# # Regresión univariable y multivariable

{#univariate-and-multivariable-regression}

Esta página muestra el uso de las funciones de regresión **básicas** de R, como glm() y el paquete **gtsummary,** para observar las asociaciones entre variables (por ejemplo, odds ratios, risk ratios y hazard ratios). También utiliza funciones como tidy() del paquete **broom** para limpiar los resultados de la regresión.

1. Univariante: tablas de dos en dos
2. Estratificado: estimaciones mantel-haenszel
3. Multivariable: selección de variables, selección de modelos, tabla final
4. gráficas forestales

Para la regresión de riesgos proporcionales de Cox, véase la página de [análisis de supervivencia](#survival-analysis).

**NOTA:** Utilizamos el término multivariable para referirnos a una regresión con múltiples variables explicativas. En este sentido, un modelo multivariable sería una regresión con varios resultados - véase este [editorial](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3518362/) para más detalles

## 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.

### Datos limpios

#### Almacenar las variables explicativas

Almacenamos los nombres de las columnas explicativas como un vector de caracteres. Esto se referenciará más adelante.

#### Convertir a 1's y 0's

A continuación convertimos las columnas explicativas de "sí"/"no", "m"/"f", y "muerto"/"vivo" a **1 / 0**, para cooperar con las expectativas de los modelos de regresión logística. Para hacer esto de manera eficiente, utilizamos across() de **dplyr** para transformar varias columnas a la vez. La función que aplicamos a cada columna es case\_when() (también de **dplyr**) que aplica la lógica para convertir los valores especificados en 1's y 0's. Vea las secciones sobre across() y case\_when() en la [página de Limpieza de datos y funciones básicas](#clean_across)).

Nota: el "." que aparece a continuación representa la columna que está siendo procesada por across() en ese momento.

#### Eliminar las filas con valores faltantes

Para eliminar las filas con valores faltantes, puede utilizar la función **tidyr** drop\_na(). Sin embargo, sólo queremos hacer esto para las filas a las que les faltan valores en las columnas de interés.

Lo primero que debemos hacer es asegurarnos de que nuestro vector explanatory\_vars incluye la columna edad (la edad habría producido un error en la operación anterior case\_when(), que sólo era para variables dicotómicas). A continuación, canalizamos linelist a drop\_na() para eliminar cualquier fila con valores faltantes en la columna de resultados o en cualquiera de las columnas explanatory\_vars.

Antes de ejecutar el código, el número de filas del listado es nrow(linelist).

El número de filas que quedan en linelist es nrow(linelist).

## Univariante

Al igual que en la página sobre [Tablas descriptivas](https://epirhandbook.com/descriptive-tables.html), su caso de uso determinará el paquete de R que utilice. Presentamos dos opciones para realizar análisis univariantes:

* Utiliza las funciones disponibles en para imprimir rápidamente los resultados en la consola. Utiliza el paquete **broom** para ordenar las salidas.
* Utiliza el paquete **gtsummary** para modelar y obtener resultados listos para su publicación

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

#### Regresión lineal

La función **base de** R lm() realiza una regresión lineal, evaluando la relación entre la respuesta numérica y las variables explicativas que se supone tienen una relación lineal.

Proporcione la ecuación como una fórmula, con los nombres de las columnas de respuesta y explicativa separados por una tilde ~. Además, especifique los datos a datos =. Defina los resultados del modelo como un objeto R, para utilizarlo más tarde.

A continuación, puede ejecutar summary() en los resultados del modelo para ver los coeficientes (estimaciones), el valor P, los residuos y otras medidas.

También se puede utilizar la función tidy() del paquete **broom** para obtener los resultados en una tabla. Lo que nos dicen los resultados es que por cada año de aumento de la edad la altura aumenta 3,5 cm y esto es estadísticamente significativo.

También puede utilizar esta regresión para añadirla a un **ggplot**, para hacer esto, primero juntamos los puntos de los datos observados y la línea ajustada en un dataframe utilizando la función augment() de **broom**.

También es posible añadir una simple recta de regresión lineal en **ggplot** utilizando la función geom\_smooth().

Consulta la sección de recursos al final de este capítulo para obtener tutoriales más detallados.

#### Regresión logística

La función glm() del paquete **stats** (parte de ) se utiliza para ajustar los modelos lineales generalizados (GLM).

glm() puede utilizarse para la regresión logística univariable y multivariable (por ejemplo, para obtener Odds Ratios). Aquí están las partes principales:

* fórmula = El modelo se proporciona a glm() como una ecuación, con el resultado a la izquierda y las variables explicativas a la derecha de una tilde ~.
* familia = Determina el tipo de modelo a ejecutar. Para la regresión logística, utiliza la familia = "binomial", para poisson utiliza la familia = "poisson". Otros ejemplos se encuentran en la tabla siguiente.
* data = Especifica tu dataframe

Si es necesario, también puede especificar la función de enlace mediante la sintaxis family = familytype(link = "linkfunction")). Puedes leer más en la documentación sobre otras familias y argumentos opcionales como pesos = y subconjunto = (?glm).

| **Familia** | **Función de enlace por defecto** |
| --- | --- |
| "binomio" | (enlace = "logit") |
| "gaussiano" | (enlace = "identidad") |
| "Gamma" | (enlace = "inverso") |
| "inverso.gaussiano" | (enlace = "1/mu^2") |
| "poisson" | (enlace = "registro") |
| "cuasi" | (enlace = "identidad", varianza = "constante") |
| "cuasibinomio" | (enlace = "logit") |
| "quasipoisson" | (enlace = "registro") |

Cuando se ejecuta glm() lo más habitual es guardar los resultados como un objeto R con nombre. A continuación, puede imprimir los resultados en su consola utilizando summary() como se muestra a continuación, o realizar otras operaciones con los resultados (por ejemplo, exponer).

Si necesita ejecutar una regresión binomial negativa, puede utilizar el paquete **MASS**; glm.nb() utiliza la misma sintaxis que glm(). Para un recorrido por diferentes regresiones, consulta la [página de estadísticas de UCLA](https://stats.idre.ucla.edu/other/dae/).

#### Univariante glm()

En este ejemplo estamos evaluando la asociación entre diferentes categorías de edad y el resultado de muerte (codificado como 1 en la sección Preparación). A continuación se muestra un modelo univariante del resultado por age\_cat. Guardamos la salida del modelo como modelo y luego la imprimimos con summary() en la consola. Observe que las estimaciones proporcionadas son las probabilidades logarítmicas y que el nivel de referencia es el primer nivel del factor age\_cat ("0-4").

Para modificar el nivel de referencia de una variable determinada, asegúrese de que la columna es del tipo Factor y mueva el nivel deseado a la primera posición con fct\_relevel() (véase la página sobre [Factores](#factors)). Por ejemplo, a continuación tomamos la columna age\_cat y establecemos "20-29" como línea de base antes de canalizar el dataframe modificado en glm().

#### Resultados de la impresión

Para la mayoría de los usos, hay que hacer varias modificaciones a los resultados anteriores. La función tidy() del paquete **broom** es conveniente para hacer presentables los resultados del modelo.

Aquí demostramos cómo combinar los resultados del modelo con una tabla de recuentos.

1. Obtenga las estimaciones de log odds ratio exponenciadas y los intervalos de confianza pasando el modelo a tidy() y estableciendo exponentiate = TRUE y conf.int = TRUE.

A continuación, se muestra el modelo de tiburón resultante:

1. Combine estos resultados del modelo con una tabla de recuentos. A continuación, creamos la tabla cruzada de recuentos con la función tabyl() de **janitor**, como se explica en la página de [tablas descriptivas](#descriptive-tables).

Este es el aspecto de este dataframe counts\_table:

Ahora podemos unir la tabla\_de\_conteo y los resultados del modelo horizontalmente con bind\_cols() (**dplyr**). Recuerde que con bind\_cols() las filas de los dos dataframes deben estar perfectamente alineadas. En este código, como estamos enlazando dentro de una cadena de tuberías, utilizamos . para representar el objeto de tuberías counts\_table mientras lo enlazamos con el modelo. Para terminar el proceso, utilizamos select() para elegir las columnas deseadas y su orden, y finalmente aplicamos la función **base** de R round() en todas las columnas numéricas para especificar 2 decimales.

Este es el aspecto del dataframe combinado, impreso de forma agradable como una imagen con una función de **flextable**. En [Tablas para la presentación](#tables-for-presentation) se explica cómo personalizar dichas tablas con **flextable**, o bien puede utilizar otros numerosos paquetes como **knitr** o **GT**.

#### Bucle de múltiples modelos univariantes

A continuación presentamos un método que utiliza glm() y tidy() para un enfoque más sencillo, véase la sección sobre **gtsummary**.

Para ejecutar los modelos en varias variables de exposición para producir odds ratios univariantes (es decir, sin controlar entre sí), puede utilizar el enfoque siguiente. Utiliza str\_c() de **stringr** para crear fórmulas univariantes (véase [Caracteres y cadenas](#characters-and-strings)), ejecuta la regresión glm() en cada fórmula, pasa cada resultado de glm() a tidy() y finalmente colapsa todos los resultados del modelo juntos con bind\_rows() de **tidyr**. Este enfoque utiliza map() del paquete **purrr** para iterar - véase la página sobre [Iteración, bucles y listas](#iteration-loops-and-lists) para más información sobre esta herramienta.

1. Cree un vector de nombres de columnas de las variables explicativas. Ya lo tenemos como explanatory\_vars de la sección de preparación de esta página.
2. Utiliza str\_c() para crear múltiples fórmulas de cadena, con el resultado a la izquierda, y un nombre de columna de explanatory\_vars a la derecha. El punto . sustituye al nombre de la columna en explanatory\_vars.
3. Pase estas fórmulas de cadena a map() y establezca ~glm() como la función a aplicar a cada entrada. Dentro de glm(), establezca la fórmula de regresión como as.formula(.x), donde .x se sustituirá por la fórmula de cadena definida en el paso anterior. map() realizará un bucle sobre cada una de las fórmulas de cadena, ejecutando regresiones para cada una.
4. Los resultados de este primer map() se pasan a un segundo comando map(), que aplica tidy() a los resultados de la regresión.
5. Por último, la salida de la segunda función map() (una lista de dataframes ordenados) se condensa con bind\_rows(), dando lugar a un dataframe con todos los resultados univariantes.

Esta vez, el objeto final modelos es más largo porque ahora representa los resultados combinados de varias regresiones univariantes. Clica para ver todas las filas del modelo.

Como antes, podemos crear una tabla de recuentos a partir del listado para cada variable explicativa, vincularla a los modelos y hacer una bonita tabla. Comenzamos con las variables, e iteramos a través de ellas con map(). Iteramos a través de una función definida por el usuario que implica la creación de una tabla de recuentos con funciones **dplyr**. Luego se combinan los resultados y se vinculan con los resultados del modelo de los modelos.

A continuación se muestra el aspecto del dataframe. Consulta la página sobre [Tablas para la presentación](#tables-for-presentation) para obtener ideas sobre cómo convertir esta tabla en una bonita salida HTML (por ejemplo, con **flextable**).

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

A continuación presentamos el uso de tbl\_uvregression() del paquete **gtsummary**. Al igual que en la página sobre [Tablas descriptivas](https://epirhandbook.com/descriptive-tables.html), las funciones de gtsummary hacen un buen trabajo al ejecutar estadísticas y producir salidas de aspecto profesional. Esta función produce una tabla de resultados de regresión univariante.

Seleccionamos sólo las columnas necesarias del listado (variables explicativas y la variable de resultado) y las introducimos en tbl\_uvregression(). Vamos a ejecutar una regresión univariante en cada una de las columnas que definimos como explanatory\_vars en la sección de preparación de datos (sexo, fiebre, escalofríos, tos, dolores, vómitos y age\_cat).

Dentro de la propia función, proporcionamos el método = como glm (sin comillas), la columna y = resultado (resultado), especificamos a method.args = que queremos ejecutar la regresión logística a través de la familia = binomial, y le decimos que exponencie los resultados.

La salida es HTML y contiene los recuentos

Hay muchas modificaciones que se pueden hacer a la salida de esta tabla, como ajustar las etiquetas de texto, poner en negrita las filas por su valor p, etc. Consulta los tutoriales [aquí](http://www.danieldsjoberg.com/gtsummary/articles/tbl_regression.html) y en otros lugares en línea.

## Estratificado

El análisis estratificado se está trabajando actualmente para **gtsummary**, esta página se actualizará a su debido tiempo.

## Multivariable

Para el análisis multivariable, volvemos a presentar dos enfoques:

* glm() y tidy()
* paquete **gtsummary**

El flujo de trabajo es similar para cada uno de ellos y sólo el último paso de elaborar una tabla final es diferente.

### Llevar a cabo la multivariable

Aquí utilizamos glm() pero añadimos más variables al lado derecho de la ecuación, separadas por símbolos de suma (+).

Para ejecutar el modelo con todas nuestras variables explicativas ejecutaríamos:

Si quiere incluir dos variables y una interacción entre ellas puede separarlas con un asterisco \* en lugar de un +. Si sólo especifica la interacción, sepárelas con dos puntos :. Por ejemplo:

Opcionalmente, puede utilizar este código para aprovechar el vector predefinido de nombres de columnas y volver a crear el comando anterior utilizando str\_c(). Esto puede ser útil si los nombres de sus variables explicativas cambian, o si no quiere escribirlos todos de nuevo.

#### Construir el modelo

Puedes construir su modelo paso a paso, guardando varios modelos que incluyan determinadas variables explicativas. Puedes comparar estos modelos con pruebas de razón de verosimilitud utilizando lrtest() del paquete **lmtest**, como se indica a continuación:

**NOTA: El** uso de la **base** anova(model1, model2, test = "Chisq) produce los mismos resultados

Otra opción es tomar el objeto modelo y aplicar la función step() del paquete **stats**. Especifique qué dirección de selección de variables desea utilizar al construir el modelo.

También puede desactivar la notación científica en su sesión de R, para mayor claridad:

Como se describe en la sección sobre el análisis univariante, pasamos la salida del modelo a tidy() para exponer las probabilidades logarítmicas y los IC. Finalmente, redondeamos todas las columnas numéricas a dos decimales. Desplácese para ver todas las filas.

Este es el aspecto del dataframe resultante:

### Combinar univariable y multivariable

#### Combinar con ****gtsummary****

El paquete **gtsummary** proporciona la función tbl\_regression(), que tomará los resultados de una regresión (glm() en este caso) y producirá una bonita tabla resumen.

Veamos la tabla:

También puede combinar varias tablas de salida diferentes producidas por **gtsummary** con la función tbl\_merge(). Ahora combinamos los resultados multivariables con los resultados univariables de **gtsummary** que creamos [anteriormente](#reg_gt_uni):

#### Combinar con ****dplyr****

Una forma alternativa de combinar los resultados univariables y multivariables de glm()/tidy() es con las funciones join **de dplyr**.

* Unir los resultados univariables de antes (univ\_tab\_base, que contiene los recuentos) con los resultados multivariables ordenados mv\_tab\_base
* Utilizar select() para mantener sólo las columnas que queremos, especificar su orden y renombrarlas
* Utiliza round() con dos decimales en toda la columna que sea de tipo Double

## forest plot

Esta sección muestra cómo producir un gráfico con los resultados de su regresión. Hay dos opciones, puedes construir un gráfico tú mismo usando **ggplot2** o usar un metapaquete llamado **easystats** (un paquete que incluye muchos paquetes).

Consulta la página sobre [los fundamentos de ggplot](#ggplot-basics) si no está familiarizado con el paquete de trazado **ggplot2**.

### Paquete ****ggplot2****

Puedes construir un gráfico de bosque con ggplot() trazando elementos de los resultados de la regresión multivariable. Añada las capas de los gráficos utilizando estos "geoms":

* estimaciones con geom\_point()
* intervalos de confianza con geom\_errorbar()
* una línea vertical en OR = 1 con geom\_vline()

Antes de trazar, es posible que desee utilizar fct\_relevel() del paquete **forcats** para establecer el orden de las variables/niveles en el eje y. ggplot() puede mostrarlos en orden alfanumérico, lo que no funcionaría bien para estos valores de categoría de edad ("30" aparecería antes de "5"). Vea la página sobre [Factores](#factors) para más detalles.

### paquetes ****easystats****

Una alternativa, si no desea el nivel de control fino que proporciona **ggplot2**, es utilizar una combinación de paquetes **easystats**.

La función model\_parameters() del paquete **parameters** hace el equivalente de la función tidy() del paquete **broom**. El paquete **see** acepta esos resultados y crea un gráfico forestal por defecto como un objeto ggplot().

## Recursos

El contenido de esta página se ha basado en estos recursos y viñetas en línea:

[Regresión lineal en R](https://www.datacamp.com/community/tutorials/linear-regression-R)

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

[Página de estadísticas de la UCLA](https://stats.idre.ucla.edu/other/dae/)

[regresión escalonada sthda](http://www.sthda.com/english/articles/36-classification-methods-essentials/150-stepwise-logistic-regression-essentials-in-r/)