# **# Tabla**s descriptivas

{#descriptive-tables}

Esta página demuestra el uso de **janitor**, **dplyr**, **gtsummary**, **rstatix** y **base** R para resumir datos y crear tablas con estadísticas descriptivas.

En esta página se explica cómo crear\* las tablas subyacentes, mientras que en la página [Tablas para la presentación](#tables-for-presentation) se explica cómo darles un buen formato e imprimirlas.\*

Cada uno de estos paquetes tiene ventajas y desventajas en cuanto a la simplicidad del código, la accesibilidad de los resultados y la calidad de los resultados impresos. Utiliza esta página para decidir qué enfoque se ajusta a su situación.

Tienes varias opciones para producir tablas de resumen de tabulación y tabulación cruzada. Algunos de los factores a tener en cuenta son la simplicidad del código, la posibilidad de personalización, la salida deseada (impresa en la consola de R, como dataframe, o como una imagen .png/.jpeg/.html "bonita"), y la facilidad de posprocesamiento. Ten en cuenta los siguientes puntos a la hora de elegir la herramienta para su situación.

* Utiliza tabyl() de **janitor** para producir y "adornar" tabulaciones y tabulaciones cruzadas
* Utiliza get\_summary\_stats() de **rstatix** para generar fácilmente dataframes de estadísticas de resumen numérico para múltiples columnas y/o grupos
* Utiliza summarise() y count() de **dplyr** para obtener estadísticas más complejas, ordenar las salidas de los dataframes o preparar los datos para ggplot()
* Utiliza tbl\_summary() de **gtsummary** para producir tablas detalladas listas para su publicación
* Utiliza table() de si no tiene acceso a los paquetes anteriores

## 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 quiere seguir el proceso, [clica para descargar linelist "limpia"](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).

A continuación se muestran las primeras 50 filas del listado.

## Buscar datos

### paquete ****skimr****

Utilizando el paquete skimr, puedes obtener una visión detallada y estéticamente agradable de cada una de las variables de tu conjunto de datos. Lee más sobre **skimr** en su [página de github](https://github.com/ropensci/skimr).

A continuación, se aplica la función skim() a todo el dataframe del listado. Se produce una visión general del dataframe y un resumen de cada columna (por tipo).

También puede utilizar la función summary(), de la **base** R, para obtener información sobre unos datos completo, pero esta salida puede ser más difícil de leer que utilizando **skimr**. Por lo tanto, la salida no se muestra a continuación, para conservar el espacio de la página.

### Estadísticas resumidas

Puedes utilizar las funciones **básicas de** R para devolver estadísticas de resumen sobre una columna numérica. Puedes devolver la mayoría de las estadísticas de resumen útiles para una columna numérica utilizando summary(), como se indica a continuación. Ten en cuenta que el nombre del dataframe también debe especificarse como se muestra a continuación.

Puedes acceder y guardar una parte específica de la misma con los corchetes de índice [ ]:

Puedes devolver estadísticas individuales con funciones básicas de R como max(), min(), median(), mean(), quantile(), sd() y range(). Consulta la página de [fundamentos de R](#r-basics) para obtener una lista completa.

**PRECAUCIÓN:** Si tus datos contienen valores faltantes, R quiere que lo sepas y por ello devolverá NA a menos que se especifique en las funciones matemáticas anteriores que quieres que R ignore los valores faltantes, mediante el argumento na.rm = TRUE.

Puedes utilizar la función get\_summary\_stats() de **rstatix** para devolver las estadísticas de resumen en un formato de dataframe. Esto puede ser útil para realizar operaciones posteriores o trazar los números. Consulta la página [Pruebas estadísticas simples](#simple-statistical-tests) para obtener más detalles sobre el paquete rstatix y sus funciones.

## paquete de ****conserjería****

Los paquetes **janitor** ofrecen la función tabyl() para producir tabulaciones y tabulaciones cruzadas, que pueden ser "adornadas" o modificadas con funciones de ayuda para mostrar porcentajes, proporciones, recuentos, etc.

A continuación, canalizamos el dataframe del listado a las funciones de **limpieza** e imprimimos el resultado. Si lo desea, también puede guardar las tablas resultantes con el operador de asignación <-.

### Tabyl simple

El uso por defecto de tabyl() en una columna específica produce los valores únicos, los recuentos y los "porcentajes" por columna (en realidad proporciones). Las proporciones pueden tener muchos dígitos. Puedes ajustar el número de decimales con adorn\_rounding() como se describe a continuación.

Como puede ver arriba, si hay valores que faltan se muestran en una fila etiquetada como <NA>. Puedes suprimirlos con show\_na = FALSE. Si no hay valores faltantes, esta fila no aparecerá. Si hay valores faltantes, todas las proporciones se dan como crudas (el denominador incluye los recuentos NA) y "válidas" (el denominador excluye los recuentos NA).

Si la columna es de tipo factor y sólo hay ciertos niveles en sus datos, todos los niveles seguirán apareciendo en la tabla. Puedes suprimir esta característica especificando show\_missing\_levels = FALSE. Lea más en la página de [Factores](#factors).

### Tabulación cruzada

Los recuentos de tabulación cruzada se consiguen añadiendo una o más columnas adicionales dentro de tabyl(). Ten en cuenta que ahora sólo se devuelven los recuentos - las proporciones y los porcentajes se pueden añadir con los pasos adicionales que se muestran a continuación.

### "Adornando" el tabyl

Utiliza las funciones de "adorno" **del conserje** para añadir totales o convertir a proporciones, porcentajes, o ajustar la visualización de otro modo. A menudo, canalizará el tabyl a través de varias de estas funciones.

| **Función** | **Resultado** |
| --- | --- |
| adornar\_totales() | Añade los totales (donde = "fila", "col", o "ambos"). Establecer nombre = para "Total". |
| adornar\_porcentajes() | Convertir los recuentos en proporciones, con denominador = "fila", "col" o "todos" |
| adorn\_pct\_formatting() | Convierte las proporciones en porcentajes. Especifique los dígitos =. Elimine el símbolo "%" con affix\_sign = FALSE. |
| adornar\_redondear() | Para redondear proporciones a dígitos = lugares. Para redondear porcentajes utiliza adorn\_pct\_formatting() con dígitos =. |
| adorn\_ns() | Añadir recuentos a una tabla de proporciones o porcentajes. Indique la posición = "atrás" para mostrar los recuentos entre paréntesis, o "adelante" para poner los porcentajes entre paréntesis. |
| adornar\_título() | Añade una cadena mediante los argumentos row\_name = y/o col\_name = |

Sea consciente del orden en que aplica las funciones anteriores. A continuación, algunos ejemplos.

Una simple tabla unidireccional con porcentajes en lugar de las proporciones por defecto.

Una tabulación cruzada con un total de filas y porcentajes de filas.

Una tabulación cruzada ajustada para que aparezcan tanto los recuentos como los porcentajes.

### Impresión del tabyl

Por defecto, el tabyl imprimirá en bruto en la consola de R.

Alternativamente, puede pasar el tabyl a **flextable** o un paquete similar para imprimirlo como una imagen "bonita" en el visor de RStudio, que podría exportarse como .png, .jpeg, .html, etc. Esto se discute en la página [Tablas de presentación](#tables-for-presentation). Ten en cuenta que si imprime de esta manera y utiliza adorn\_titles(), debe especificar placement = "combined".

### Uso en otras mesas

Puedes utilizar las funciones adorn\_\*() de **janitor** en otras tablas, como las creadas por summarise() y count() de **dplyr**, o table() de . Por ejemplo:

### Guardar el tabyl

Si convierte la tabla en una imagen "bonita" con un paquete como **flextable**, puede guardarla con funciones de ese paquete - como save\_as\_html(), save\_as\_word(), save\_as\_ppt(), y save\_as\_image() de **flextable** (como se discute más ampliamente en la página [Tablas para la presentación](#tables-for-presentation)). A continuación, la tabla se guarda como un documento de Word, en el que se puede seguir editando a mano.

### Estadísticas

Puedes aplicar pruebas estadísticas a las tabillas, como chisq.test() o fisher.test() del paquete **stats**, como se muestra a continuación. Ten en cuenta que los valores faltantes no están permitidos, por lo que se excluyen de la tabulación con show\_na = FALSE.

Consulta la página sobre [Tests estadísticos sencillos](#simple-statistical-tests) para obtener más código y consejos sobre estadística.

### Otros consejos

* Incluya el argumento na.rm = TRUE para excluir los valores faltantes de cualquiera de los cálculos anteriores.
* Si aplica cualquier función de ayuda adorn\_\*() a tablas no creadas por tabyl(), puede especificar una(s) columna(s) particular(es) para aplicarlas como adorn\_percentage(,,,c(cases,deaths)) (especifíquelos en el cuarto argumento sin nombre). La sintaxis no es sencilla. Considere la posibilidad de utilizar summarise() en su lugar.
* Puedes leer más detalles en la [página del conserje](https://cran.r-project.org/web/packages/janitor/vignettes/janitor.html) y en esta [viñeta del tabyl](https://cran.r-project.org/web/packages/janitor/vignettes/tabyls.html).

## paquete ****dplyr****

**dplyr** forma parte de los paquetes **tidyverse** y es una herramienta de gestión de datos muy común. La creación de tablas con las funciones de **dplyr** summarise() y count() es un enfoque útil para calcular estadísticas de resumen, resumir por grupos o pasar tablas a ggplot().

summarise() crea un nuevo dataframe de resumen. Si los datos no están agrupados, devolverá un dataframe de una fila con las estadísticas de resumen especificadas de todo el dataframe. Si los datos están agrupados, el nuevo dataframe tendrá una fila por grupo (véase la página [Agrupar datos](#grouping-data)).

Dentro del paréntesis de summarise(), se proporcionan los nombres de cada nueva columna de resumen, seguidos de un signo de igualdad y de una función estadística a aplicar.

**SUGERENCIA:** La función summarise funciona tanto con la ortografía británica como con la estadounidense (summarise() y summarize()).

### Obtener recuentos

La función más sencilla de aplicar dentro de summarise() es n(). Deje los paréntesis vacíos para contar el número de filas.

Esto se vuelve más interesante si hemos agrupado los datos de antemano.

El comando anterior se puede acortar utilizando la función count() en su lugar. count() hace lo siguiente:

1. Agrupa los datos por las columnas que se le proporcionan
2. Los resume con n() (creando la columna n)
3. Desagrupa los datos

Puedes cambiar el nombre de la columna de recuentos de la n por defecto a otra cosa especificando a nombre =.

Los recuentos tabulados de dos o más columnas de agrupación se siguen devolviendo en formato "largo", con los recuentos en la columna n. Consulta la página sobre [Pivoteo de datos](#pivoting-data) para conocer los formatos de datos "largos" y "anchos".

### Mostrar todos los niveles

Si está tabulando una columna de de tipo factor, puede asegurarse de que se muestren todos los niveles (no sólo los niveles con valores en los datos) añadiendo .drop = FALSE en el comando summarise() o count().

Esta técnica es útil para estandarizar sus tablas/trazados. Por ejemplo, si está creando cifras para varios subgrupos, o creando repetidamente la cifra para informes de rutina. En cada una de estas circunstancias, la presencia de valores en los datos puede fluctuar, pero puede definir niveles que permanezcan constantes.

Para más información, consulta la página sobre [los factores](#factors).

### Proporciones

Las proporciones pueden añadirse pasando la tabla por mutate() para crear una nueva columna. Defina la nueva columna como la columna de recuentos (n por defecto) dividida por la suma() de la columna de recuentos (esto devolverá una proporción).

Ten en cuenta que en este caso, sum() en el comando mutate() devolverá la suma de toda la columna n para utilizarla como denominador de la proporción. Como se explica [en la página Agrupar datos](#group_summarise), si sum() se utiliza en datos agrupados (por ejemplo, si el comando mutate() siguió inmediatamente a un comando group\_by()), devolverá sumas por grupo. Como se acaba de indicar, count() termina sus acciones desagrupando. Por lo tanto, en este escenario obtenemos proporciones de columnas completas.

Para mostrar fácilmente los porcentajes, puede envolver la proporción en la función percent() del paquete **scales** (Ten en cuenta que se convierte en tipo carácter).

A continuación se presenta un método para calcular las proporciones dentro de los grupos. Se basa en diferentes niveles de agrupación de datos que se aplican y eliminan selectivamente. En primer lugar, los datos se agrupan en función del resultado mediante group\_by(). A continuación, se aplica count(). Esta función agrupa además los datos por age\_cat y devuelve los recuentos para cada combinación de resultado-age-cat. Es importante destacar que, al finalizar su proceso, count() también desagrupa la agrupación age\_cat, por lo que la única agrupación de datos que queda es la agrupación original por resultado. Por lo tanto, el paso final del cálculo de las proporciones (denominador sum(n)) sigue estando agrupado por resultado.

### Trazando

Mostrar una tabla "larga" como la anterior con ggplot() es relativamente sencillo. Los datos están naturalmente en formato "largo", que es aceptado naturalmente por ggplot(). Vea más ejemplos en las páginas [ggplot basics](#ggplot-basics) y [ggplot tips](#ggplot-tips).

### Estadísticas resumidas

Una de las principales ventajas de **dplyr** y de summarise() es la capacidad de devolver resúmenes estadísticos más avanzados como la mediana(), la media(), el máximo(), el mínimo(), la sd() (desviación estándar) y los percentiles. También puede utilizar sum() para devolver el número de filas que cumplen ciertos criterios lógicos. Al igual que en el caso anterior, estas salidas pueden producirse para todo el conjunto de dataframes o por grupos.

La sintaxis es la misma: dentro de los paréntesis de summarise() se proporcionan los nombres de cada nueva columna de resumen, seguidos de un signo de igualdad y de una función estadística para aplicar. Dentro de la función estadística, indique la(s) columna(s) con la(s) que se va a operar y cualquier argumento relevante (por ejemplo, na.rm = TRUE para la mayoría de las funciones matemáticas).

También puede utilizar sum() para devolver el número de filas que cumplen un criterio lógico. La expresión que contiene se cuenta si se evalúa como TRUE. Por ejemplo:

* sum(edad\_años < 18, na.rm=T)
* sum(género == "masculino", na.rm=T)
* sum(respuesta %in% c("Probable", "Muy probable"))

A continuación, se resumen los datos del listado para describir los días de retraso desde el inicio de los síntomas hasta el ingreso en el hospital (columna days\_onset\_hosp), por hospital.

Algunos consejos:

* Utilizar sum() con una sentencia lógica para "contar" las filas que cumplen ciertos criterios (==)
* Ten en cuenta el uso de na.rm = TRUE dentro de funciones matemáticas como sum(), de lo contrario se devolverá NA si hay algún valor faltante
* Utiliza la función percent() del paquete **scales** para convertir fácilmente a porcentajes
  + Ajuste la precisión = a 0,1 o 0,01 para garantizar 1 o 2 decimales respectivamente
* Utilizar round() de **base** R para especificar los decimales
* Para calcular estas estadísticas en todo el set de datos, utiliza summarise() sin group\_by()
* Puedes crear columnas para los propósitos de cálculos posteriores (por ejemplo, denominadores) que eventualmente se eliminan de su dataframe con select().

### Estadísticas condicionales

Es posible que desee devolver estadísticas condicionales, por ejemplo, el máximo de filas que cumplen ciertos criterios. Esto se puede hacer sub-configurando la columna con corchetes [ ]. El ejemplo siguiente devuelve la temperatura máxima de los pacientes clasificados con o sin fiebre. Sin embargo, Ten en cuenta que puede ser más apropiado añadir otra columna al comando group\_by() y pivot\_wider() (como se demuestra [a continuación](#tbls_pivot_wider)).

### Pegado

La función str\_glue() de **stringr** es útil para combinar valores de varias columnas en una nueva columna. En este contexto, se suele utilizar después del comando summarise().

En la página [Caracteres y cadenas](#characters-and-strings), se discuten varias opciones para combinar columnas, incluyendo unite(), y paste0(). En este caso de uso, abogamos por str\_glue() porque es más flexible que unite() y tiene una sintaxis más sencilla que paste0().

A continuación, el dataframe de summary\_table (creado anteriormente) se muta de manera que las columnas delay\_mean y delay\_sd se combinan, se añade el formato de paréntesis a la nueva columna y se eliminan sus respectivas columnas antiguas.

Luego, para hacer la tabla más presentable, se añade una fila de totales con adorn\_totals() de **janitor** (que ignora las columnas no numéricas). Por último, utilizamos select() de **dplyr** para reordenar y renombrar los nombres de las columnas.

Ahora puede pasar a **flextable** e imprimir la tabla a Word, .png, .jpeg, .html, Powerpoint, RMarkdown, etc. (ver la página de [Tablas para la presentación](#tables-for-presentation)).

#### Percentiles

Los percentiles y cuartiles en **dplyr** merecen una mención especial. Para devolver los cuantiles, utiliza quantile() con los valores predeterminados o especifique el valor o los valores que desea con probs =.

Si desea devolver cuantiles por grupo, puede encontrar salidas largas y menos útiles si simplemente añade otra columna a group\_by(). Por lo tanto, pruebe este enfoque en su lugar: cree una columna para cada nivel de cuantil deseado.

Aunque **dplyr** summarise() ofrece ciertamente un control más fino, puede encontrar que todas las estadísticas de resumen que necesita pueden producirse con get\_summary\_stat() del paquete **rstatix**. Si se opera con datos agrupados, if devolverá 0%, 25%, 50%, 75% y 100%. Si se aplica a datos no agrupados, puede especificar los percentiles con probs = c(.05, .5, .75, .98).

### Resumir los datos agregados

Si comienza con datos agregados, al utilizar n() devuelve el número de filas, no la suma de los recuentos agregados. Para obtener sumas, utiliza sum() en la columna de recuentos de los datos.

Por ejemplo, digamos que se empieza con el dataframe de recuentos que se muestra a continuación, llamado linelist\_agg - muestra en formato "largo" los recuentos de casos por resultado y género.

A continuación creamos este dataframe de ejemplo de recuentos de casos del listado por resultado y sexo (se eliminan los valores faltantes para mayor claridad).

Para sumar los recuentos (en la columna n) por grupo, puede utilizar summarise() pero establecer la nueva columna igual a sum(n, na.rm=T). Para añadir un elemento condicional a la operación de suma, puede utilizar la sintaxis del subconjunto [ ] en la columna de recuentos.

### across() varias columnas

Puedes utilizar summarise() en varias columnas utilizando across(). Esto facilita la vida cuando se desea calcular las mismas estadísticas para muchas columnas. Coloque across() dentro de summarise() y especifique lo siguiente:

* .cols = como un vector de nombres de columnas c() o funciones de ayuda "tidyselect" (explicadas más adelante)
* .fns = la función a realizar (sin paréntesis) - puede proporcionar varias dentro de una lista()

A continuación, mean() se aplica a varias columnas numéricas. Se nombra explícitamente un vector de columnas a .cols = y se especifica una única función media (sin paréntesis) a .fns =. Cualquier argumento adicional para la función (por ejemplo, na.rm=TRUE) se proporciona después de .fns =, separado por una coma.

Puedes ser difícil conseguir el orden correcto de los paréntesis y las comas cuando se utiliza across(). Recuerde que dentro de across() debe incluir las columnas, las funciones y cualquier argumento extra necesario para las funciones.

Se pueden ejecutar varias funciones a la vez. A continuación se proporcionan las funciones mean y sd a .fns = dentro de una lista(). Tiene la oportunidad de proporcionar nombres de caracteres (por ejemplo, "mean" y "sd") que se añaden en los nuevos nombres de columna.

Aquí están esas funciones de ayuda "tidyselect" que puede proporcionar a .cols = para seleccionar columnas:

* everything() - todas las demás columnas no mencionadas
* last\_col() - la última columna
* where() - aplica una función a todas las columnas y selecciona las que son TRUE
* starts\_with() - coincide con un prefijo especificado. Ejemplo: starts\_with("date")
* ends\_with() - coincide con un sufijo especificado. Ejemplo: ends\_with("\_end")
* contains() - columnas que contienen una cadena de caracteres. Ejemplo: contains("tiempo")
* matches() - para aplicar una expresión regular (regex). Ejemplo: contains("[pt]al")
* num\_range() -
* any\_of() - coincide con el nombre de la columna. Es útil si el nombre puede no existir. Ejemplo: any\_of(date\_onset, date\_death, cardiac\_arrest)

Por ejemplo, para devolver la media de cada columna numérica utiliza where() y proporcione la función as.numeric() (sin paréntesis). Todo esto queda dentro del comando across().

### Pivote más amplio

Si prefiere su tabla en formato "ancho" puede transformarla utilizando la función **tidyr** pivot\_wider(). Es probable que tenga que renombrar las columnas con rename(). Para más información, consulta la página sobre [Pivoteo de datos](#pivoting-data).

El ejemplo siguiente comienza con la tabla "larga" age\_by\_outcome de la [sección de proporciones](#tbl_dplyr_prop). La creamos de nuevo y la imprimimos, para mayor claridad:

Para pivotar más ampliamente, creamos las nuevas columnas a partir de los valores de la columna existente age\_cat (estableciendo names\_from = age\_cat). También especificamos que los nuevos valores de la tabla provendrán de la columna existente n, con values\_from = n. Las columnas no mencionadas en nuestro comando de pivoteo (resultado) permanecerán sin cambios en el extremo izquierdo.

### Total de filas

Cuando summarise() opera con datos agrupados no produce automáticamente estadísticas "totales". A continuación, se presentan dos enfoques para añadir una fila de totales:

#### adornar\_totales() del ****conserje****

Si su tabla consiste sólo en recuentos o proporciones/porcentajes que pueden sumarse en un total, entonces puede añadir totales de suma usando adorn\_totals() de **janitor** como se describe en la sección anterior. Ten en cuenta que esta función sólo puede sumar las columnas numéricas - si desea calcular otras estadísticas de resumen total, vea el siguiente enfoque con **dplyr**.

A continuación, linelist se agrupa por género y se resume en una tabla que describe el número de casos con resultado conocido, los fallecidos y los recuperados. Al pasar la tabla por adorn\_totals() se añade una fila total en la parte inferior que refleja la suma de cada columna. Las funciones posteriores adorn\_\*() ajustan la visualización como se indica en el código.

#### resumir() en los datos "totales" y luego bind\_rows()

Si su tabla consta de estadísticas de resumen como la mediana(), la media(), etc., el enfoque adorn\_totals() mostrado anteriormente no será suficiente. En su lugar, para obtener los estadísticos de resumen de todo el set de datos debe calcularlos con un comando summarise() separado y luego vincular los resultados a la tabla de resumen agrupada original. Para hacer el enlace puede utilizar bind\_rows() de **dplyr** s descrito en la página de [unión de datos](#joining-data). A continuación se muestra un ejemplo:

Se puede hacer una tabla resumen de resultados por hospital con group\_by() y summarise() así:

Para obtener los totales, ejecuta el mismo comando summarise() pero agrupando los datos sólo por resultado (no por hospital), de la siguiente manera:

Podemos unir estos dos dataframes. Ten en cuenta que by\_hospital tiene 4 columnas, mientras que los totales tienen 3 columnas. Al utilizar bind\_rows(), las columnas se combinan por nombre, y cualquier espacio extra se rellena con NA (por ejemplo, los valores de la columna hospital para las dos nuevas filas de totales). Después de enlazar las filas, convertimos estos espacios vacíos en "Total" utilizando replace\_na() (véase la página de [limpieza de datos y funciones básicas](#cleaning-data-and-core-functions)).

Aquí está la nueva tabla con las filas "Total" en la parte inferior.

Esta tabla tiene un formato "largo", que puede ser lo que quieres. Opcionalmente, puedes pivotar esta tabla más ampliamente para hacerla más legible. Vea la sección sobre pivoteo más amplio arriba, y la página [Pivoteo de datos](#pivoting-data). También puede añadir más columnas, y organizarla de forma agradable. Este código está abajo.

Y luego puede imprimir esto muy bien como una imagen - abajo está la salida impresa con **flextable**. Puedes leer más en profundidad sobre este ejemplo y cómo lograr esta tabla "bonita" en la página [Tablas para la presentación](#tables-for-presentation).

## paquete ****gtsummary****

Si deseas imprimir tus estadísticas de resumen en un gráfico bonito y listo para su publicación, puede utilizar el paquete **gtsummary** y su función tbl\_summary(). El código puede parecer complejo al principio, pero los resultados se ven muy bien y se imprimen en su panel de RStudio Viewer como una imagen HTML. Lea una [viñeta aquí](http://www.danieldsjoberg.com/gtsummary/articles/tbl_summary.html).

También puede añadir los resultados de las pruebas estadísticas a las tablas de gtsummary. Este proceso se describe en la sección **gtsummary de la** página [Pruebas estadísticas simples](#stats_gt).

Para introducir tbl\_summary() mostraremos primero el comportamiento más básico, que realmente produce una tabla grande y bonita. Luego, examinaremos en detalle cómo hacer ajustes y tablas más a medida.

### Cuadro resumen

El comportamiento por defecto de tbl\_summary() es bastante increíble: toma las columnas que proporcionas y crea una tabla de resumen en un solo comando. La función imprime las estadísticas apropiadas para el tipo de columna: mediana y rango intercuartil (IQR) para las columnas numéricas, y recuentos (%) para las columnas categóricas. Los valores faltantes se convierten en "Desconocidos". Se añaden notas a pie de página para explicar las estadísticas, mientras que el N total se muestra en la parte superior.

### Ajustes

Ahora explicaremos cómo funciona la función y cómo hacer los ajustes. Los argumentos clave se detallan a continuación:

**by =**   
Puedes estratificar su tabla por una columna (por ejemplo, por resultado), creando una tabla de dos vías.

**statistic =Usa**   
una ecuación para especificar qué estadísticas mostrar y cómo mostrarlas. La ecuación tiene dos lados, separados por una tilde ~. En el lado derecho, entre comillas, está la visualización estadística deseada, y en el izquierdo están las columnas a las que se aplicará esa visualización.

* El lado derecho de la ecuación utiliza la sintaxis de str\_glue() de **stringr** (véase [Caracteres y cadenas](#characters-and-strings)), con la cadena de visualización deseada entre comillas y los propios estadísticos entre llaves. Puedes incluir estadísticas como "n" (para los recuentos), "N" (para el denominador), "media", "mediana", "sd", "max", "min", percentiles como "p##" como "p25", o porcentaje del total como "p". Consulta ?tbl\_summary para obtener más detalles.
* Para el lado izquierdo de la ecuación, puede especificar las columnas por su nombre (por ejemplo, edad o c(edad, sexo)) o utilizando ayudantes como all\_continuous(), all\_categorical(), contains(), starts\_with(), etc.

Un ejemplo sencillo de una ecuación estadística = podría ser como el siguiente, para imprimir sólo la media de la columna edad\_años:

Una ecuación un poco más compleja podría tener el aspecto de "({min}, {max})", incorporando los valores máximo y mínimo entre paréntesis y separados por una coma:

También puede diferenciar la sintaxis para columnas separadas o tipos de columnas. En el ejemplo más complejo de abajo, el valor proporcionado a statistc = es una **lista** que indica que para todas las columnas continuas la tabla debe imprimir la media con la desviación estándar entre paréntesis, mientras que para todas las columnas categóricas debe imprimir el n, el denominador y el porcentaje.

**dígitos =Ajustar**   
los dígitos y el redondeo. Opcionalmente, se puede especificar que sea sólo para columnas continuas (como a continuación).

**label =Ajustar**   
cómo debe mostrarse el nombre de la columna. Proporcione el nombre de la columna y su etiqueta deseada separados por una tilde. El valor por defecto es el nombre de la columna.

**missing\_text =Ajustar**   
cómo se muestran los valores faltantes. El valor por defecto es "Desconocido".

**type**   
=Se utiliza para ajustar cuántos niveles de la estadística se muestran. La sintaxis es similar a la de la estadística = en el sentido de que se proporciona una ecuación con columnas a la izquierda y un valor a la derecha. Dos escenarios comunes incluyen:

* type = all\_categorical() ~ "categorical" Fuerza a las columnas dicotómicas (por ejemplo, fiebre sí/no) a mostrar todos los niveles en lugar de sólo la fila "sí
* type = all\_continuous() ~ "continuous2" Permite estadísticas de varias líneas por variable, como se muestra en una sección posterior

En el siguiente ejemplo, cada uno de estos argumentos se utiliza para modificar la tabla resumen original:

### Estadísticas de varias líneas para variables continuas

Si desea imprimir varias líneas de estadísticas para variables continuas, puede indicarlo estableciendo el tipo = a "continuous2". Puedes combinar todos los elementos mostrados anteriormente en una tabla eligiendo qué estadísticas quiere mostrar. Para ello, debe indicar a la función que desea obtener una tabla introduciendo el tipo como "continuous2". El número de valores faltantes se muestra como "Desconocido".

Hay muchas otras formas de modificar estas tablas, incluyendo la adición de valores p, el ajuste del color y los títulos, etc. Muchas de ellas se describen en la documentación (introduzca ?tbl\_summary en Console), y algunas se dan en la sección de [pruebas estadísticas](https://epirhandbook.com/simple-statistical-tests.html).

## ****base**** R

Puedes utilizar la función table() para tabular y cruzar las columnas. A diferencia de las opciones anteriores, debe especificar el dataframe cada vez que haga referencia a un nombre de columna, como se muestra a continuación.

**ATENCIÓN:** Los valores NA (missing) **no** se tabularán a menos que se incluya el argumento useNA = "always" (que también podría establecerse como "no" o "ifany").

**CONSEJO:** Puedes utilizar el %$% de **magrittr** para eliminar la necesidad de repetir las llamadas al dataframe dentro de las funciones **base**. Por ejemplo, lo siguiente podría escribirse tabla %$%(resultado, useNA = "siempre")

Se pueden cruzar varias columnas enumerándolas una tras otra, separadas por comas. Opcionalmente, se puede asignar a cada columna un "nombre" como Outcome = linelist$outcome.

### Proporciones

Para devolver las proporciones, pase la tabla anterior a la función prop.table(). Utiliza el argumento márgenes = para especificar si desea que las proporciones sean de filas (1), de columnas (2) o de toda la tabla (3). Para mayor claridad, canalizamos la tabla a la función round() de **base** R, especificando 2 dígitos.

### Totales

Para añadir los totales de filas y columnas, pase la tabla a addmargins(). Esto funciona tanto para recuentos como para proporciones.

### Convertir en dataframe

Convertir un objeto table() directamente en un dataframe no es sencillo. A continuación se muestra un enfoque:

1. Cree la tabla, sin utilizar useNA = "always". En su lugar, convierta los valores NA en "(Missing)" con fct\_explicit\_na() de **forcats**.
2. Añade los totales (opcional) pasando por addmargins()
3. Pipe a la función **base de** R as.data.frame.matrix()
4. Enviar la tabla a la función **tibble** rownames\_to\_column(), especificando el nombre de la primera columna
5. Imprima, visualice o exporte según desee. En este ejemplo utilizamos flextable() del paquete **flextable** como se describe en la página [Tablas para la presentación](#tables-for-presentation). Esto imprimirá en el panel de visualización de RStudio como una bonita imagen HTML.

## Recursos

Gran parte de la información de esta página está adaptada de estos recursos y viñetas en línea:

[gtsummary](http://www.danieldsjoberg.com/gtsummary/articles/tbl_summary.html)

[dplyr](https://dplyr.tidyverse.org/articles/grouping.html)