continua en función del ancho simétrico del "violín".
  + Los **gráficos de Sina**, son una combinación de los gráficos de jitter y de violín, donde se muestran los puntos individuales pero con la forma simétrica de la distribución (a través del paquete **ggforce**).
* **Gráfico de dispersión** para dos variables continuas.
* **Gráficos de calor** para tres variables continuas (enlazado a la página de [gráficos de calor](#heat-plots))

### Histogramas

Los histogramas pueden parecerse a los gráficos de barras, pero son distintos porque miden la distribución de una variable continua. No hay espacios entre las "barras", y sólo se proporciona una columna a geom\_histogram().

A continuación se muestra el código para generar **histogramas**, que agrupan datos continuos en rangos y se muestran en barras adyacentes de altura variable. Esto se hace utilizando geom\_histogram(). Consulte la [sección "Diagrama de barras"](#ggplot_basics_bars) de la página de fundamentos de ggplot para entender la diferencia entre geom\_histogram(), geom\_bar() y geom\_col().

Vamos a mostrar la distribución de las edades de los casos. Dentro de mapping = aes() especifique la columna de la que quiere ver la distribución. Puedes asignar esta columna al eje x o al eje y.

Las filas serán asignadas a "bins" basados en su edad numérica, y estos bins serán representados gráficamente por barras. Si se especifica un número de bins con la estética de gráfico bins =, los puntos de ruptura se espacian uniformemente entre los valores mínimos y máximos del histograma. Si no se especifica bins =, se adivinará un número apropiado de bins y aparecera este mensaje después del gráfico:

## `stat\_bin()` usando `bins = 30`. Elija un valor mejor con `binwidth`.

Si no quieres especificar un número de bins a bins =, puedes especificar alternativamente binwidth = en las unidades del eje. Damos algunos ejemplos que muestran diferentes bins y anchos de bins:

Para obtener proporciones suavizadas, puede utilizar geom\_density():

Para obtener un histograma "apilado" (de una columna continua de datos), puede hacer una de las siguientes cosas:

1. Utiliza geom\_histogram() con el argumento fill = dentro de aes() y asignado a la columna de agrupación, o
2. Utiliza geom\_freqpoly(), que probablemente sea más fácil de leer (aún puedes establecer binwidth =)
3. Para ver las proporciones de todos los valores, asigna el y = after\_stat(density) (utilice esta sintaxis exactamente - no se ha modificado para sus datos). Nota: estas proporciones se mostrarán por grupo.

Cada uno se muestra a continuación (\*nótese el uso de color = versus fill = en cada uno):

Si quieres divertirte un poco, prueba con geom\_density\_ridges del paquete **ggridges** ([vignette aquí](https://cran.r-project.org/web/packages/ggridges/vignettes/introduction.html)).

Lea más en detalle sobre los histogramas en la [página de](https://ggplot2.tidyverse.org/reference/geom_histogram.html) **tidyverse** [sobre geom\_histogram()](https://ggplot2.tidyverse.org/reference/geom_histogram.html).

### Gráficos de caja

Los gráficos de caja son habituales, pero tienen limitaciones importantes. Pueden ocultar la distribución real - por ejemplo, una distribución bimodal. Consulte esta [galería de gráficos de R](https://www.r-graph-gallery.com/boxplot.html) y este [artículo sobre la visualización de datos](https://www.data-to-viz.com/caveat/boxplot.html) para obtener más detalles. Sin embargo, muestran muy bien el rango intercuartil y los valores atípicos, por lo que pueden superponerse a otros tipos de gráficos que muestran la distribución con más detalle.

A continuación te recordamos los distintos componentes de un boxplot:

Cuando se utiliza geom\_boxplot() para crear un gráfico de caja, generalmente se asigna sólo un eje (x o y) dentro de aes(). El eje especificado determina si los gráficos son horizontales o verticales.

En la mayoría de los geoms, se crea un gráfico por grupo asignando una estética como color = o fill = a una columna dentro de aes(). Sin embargo, en el caso de los gráficos de caja, esto se consigue asignando la columna de agrupación al eje no asignado (x o y). A continuación se muestra el código para un diagrama de caja de todos los valores de edad en el conjunto de datos, y un segundo trozo de código para mostrar un diagrama de caja para cada género (no ausente) en el conjunto de datos. Ten en cuenta que los valores NA (valores faltantes) aparecerán como un gráfico de caja separado a menos que se eliminen. En este ejemplo, también hemos asignado el fill de la columna outcome para que cada gráfico tenga un color diferente, pero esto no es necesario.

Para ver el código para añadir un gráfico de caja a los bordes de un gráfico de dispersión (gráficos "marginales"), consulta la página [ggplot tips](#ggplot-tips).

### Gráficos de violín, jitter y sina

A continuación se muestra el código para crear gráficos de **violín** (geom\_violin) y **gráficos de fluctuación** (geom\_jitter) para mostrar distribuciones. Se puede especificar que el relleno o el color también estén determinados por los datos, insertando estas opciones dentro de aes().

Puedes combinar los dos tipos de grafico usando la función geom\_sina() del paquete **ggforce**. La sina traza los puntos de jitter en la forma del gráfico de violín. Cuando se superpone al gráfico de violín (ajustando las transparencias) puede ser más fácil de interpretar visualmente.

### Dos variables continuas

Siguiendo una sintaxis similar, geom\_point() te permitirá trazar dos variables continuas en un **gráfico de dispersión**. Esto es útil para mostrar los valores reales en lugar de sus distribuciones. En (A) se muestra un gráfico de dispersión simple de la edad frente al peso. En (B) volvemos a utilizar facet\_grid() para mostrar la relación entre dos variables continuas.

### Tres variables continuas

Puedes mostrar tres variables continuas utilizando el argumento fill = para crear un gráfico de calor. El color de cada "celda" reflejará el valor de la tercera columna continua de datos. Consulte la página de [consejos de ggplot](#ggplot-tips) y la página de [gráficos de calor](#heat-plots) para obtener más detalles y varios ejemplos.

Hay formas de hacer gráficos en 3D en R, pero para la epidemiología aplicada suelen ser difíciles de interpretar y, por tanto, menos útiles para la toma de decisiones.

## Trazar datos categóricos

Los datos categóricos pueden ser valores de carácter, pueden ser lógicos (TRUE/FALSE), o factores (ver la página de [Factores](#factors)).

### Preparación

#### Estructura de datos

Lo primero que hay que entender sobre tus datos categóricos es si existen como observaciones en bruto, como una lista de casos, o como un dataframe de resumen o agregado que contiene recuentos o proporciones. El estado de sus datos afectará a la función de trazado que utilice:

* Si tus datos son observaciones en bruto con una fila por observación, es probable que utilices geom\_bar()
* Si sus datos ya están agregados en recuentos o proporciones, es probable que utilices geom\_col()

#### Tipo de columna y ordenación de valores

A continuación, examine el tipo o clase de las columnas (variables) que desea trazar. Veamos hospital, primero usando class() de R **base**, y luego con tabyl() de **janitor**.

Podemos ver que los valores de la variable hospital son caracteres, ya que son nombres de hospitales, y por defecto están ordenados alfabéticamente. Hay valores "otros" y "faltantes", que preferiríamos que fueran las últimas subcategorías al presentar los desgloses. Así que convertimos esta columna a clase factor y la reordenamos. Esto se trata con más detalle en la página de [Factores](#factors).

### geom\_bar()

Utilice geom\_bar() si desea que la altura de las barras (o la altura de los componentes de las barras apiladas) refleje el número de filas relevantes de los datos. Estas barras tendrán huecos entre ellas, a menos que se ajuste la estética de width =.

* Proporcione sólo una asignación de columna de eje (normalmente el eje x). Si proporciona x e y, obtendrá Error: stat\_count() can only have an x or y aesthetic.
* Puedes crear barras apiladas añadiendo una asignación de Columba usando fill = dentro de mapping = aes()
* El eje opuesto se llamará por defecto “count” (recuento), ya que representa el número de filas

A continuación, hemos asignado el resultado (outcome) al eje Y, pero podría estar fácilmente asignarse al eje X. Si tienes valores de caracteres más largos, a veces el grafico se ve mejor si las barras se grafican de manera horizontal en vez de vertical ,poniendo la leyenda en la parte inferior del gráfico. Esto puede afectar el orden en el que aparecen los factores - en este caso los invertimos con fct\_rev() para poner la categoría de ‘faltantes’ y ‘otros’ en la parte inferior.

### geom\_col()

Utilice geom\_col() si desea que la altura de las barras (o la altura de los componentes de las barras apiladas) refleje valores precalculados que existen en los datos. A menudo, se trata de recuentos sumarios o "agregados", o de proporciones.

Proporcione las asignaciones de columna para ambos ejes a geom\_col(). Normalmente la columna del eje x es discreta y la del eje y es numérica.

Digamos que tenemos los un conjunto de datos denominado outcomes:

A continuación se muestra un código que utiliza geom\_col para crear gráficos de barras sencillos que muestren la distribución de los resultados de los pacientes con Ebola. Con geom\_col, es necesario especificar tanto x como y. Aquí x es la variable categórica a lo largo del eje x, e y es la columna de proporciones precalculada denominada proportion.

Para mostrar los desgloses por hospital, necesitaríamos que nuestra tabla contuviera más información, y que estuviera en formato "largo". Creamos esta tabla con las frecuencias de las categorías combinadas outcome y hospital (véase la página de [Agrupar datos](#grouping-data) para obtener consejos de agrupación).

A continuación, creamos el ggplot añadiendo algunas asignaciones de formato:

* **Cambio de posición del eje**: Intercambiamos los ejes con coord\_flip() para poder leer los nombres de los hospitales.
* **Columnas de lado a lado**: Se ha añadido un argumento de position = "dodge" para que las barras de muerte (Death) y recuperación (Recover) se presenten una al lado de la otra en lugar de apiladas. Ten en cuenta que las barras apiladas aparecen de manera predeterminada.
* **Ancho de columna**: Se especifica el "ancho", para graficar las columnas a la mitad del ancho posible.
* **Orden de las columnas**: Se ha invertido el orden de las categorías en el eje y para que "Otros" y "Faltantes" aparezcan últimos, con scale\_x\_discrete(limits=rev). Ten en cuenta que usamos eso en lugar de scale\_y\_discrete porque la variable hospital se asigno al eje x en el en el argumento de aes(), aunque visualmente se vea en el eje y. Hacemos esto porque ggplot parece presentar las categorías al revés a menos que le digamos que no lo haga.
* **Otros detalles**: Añadimos etiquetas/títulos y colores dentro de las funciones de labs y scale\_fill\_color respectivamente.

Ten en cuenta que las proporciones son binarias, por lo que podemos preferir omitir "recuperar" y mostrar sólo la proporción que murió. Esto es sólo a título ilustrativo.

Si se utiliza geom\_col() con datos de fechas (por ejemplo, una epicurva a partir de datos agregados) - querrá ajustar el argumento width = para eliminar los "huecos" entre las barras. Si se utilizan datos diarios (en días), ajuste width= 1. Si se trata de datos semanales, la anchura seria width = 7. Los meses no son posibles de representar porque cada mes tiene un número diferente de días.

### geom\_histogram()

Los histogramas pueden parecerse a los gráficos de barras, pero son distintos porque miden la distribución de una variable continua. No hay huecos o espacios entre las "barras" y sólo se asigna una columna a geom\_histogram(). Hay argumentos específicos para los histogramas como bin\_width = y breaks = para especificar cómo se deben dividir los datos. La sección anterior sobre datos continuos y la página sobre [curvas epidémicas](#epidemic-curves) proporcionan detalles adicionales.

## Recursos

Hay una gran cantidad de ayuda en línea, especialmente sobre ggplot. Ver:

* [hoja de trucos de ggplot2](http://r-statistics.co/ggplot2-cheatsheet.html)
* [otra hoja de trucos](https://biostats.w.uib.no/the-ggplot2-cheat-sheet-by-rstudio/)
* [página de fundamentos de tidyverse ggplot](https://ggplot2.tidyverse.org/reference/)
* [trazado de variables continuas](http://www.sthda.com/english/articles/32-r-graphics-essentials/131-plot-two-continuous-variables-scatter-graph-and-alternatives/)
* páginas de R for Data Science sobre [visualización de datos](https://r4ds.had.co.nz/data-visualisation.html)
* [gráficos para la comunicación](https://r4ds.had.co.nz/graphics-for-communication.html)