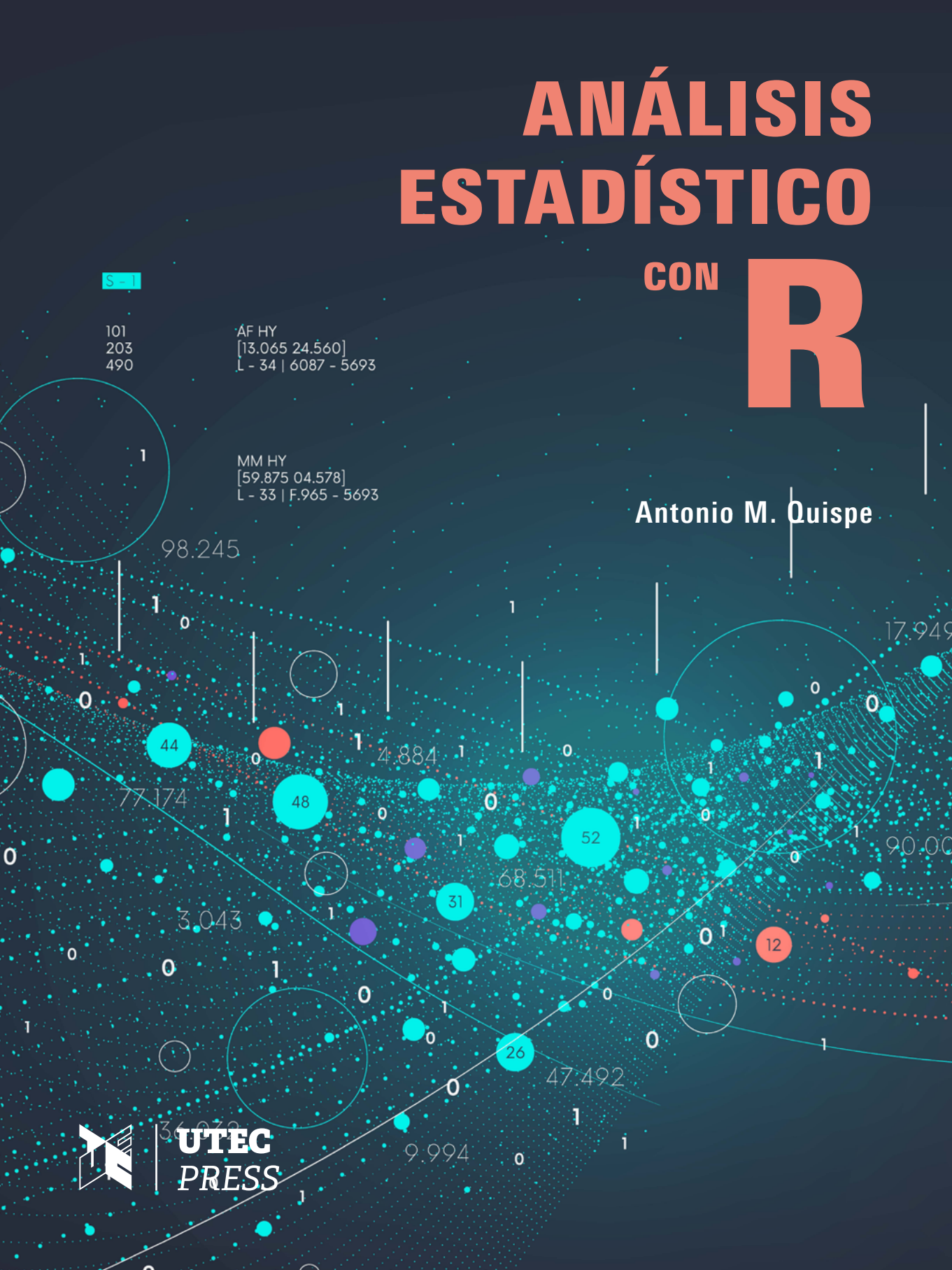
ANÁLISIS ESTADÍSTICO CON R

Antonio M. Quispe



UTEC PRESS

Antonio M. Quispe, MD, Ph.D.

Médico epidemiólogo, profesor e investigador

Universidad de Ingeniería y Tecnología
UTEC, Lima, Perú

Médico epidemiólogo, docente, estadístico, investigador (con certificación RENACYT), consultor internacional y comunicador científico, que en la última década ha contribuido a la misión de diferentes organizaciones nacionales e internacionales, tanto en el sector público como en el sector privado. Se ha desempeñado como investigador principal de sus propios *grants* en el U.S. Naval Medical Research Unit No 6 y en la Bill & Melinda Gates Foundation; como consultor en FHI-360, Food for Hungry y la Organización Mundial de Salud Animal; como médico epidemiólogo en el Grupo de Trabajo Prospectiva, como asesor del ministro de Salud y como presidente del Grupo de Trabajo de Inteligencia Estratégica, en su momento encargados de asesorar al gobierno del Perú en su lucha por mitigar el impacto de la pandemia del COVID-19; y como docente en la Universidad de Ingeniería y Tecnología, la Universidad Continental y la Universidad Peruana Cayetano Heredia, entre otras. En la actualidad, a la par de su carrera como docente e investigador, realiza diversos esfuerzos como activista científico, brigadista y activista de derechos humanos, liderando diferentes iniciativas para informar la toma de decisiones y contribuir a la salud pública, un país más justo y con menos inequidad de acceso a la salud y a una vida plena.

ANÁLISIS ESTADÍSTICO CON R

Antonio M. Quispe

UTEC PRESS

Jr. Medrano Silva 165, Barranco, Lima 04, Perú
Telf: (51-1) 2305020
Correo-e: utecpress@utec.edu.pe
URL: http://www.utec.edu.pe
ISBN eBook: 978-612-48515-4-4

Primera edición digital: Lima, UTEC Press, mayo 2023

Hecho el depósito legal en la Biblioteca Nacional del Perú: 2023-03806

Diseño y diagramación: James Hart Arbulú
Revisión de texto y cuidado de edición: Carlo Trivelli Avila

© Quispe, Antonio M.
Análisis estadístico con R. Lima, UTEC Press, 2023.

ÍNDICE

A mis padres, a mi familia, a mis queridos Renato y Alessandro, a mis estudiantes y a mi Estrella. Ustedes son mi inspiración y fuerza para continuar sumando en la lucha por una toma de decisiones basada en evidencias y por una salud pública de calidad en el Perú y en la región.

Este es el segundo libro de la trilogía de libros de texto del Club de Redacción de Artículos Científicos, que tiene por objeto constituirse en una intervención estructural para romper la inequidad de acceso a recursos académicos que afecta a los jóvenes investigadores en el Perú. El primero de estos libros aborda las bases de la metodología en investigación científica. En este, el segundo, aprenderán las bases del análisis estadístico con R. Y en el tercero, las bases de la epidemiología clínica y de campo. Juntos, estos tres libros les permitirán dar un salto cualitativo y cuantitativo a quienes quieren formarse como investigadores.

Como becario Fogarty, mi sueño era regresar a mi país a formar a la próxima generación de investigadores mientras encontraba un nicho donde sacarle lustre a mis habilidades como médico, estadístico, epidemiólogo, docente e investigador y dar el siguiente salto en mi carrera. Pero mi principal justificación era pagar lo que entendía era mi deuda social, ya que casi la totalidad de mis más de tres décadas de estudios fueron financiadas con becas públicas o privadas. Sin embargo, el reto resultó más grande y duro de lo que esperaba. Y es que ¿cómo hace un expatriado para encontrar aprendices o *mentees* a quienes formar y convencerlos de que investigar y publicar era una de las mejores inversiones que podían hacer con su tiempo libre en la universidad?

Por encima de cualquier otra consideración, mi anhelo era escribir libros que tuvieran el mismo estándar de los libros de texto con los que mis profesores me entrenaron en el doctorado de Epidemiología y Control de Enfermedades Globales en la Escuela de Salud Pública Bloomberg de la Universidad de Johns Hopkins, considerada la mejor escuela de salud pública del mundo desde 1994. Y es que no hay nada más aleccionador que aprender de los autores del libro de texto que uno está cursando y, sobre todo, de quienes llevan décadas investigando las materias de cada uno de nuestros cursos. Ese estándar era precisamente el que quería imprimir para mis propios cursos y la principal motivación para redactar la trilogía de libros de texto de mi Club de Redacción de Artículos Científicos, el CRAC.

El CRAC, desde su concepción, fue planificado como una intervención estructural dirigida a contribuir a romper esa horrible inequidad de acceso a recursos de investigación en el Perú, que beneficiaba a algunas universidades en detrimento de la mayoría. Y es que luego de los primeros talleres del CRAC, me quedó claro que en el Perú las universidades privadas o de Lima tenían un mayor acceso a recursos de investigación que las universidades nacionales del resto del país, desigualdad que al ser injusta desde el punto de vista social, se convertía en una inequidad perpetuada por años. Tal inequidad, en la práctica, condenaba a los estudiantes de ciencias de la salud de la gran mayoría de las universidades al ostracismo científico y mermaba severamente sus probabilidades de contribuir al conocimiento científico. Y lo que es peor, los convertía en víctimas de las pseudociencias y de la charlatanería de docentes de cursos de investigación que nunca habían publicado un manuscrito científico.

El primer taller del CRAC me di el gusto de dictarlo en mi *alma mater*, la Universidad Nacional Mayor de San Marcos, en el año 2019, en el pequeño auditorio de la Sociedad Científica de San Fernando, a un pequeño grupo de soñadores que, como yo, creían que el cielo era el límite y que el mundo estaba ahí para conquistarlo. Esta selección "natural" de talento nacional que son los estudiantes de la Facultad de Medicina de la Universidad Nacional Mayor de San Marcos me enseñó que si bien las ganas nunca faltan, lo cierto es que el reto de formarlos como investigadores jóvenes requería un programa validado que verdaderamente les permitiera adquirir las habilidades mínimas necesarias para investigar y publicar.

Enseñar a investigar y publicar no es un reto menor. Requiere de tiempo, dedicación, un programa curado y contenidos debidamente validados, así como un instructor certificado como docente e investigador. Validar los contenidos del CRAC tomó en total ocho iteraciones y validar y optimizar el formato, cinco iteraciones más. Terminado el proceso se identificaron ocho habilidades esenciales para investigar y publicar: 1) planteamiento de la pregunta de investigación; 2) revisión sistemática de literatura científica; 3) buenas prácticas de investigación; 4) diseño de estudios cuantitativos; 5) cálculo de tamaño de muestra; 6) análisis estadístico de datos; 7) redacción científica; y 8) lectura crítica.

Este segundo libro de la trilogía que comprende este proyecto incluye la guía de análisis estadístico con R para investigadores jóvenes y el mismo tiene por objetivo servir como libro de texto del curso de estadística del CRAC. Quienes lean y practiquen con este segundo libro adquirirán las siguientes habilidades: 1) cómo instalar

y redactar códigos de análisis de datos en R y R Studio; 2) manejo de datos en R; 3) visualización de datos; 4) análisis exploratorio de datos; 5) análisis bivariado y prueba de hipótesis; 6) análisis de regresión lineal; 7) análisis de regresión logística; 8) análisis de regresión de Poisson; 9) análisis de regresión binomial; 10) análisis de regresión de Cox. Esta son habilidades que todo investigador bien formado debe conocer y de las cuales no puede prescindir. Ahora, más que solo una guía, esperamos que este libro sea una inspiración para todos los jóvenes que desean iniciarse en el fascinante mundo de la investigación y no solo aprendan cómo publicar sino que eventualmente puedan contribuir significativamente al conocimiento científico.

Abril del 2023

CAPÍTULO 1

INTRODUCCIÓN A R Y RSTUDIO

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Diferenciar R de RStudio y conocer la funcionalidad de los paneles o ventanas de RStudio.
- Definir correctamente las variables y diferenciar los tipos de datos que reconoce R.
- Utilizar los operadores y funciones más comunes.
- Realizar anotaciones en los códigos.
- Instalar y cargar paquetes.
- Configurar tu directorio de trabajo.
- Importar cualquier tipo de base de datos desde R.
- Redactar de forma eficiente en los scripts.

Primeros pasos con R

¿Qué es R?

R es un lenguaje de programación para hacer análisis estadísticos desarrollado por Ross Ihaka y Robert Gentleman en la Universidad de Auckland, Nueva Zelanda, a principios de los años noventa. Desde que fue establecido como un lenguaje de código abierto en 1995, R ha ido ganando adeptos entre analistas de datos de distintos ámbitos (desde la bioinformática al análisis financiero o los modelos climáticos) en todo el mundo.

R es usado para explorar, analizar y comprender datos provenientes de cualquier tipo de fuente. Una de sus ventajas más importantes es que ofrece la posibilidad de explorar, manipular y mostrar datos gráficamente con facilidad. Otra de

sus ventajas es que permite automatizar las tareas analíticas repetitivas a través de la creación de funciones simples, por lo que permite realizar, de manera eficiente y efectiva, análisis que de otro modo requerirían de una gran de cantidad de tiempo y programación.

Como proyecto de código abierto, R tiene su centro en R Core, un grupo de una veintena de desarrolladores que mantienen el lenguaje y guían su evolución y que cuentan con el soporte de la R Foundation, una organización sin fines de lucro que les da sostenibilidad. Alrededor de R hay toda una comunidad de usuarios que ha ido desarrollando paquetes, también de libre acceso, que agregan funcionalidades a R. Estos desarrollos, junto con las nuevas versiones de R y la documentación y manuales que las acompañan se encuentran reunidos en la Comprehensive R Archive Network (CRAN), la red que aloja el archivo integral o comprehensivo de R (la puedes encontrar en https://cran.r-project.org/).

¿Por qué instalar RStudio?

RStudio es un entorno de desarrollo integrado (integrated development environment o IDE en inglés) para crear y ejecutar código en R. Si bien el análisis de datos y la presentación gráfica de resultados se puede realizar directamente en R, sin recurrir a un IDE, RStudio facilita enormemente el trabajo con R. RStudio también es un programa de código abierto, de libre acceso, y se lo encuentra en versiones para Linux, Windows y macOS. Como veremos enseguida, cuenta con cuatro paneles: un editor de código, una consola para la ejecución directa de código, un visor del entorno y un visor de archivos, gráficos, paquetes y ayuda.

Instalación de R y RStudio

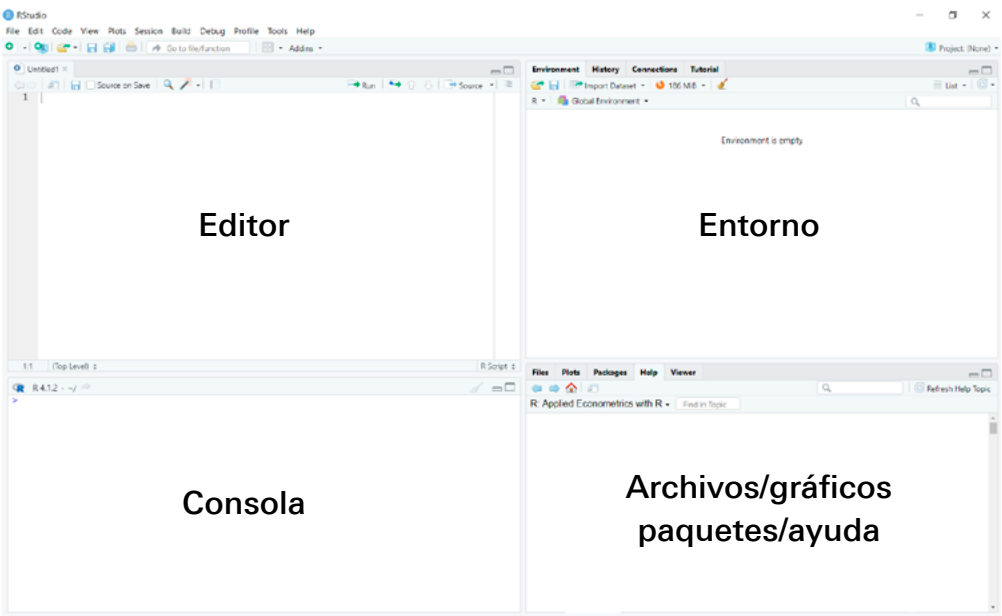
Para instalar R en Windows, macOS o Linux, sigue los siguientes pasos:

- Dirígete a https://www.r-project.org/
- Haz click en “download R”, texto que se encuentra en el primer párrafo, bajo el título “Getting Started”. Esto te llevará a una página en la que verás un listado de los sitios-espejo del CRAN.
- Selecciona una ubicación CRAN cercana a ti y haz click en el enlace correspondiente.

- En la parte superior de la página aparecerán las opciones para descargar R, ya sea para Linux, macOS o Windowns. Selecciona el sistema operativo apropiado y descarga el archivo que aparece en la parte superior de la página.
- Ejecuta el programa de instalación y acepta las opciones de instalación predeterminadas.
- Después de instalar R, instala RStudio. Puedes descargar gratuitamente la última versión de https://www.rstudio.com/products/rstudio/download/#download

Conociendo nuestro entorno de trabajo en RStudio

Al abrir Rstudio por primera vez, el entorno de trabajo presenta tres paneles. Para trabajar, sin embargo, necesitamos abrir el cuarto panel (el editor). Para hacerlo vamos al menú de la parte superior de la pantalla y abrimos el menú desplegable "File" y escogemos la primera opción: "New File". Esta, a su vez, despliega otro menú, en el que se presentan los distintos formatos que puede adquirir nuestro archivo de trabajo (por ejemplo, R Script, R Notebook o R Markdown). Volveremos sobre estas opciones más adelante. Por el momento escoge la primera opción: R Script. Al hacerlo aparecerá el panel del editor y tu entorno de trabajo se verá así:



Editor

Se ubica en la parte superior izquierda. En el panel de edición puedes escribir y editar los comandos en R que juntos forman un script. Antes de comenzar a escribir un script R sin título, siempre debes guardar el archivo con un nuevo nombre de archivo. De esta manera, si algo en tu computadora falla mientras estás trabajando, RStudio tendrá tu código esperándote cuando vuelvas a abrir el programa.

Entorno

Ubicado en la parte superior derecha, este panel muestra los nombres de los objetos de datos que vas creando (vectores, matrices y marcos de datos). Al hacer click sobre uno de los objetos puedes visualizar el número de filas y columnas de sus datos. Puedes encontrar también la pestaña "History" que guarda el historial con todos los comandos ejecutados en la consola.

Consola

La consola es la parte central de R, ya que ahí se muestran los resultados que se obtienen al ejecutar los diferentes comandos. Puedes escribir el código directamente en la consola y obtener una respuesta inmediata.

Archivos/gráficos/paquetes/ayuda

Se encuentra en la parte inferior derecha y tiene muchas pestañas útiles.

- La pestaña "Files" permite ver el historial de archivos con los que trabajaste en el programa; además, desde ahí puedes configurar tu directorio de trabajo (navega hasta que encuentres la carpeta en la que deseas guardar tus archivos y haz clic en "More" y luego en "Set As Working Directory").
- La pestaña "Plots" permite visualizar los gráficos que se generen en el editor; desde esta pestaña puedes exportar tus gráficos con distintas extensiones, por ejemplo, pdf o jpg.
- La pestaña "Packages" permite ver los paquetes descargados y guardados en el disco duro, así como instalar y actualizar paquetes. Los paquetes que presentan el símbolo de "check" son los que están cargados.
- La ventana "Help" brinda información sobre los paquetes o funciones y ofrece manuales de guía para el usuario.

- La ventana "Viewer" muestra los resultados que se obtienen al generar reportes mediante funcionalidades como R Markdown (sobre esto volveremos más adelante).

Ejecución de códigos

Los comandos se pueden escribir tanto en el editor como en la consola. Pero hay importantes diferencias entre hacerlo en uno u otro panel. En el editor los comandos no se ejecutan automáticamente, mientras que en la consola sí. Por otro lado, lo que escribes en el editor se puede guardar como un archivo para volver a él más tarde o para compartirlo, mientras que lo que se escribe en la consola no se puede guardar y se pierde cuando cierras la sesión de Rstudio.

En el editor, los comandos se pueden ejecutar de las siguientes formas:

- Hacer click antes del código y presionar ctrl + Enter.
- Resaltar todo el código que se desea ejecutar y presionar Enter.
- Copiar y pegar el código directamente en la consola después de `(>)` y presionar Enter.

El código en el editor se ve así:

```
1+2
## [1] 3
```

Si ejecutas el comando, este aparecerá en la consola del siguiente modo:

```
> 1+2
[1] 3
```

Como puedes ver, lo que escribes en la consola no se ve reflejado en el editor, mientras que los comandos que ejecutas en el editor sí aparecen en la consola.

Se recomienda escribir en el script para posteriormente guardar el código con un nombre de archivo, ya que, en la consola, los códigos ejecutados se pierden cuando cierra sesión.

Variables y tipos de datos

¿Qué es una variable?

Una variable expresa una cantidad, cualidad o propiedad que se puede medir. Al programar, se utilizan las variables para almacenar datos, por lo que también se les denomina contenedores de datos. Los datos que almacena una variable pueden ser de diferentes tipos. Por último, el estado de una variable cuando esta se mide es su valor (y este valor puede cambiar de una medición a otra).

◊ *Reglas para nombrar variables en R*

- Los nombres de variables no pueden contener espacios.
- Un nombre de variable puede comenzar con un punto, pero el punto no debe seguir al número. Si el punto inicial no va seguido de un número, entonces es válido.
- Un nombre de variable no debe comenzar con un número.
- Un nombre de variable puede contener letras, números, guiones bajos y puntos.
- Los guiones bajos no pueden ser el primer caracter del nombre de la variable.
- Hay palabras que no se pueden usar tales como TRUE, FALSE, NULL, entre otras.

Tipos de datos

Es importante conocer los tipos de datos con los que se puede trabajar en R. En algunos casos no conocer estas diferencias puede generar entradas de datos defectuosas por lo que debemos ser conscientes de los diferentes tipos de datos para identificar y corregir los errores que se nos puedan presentar. Por ejemplo, puede ocurrir que, debido a un problema en el ingreso de información en una base de datos, la variable “fecha” sea analizada como una secuencia de caracteres o que una variable numérica sea analizada como una secuencia de caracteres debido a errores tipográficos.

◊ *Datos numéricos (continuos)*

R puede leer como `numeric` o `num` los valores que son números o tienen decimales. Los números enteros son números sin decimales (p. ej., 3, 4 y 5), mientras que `double` significa formato de "coma flotante de doble precisión" (p. ej., 1.569, 4.5747).

Por lo general, no importa si R está clasificando sus datos continuos como `numeric/num/double/int`, pero es bueno tener en cuenta estos diferentes términos, ya que los verás en los mensajes de R.

◊ *Datos de caracter*

Se utiliza para especificar categorías, caracteres o valores de cadena en una variable. En R los caracteres son palabras o letras. Este tipo de datos también se conoce como tipo de datos de cadena. La cadena (texto) tiene que estar entre comillas simples '…' o entre comillas dobles "…"

◊ *Datos factoriales*

Los datos factoriales son un tipo particular de datos de caracter. Como estos, son valores expresados en caracteres, pero los valores de los datos factoriales son inventarios cerrados, como en el caso de que establezcamos una variable para el sexo de los habitantes de un país: esta solo presentará dos valores posibles: "masculino" y "femenino"; lo mismo puede suceder con la edad de las personas: en vez de medirla en años, podemos medirla utilizando grupos de edad, como "niños", "jóvenes", "adultos" y "adultos mayores". Estos cuatro valores posibles de nuestra variable se conocen como niveles.

Los datos factoriales son útiles para el modelado estadístico y para presentar la información de manera más clara. Por eso, en muchos casos es conveniente convertir datos numéricos en datos factoriales; por ejemplo, podemos tener el índice de masa corporal (IMC) para un conjunto de personas, pero estar interesados solo en destacar cuántos tienen sobrepeso y, por lo tanto, ven su salud en riesgo. Así, si reducimos la gran variabilidad de valores del IMC que encontramos entre la población que estamos estudiando (los datos, con seguridad, serán todos distintos pues es difícil que dos personas tengan exactamente el mismo IMC) a solo dos valores

factoriales (“normal” y “sobrepeso”) podremos presentar los datos con mayor claridad para nuestros propósitos (como determinar el porcentaje de personas con sobrepeso o su número total).

◊ *Datos de fecha*

Los datos de fecha representan a menudo un desafío en el manejo de datos. Las funciones `as.Date`, `POSIXct` y `POSIXlt` se pueden utilizar para crear este tipo de datos. Antes de conocer los diferentes usos de dichas funciones, veamos la siguiente tabla con las especificaciones del formato de fecha más utilizadas:

Símbolo	Descripción	Ejemplo
%S	Segundos (00-59)	15
%M	Minutos (00-59)	23
%H	Hora (00-23)	17
%d	Día del mes	07
%m	Mes en número	10
%b	Mes (abreviado)	Jul
%B	Mes (nombre completo)	Julio
%y	Año (dos dígitos)	22
%Y	Año (cuatro dígitos)	2022

as.Date

Esta función es útil para manejar fechas sin horas. Su argumento `format` permite especificar una variedad de formatos de fecha. De manera predeterminada, muestra el año con cuatro dígitos, seguido de un mes y un día, todos ellos separados por guiones.

En el siguiente ejemplo, agregamos una fecha y la guardamos en el objeto **date** (sobre objetos, funciones y argumentos, daremos detalles más adelante). Esta fecha no se encuentra en el formato predeterminado, así que nos apoyaremos en la tabla anteriormente presentada.

```
date <- as.Date('Julio 5, 2022',format='%B %d, %Y')
```

Observamos

```
date
## [1] “2022-07-05”
```

Si queremos obtener el nombre completo del mes, entonces realizamos lo siguiente:

```
format(date, format =”%B”)
## [1] “Julio”
```

POSIX

Las clases `POSIX` son útiles para manejar fechas y horas con control de zonas horarias. `POSIXct` representa la cantidad de segundos desde el inicio del 1 de enero de 1970. La cantidad de segundos antes de ese tiempo se representan como números negativos y los posteriores como positivos. Además, esta clase representa la hora del calendario.

Por otro lado, las entradas que tiene `POSIXlt` son segundos, minutos, horas, día del mes, mes del año, años desde 1900, día de la semana, día del año e indicador del horario de verano; asimismo, representa la hora local en forma de listas.

Veamos un ejemplo de cómo manipular fechas que no tienen el formato apropiado. Primero, creamos un ejemplo de fecha y hora y lo guardamos en el objeto denominado **fecha**.

```
fecha <- “22-10-2022  09:27”
```

Observamos

```
fecha
## [1] “22-10-2022  09:27”
```

Utilizamos la función `as.POSIXlt` para convertir el dato en el formato correcto

```
fecha_POSIXa <- as.POSIXlt(fecha)
```

Observamos

```
fecha_POSIXa
## [1] "0022-10-20 LMT"
```

Vemos que la salida es incorrecta, la fecha no está en orden y nuestro objeto no contiene la hora. Esto se debe a que tenemos que especificar la fecha y hora dentro de la función.

```
fecha_POSIXb <-  as.POSIXlt (fecha,
                        format =  "%d-%m-%Y %H:%M" )
```

Observamos

```
fecha_POSIXb
## [1] "2022-10-22 09:27:00 -05"
```

Ahora sí nos aparecen la fecha y hora. Se puede comprobar que tiene el formato adecuado utilizando la función `str` que nos permite conocer la estructura de nuestros datos. Vamos a comparar los objetos **fecha** y **fecha_POSIXb**.

```
str(fecha)
##  chr "22-10-2022  09:27"
str(fecha_POSIXb)
##  POSIXlt[1:1], format: "2022-10-22 09:27:00"
```

El objeto **fecha** está en formato de caracter, mientras que el segundo objeto tiene el formato apropiado de fecha.

Como hemos visto, con R se puede conocer la cantidad de días, semanas y meses, también las zonas horarias y analizar las fechas en prácticamente cualquier formato, pero para ello tienes que indicarle a R cómo se formatea su fecha (p. ej., el nombre del mes abreviado o sin abreviar, el año en 2 o 4 dígitos, etc.). Cabe agregar que es importante saber si R ha entendido cuáles de sus columnas contienen información de fecha u hora y cuáles son solo caracteres normales.

Objetos

El objeto es el concepto principal sobre el cual se fundamenta la programación orientada a objetos (POO) que caracteriza a un lenguaje como R. Un objeto puede ser visto como una entidad que posee determinados atributos y efectúa acciones. Manejar de forma adecuada los objetos permite aminorar la complejidad de trabajar con R. El objeto más común es el conjunto de datos que es representado como una matriz de filas y columnas. Sin embargo, R te permitirá guardar diferentes tipos de datos almacenándolos dentro de un nombre u objeto, de esta manera solo llamando al nombre del objeto podrás ver los datos guardados en su interior.

¿Existe una diferencia entre las variables y los objetos?

No existe diferencia entre las variables y los objetos desde la perspectiva del tipo de datos, puesto que todo en R es un objeto: un número, un texto, un valor lógico, un conjunto de datos, una matriz, etc.

En el panel "Entorno" de RStudio se guardan los objetos, pero requieren tener un nombre para que puedas verlos. Al igual que las variables, los nombres de los objetos no pueden estar separados por espacios, por lo que se recomienda usar un guion bajo para separar palabras.

R como calculadora

R se puede utilizar también como una calculadora interactiva. Utilizamos el paréntesis cada vez que sea necesario agrupar operaciones. Por ejemplo, si deseamos calcular la siguiente operación matemática (2+5)^3 - 2*40/4, la escribimos en el editor y ejecutamos el comando. Obtenemos el siguiente resultado:

```
(2+5)^3 - 2*40/4
## [1] 323
```

Operadores

Una de las ventajas del lenguaje R es que podemos automatizar los procesos para evitar repeticiones innecesarias, los operadores, en ese sentido, son muy útiles. Si queremos trabajar con objetos es importante conocer los operadores. Estos son símbolos que le dicen a R cómo manejar diferentes datos u objetos. Veremos los operadores que son más útiles al momento de trabajar con objetos.

Operador de asignación: `<-`

Si queremos utilizar el resultado de la operación matemática anterior en un segundo cálculo, en lugar de escribir nuevamente la operación matemática cada vez que la necesitemos, podemos crear un nuevo objeto que almacene dicho resultado. El resultado anterior lo vamos a guardar o asignar en un nuevo objeto llamado **x**. El operador de asignación es el símbolo "menor que", seguido del signo "menos" (piensa este operador de asignación como una flecha que apunta a la izquierda). La función de este operador se puede entender como *asignar el valor del lado derecho de la flecha al nombre de la variable que está al lado izquierdo de la flecha*. Se ve de esta manera:

```
x <-(2+5)^3 - 2*40/4
```

Para ver el contenido de la variable **x**, simplemente escribimos `x` y presionamos "enter".

```
x
## [1] 323
```

Ahora, almacenaremos el resultado de **x-5** en un objeto llamado **y**.

```
y <- x-5
y
## [1] 318
```

La flecha de asignación también puede usarse apuntando hacia la derecha, pero, por lo general, no es utilizada de esa forma. Si deseas utilizar la asignación derecha, el objeto que quieras almacenar debe estar a la izquierda, de lo contrario se producirá un error. Cabe agregar que el operador `=` funciona también como operador de asignación.

Operador de encadenamiento: `%>%`

El beneficio de `%>%` es que nos permite encadenar múltiples operaciones de forma lineal. El operador permite escribir una secuencia de operaciones de izquierda a derecha.

Por ejemplo, si queremos seleccionar determinados datos, agruparlos por categorías y luego ordenarlos, lo podemos hacer usando un código de línea con el operador `%>%`. La idea es que estas funciones sean bloques de construcción para producir el resultado deseado. Veremos con mayor detalle en el siguiente capítulo la aplicación de este operador.

Otros operadores

- `$` : Subconjunto de un data frame
- `o` : Lista con nombres
- `:` : Generador de secuencias
- `~` : Formulación de modelos
- `==` : Igual a
- `!=` : No igual a

Funciones

Las funciones son fundamentales para automatizar un conjunto de comandos, incluidos los operadores. Las funciones se ejecutan escribiendo su nombre entre corchetes, en los que proporcionamos uno o más parámetros (argumentos) separados por comas. Hay muchas funciones que están predefinidas o se encuentran dentro de los paquetes de R (esto se abordará más adelante).

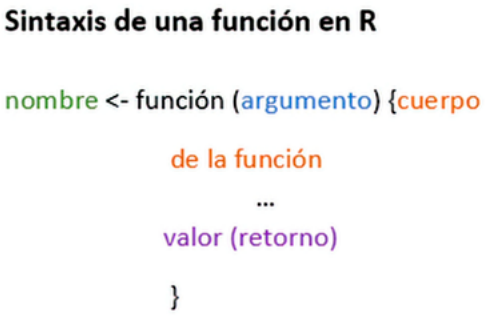
Funciones útiles

ls	Lista de objetos de su espacio de trabajo
c	Concatenar o combinar datos
rm	Eliminar objeto
help	Abrir instrucciones de ayuda
getwd	Directorio de trabajo actual
setwd	Establecer un nuevo directorio

Argumentos de la función

Los argumentos son los parámetros proporcionados a una función y ésta puede tener múltiples argumentos. En R, podemos usar los argumentos que queramos mientras estén separados por una coma. No existe un límite en la cantidad de argumentos en una función en R.

Como se observa en la siguiente imagen, los argumentos contienen valores que son proporcionados a la función. Las expresiones dentro del cuerpo de la función son ejecutadas cada vez que la función es llamada y, con la última línea del código, la función devuelve el resultado.



Comentarios y anotaciones con `#`

Por lo general, se colocan para explicar qué hace una línea de código o qué se supone que debe producir. Es útil para hacer notas y recordar, de esta manera, el propósito

de una línea de código o ayudar a alguien a consultar el mismo. También se puede utilizar el signo `#` para impedir la ejecución de una línea de código (al detectar el signo `#`, R asume que lo que viene no es ejecutable). R solo admite códigos de una sola línea, por lo que si deseas agregar más comentarios tendrá que utilizar `#` cada vez que inicias un comentario en una nueva línea. Aquí un par de ejemplos:

```
# asignando valores a la variable "a"
a <- c(7, 8, 9, 10)
#asignando valores a la variable "a"
#usamos la función c() para asignar más de un valor a nuestro
objeto
a <- c(7, 8, 9, 10)
```

Paquetes

Paquetes predeterminados

Los paquetes en R contienen conjuntos de datos y funciones específicas para resolver cuestiones concretas. R viene con un "paquete base" que contiene las funciones básicas y los conjuntos de datos, así como las funciones estadísticas y gráficas estándar que permiten el funcionamiento de R.

Gran parte de la funcionalidad de R proviene de este paquete. Si queremos conocer la lista de funciones predeterminadas que nos provee R, realizamos lo siguiente:

```
help(package ="base")
```

Instalar paquetes de descarga

Existen, también, otros paquetes disponibles para su descarga e instalación desde CRAN o los repositorios de GitHub. Para instalar paquetes desde CRAN usamos la función `install.packages()` y asignamos el nombre del paquete que deseamos instalar entre comillas.

Por ejemplo, vamos a instalar el paquete llamado `readr`. Este paquete lee datos de archivos delimitados por comas (CSV) o tabuladores (TSV). Sus funciones más conocidas son `read_delim()`, `read_csv()`, `read_tsv()` y `read_table`.

```
#install.packages("readr")
```

Carga de paquetes

Una vez que se ha descargado un paquete, es necesario incorporarlo a la biblioteca para que esté activo y podamos utilizar sus funcionalidades. Para ello usamos `library()` e incluimos el nombre del paquete al interior de los paréntesis. Para cargar el paquete `readr` ejecuta lo siguiente:

```
library(readr)
```

Espacio de trabajo

Configuración del directorio de trabajo

El directorio de trabajo es el lugar en el que guardamos los distintos archivos que utilizaremos para trabajar en R. Estos archivos pueden tener distintos formatos (.xlsx, .xls, .sav, .csv y .dta). R se encargará de buscar las bases de datos en este lugar para importarlas y, en caso lo deseemos, exportarlas. Así que el primer paso es establecer nuestro directorio de trabajo.

◊ *¿Cuál es nuestro directorio de trabajo actual?*

Para conocer el directorio de trabajo actual utilizamos la función `getwd()`. Nótese que esta función simple no acepta argumentos entre los paréntesis, así que ingrésala y ejecútala sin añadir ningún valor o argumento. Verás un resultado parecido a este:

```
getwd() #Muestra el directorio actual
```

```
## [1] "C:/Users/Documents/bookdown-word"
```

◊ *Establecer un nuevo directorio de trabajo*

Si deseamos cambiar de directorio, usamos la función `setwd()`. Dentro del paréntesis se escribe la nueva dirección, la cual debe estar entre comillas. Considera que la barra oblicua o slash "/" debe tener la parte superior orientada hacia la derecha, de lo contrario la función no leerá la dirección que pusiste entre paréntesis.

```
#Ingresa el directorio que desees.
setwd("D:/aquispe/Book R")
```

Para corroborar que la dirección ha cambiado, podemos utilizar nuevamente la función `getwd()` a fin de obtener la dirección actual de trabajo

```
getwd()
## [1] "D:/aquispe/Book R"
```

◊ *¿Cuántos archivos tenemos en nuestro directorio actual?*

Podemos usar la función `list.files()` para inspeccionar los nombres de los archivos dentro del directorio. Antes de la función colocamos la palabra `length` (*longitud* en inglés), R nos muestra el total de archivos que tenemos.

```
length(list.files())
## [1] 167
```

Si deseamos visualizar los primeros seis nombres de archivo, antes de la función `list.files()` colocamos `head`, la cual de forma predeterminada nos brinda los seis primeros archivos.

```
head(list.files())
## [1] "~$-cap2_new version.docx" "~$-cap3_.docx"
## [3] "~$-cap4._docx.docx"       "~$-cap6_.docx"
## [5] "~$EC_P03_.docx"           "~$EC_P04_.docx"
```

Importación de datos

Una vez establecido el directorio de trabajo, el siguiente paso para realizar cualquier tipo de análisis es importar los datos que vamos a analizar. Para descargar todas las bases de datos que se utilizarán en este primer capítulo ingresar a Base de datos Cap1

Importar desde SPSS

El Paquete Estadístico para las Ciencias Sociales, conocido por sus siglas en inglés: SPSS, es un sofware estadístico que brinda herramientas que permiten consultar datos y formular hipótesis. Para importar este tipo de archivo se tienen que instalar, en primer lugar, los paquetes `haven` o `foreign`. El primer paquete contiene la función `read_sav()` que se encarga de leer archivos SPSS. Mientras que el paquete `foreign` contiene la función `read.spss`, como ambos leen archivos SPSS, optamos por uno solo; en este caso, el primero. Instalamos y cargamos el paquete.

```
#install.packages ("haven")
library(haven)
```

Una vez cargado el paquete, utilizamos la función `read_sav()`. Dentro del paréntesis de la función introduciremos el nombre y la extensión del archivo que queremos cargar, todo ello entre comillas, y lo guardaremos en el objeto denominado **dataSpss** (puedes ponerle el nombre que desees).

```
dataSPSS <- read_sav("LAPOP2014.sav")
```

Importar desde Stata

Al igual que SPSS, Stata es un sofware estadístico que permite analizar, gestionar y graficar datos. Para importar archivos de Stata (de extensión dta), vamos a utilizar la función `read.dta()` del paquete `haven`. Dentro del paréntesis del comando introducimos el nombre y la extensión del archivo entre comillas y lo guardamos en el objeto **dataStata**.

```
dataStata <-read_dta("Peru_LAPOP2017.dta")
```

Importar archivo CSV

Un CSV es un archivo de valores separados por comas (eso significan sus siglas en inglés: *comma separated values*). Para cargar este tipo de archivo vamos a utilizar la función `read.csv()` del paquete `readr` antes instalado y procederemos de igual forma que en los casos anteriores.

```
dataCSV<-read.csv("TB_ATEN_COVID19.csv")
```

Importar desde Excel

Para importar un archivo con extensión .xlsx o .xls, primero instalamos y cargamos el paquete `readxl`.

```
#install.packages("readxl")
library(readxl) #Lee archivos excel
```

Utilizamos la función `read_xlsx` o `read_xls`, dependiendo el tipo de extensión de nuestro archivo. En este caso leeremos un archivo de extensión xlsx.

```
dataXLSX<-read_xlsx("SALUD_ATENCIONES.xlsx")
```

Importar desde un sitio web

Si tenemos la url de una base de datos alojada en la red, podemos utilizar cualquiera de los comandos arriba señalados, siempre según la extensión del archivo que queramos descargar. Veamos una base de datos de un archivo Stata ubicado en los servidores de la Universidad de California:

```
datos<-read_dta("http://stats.idre.ucla.edu/stat/stata/dae/
binary.dta")
```

Beautiful coding: recomendaciones prácticas sobre cómo redactar *scripts* en R

Mantén un registro de su código

Comienza tu *script* con una breve descripción de lo que hace cuando se ejecuta, de esta forma sabrás en un futuro el propósito que tuvo cada *script* que hayas guardado. Por ejemplo:

```
#Este código grafica la distribución de la variable "x"
```

Cargar al inicio todos los paquetes a utilizar

Es importante que cargues los paquetes que vas a necesitar en la parte inicial de tu *script*, ya que puede ser tedioso tener que recorrer línea por línea para ubicar los paquetes que se carguen desordenadamente. Además, de esta forma es más sencillo saber si un paquete no se instaló.

Códigos cortos y segregados

Haz un esfuerzo por limitar tus *scripts*. Si son demasiado largos, considera buscar formas de dividirlos en partes más pequeñas. Además, es una buena práctica ir intercalando breves anotaciones entre las líneas de comandos del *script* para separarlas en secciones y facilitar la búsqueda de partes específicas de código cuando regreses sobre el *script* para mejorarlo o replicarlo.

Resumen del capítulo

Como hemos visto, R es un lenguaje en el que escribes y ejecutas comandos para que el programa los lea y muestre el resultado. Los componentes más importantes del programa R son los objetos y las funciones, que almacenan y manipulan datos respectivamente. Los objetos pueden contener datos de diferentes tipos y es importante reconocerlos para poder trabajar con ellos.

R utiliza también una serie de operadores que hacen fácil realizar tareas básicas. Estos operadores pueden ser útiles para hacer cálculos matemáticos o mecanizar tareas. Conocer cómo instalar y cargar paquetes es, de igual forma, un proceso indispensable. Las funciones que contienen los paquetes permiten que los comandos sean ejecutados y brinden los resultados esperados.

Para iniciar un análisis estadístico en R, es importante conocer, en primer lugar, nuestro directorio de trabajo pues en él se encuentran los archivos con los que vamos a trabajar. Una vez establecido nuestro directorio, importamos los archivos que nos interesa analizar. De acuerdo a su extensión, estos archivos se importan utilizando determinadas funciones que permitan llamar a los datos.

Finalmente, describir tu *script* con comentarios, mencionar los paquetes necesarios para su funcionamiento y limitar lo más posible la extensión de tus *scripts* son buenas prácticas que todo usuario de R debe considerar.

CAPÍTULO 2

MANEJO DE DATOS EN R

Objetivos de aprendizaje

Después de finalizar este capítulo, deberías ser capaz de:

- Realizar un diagnóstico general de tu base de datos.
- Identificar y remover valores perdidos.
- Manipular datos.
- Exportar bases de datos desde R.

Paquetes

Instalación

```
#install.packages("dplyr")
#install.packages("reshape")
#install.packages("readr")
#install.packages("tidyverse")
#install.packages("lubridate")
#install.packages("aweek")
#install.packages("openxlsx")
```

Carga

```
library(dplyr) #Manipular datos
library(reshape)#Renombrar
library(readr) #Leer archivos
library(tidyverse) #Manipular la data
library(lubridate) #Manipular las fechas
```

```
library(aweek) #Manipular las fechas
library(openxlsx) #Exportar archivo
```

Construcción de un marco de datos

Podemos construir un conjunto de datos simple de dos formas: creando variables y asignándoles los datos para, finalmente, unirlas en un marco de datos o *data frame*, o utilizando la función `tribble` del paquete `dplyr`.

Primera opción

La función `c()` nos permite concatenar múltiples datos. Haremos eso y los guardaremos en los objetos **Nombre** y **Peso**. Estos objetos son nuestras variables, sin embargo para unirlas en un marco de datos debemos utilizar la función `data.frame`. Veamos el proceso a continuación:

```
#primer conjunto de datos
Nombre <- c("Carlos","Cecilia", "Juan", "Enrique","Gimena",
"Jorge")
Peso <- c(80,55,60,93,65,48)

## unir las variables para crear el marco de datos
data1 <- data.frame(Nombre, Peso)
data1
##    Nombre Peso
## 1  Carlos   80
## 2 Cecilia   55
## 3    Juan   60
## 4 Enrique   93
## 5  Gimena   65
## 6   Jorge   48
#segundo conjunto de datos
Nombre <- c("Carlos","Cecilia", "Juan", "Enrique","Gimena",
```

```
"Bruno")
Altura <- c(1.78,NA,1.62,1.85, 1.60, 1.55)

data2 <- data.frame(Nombre, Altura)
data2
##    Nombre Altura
## 1  Carlos   1.78
## 2 Cecilia     NA
## 3    Juan   1.62
## 4 Enrique   1.85
## 5  Gimena   1.60
## 6   Bruno   1.55
```

Como puedes ver, en nuestro segundo conjunto de datos, el segundo valor de la variable "Altura" es NA. Este valor se incluye en las bases de datos cuando nos falta un valor para una instancia determinada (representa las siglas en inglés de "not applicable, "not availabe" o "no answer"). Más adelante veremos cómo identificar estos datos incompletos y removerlos fácilmente.

Segunda opción

Con la función `tribble` del paquete `dplyr` creamos de otra manera nuestro marco de datos. Este paquete tiene muchas funcionalidades para el manejo de datos que veremos más adelante, por el momento usaremos la función `tribble` que nos permite unir datos de forma más eficiente. Dentro de la función se coloca el nombre de las variables después del símbolo `~` y le asignamos los datos. Guardamos el resultado en objetos del mismo nombre que los que usamos en el ejemplo anterior.

```
data1 <- tribble(
  ~Nombre, ~Peso,
   "Carlos", 80,
   "Cecilia", 55,
   "Juan", 60,
   "Enrique", 93,
```

```
   "Gimena", 65,
   "Jorge", 48)
data2 <- tribble(
  ~Nombre, ~Altura,
   "Carlos", 1.78,
   "Cecilia", NA,
   "Juan", 1.62,
   "Enrique", 1.85,
   "Gimena", 1.60,
   "Bruno", 1.55)
```

Diagnóstico de la base de datos

Dimensión de nuestra base de datos (columnas y filas)

Conocer el número de columnas y filas es importante para trabajar de forma más rápida y eficiente en R. Además de conocer mejor la base de datos con la que trabajas, algunas tareas en R requieren conocer el número de columnas y filas para obtener determinados resultados. Para ello podemos utilizar las siguientes funciones: `ncol`, `nrow` y `dim`.

```
# Número de columnas
ncol(data1)
## [1] 2
#Número de filas u observaciones
nrow(data1)
## [1] 6
#Dimension del data frame (número de filas y columnas)
dim(data1)
## [1] 6 2
```

Estructura y tipo de datos

Para comprender mejor el tipo de datos que tenemos de una variable o un conjunto de datos, utilizamos la función `class`.

```
# Estructura de una base de datos
class(data1)
## [1] "tbl_df"     "tbl"        "data.frame"
```

Observamos que la base de datos "data1" es un marco de datos o *data frame*. Si deseamos conocer el tipo de datos de la variable "Nombre", utilizamos el símbolo del dólar del siguiente modo:

```
# Tipo de datos de una variable
class(data1$Nombre)
## [1] "character"
```

Notamos que la variable "Nombre" alberga datos de caracter.

Si queremos conocer la estructura interna de todas las variables de una base de datos, utilizamos la función `str`.

```
str(data1)
## tibble [6 x 2] (S3: tbl_df/tbl/data.frame)
##  $ Nombre: chr [1:6] "Carlos" "Cecilia" "Juan" "Enrique" ...
##  $ Peso  : num [1:6] 80 55 60 93 65 48
```

Nos muestra que la base de datos es un *data frame*, que la variable "Nombre" contiene datos de caracter y la variable "Peso" contiene datos numéricos.

Otra forma de visualizar la estructura de nuestras variables es utilizando la función `glimpse`.

```
glimpse (data1)
## Rows: 6
## Columns: 2
```

```
## $ Nombre <chr> “Carlos”, “Cecilia”, “Juan”, “Enrique”,
“Gimena”, “Jorge”
## $ Peso   <dbl> 80, 55, 60, 93, 65, 48
```

Lectura instantánea de datos

◊ *Observación de las primeras filas*

La función `head` facilita la observación de las primeras filas de un conjunto de datos. De forma predeterminada, muestra las primeras 6 filas, sin embargo, si deseamos ver, por ejemplo, las primeras 2 filas, podemos especificarlo en el argumento de la función.

```
# Nos muestra las primeras 6 filas
head(data1)
## # A tibble: 6 x 2
##   Nombre   Peso
##   <chr>   <dbl>
## 1 Carlos     80
## 2 Cecilia    55
## 3 Juan       60
## 4 Enrique    93
## 5 Gimena     65
## 6 Jorge      48
# Nos muestra las primeras 2 filas
head(data1, n = 2)
## # A tibble: 2 x 2
##   Nombre   Peso
##   <chr>   <dbl>
## 1 Carlos     80
## 2 Cecilia    55
```

◊ *Observación de las últimas filas*

Si queremos observar, esta vez, las últimas 2 filas, utilizamos la función `tail`.

```
#Nos muestras las últimas 2 filas
tail(data1, n = 2)
## # A tibble: 2 x 2
##   Nombre  Peso
##   <chr>  <dbl>
## 1 Gimena    65
## 2 Jorge     48
```

Manejo de datos perdidos

Diagnóstico

Es importante saber si nuestra base de datos tiene valores perdidos, porque la presencia de estos puede sesgar el análisis.

Para saber cuántos valores faltantes hay, utilizamos la función `complete.cases()` y dentro del paréntesis colocamos el nombre de nuestra base de datos. Usamos antes la función `table` para que nos muestre la frecuencia de los datos en forma de tabla. **FALSE** es igual al número de casos perdidos.

```
#base de datos "data1"
table (complete.cases(data1))
##
## TRUE
##    6
```

La base de datos "data1" no tiene ningún valor perdido. Veamos ahora la base de datos "data2".

```
#base de datos "data2"
table(complete.cases(data2))
```

```
##
## FALSE  TRUE
##     1     5
```

La base de datos "data2" tiene un valor perdido. Si queremos observar en qué fila y columna se encuentra dicho dato perdido, utilizamos corchetes y el signo de admiración, el cual permite obtener la respuesta negativa inversa de la función `complete.cases()` o, lo que es lo mismo, solo los valores incompletos.

```
data2[!complete.cases(data2), ]
## # A tibble: 1 x 2
##   Nombre  Altura
##   <chr>    <dbl>
## 1 Cecilia     NA
```

Notamos que el valor o dato faltante es la altura de Cecilia.

Limpieza de datos

Para remover los valores perdidos y quedarnos solo con los casos completos utilizamos la sintaxis básica de la función `complete.cases` y le pedimos que guarde los casos completos de la base de datos "data2" en la misma base de datos. Podemos guardarlos en un nuevo objeto, sin embargo, para evitar crear más objetos, lo guardaremos en el mismo lugar.

```
#Nos quedamos con los casos completos
data2 <-data2[complete.cases(data2), ]

table(complete.cases(data2)) #Comprobamos
##
## TRUE
##    5
```

La nueva base de datos "data2" se encuentra ahora sin datos perdidos. Para evitar eliminar de forma arbitraria los datos perdidos, se recomienda realizar previamente un análisis de sensibilidad. Este es un método que permite analizar un conjunto de datos a fin de evaluar si la alteración de algunas de las suposiciones realizadas lleva a diferentes interpretaciones o conclusiones finales. En otras palabras, se pregunta qué pasaría si los supuestos cambiaran. Realizar un análisis de este tipo puede dar solidez a nuestras conclusiones.

Manejo de datos

Ordenar datos

Utilizamos la función `arrange` para ordenar los datos tanto de forma ascendente como descendente, de acuerdo a una o más variables. Por defecto, `arrange` ordena las filas de forma ascedente; para cambiar a forma descendente agregamos el argumento `desc` antes de la variable a partir de la cual deseamos ordenar el contenido del marco de datos. Para guardarlo, asignamos este nuevo orden al mismo objeto.

```
data1 <-arrange(data1, Peso) #De manera ascendente
data1
## # A tibble: 6 x 2
##   Nombre   Peso
##   <chr>   <dbl>
## 1 Jorge      48
## 2 Cecilia    55
## 3 Juan       60
## 4 Gimena     65
## 5 Carlos     80
## 6 Enrique    93
data2 <- arrange(data2,desc(Altura)) #De manera descendente
data2
## # A tibble: 5 x 2
##   Nombre  Altura
##   <chr>    <dbl>
## 1 Enrique   1.85
```

```
## 2 Carlos    1.78
## 3 Juan      1.62
## 4 Gimena    1.6
## 5 Bruno     1.55
```

Renombrar variables

Para conocer, en primer lugar, los nombres de todas las variables de nuestra base de datos, utilizamos la función `names()` que pertenece al paquete base de R.

```
names(data1)
## [1] "Nombre" "Peso"
```

Si deseamos cambiar el nombre de una de las variables, utilizamos la misma función. Se coloca entre paréntesis el nombre de la base de datos en que se encuentra la variable cuyo nombre deseamos modificar; seguido de esto y entre corchetes se coloca el número de la columna donde se ubica la variable. En este caso, queremos cambiar a minúscula la variable "Nombre" de nuestra base de datos "data1", la cual se ubica en la primera columna. Le asignamos el nuevo nombre entre comillas.

```
names(data1)[1] = "nombre"
names(data1) #Comprobamos
## [1] "nombre" "Peso"
```

En caso tuviéramos una base de datos con muchas columnas y se nos dificultara conocer la ubicación exacta de la variable que queremos renombrar, podemos cambiar el nombre de la variable de otra manera. Con la misma función introducimos el nombre actual de la variable más el nuevo nombre con el cual deseamos reemplazarlo. Esta vez cambiaremos la misma variable pero de la base de datos "data2". El símbolo `==` se usa para pruebas de igualdad, pero también significa lo mismo que `=`, dependerá del contexto.

```
names (data2)[names(data2)== "Nombre"] <- "nombre"
```

```
names(data2)
## [1] “nombre” “Altura”
```

Con la función `rename` del paquete `reshape` podemos también cambiar de nombre a nuestras variables. Pondremos en minúscula las variables “Peso” y “Altura” de la base de datos “data 1” y “data2”

```
data1 <- rename(data1, c(Peso=”peso”))
data2 <- rename (data2, c(Altura=”altura”))
#Comprobamos
data1
## # A tibble: 6 x 2
##   nombre   peso
##   <chr>   <dbl>
## 1 Jorge      48
## 2 Cecilia    55
## 3 Juan       60
## 4 Gimena     65
## 5 Carlos     80
## 6 Enrique    93
data2
## # A tibble: 5 x 2
##   nombre  altura
##   <chr>    <dbl>
## 1 Enrique   1.85
## 2 Carlos    1.78
## 3 Juan      1.62
## 4 Gimena    1.6
## 5 Bruno     1.55
```

Unir datos

Para fusionar dos conjuntos de datos, utilizamos la función `merge` del paquete base de R. Se unen dos bases de datos mediante una o más variables comunes. Unimos

entonces las bases de datos “data1” y “data2” por la variable en común “nombre”. Y guardamos el resultado en un nuevo objeto denominado **data**.

```
data <- merge(data1, data2, by=”nombre”)
data
##    nombre peso altura
## 1  Carlos   80   1.78
## 2 Enrique   93   1.85
## 3  Gimena   65   1.60
## 4    Juan   60   1.62
```

Observamos que las variables “peso” y “altura” están en una sola base de datos unidas por la variable en común “nombre”.

Crear nuevas variables

Para agregar columnas a la base de datos usamos la función `mutate()` del paquete `dplyr`. En este caso, agregaremos una columna para el índice de masa corporal (“IMC”) a la base de datos **data**. Además, haremos uso del operador `%>%`. Si bien en este ejemplo no se nota una gran diferencia, veremos en los siguientes apartados que hacer uso de este operador simplifica nuestro código y nos facilita las tareas.

```
#sin %>%
data <-mutate(data, IMC=peso/(altura)^2)
#con %>%
data <- data %>% mutate(IMC=peso/(altura)^2)
data
##    nombre peso altura      IMC
## 1  Carlos   80   1.78 25.24934
## 2 Enrique   93   1.85 27.17312
## 3  Gimena   65   1.60 25.39062
## 4    Juan   60   1.62 22.86237
```

La nueva columna creada aparece por defecto al final de la base de datos, sin embargo, si deseamos agregarla antes de una columna en particular, agregamos el argumento `.before` En este caso deseamos que "IMC" esté ubicada antes de la columna "peso".

```
data %>% mutate(IMC= peso/(altura)^2,.before = peso)
##    nombre peso altura      IMC
## 1  Carlos   80   1.78 25.24934
## 2 Enrique   93   1.85 27.17312
## 3  Gimena   65   1.60 25.39062
## 4    Juan   60   1.62 22.86237
data
##    nombre        IMC peso altura
## 1  Carlos   25.24934   80   1.78
## 2 Enrique   27.17312   93   1.85
## 3  Gimena   25.39062   65   1.60
## 4    Juan   22.86237   60   1.62
```

De igual forma, si lo que queremos es añadir la nueva columna después de una determinada variable, en lugar de `.before` colocamos `.after`.

Convertir variables

Si queremos cambiar el tipo de datos de nuestras variables –como por ejemplo, convertir valores numéricos en valores de caracter o viceversa–, utilizamos `as.character` o `as.numeric`, según corresponda. Trabajaremos, en este caso, con la variable "peso"; la convertiremos en una variable de datos de caracter y, posteriormente, la convertiremos de nuevo en una variable de datos numéricos.

◊ *Conversión a datos categóricos*

```
data$peso <- as.character (data$peso)
str(data$peso) #Comprobamos
##  chr [1:4] "80" "93" "65" "60"
```

Vemos que los valores de peso se convirtieron en datos de caracter, razón por la cual aparecen entre comillas.

◊ *Conversión a datos numéricos*

```
data$peso <- as.numeric (data$peso)
str(data$peso) #Comprobamos
##  num [1:4] 80 93 65 60
```

Manejo de fechas

Para aprender a trabajar con fechas, en primer lugar, creamos una nueva base de datos con el año, mes y día de nacimiento de las personas de nuestra base de datos anterior. Para manejar fechas de forma sencilla, trabajaremos con la función `make_date`. Lo hacemos de la siguiente manera:

```
fecha <- tribble(
  ~nombre, ~anio, ~mes, ~dia,
  "Carlos", "1997", 2,  23,
"Enrique", "1967", 7, 1,
 "Gimena", "2000", 5, 11,
   "Juan", "1988", 11, 30)
```

Ahora procedemos a crear la variable "fecha_nac" en la base de datos "fecha". Con la función `make_date` del paquete `lubridate` nos aseguramos de que las variables que unimos ("anio", "mes" y "día") estén en formato de fecha. Para crear fechas y horas, utilizamos `make_datetime`.

```
fecha <- fecha %>% mutate(fecha_nac= make_date(anio, mes, dia))

fecha
## # A tibble: 4 x 5
##   nombre  anio    mes   dia fecha_nac
```

```
##   <chr>   <chr> <dbl> <dbl> <date>
## 1 Carlos  1997      2    23 1997-02-23
## 2 Enrique 1967      7     1 1967-07-01
## 3 Gimena  2000      5    11 2000-05-11
## 4 Juan    1988     11    30 1988-11-30
str(fecha$fecha_nac) #Comprobamos que tenga el formato fecha
##  Date[1:4], format: "1997-02-23" "1967-07-01" "2000-05-11"
"1988-11-30"
```

Lo siguiente que haremos será unir ambas bases de datos y guardar el resultado en un nuevo objeto denominado **nueva_data**.

```
nueva_data <- merge(data, fecha, by="nombre")
view(nueva_data) #observamos la nueva base de datos
str(nueva_data)
## 'data.frame':    4 obs. of  8 variables:
##  $ nombre   : chr  "Carlos" "Enrique" "Gimena" "Juan"
##  $ peso     : num  80 93 65 60
##  $ altura   : num  1.78 1.85 1.6 1.62
##  $ IMC      : num  25.2 27.2 25.4 22.9
##  $ anio     : chr  "1997" "1967" "2000" "1988"
##  $ mes      : num  2 7 5 11
##  $ dia      : num  23 1 11 30
##  $ fecha_nac: Date, format: "1997-02-23" "1967-07-01" ...
```

Notamos que la variable "anio" tiene datos de caracter, así que los convertimos en datos numéricos.

```
nueva_data$anio <- as.numeric(nueva_data$anio)
str(nueva_data$anio) #Comprobamos
##  num [1:4] 1997 1967 2000 1988
```

Selección de variables

Para seleccionar solo las variables con las que deseamos trabajar, utilizamos la función `select()`. En este caso, queremos quedarnos solo con las variables "nombre", "IMC", "anio" y "fecha_nac".

```
nueva_data <- nueva_data %>% select(nombre, IMC, anio, fecha_nac)
nueva_data #Comprobamos
##    nombre      IMC anio  fecha_nac
## 1  Carlos 25.24934 1997 1997-02-23
## 2 Enrique 27.17312 1967 1967-07-01
## 3  Gimena 25.39062 2000 2000-05-11
## 4    Juan 22.86237 1988 1988-11-30
```

Filtrar un subconjunto de datos

Queremos reunir los datos de las personas que nacieron a partir de 1990. Para ello utilizamos la función `filter` del paquete `dplyr`. Guardamos el resultado en un nuevo objeto llamado **data_final**.

```
data_final <- nueva_data %>% filter(anio >=1990)
data_final
##   nombre      IMC anio  fecha_nac
## 1 Carlos 25.24934 1997 1997-02-23
## 2 Gimena 25.39062 2000 2000-05-11
```

Exportación de bases de datos

Vamos a exportar nuestro conjunto de datos de R a archivos CSV y Excel.

Exportar a CSV

```
write.csv(data_final, "data_final.csv")
```

Exportar a Excel

```
write.xlsx (data_final, "data_final.xlsx")
```

Ejercicio práctico

La mortalidad en el Perú en el contexto de la pandemia por la COVID-19

El Perú fue uno de los pocos países que aplicó medidas drásticas e inmediatas para contener la expansión de la COVID-19: apenas nueve días después de confirmarse el primer caso y sin que se registrara aún la primera muerte por el virus en el país. A pesar de que al inicio de la pandemia se siguieron las medidas más eficientes y recomendadas para frenar la expansión del virus, estas no consiguieron detener el crecimiento exponencial de casos y de fallecidos por el COVID-19 y el Perú llegó a liderar la lista de los países con la mayor tasa de mortalidad en el mundo.

Para este ejercicio, analizaremos la tasa de mortalidad en el Perú según departamento y semana epidemiológica durante el año 2021. En primer lugar, realizamos un diagnóstico de la base de datos. Luego, manipulamos nuestros datos (eliminamos datos perdidos, creamos nuevas variables, uniremos bases de datos, etc.) utilizando distintas funciones. Por último, exportaremos la base de datos ya limpia y tratada.

Consideramos dos bases de datos: **fallecidos_sinadef** y **población21**, las mismas que puedes obtener ingresando al siguiente enlace Bases de datos Cap2

Carga de archivos

◊ *Fallecidos 2021*

Se carga el archivo llamándolo desde nuestro directorio. Utilizamos la función `read_delim` del paquete `readr` para que R lea la base de datos. Esta función permite importar archivos que están separados por tabs, comas o puntos y comas, como es el caso de nuestra base de datos. La base de datos será guardada en **fallecidos_sinadef**.

```
fallecidos_sinadef <- read_delim("fallecidos_sinadef.csv",
    delim = ";", escape_double = FALSE, trim_ws = TRUE)
```

◊ Población 2021

```
poblacion21 <- read_csv("TB_POBLACION_INEI.csv")
```

Diagnóstico de datos

◊ Tipo de datos

```
# Base de datos
class(fallecidos_sinadef)
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
# Solo una variable
class(fallecidos_sinadef$FECHA)
## [1] "Date"
```

◊ Dimensión de nuestro data frame

```
# Número de columnas
ncol(fallecidos_sinadef)
## [1] 4
#Número de filas u observaciones
nrow(fallecidos_sinadef)
## [1] 830797
#Dimension del data frame (filas y columnas)
dim(fallecidos_sinadef)
## [1] 830797      4
```

◊ *Primeras filas de nuestra base de datos*

```
# Nos muestra los primeras 6 observaciones
head(fallecidos_sinadef)
## # A tibble: 6 x 4
##   `PAIS DOMICILIO` `DEPARTAMENTO DOMICILIO` FECHA        AÑO
##   <chr>            <chr>                    <date>     <dbl>
## 1 PERU             AREQUIPA                 2020-11-18  2020
## 2 PERU             LA LIBERTAD              2021-01-19  2021
## 3 PERU             <NA>                     2020-01-27  2020
## 4 PERU             <NA>                     2020-01-24  2020
## 5 PERU             CALLAO                   2021-01-30  2021
## 6 PERU             CUSCO                    2020-11-17  2020
# Nos muestra las primeras 15 observaciones
head(fallecidos_sinadef, n = 15)
## # A tibble: 15 x 4
##    `PAIS DOMICILIO` `DEPARTAMENTO DOMICILIO` FECHA        AÑO
##    <chr>            <chr>                    <date>     <dbl>
##  1 PERU             AREQUIPA                 2020-11-18  2020
##  2 PERU             LA LIBERTAD              2021-01-19  2021
##  3 PERU             <NA>                     2020-01-27  2020
##  4 PERU             <NA>                     2020-01-24  2020
##  5 PERU             CALLAO                   2021-01-30  2021
##  6 PERU             CUSCO                    2020-11-17  2020
##  7 PERU             HUANUCO                  2020-10-22  2020
##  8 PERU             LIMA                     2021-01-10  2021
##  9 PERU             LIMA                     2020-02-01  2020
## 10 PERU             PIURA                    2020-01-08  2020
## 11 PERU             LIMA                     2021-01-17  2021
## 12 VENEZUELA        EXTRANJERO               2020-11-04  2020
## 13 PERU             LIMA                     2020-11-25  2020
## 14 PERU             LORETO                   2021-01-24  2021
## 15 PERU             LORETO                   2020-01-03  2020
```

◊ *Estructura de las variables*

```
str(fallecidos_sinadef)
## spec_tbl_df [830,797 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.
frame)
##  $ PAIS DOMICILIO        : chr [1:830797] “PERU” “PERU” “PERU”
“PERU” ...
##  $ DEPARTAMENTO DOMICILIO: chr [1:830797] “AREQUIPA” “LA
LIBERTAD” NA NA ...
##  $ FECHA                 : Date[1:830797], format: “2020-11-
18” “2021-01-19” ...
##  $ AÑO                   : num [1:830797] 2020 2021 2020 2020
2021 ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   `PAIS DOMICILIO` = col_character(),
##   ..   `DEPARTAMENTO DOMICILIO` = col_character(),
##   ..   FECHA = col_date(format = “”),
##   ..   AÑO = col_double()
##   .. )
##  - attr(*, “problems”)=<externalptr>
glimpse(fallecidos_sinadef)
## Rows: 830,797
## Columns: 4
## $ `PAIS DOMICILIO`         <chr> “PERU”, “PERU”, “PERU”,
“PERU”, “PERU”, “PERU~
## $ `DEPARTAMENTO DOMICILIO` <chr> “AREQUIPA”, “LA LIBERTAD”,
NA, NA, “CALLAO”, ~
## $ FECHA                    <date> 2020-11-18, 2021-01-19,
2020-01-27, 2020-01-~
## $ AÑO                      <dbl> 2020, 2021, 2020, 2020, 2021,
2020, 2020, 202~
```

◊ *Nombre de nuestras variables*

```
names(fallecidos_sinadef)
## [1] "PAIS DOMICILIO"         "DEPARTAMENTO DOMICILIO" "FECHA"
## [4] "AÑO"
```

Manejo de datos

◊ *Datos perdidos*

Exploramos cuántos datos con alguna NA tenemos (recuerda que `FALSE` es igual al número de valores perdidos).

```
table(complete.cases(fallecidos_sinadef))
##
##  FALSE   TRUE
##  18681 812116
```

Observamos solo los datos con valores perdidos.

```
fallecidos_sinadef[!complete.cases(fallecidos_sinadef), ]
## # A tibble: 18,681 x 4
##    `PAIS DOMICILIO` `DEPARTAMENTO DOMICILIO` FECHA        AÑO
##    <chr>            <chr>                    <date>     <dbl>
##  1 PERU             <NA>                     2020-01-27  2020
##  2 PERU             <NA>                     2020-01-24  2020
##  3 PERU             <NA>                     2020-01-13  2020
##  4 <NA>             <NA>                     2020-01-04  2020
##  5 PERU             <NA>                     2020-01-30  2020
##  6 PERU             <NA>                     2020-01-08  2020
##  7 PERU             <NA>                     2020-01-01  2020
##  8 PERU             <NA>                     2020-01-24  2020
##  9 PERU             <NA>                     2020-01-28  2020
## 10 PERU             <NA>                     2020-01-31  2020
```

```
## # ... with 18,671 more rows
```

Guardamos solo los casos completos de la base de datos **fallecidos_sinadef** en la misma base de datos.

```
#Quedarnos con casos completos
fallecidos_sinadef<-fallecidos_sinadef[complete.cases(fallecidos_
sinadef), ]

#Comprobar que ya no haya casos perdidos
##Solo debe haber TRUE, ningún FALSE
table(complete.cases(fallecidos_sinadef))
##
##   TRUE
## 812116
```

◊ *Filtro*

Solo trabajaremos con los fallecidos en territorio peruano durante el año 2021. Para ello filtramos lo que no se utilizará en el análisis. Las variables que tienen datos de tipo caracter se colocan entre comillas dobles (“…”) o simples (‘…’), las numéricas como “año” no lo requieren.

```
# Eliminamos los fallecidos en el extranjero y sin registro
fallecidosfiltrado <- fallecidos_sinadef %>%
  filter(fallecidos_sinadef$`DEPARTAMENTO DOMICILIO`!=
“EXTRANJERO”)

fallecidosfiltrado <- fallecidosfiltrado %>%
  filter(fallecidosfiltrado$`DEPARTAMENTO DOMICILIO`!= “SIN
REGISTRO”)

#Eliminamos los fallecidos antes del 2021
fallecidosfiltrado <- fallecidosfiltrado %>%
```

```
  filter(AÑO == 2021)
```

◊ *Renombrar variables*

Cambiamos el nombre de la variable "DEPARTAMENTO DOMICILIO" por "DEPARTAMENTO"

```
names (fallecidosfiltrado)[2] = "DEPARTAMENTO"
names (fallecidosfiltrado)
## [1] "PAIS DOMICILIO" "DEPARTAMENTO"   "FECHA"          "AÑO"
```

◊ *Crear nuevas variables*

Crearemos variables de fecha: "dia" (día de la semana), "semana_epi" (semana epidemiológica) y "anio_epi" (año epidemiológico). Para esto, y por tratarse de varias funciones que usaremos a la vez, se destacará la importancia del operador `%>%`.

La función `mutate` nos permite crear las variables y el paquete `lubridate` brinda facilidades para trabajar con fechas. Su función `epiweek()` nos brindará las semanas epidemiológicas que comienzan el día domingo, de acuerdo a la versión de semana epidemiológica de EE.UU. que maneja el paquete. Trabajar con semanas epidemiológicas facilita la comparación de eventos epidemiológicos acaecidos en una fecha determinada, además de que facilita la comparación con otros países.

Estas variables se crearán tomando como referencia la variable "FECHA" de nuestra base de datos **fallecidosfiltrado**, es decir, la función convertirá los datos de la variable "FECHA" en días, semanas y año epidemiológico. Asimismo, usamos la función `filter` para que solo nos arroje las semanas del año 2021. Guardamos todo ello en el objeto **SEGUNDA_OLA**.

```
SEGUNDA_OLA <- fallecidosfiltrado %>%
  mutate(dia= lubridate::wday(FECHA),
         semana_epi =lubridate:: epiweek(FECHA),
         anio_epi = lubridate::epiyear(FECHA)) %>%
  filter(anio_epi == 2021)
```

◊ *Manipulación de la base de datos de población*

En la base de datos original tenemos a la población dividida por grupo etario y sexo; es por ello que solo seleccionamos las variables "Departamento" y "Cantidad". Deseamos agrupar a la población según el departamento, así que utilizamos la función `group_by` del paquete `dplyr`. Asimismo, para realizar el conteo le pedimos a R que sume las cantidades con la función `summarize` y que se cree la nueva variable "sumpoblacion" para que sea igual a la suma de esas cantidades. De esta forma se realizará el conteo de la población por departamento, que estará guardada en la nueva base de datos **poblaciondepartamento21**, tal como se muestra a continuación:

```
poblaciondepartamento21 <- poblacion21 %>%
  select(`Departamento`, Cantidad)%>%
  group_by(`Departamento`) %>%
  summarize(sumpoblacion =  sum(Cantidad))
```

◊ *Ordenar base de datos*

Utilizamos la función `arrange` para ordenar datos tanto de forma ascendente como descendente.

```
#De manera ascendente
arrange(poblaciondepartamento21,sumpoblacion)
## # A tibble: 25 x 2
##    Departamento  sumpoblacion
##    <chr>                <dbl>
##  1 MADRE DE DIOS       150181
##  2 MOQUEGUA            187941
##  3 TUMBES              248877
##  4 PASCO               312777
##  5 TACNA               358314
##  6 AMAZONAS            427041
##  7 APURIMAC            466936
```

```
##  8 HUANCAVELICA        509117
##  9 UCAYALI             518190
## 10 AYACUCHO            718634
## # ... with 15 more rows
#De manera descendente
arrange(poblaciondepartamento21,desc(sumpoblacion))
## # A tibble: 25 x 2
##    Departamento sumpoblacion
##    <chr>              <dbl>
##  1 LIMA             10458367
##  2 LA LIBERTAD       1956389
##  3 PIURA             1901896
##  4 CAJAMARCA         1543104
##  5 PUNO              1471405
##  6 JUNIN             1389850
##  7 AREQUIPA          1350676
##  8 CUSCO             1346373
##  9 LAMBAYEQUE        1300720
## 10 ANCASH            1171756
## # ... with 15 more rows
```

◊ *Unir dos bases de datos*

Nuestro análisis requiere que podamos ver el número de habitantes por departamento y el número de fallecidos en una misma base de datos, por ello utilizamos la función `merge`. En los dos primeros argumentos de este comando se coloca el nombre de las bases de datos que queremos unir y en los siguientes dos el criterio bajo el cual se unen. En este caso, la variable común a ambas bases de datos es el departamento. Sin embargo, como los nombres de las variables difieren por el uso de mayúsculas y minúsculas, especificamos las columnas utilizadas agregando el nombre con los argumentos `by.y` y `by.x`.

```
#Unir las bases de datos SEGUNDA_OLA y poblacióndepartamento21

SEGUNDA_OLA <- merge(SEGUNDA_OLA, poblaciondepartamento21,
      by.x = "DEPARTAMENTO", by.y = "Departamento", all.x = T)
```

◊ Calcular el número de muertos

Es necesario tener en cuenta que cada línea en la base de datos es una persona fallecida; en consecuencia, nos conviene agrupar a los fallecidos por semana. En la siguiente línea de código seleccionamos las variables de interés, agrupamos y realizamos el conteo de fallecidos. Para esto último usamos la función `count()` que permite contar los valores únicos de las variables que decidimos agrupar. Esta función es equivalente a `summarize(n=n())`. Los cambios se verán reflejados al visualizar la data **SEGUNDA_OLA**, en la cual se observará que se crea una nueva variable llamada "n" que contiene el conteo de fallecidos.

```
SEGUNDA_OLA <- SEGUNDA_OLA %>%
select(DEPARTAMENTO, semana_epi, sumpoblacion)%>%
group_by(DEPARTAMENTO, semana_epi, sumpoblacion) %>%
count()
```

◊ Calcular la mortalidad semanal

Para calcular la mortalidad multiplicamos cada dato de la variable "n" por 100 000 y lo dividimos por la población. Asimismo, es necesario contar con números enteros ya que estamos analizando el número de fallecidos, por eso usamos la función `as.integer`. El asterisco (`*`) y la barra oblicua (`/`) son operadores matemáticos que nos permiten multiplicar y dividir, respectivamente. El resultado de dicha operación se guarda en la nueva variable "Mortalidad", el `=` nos permite hacerlo.

```
SEGUNDA_OLA$Mortalidad=
  as.integer(SEGUNDA_OLA$n*100000/
               SEGUNDA_OLA$sumpoblacion)
```

◊ *Cambiar los datos a forma horizontal*

Para una visualización horizontal de los datos, primero seleccionamos las variables que nos son útiles y las guardamos en un nuevo objeto. Luego usamos la función `pivot_wider()` para transponer los datos. Queremos visualizar la mortalidad por semana epidemiológica, es decir, que las semanas epidemiológicas aparezcan como variables en la parte superior. Para ello, igualamos la variable "semana_epi" al argumento `names_from` y los valores de la mortalidad al argumento `values_from`.

```
SegundaOlaHorizonal <- SEGUNDA_OLA %>% select(1,2,3,5) %>%
pivot_wider(names_from=semana_epi, values_from = Mortalidad)
```

◊ *Exportar datos*

Exportamos los datos en formato Excel.

```
write.xlsx(SegundaOlaHorizonal,"segunda_olaH.xlsx" )
```

Resumen del capítulo

Un paso previo para realizar un análisis estadístico en R es diagnosticar, limpiar y manipular nuestra base de datos. El diagnóstico nos permite evaluar el tipo de datos que tenemos, conocer la estructura interna de la base de datos e identificar si hay valores faltantes en ella. Tener conocimiento de estas características de nuestros datos es esencial para manipularlos y tener una base de datos limpia y ordenada sobre la cual trabajar en R. Una vez que hemos modelado nuestros datos de una forma que nos sea útil para nuestro propósito, podemos exportarlos en diferentes formatos de archivo desde R.

El ejercicio práctico nos ha permitido seguir todos estos pasos, además de realizar cálculos que nos dieron como resultado la obtención de la mortalidad por semana epidemiológica durante el año 2021 a partir de dos bases de datos que fueron previamente limpiadas y manipuladas.

CAPÍTULO 3

VISUALIZACIÓN DE DATOS

En este capítulo aprenderás desde crear gráficos básicos hasta cómo exportarlos desde R. En primer lugar, comprenderás las funcionalidades del paquete `ggplot2`, el paquete para la visualización de datos por excelencia. Seguidamente aprenderás a realizar gráficos univariados a fin de conocer la distribución de una única variable, ya sea categórica o numérica. Aprenderás a construir gráficos bivariados y multivariados y te ejercitarás en el uso de funciones para personalizar tus gráficos de forma que queden listos para exportarlos desde R en el formato que desees.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Comprender las funcionalidades de `ggplot2`.
- Realizar gráficos univariados, bivariados y multivariados de variables categóricas y numéricas.
- Personalizar tus gráficos.
- Guardar y exportar gráficos en el formato de tu preferencia.

Paquetes

Instalación

```
#install.packages(ggplot2)
#install.packages(ggmosaic)
#install.packages(treemapify)
#install.packages(dplyr)
#install.packages(scales)
#install.packages(ggridges)
```

Carga

```
library(ggplot2) #gráficos
library(ggmosaic) #mosaicos
library(treemapify) #diagrama de árbol
library(dplyr) #función triblee
library(scales) #contiene "label=comma"
library(ggridges) #gráfico distribución
```

Introducción a ggplot2

`ggplot2` es un paquete utilizado para la visualización de datos, su función es `ggplot`. Si bien se pueden generar gráficos básicos, por ejemplo, con las funciones `plot`, `density`, `hist`, `boxplot` o `barplot`, `ggplot2` es la más versátil ya que permite generar gráficos de mejor calidad y visualmente más estéticos.

Veamos un ejemplo simple creando un marco de datos:

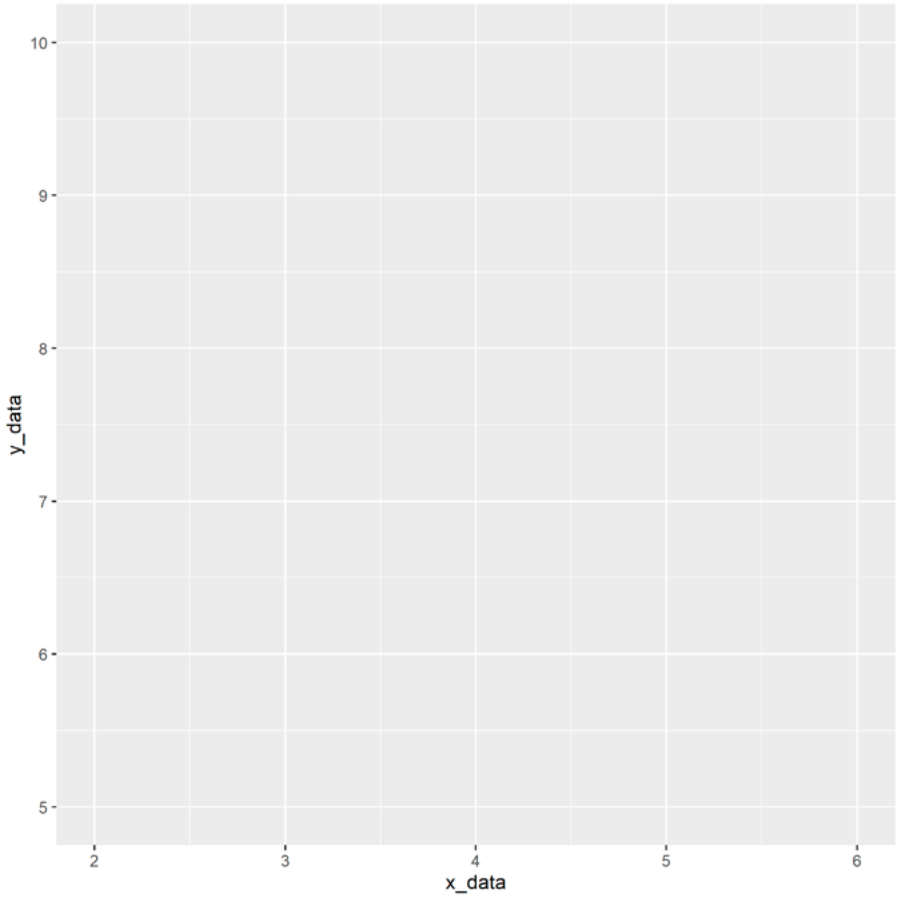
```
data <- tribble(
  ~x_data, ~y_data,~grupo,
   2, 5, "A",
   3, 7, "B",
   4, 9, "C",
   6, 10, "D")
```

En una pestaña separada de nuestro entorno, se visualiza de la siguiente manera:

	x_data	y_data	grupo
1	2	5	A
2	3	7	B
3	4	9	C
4	6	10	D

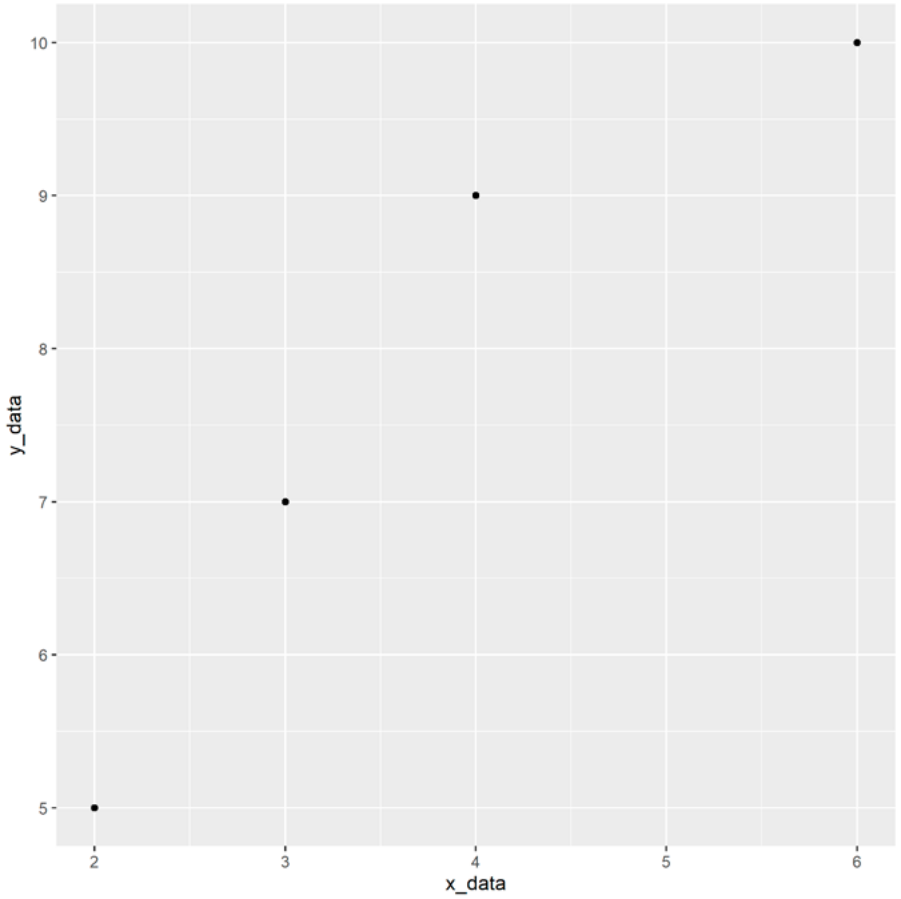
Creado el marco de datos, usamos la sintaxis básica de la función `ggplot` que utiliza el comando `aes` para generar una visualización estética de los datos que escojamos de nuestro marco de datos:

```
ggplot(data, aes(x=x_data, y=y_data))
```



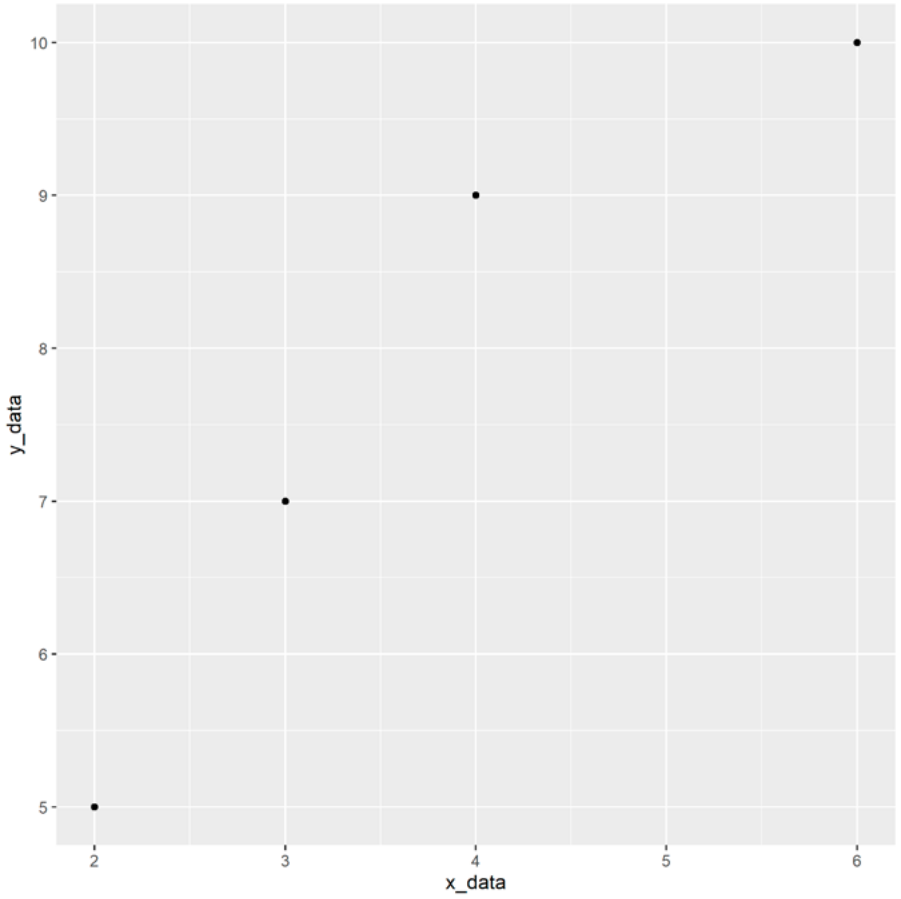
Se requiere, además, especificar el tipo de gráfico que queremos que se visualice. Agregaremos `geom_point` para que aparezcan puntos y dé como resultado un gráfico de dispersión.

```
ggplot(data, aes(x=x_data, y=y_data)) +
  geom_point()
```



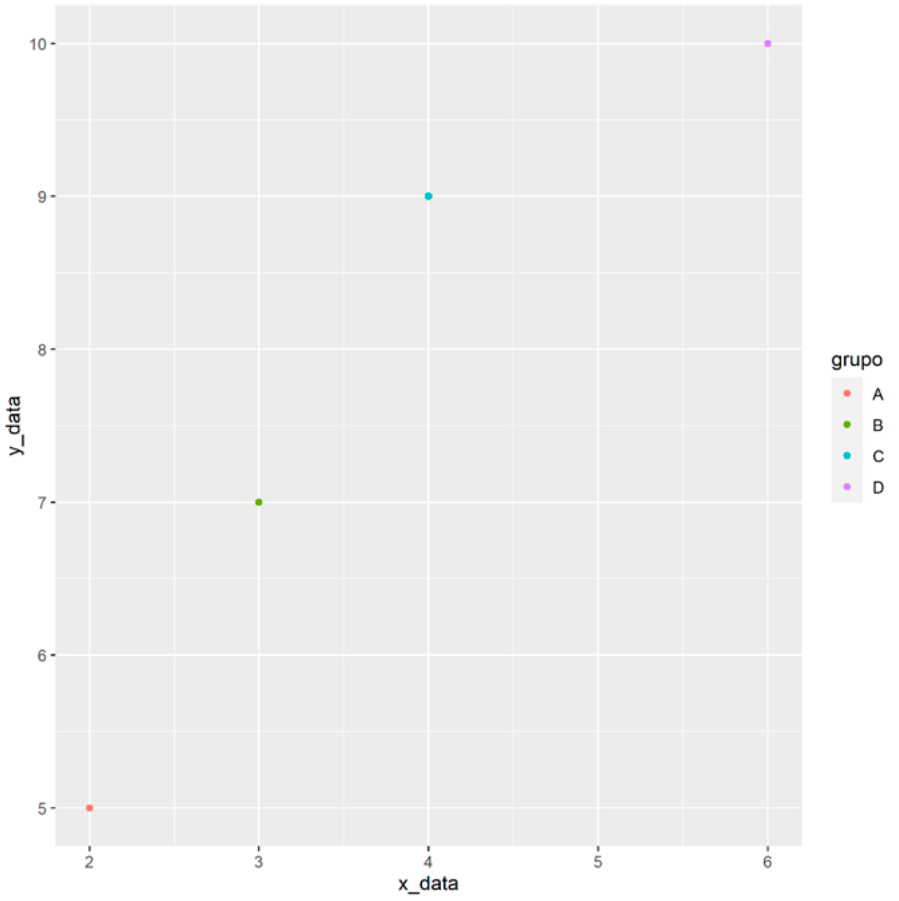
Para evitar tener que reescribir todo el código, lo guardamos en el objeto **m**.

```
m <- ggplot(data, aes(x=x_data, y=y_data))

m +  geom_point()
```



Vemos que reproduce el mismo resultado, pero con un comando más simple. Ahora, si deseamos incluir la variable "grupo" de nuestro marco de datos en el gráfico, podemos asignarle color a los puntos que hemos trazado. Para hacerlo, ingresamos el siguiente comando:

```
m + geom_point(aes(colour=grupo))
```



`ggplot2` nos permite manejar lo que se conoce como gramática de gráficos. Como se puede observar, nuestros gráficos están compuestos por varias capas, en las cuales se van agregando ejes, puntos, etc.

Estudio de caso

Usaremos el conjunto de datos `diamonds` que viene incorporado en el paquete `ggplot2`. Cada fila del conjunto de datos representa un diamante diferente. Hay 53,940 filas de datos y 10 variables.

Veamos el conjunto de datos `diamonds`

```
View(diamonds)
```

diamonds ×

Filter

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39

Hay 10 variables que ofrecen distintos tipos de información sobre los diamantes:

Variable	Descripción	Tipo de variable
price	Precio en dólares ($326-$18 823)	Cuantitativa
carat	Peso del diamante (0.2-5.0)	Cuantitativa
cut	Calidad del corte (Regular, bueno, muy bueno, premium, ideal)	Cualitativa
color	Color del diamante, de D (mejor) a J (peor)	Cualitativa
clarity	Medida de claridad del diamante (I1 (peor), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (mejor))	Cualitativa
x	Longitud en mm (0–10.74)	Cuantitativa
y	ancho en mm (0–58.9)	Cuantitativa
z	Profundidad en mm (0–31.8)	Cuantitativa
depth	porcentaje de profundidad total = z / media (x, y) = 2 * z / (x + y) (43–79)	Cuantitativa
table	Ancho de la parte superior del diamante en relación con el punto más ancho (43–95)	Cuantitativa

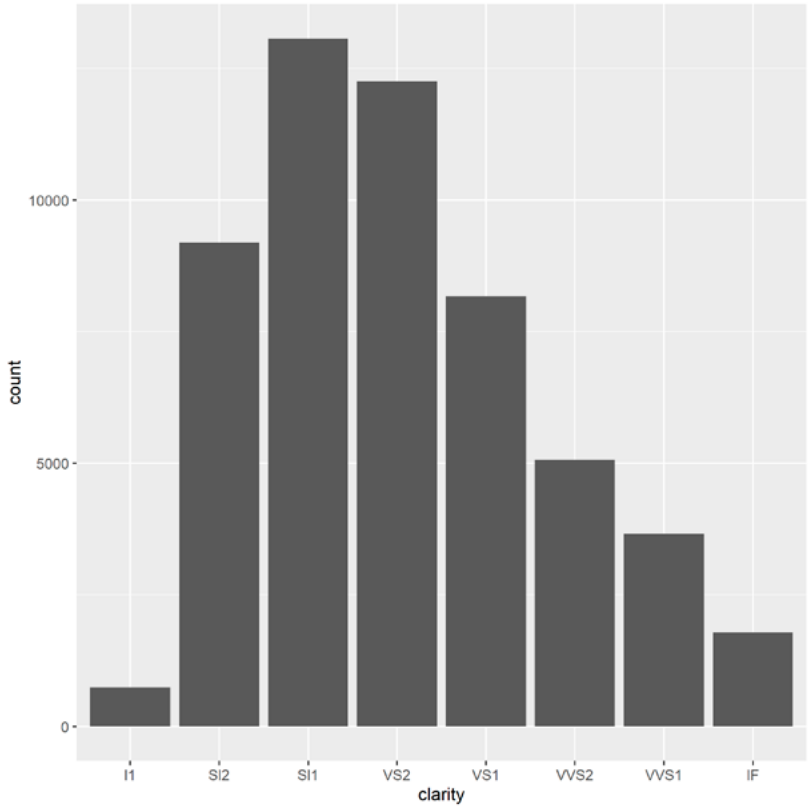
Gráficos univariados

Categórico

◊ *Gráfico de barras*

Generamos un gráfico de barras para ver la distribución de los diamantes según la medida de claridad de los mismos.

```
ggplot(diamonds, aes(x = clarity)) +
  geom_bar()
```

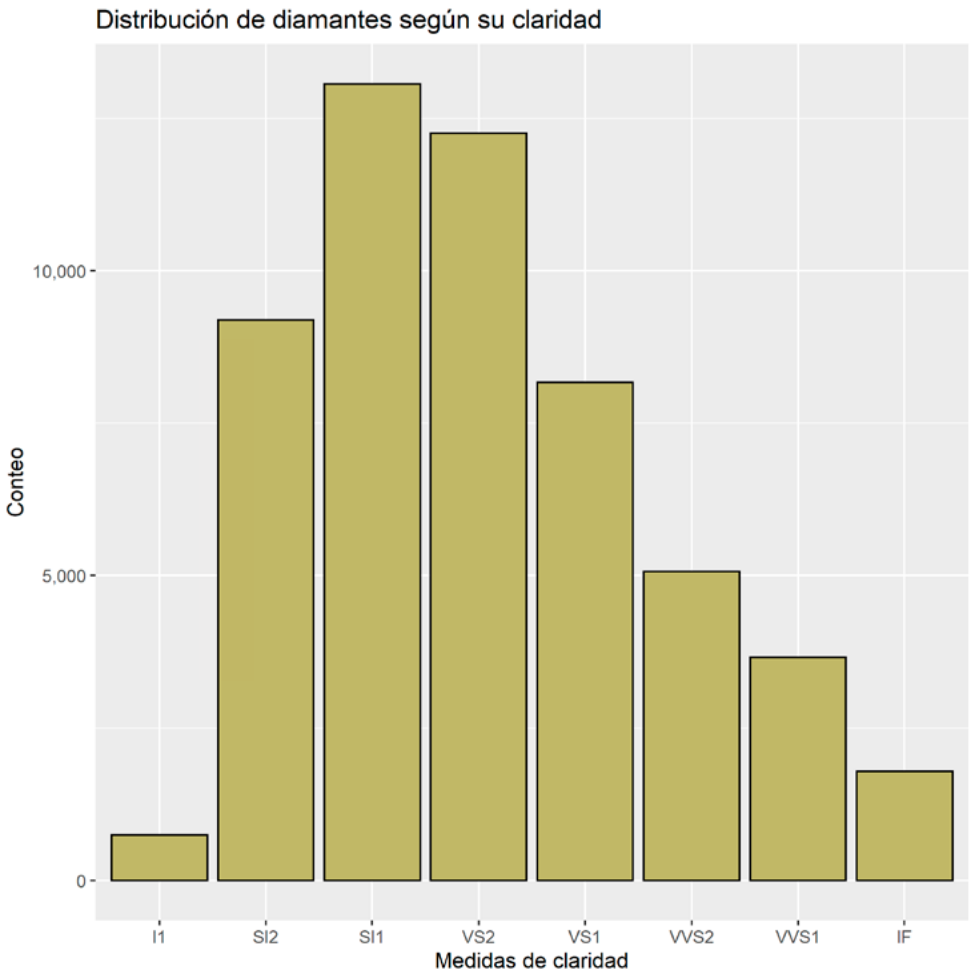


La mayoría de los diamantes tienen una claridad de SI1, seguidos por los que tienen una claridad de VS2 y hay muy pocos con claridad de I1 y IF; estos últimos representan los diamantes de peor y mejor claridad, respectivamente.

Color, etiquetas, título y comas

El gráfico anterior es útil, pero podemos presentar los datos de una manera más pulcra y explícita si agregamos color y bordes a las barras, títulos a los ejes y a todo el gráfico y comas a los valores del eje `y`. Todas estas opciones las podemos agregar con la función `geom_bar`. El argumento `fill=` establece el color del relleno de las barras del gráfico y `color=` el color del borde de las mismas. Con el argumento `labs` podemos asignarle un título a cada uno de los ejes del gráfico (en este caso, "Distribución de diamantes según su claridad" como título del gráfico, "Medidas de claridad" para el eje *x* y "Conteo" para el eje *y*). Además, para facilitar la lectura de las cantidades del eje *y*, podemos agregar comas para marcar los millares. Esto último lo hacemos con la función `scale_y_continuous` y el argumento `labels=` con el valor `comma`.

```
ggplot(diamonds, aes(x = clarity)) +
  geom_bar(fill= "darkkhaki", color= "black") +
    labs(x = "Medidas de claridad",
       y = "Conteo",
       title = "Distribución de diamantes según su claridad") +
  scale_y_continuous(labels = comma)
```

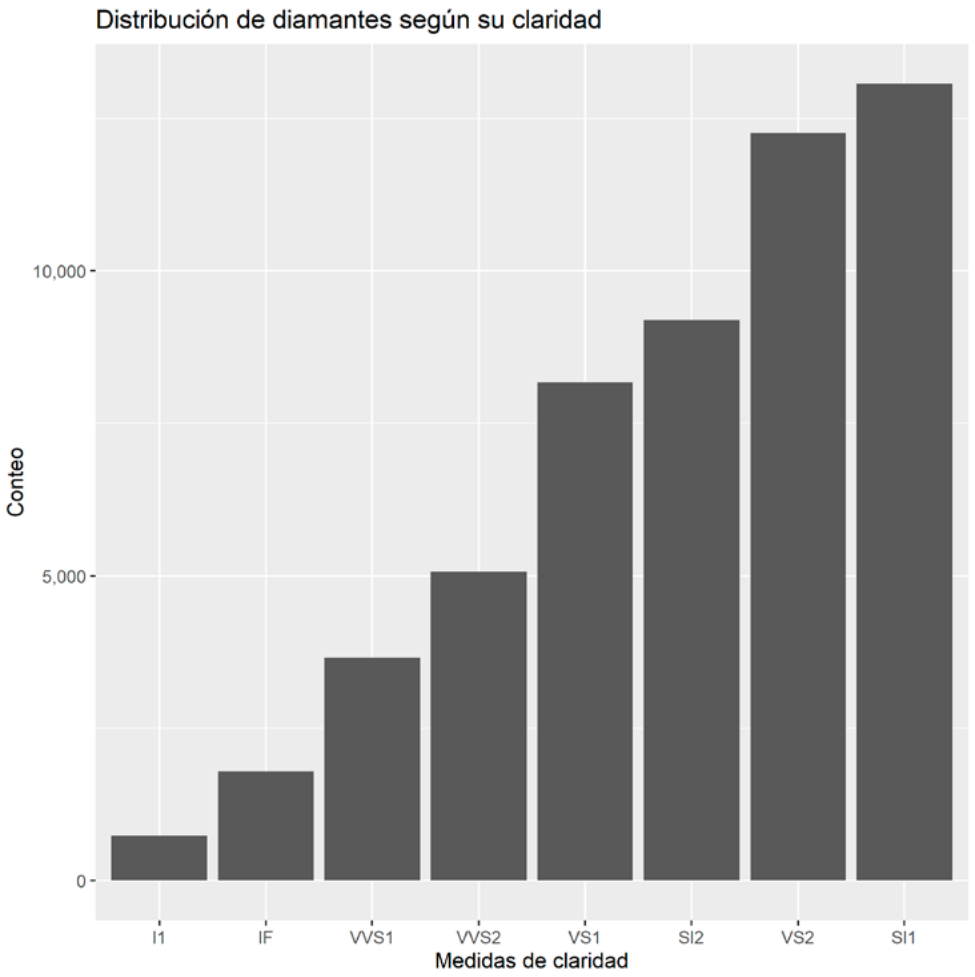

Distribución de diamantes según su claridad
Conteo
10,000
5,000
0
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
Medidas de claridad

Reordenar las barras

Podemos reordenar las barras de distintas maneras; a continuación mostramos cómo hacerlo de tres formas distintas:

Con la función `reorder` del paquete `dplyr` podemos hacer que R cambie el orden de los elementos de un marco de datos. En este caso, creamos un marco de datos que tenga como variables la claridad de los diamantes y la cantidad de diamantes por cada medida de claridad utilizando la función `count`. Luego asignamos ese marco de datos a un objeto (en este caso, **s**) y lo reordenamos con `reorder`:

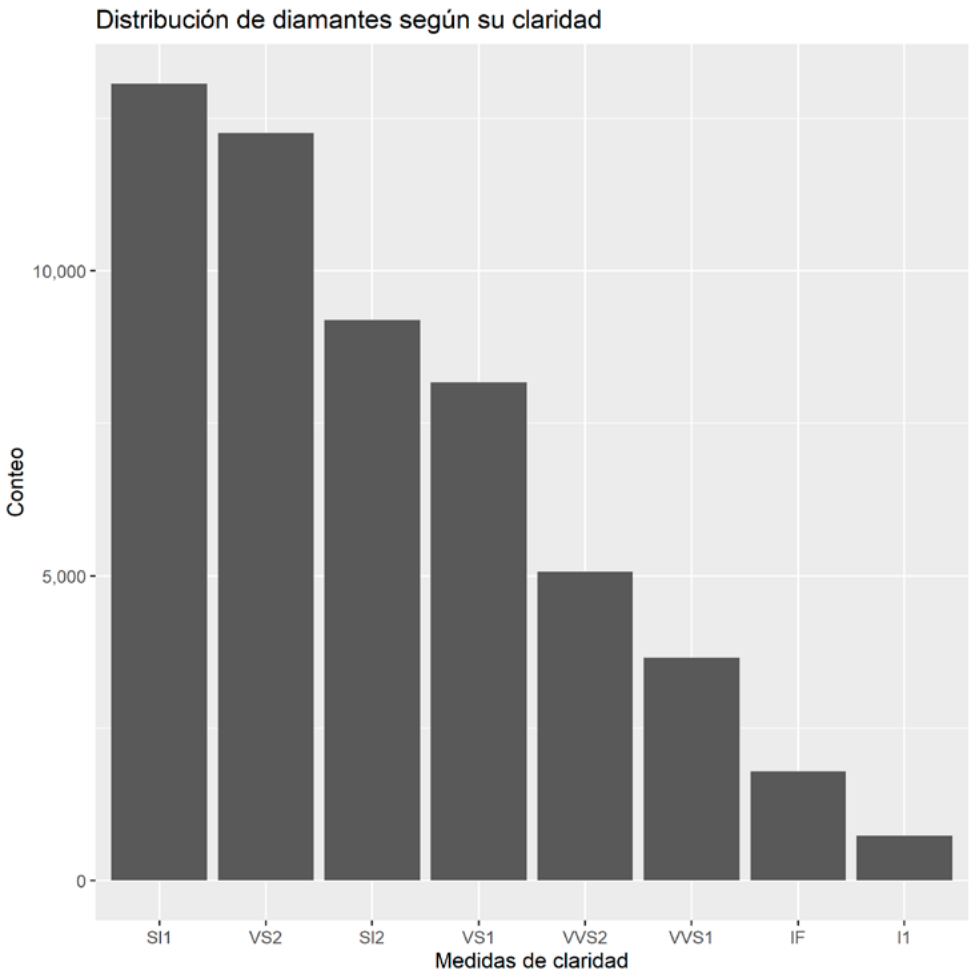
```
s <- diamonds %>% count(clarity)
s
## # A tibble: 8 x 2
##   clarity     n
##   <ord>   <int>
## 1 I1        741
## 2 SI2      9194
## 3 SI1     13065
## 4 VS2     12258
## 5 VS1      8171
## 6 VVS2     5066
## 7 VVS1     3655
## 8 IF       1790
ggplot(s, aes(x = reorder(clarity, n),
           y = n)) +
  geom_bar(stat = "identity") +
    labs(x = "Medidas de claridad",
       y = "Conteo",
       title = "Distribución de diamantes según su claridad") +
  scale_y_continuous(labels = comma)
```


Distribución de diamantes según su claridad
Conteo
10,000
5,000
0
I1
IF
VVS1
VVS2
VS1
SI2
VS2
SI1
Medidas de claridad

Las barras se reordenaron por defecto de forma ascendente. Para reordenarlas de forma descendente utilizamos `reorder(clarity, -n)`

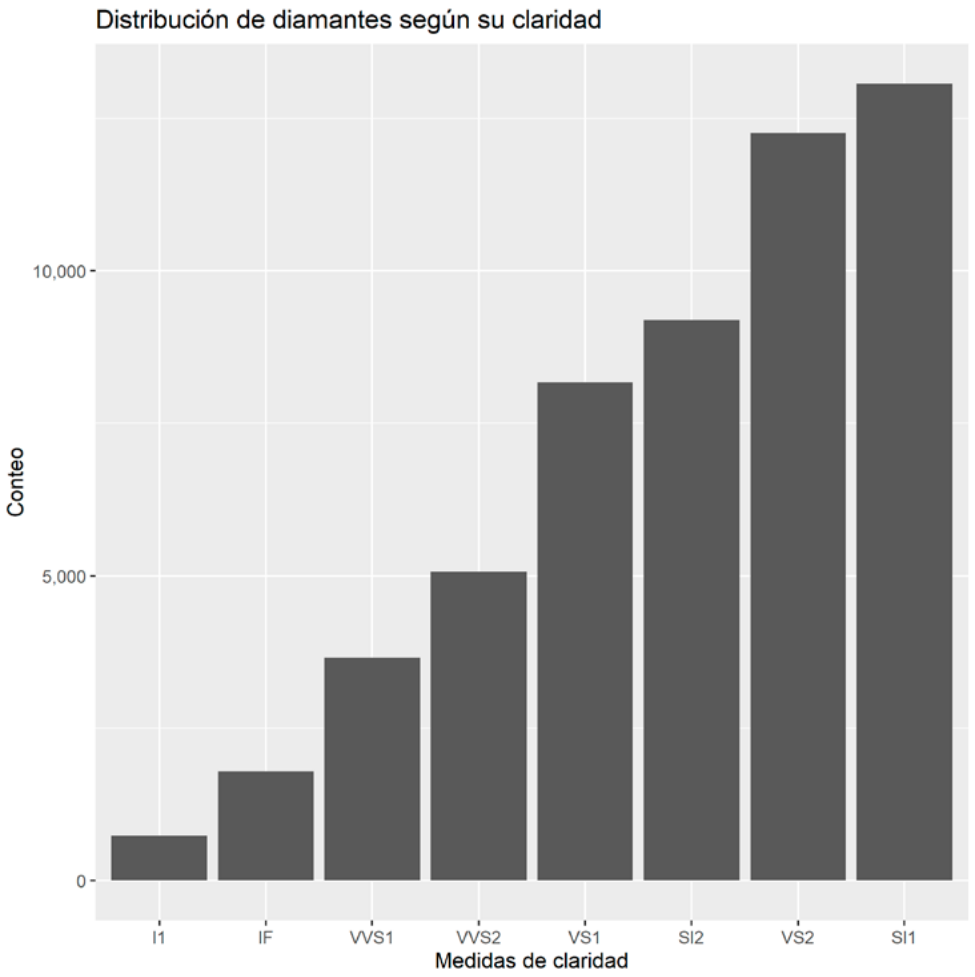
Otra opción para reordenar las barras es utilizando la función `levels`. Con esta función establecemos los niveles de la variable "clarity" y los ordenamos como queremos con la función `sort`, especificando el orden descendente con `decreasing`. Si deseamos que las barras se vean de forma ascendente lo cambiamos por `increasing`. Utilizamos la función `within` al inicio para que se calcule el resultado de la expresión dentro de sí misma. Guardamos en el objeto **p**.

```
#establecemos los niveles en el orden que queremos
p <- within(diamonds, clarity <- factor(clarity,
                  levels=names(sort(table(clarity),
                  decreasing=TRUE))))

#graficamos el objeto con ggplot
ggplot(p, aes(x = clarity)) +
  geom_bar() +
    labs(x = "Medidas de claridad",
       y = "Conteo",
       title = "Distribución de diamantes según su claridad") +
  scale_y_continuous(labels = comma)
```


Distribución de diamantes según su claridad
Conteo
10,000
5,000
0
SI1
VS2
SI2
VS1
VVS2
VVS1
IF
I1
Medidas de claridad

Por último, podemos reordenar los niveles asignándoles un orden manualmente con la función `levels`. La función `c()` nos permite vincular o encadenar estos niveles en el orden de nuestra preferencia.

```
#los regresamos al orden ascendente manualmente
diamonds$clarity <- factor(diamonds$clarity,
                           levels = c("I1", "IF", "VVS1", "VVS2",
"VS1", "SI2", "VS2","SI1"))


ggplot(diamonds, aes(x = clarity)) +
  geom_bar() +
    labs(x = "Medidas de claridad",
       y = "Conteo",
       title = "Distribución de diamantes según su claridad") +
  scale_y_continuous(labels = comma)
```


Distribución de diamantes según su claridad
Conteo
10,000
5,000
0
I1
IF
VVS1
VVS2
VS1
SI2
VS2
SI1
Medidas de claridad

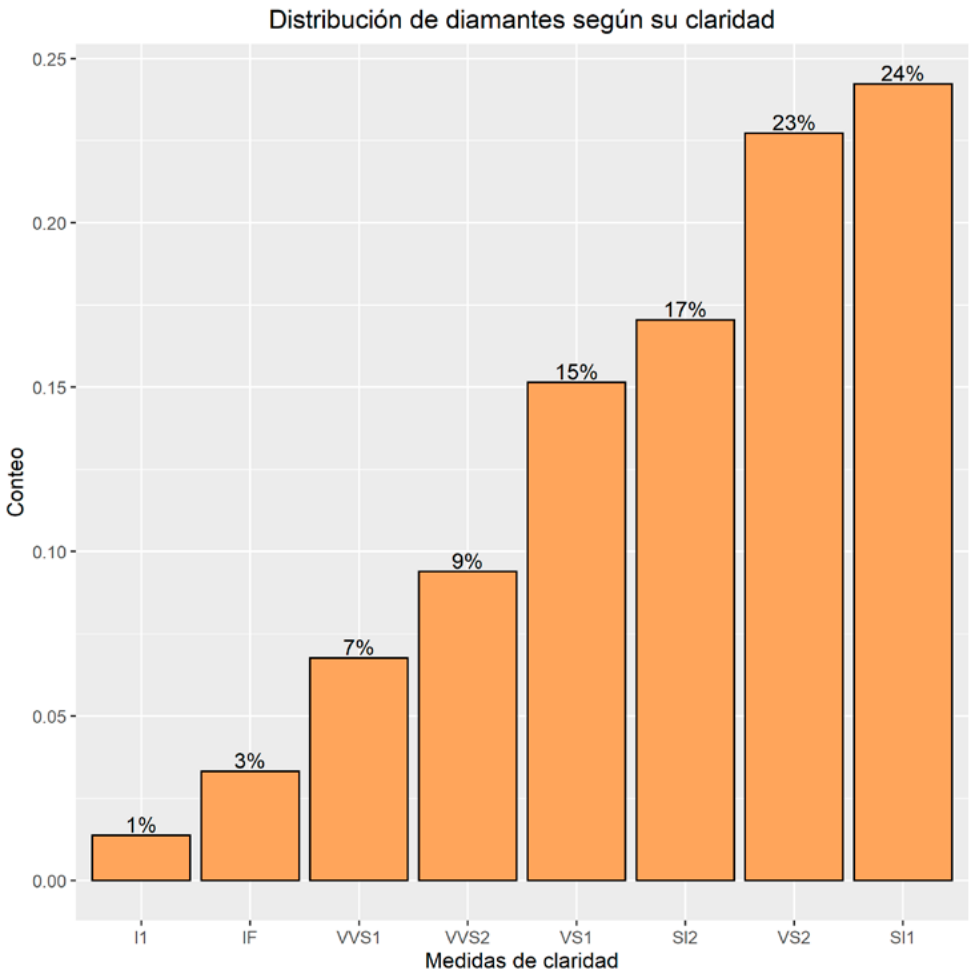
Mostrar valores en las barras

Con `geom_text` agregamos las etiquetas de los valores y con `vjust` y `hjust` (justificación vertical y horizontal, respectivamente) controlamos la posición de las etiquetas.

Primero, realizamos el conteo de cada una de las medidas de claridad y con `mutate` creamos las nuevas variables que contengan tanto la suma de los conteos (`count`) como el porcertaje redondeado (`countlabel`); los valores de `count` los multiplicamos por 100 y con `round` redondeamos el resultado. La función `paste0` nos permite concatenar dicho resultado con el símbolo "%".

Los argumentos `vjust` y `hjust` se usan para posicionar un título o etiqueta. Sus valores oscilan entre 0 y 1 donde 0 significa justificado a la izquiera y 1 a la derecha. En este ejemplo que se muestra a continuación, utilizamos los argumentos `vjust` para los valores de la variable "countlabel" y `hjust` para ajustar el título del gráfico. La función `theme` justamente nos permite personalizar la estética de nuestros gráficos sin cambiar las propiedades de los datos, es decir, se enfoca en el manejo de los títulos, etiquetas, fuentes, tamaño, fondo, cuadrículas y leyenda. El argumento `plot.title` hace referencia a la apariencia del título y `element.text` permite asignarle valores, en este caso, la justificación horizontal con `hjust` el cual contiene los argumentos `vjust` y `hjust` que nos permiten posicionar el título del gráfico.

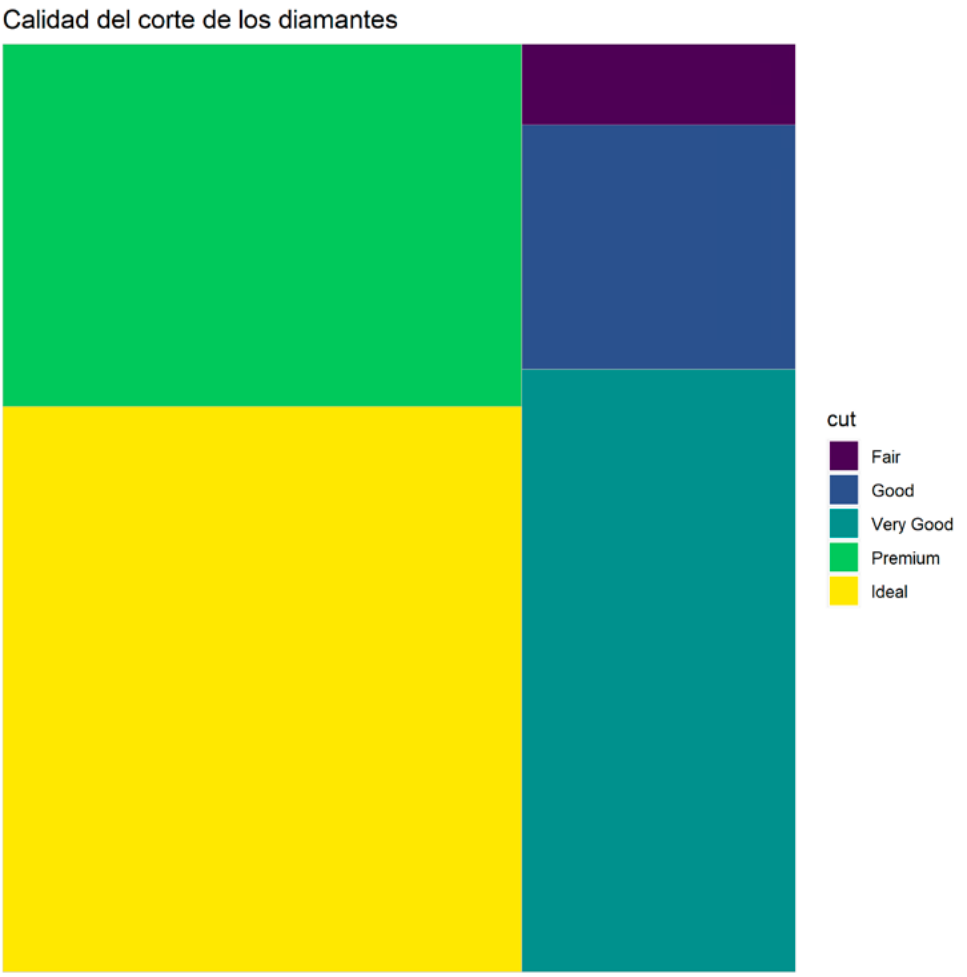
```
plot <- diamonds %>% count(clarity) %>%
  mutate(count=n/sum(n), countlabel= paste0(round(count*100),
"%"))


ggplot(plot, aes(x = clarity, y=count)) +
  geom_bar(stat="identity", fill= "sandybrown",
           color= "black") +
    labs(x = "Medidas de claridad",
       y = "Conteo",
       title = "Distribución de diamantes según su claridad") +
  scale_y_continuous(labels = comma) +
geom_text(aes(label=countlabel), vjust=-0.2)  +
  theme(plot.title = element_text(hjust = 0.5)) #centrar título
```


Distribución de diamantes según su claridad
Conteo
0.25
0.20
0.15
0.10
0.05
0.00
1%
3%
7%
9%
15%
17%
23%
24%
I1
IF
VVS1
VVS2
VS1
SI2
VS2
SI1
Medidas de claridad

◊ *Diagrama de árbol*

El *treemap* o diagrama de árbol ofrece otra forma de representar los datos de manera jerárquica: cada factor de una variable es representado por un rectángulo cuya área es proporcional a su valor en relación con la suma total de los valores de todos los factores de la variable que estamos graficando. En el siguiente ejemplo hacemos un diagrama de árbol de los diamantes de acuerdo con la calidad de su corte. A cada tipo de corte se le asigna un color y la cantidad de diamantes con ese tipo de corte determina el tamaño del rectángulo que lo representa.

```
graph_arbol <- diamonds %>% count(cut)

ggplot(graph_arbol,
       aes(fill = cut, area=n)) +
  geom_treemap() +
  labs(title = “Calidad del corte de los diamantes”)
```



Observamos que la mayoría de los diamantes tienen un corte ideal.

Agregamos etiquetas dentro del gráfico

En el argumento de `ggplot` añadimos `label` y con la función `geom_treemap_text` damos color y centramos las etiquetas. Borramos la leyenda para una mejor visualización.

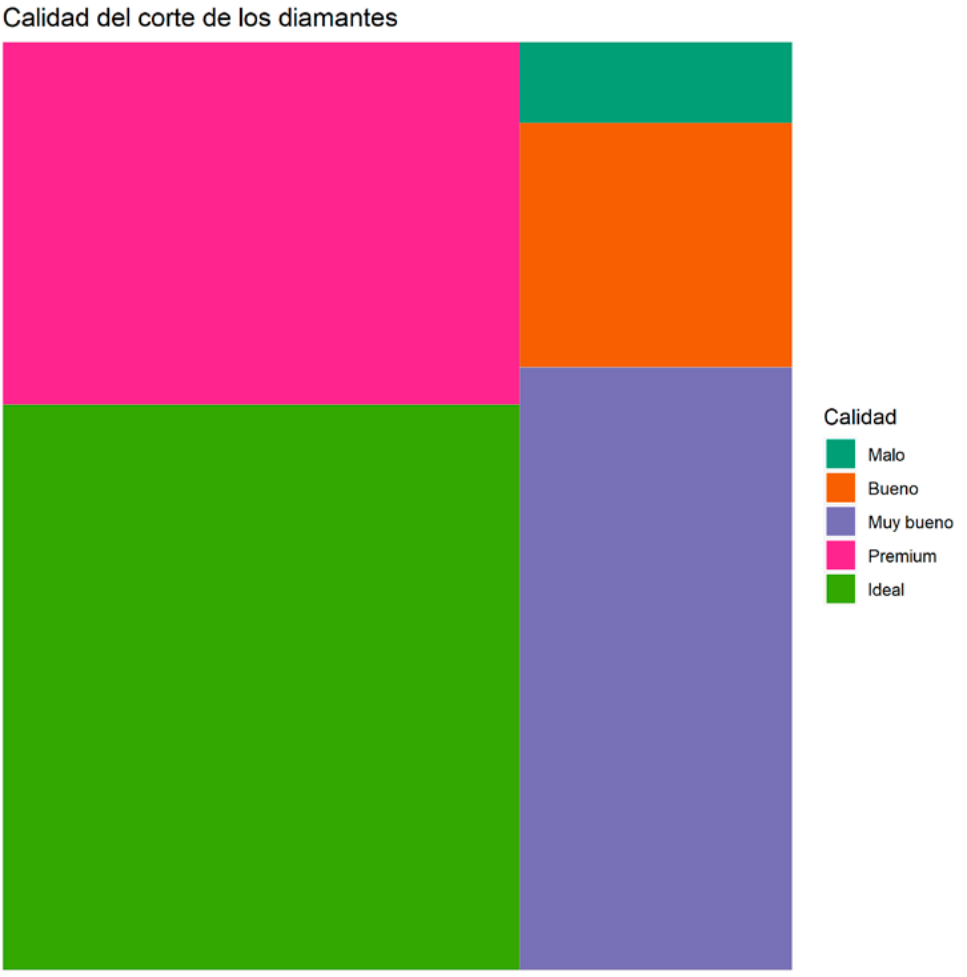
```
ggplot(graph_arbol,
       aes(fill = cut, area=n, label= cut)) +
  geom_treemap() + geom_treemap_text(colour="white",
                                     place = "centre") +
  labs(title = "Calidad del corte de los diamantes") +
  theme(legend.position = "none") #eliminanos la leyenda
```

Calidad del corte de los diamantes

Leyenda

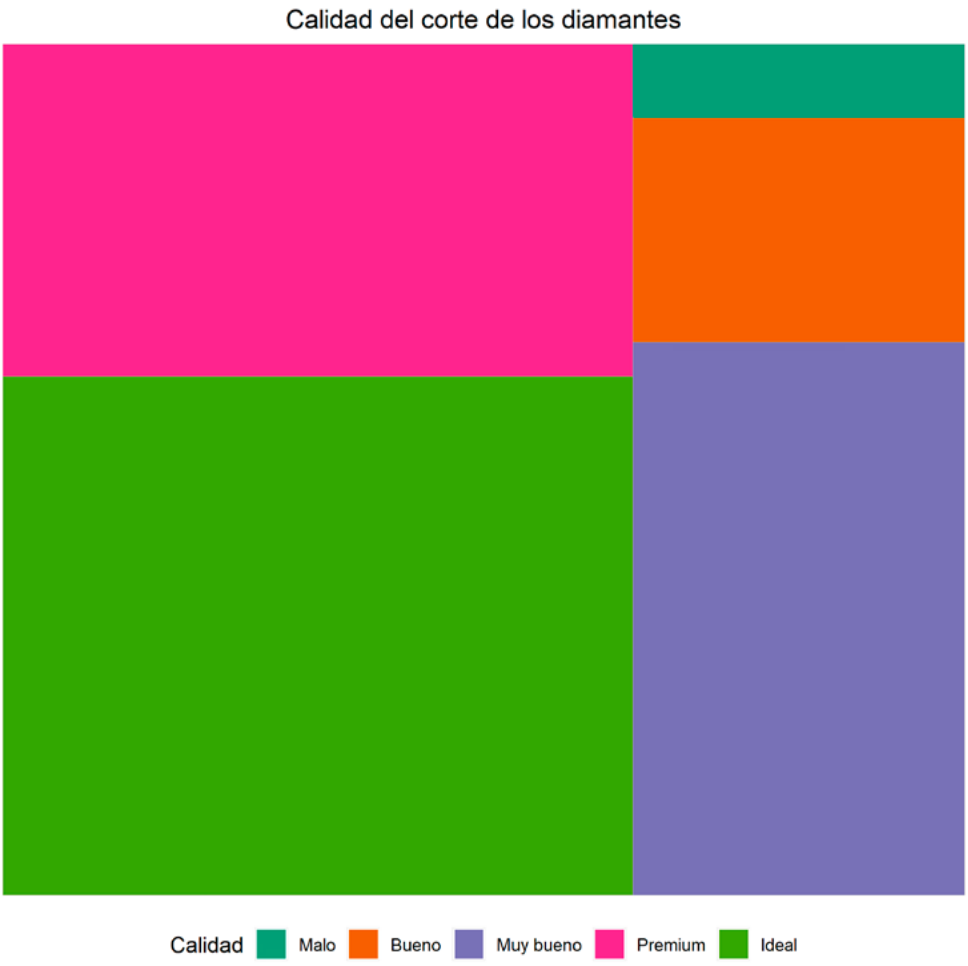
Para cambiar el título y las etiquetas usamos la función `scale_fill_brewer`. Esta función, además, nos permite cambiar la paleta de color de nuestro gráfico.

```
ggplot(graph_arbol,
       aes(fill = cut, area=n)) +
  geom_treemap() +
  labs(title = “Calidad del corte de los diamantes”,
       fill=”Calidad”) +
  scale_fill_brewer(palette = “Dark2”, name=”Calidad”,
                    labels = c(“Malo”, “Bueno”,
                               “Muy bueno”, “Premium”, “Ideal”))
```


Calidad del corte de los diamantes
Calidad
Malo
Bueno
Muy bueno
Premium
Ideal

Con la función `theme`, además de centrar el título de nuestro gráfico, cambiamos de posición la leyenda. Con el argumento `legend.position`, que ya usamos arriba, establecemos la posicion que queremos: en este caso la colocamos debajo del gráfico con `bottom`.

```
ggplot(graph_arbol,
       aes(fill = cut, area=n)) +
  geom_treemap() +
  labs(title = “Calidad del corte de los diamantes”,
fill=”Calidad”) +
  scale_fill_brewer(palette = “Dark2”, name=”Calidad”,
                    labels = c(“Malo”, “Bueno”,
               “Muy bueno”, “Premium”, “Ideal”)) +
  theme(plot.title=element_text(hjust=0.5),legend.
position=”bottom”)#posición
```


Calidad del corte de los diamantes
Calidad
Malo
Bueno
Muy bueno
Premium
Ideal

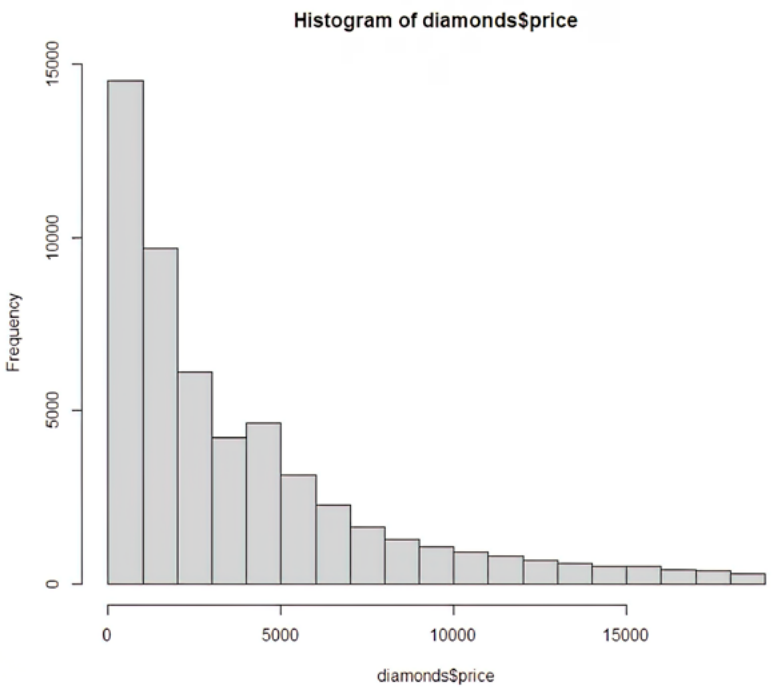
Numéricos

◊ *Histograma*

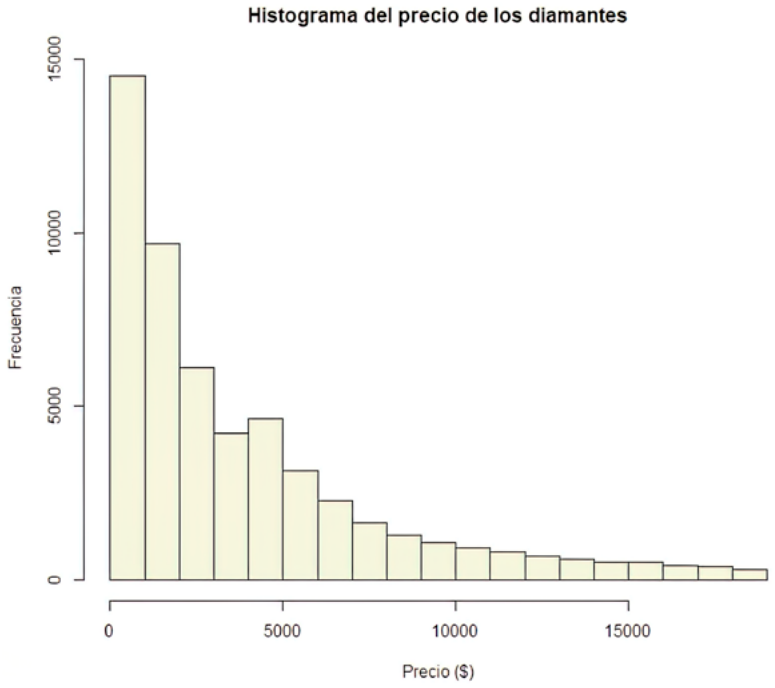
Muestra la distribución de un conjunto de datos. Sirve para obtener una vista general o panorama de la distribución de una población o muestra respecto a una característica. En este caso, la característica es la variable precio.

Podemos hacer un histograma simple utilizando el comando `hist( )`.

```
hist(diamonds$price)
```

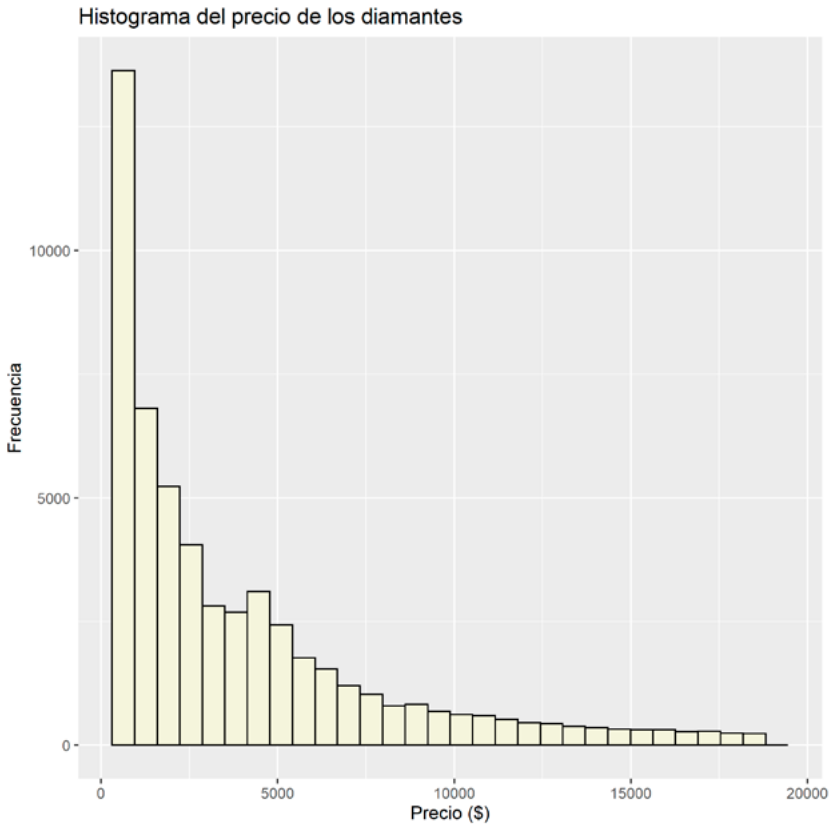


```
#Mejorando el gráfico de histograma

hist(diamonds$price, main =
       "Histograma del precio de los diamantes"
     #para cambiar el título del grafico
     ,xlab = "Precio ($)" #para renombrar el eje X
     ,ylab = "Frecuencia" #para renombrar el eje Y
     ,col = "beige" #para darle color al grafico
     ,border = "black") #para darle color a los bordes
```



También podemos hacer un histograma con las herramientas que nos da `ggplot`.

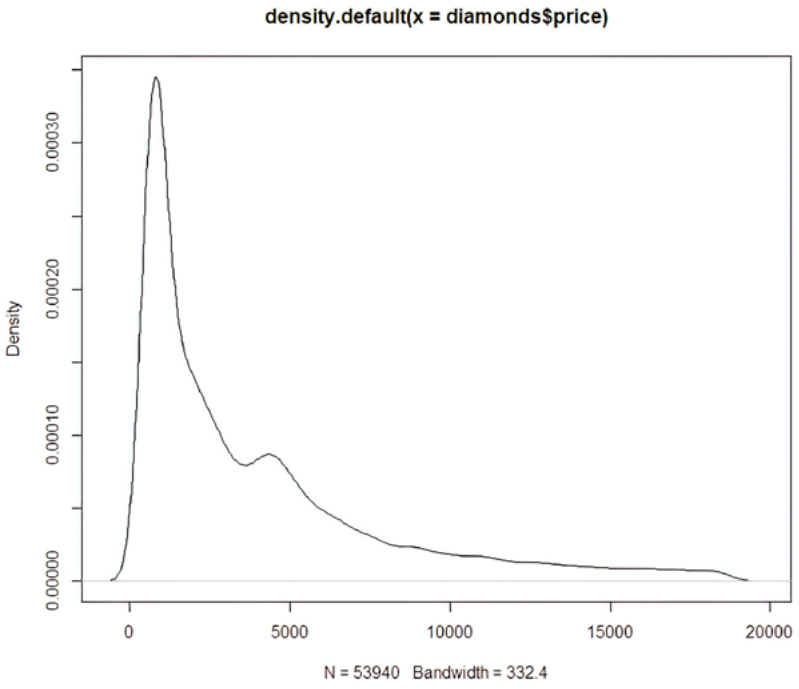
```
#con ggplot

ggplot(diamonds, aes(x = price)) +
  geom_histogram(fill= “beige”, color=”black”) +
  labs(title = “Histograma del precio de los diamantes”,
       x = “Precio ($)”,
       y= “Frecuencia”)
```



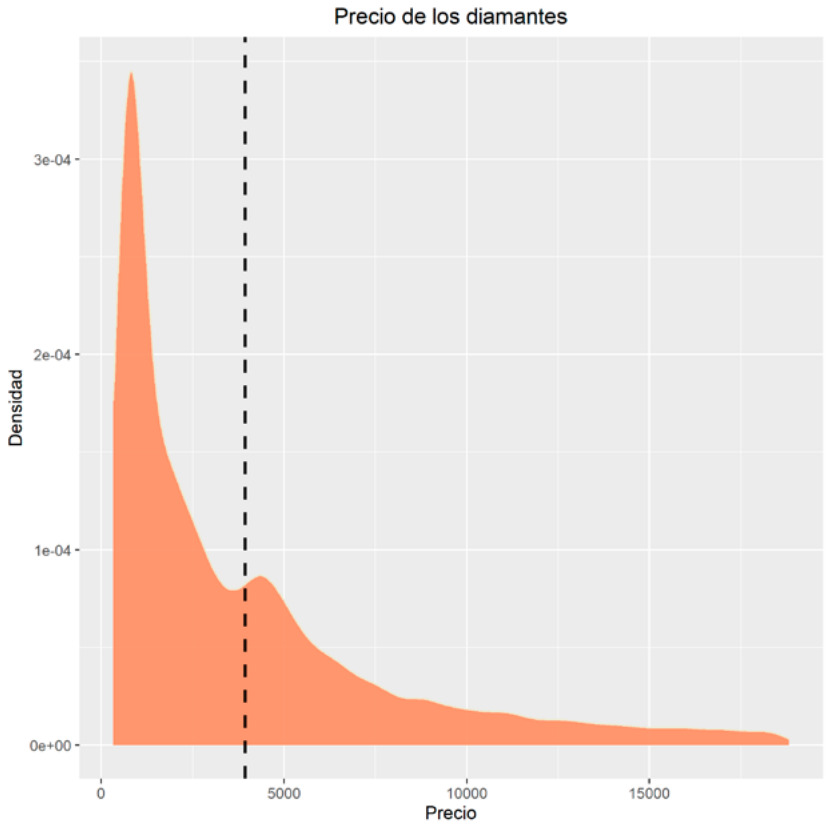
Densidad

Este gráfico es una variación del histograma. Se diferencia porque usa el suavizado de cerner para trazar valores. Esto permite observar distribuciones más suaves.

```
dens <- density(diamonds$price)
plot(dens)
```



```
ggplot(diamonds,aes(x=price))+
  geom_density(fill="coral", color="cornsilk2"
               , alpha=0.8) +
  labs (y= "Densidad", x="Precio") +
  ggtitle("Precio de los diamantes")  +
  geom_vline(aes(xintercept=mean(price)), #Agregamos la línea
media
   color="black", linetype="dashed", size=1) +
  theme(plot.title = element_text(hjust = 0.5)) #centramos título
```



También podemos superponer el histograma y la densidad.

```
#Le agregamos un relleno semitransparente con alpha
ggplot(diamonds, aes(x = price)) +
  geom_histogram(aes(y = ..density..),
                 colour = 1, fill = "white") +
    geom_density(alpha=.2, fill="darkgoldenrod2") +
  labs (y= "Densidad", x="Precio ($)")
```

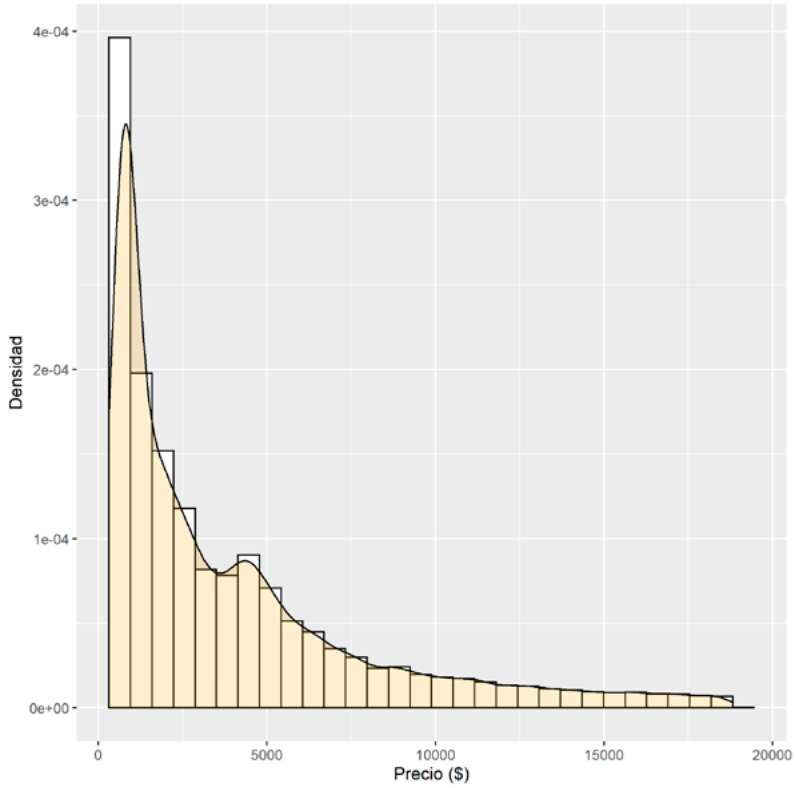
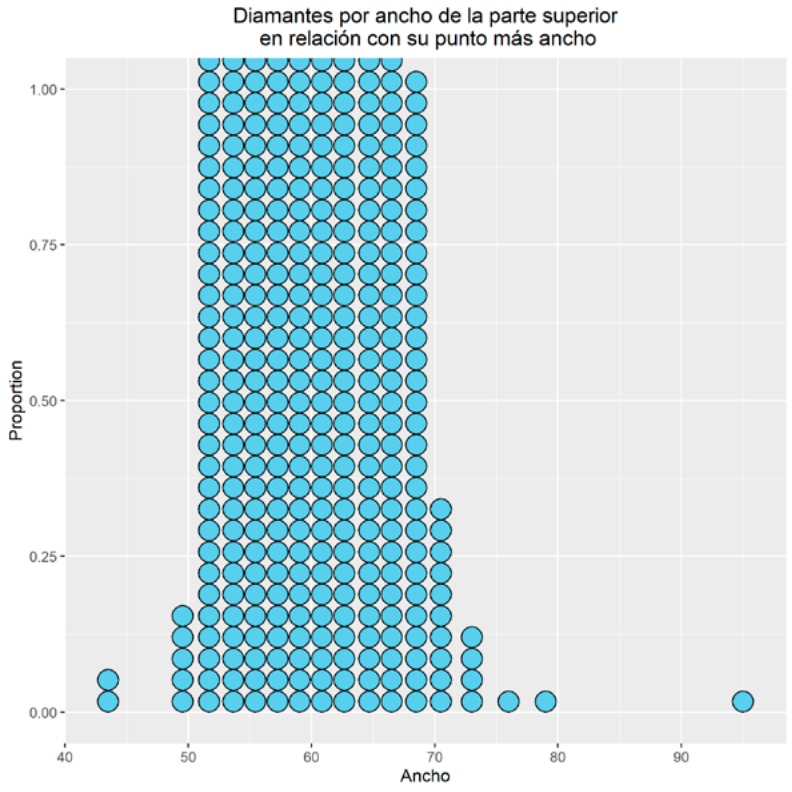


Gráfico de puntos

El gráfico de puntos es una alternativa al histograma, solo que en lugar de barras, cada observación o fila se representa con un punto. Se recomienda este tipo de gráficos cuando el número de observaciones es pequeño. Por ese motivo, optamos por graficar la variable "table".

Cuando el título es demasiado largo, podemos dividirlo en dos líneas usando `\n`.

```
ggplot(diamonds, aes(x = table)) +
  geom_dotplot(fill= "skyblue") +
  labs(title = "Diamantes por ancho de la parte superior\n en
relación con su punto más ancho",
       y = "Proportion",
       x = "Ancho")  +
  theme(plot.title = element_text(hjust = 0.5)) #centrar título
```

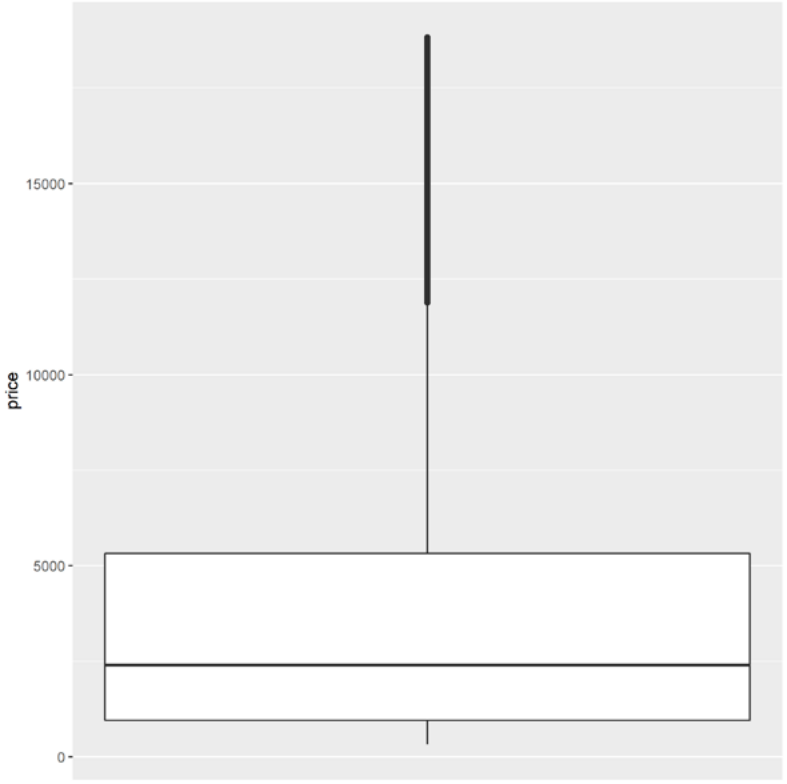


◊ *Gráfico de cajas*

El gráfico de cajas muestra también la distribución de los datos numéricos, sin embargo lo hace mediante el resumen de cinco números: el mínimo, primer cuartil, mediana, tercer cuartil y el máximo. Debido al resumen que realiza, el diagrama de cajas puede usarse para manejar una gran cantidad de datos. Además, muestra también los valores atípicos en caso los hubiera.

En el siguiente ejemplo, agregamos la función `scale_x_continous` y colocamos `NULL` en el argumento `breaks` a fin de que no aparezcan las escalas del eje *x* en el gráfico.

```
ggplot(diamonds, aes( y = price)) +
  geom_boxplot()  + scale_x_continuous( breaks=NULL)
```



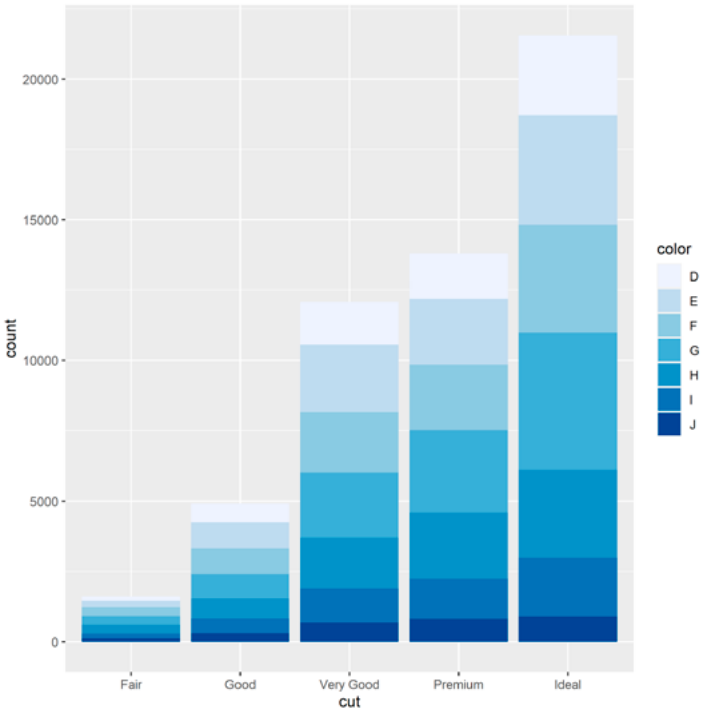
Gráficos bivariados

Categórico vs categórico

◊ *Barras apiladas*

En un gráfico de barras apiladas veremos la relación que existe entre el corte de los diamantes y el color. Agregamos el argumento `fill` o relleno y hacemos que sea equivalente a la variable "color". Asimismo, en la función `geom_bar` especificamos la posición `stack` que quiere decir apilar las barras una encima de la otra. Por último, añadimos la función `scale_fill_brewer` y elegimos la paleta de color de nuestra preferencia.

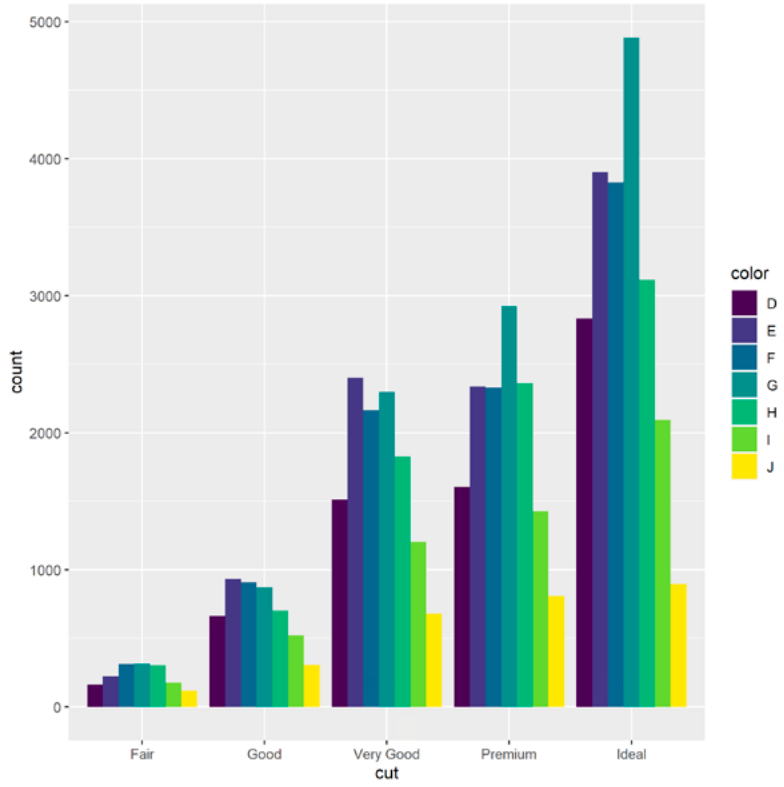
```
ggplot(diamonds,
       aes(x = cut,
           fill = color)) +
  geom_bar(position = "stack") +
  scale_fill_brewer(palette=1)
```



◊ *Barras agrupadas*

El gráfico de barras agrupadas coloca las variables categóricas una al lado de la otra. Para realizar este gráfico usamos `position="dodge"` dentro de la función `geom_bar`.

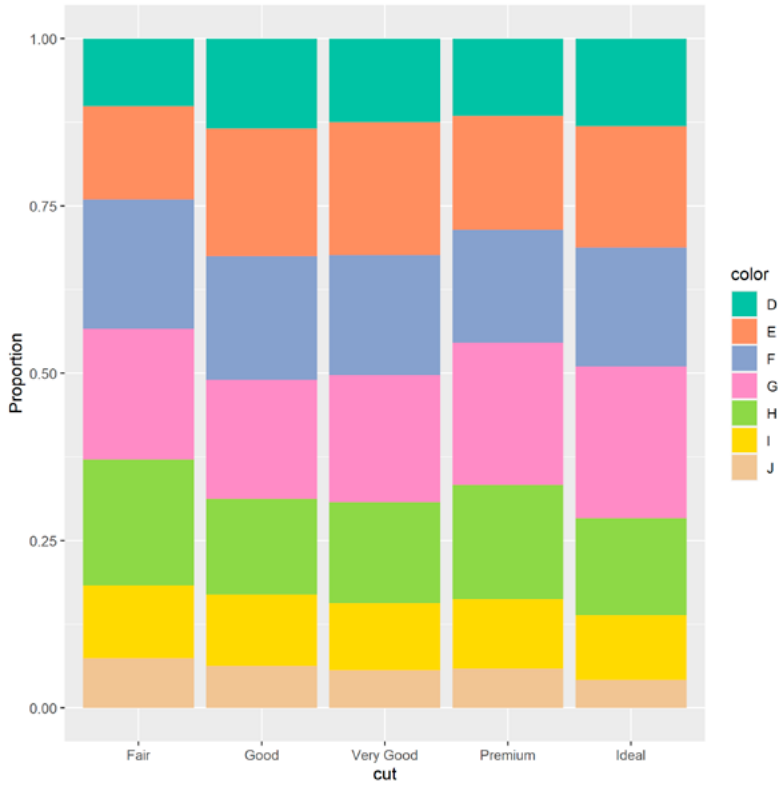
```
ggplot(diamonds,
       aes(x = cut,
           fill = color)) +
  geom_bar(position = "dodge")
```



◊ *Barras segmentadas*

Este gráfico, al igual que los anteriores, nos permite comparar dos categorías, sin embargo, lo hace dentro de un conjunto de datos donde cada barra representa el 100%. Usamos la opción `position=fill`.

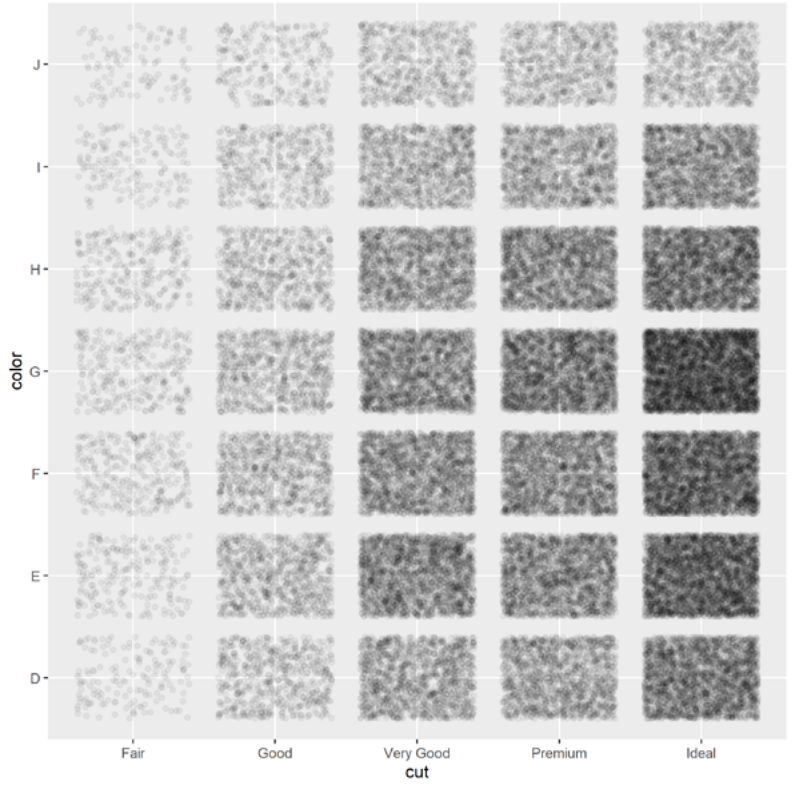
```
ggplot(diamonds,
       aes(x = cut,
           fill = color)) +
  geom_bar(position = "fill") +
  labs(y = "Proportion") + scale_fill_brewer(palette="Set2")
```



◊ *Gráfico con fluctuaciones*

Este tipo de gráfico facilita la comparación entre todas las combinaciones de las variables “color” y “cut”.

```
ggplot(diamonds, aes(cut, color)) +
  geom_jitter(alpha = 0.05)
```



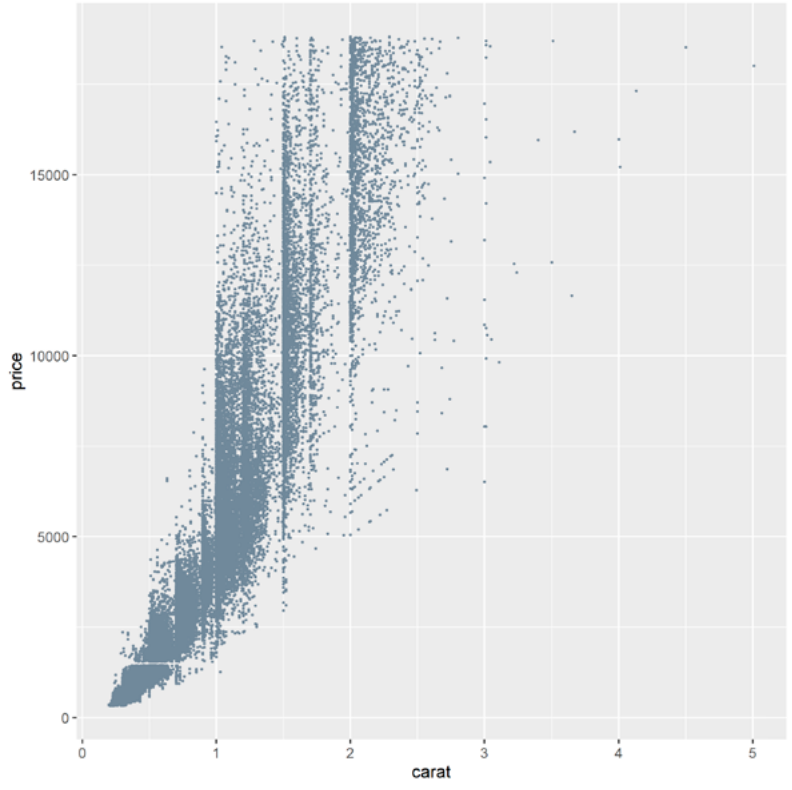
Cuantitativo vs cuantitativo

◊ *Gráfico de dispersión*

Los gráficos de dispersión muestran la correlación entre dos variables. Existe correlación cuando los puntos se concentran cerca de una línea recta, mientras que si se observa los puntos dispersos aleatoriamente sobre las coordenadas, entonces no existe correlación.

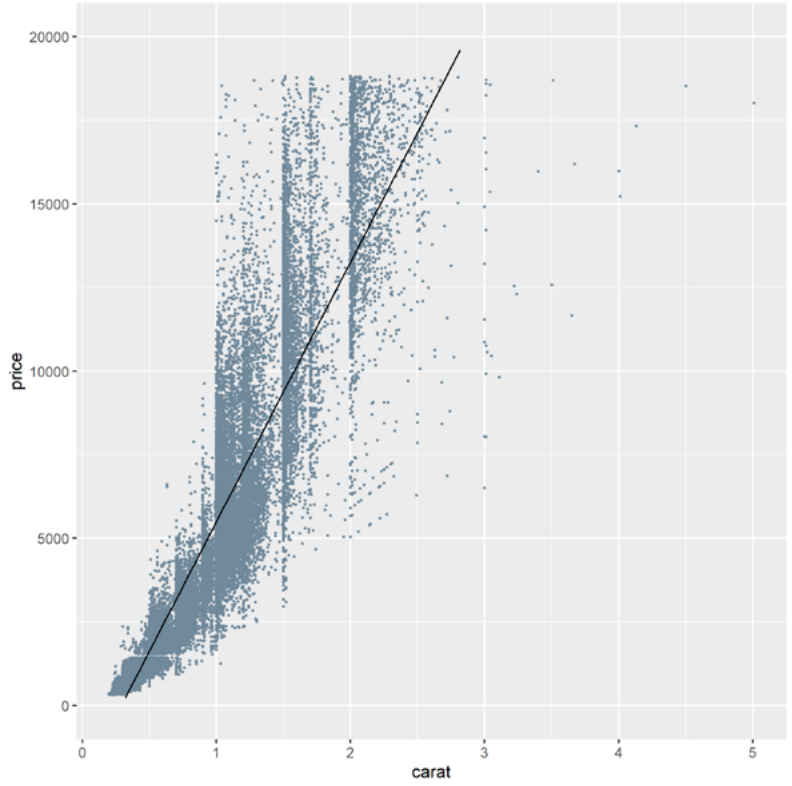
El tamaño de los puntos se puede controlar con `size`. El valor predeterminado de tamaño es 2; lo cambiaremos a 0.5 para tener puntos más pequeños.

```
ggplot(diamonds,
       aes(x = carat,
           y = price)) +
  geom_point(color=”lightslategrey”,size= 0.5)
```



Agregamos una línea de ajuste con la función `geom_smooth` y elegimos el método `"lm"` para ajustar el modelo lineal. Asímismo, con `ylim` definimos la escala del eje *y* para que tenga la misma presentación que el gráfico anterior.

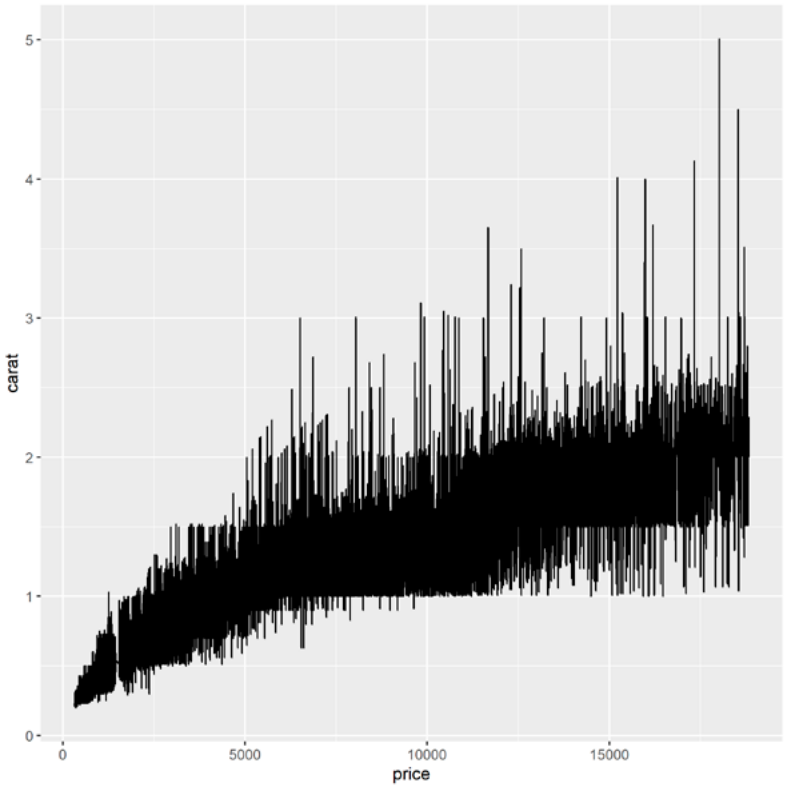
```
ggplot(diamonds,
       aes(x = carat,
           y = price)) +
  geom_point(color= "lightslategrey", size=0.5) +
  geom_smooth(method = "lm", colour="black", size = 0.5) +
ylim(0, 20000)
```



◊ *Gráfico de líneas*

A diferencia de los gráficos de dispersión que pueden ayudarnos a conocer el grado de correlación entre dos variables mediante la distribución de los puntos, el gráfico de líneas nos muestra tendencias y patrones. Se suele usar con datos de un tiempo en específico.

```
ggplot(diamonds,
       aes(x = price,
           y = carat)) +
  geom_line()
```

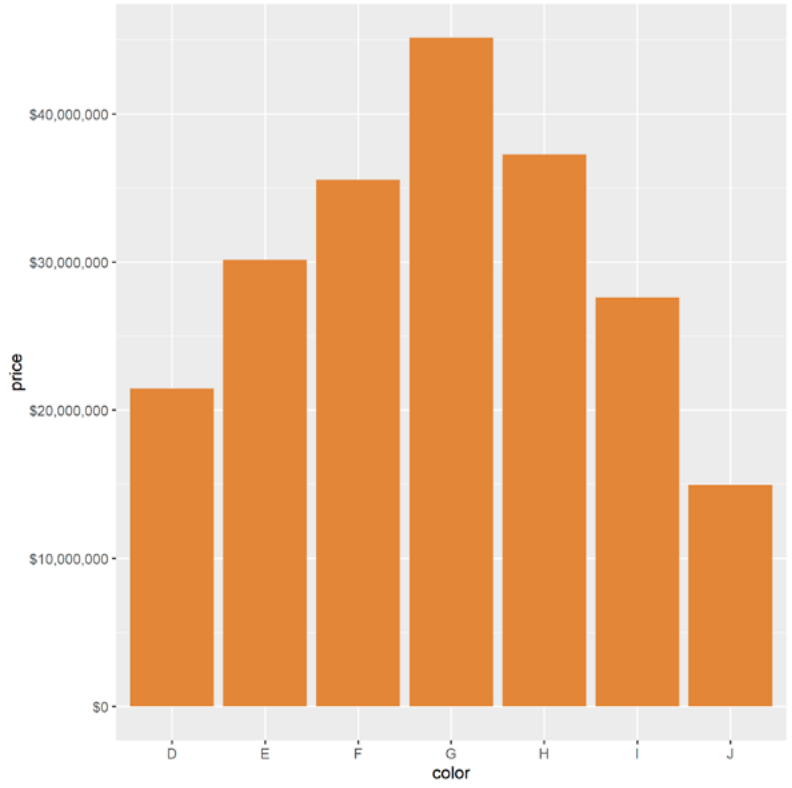


Categórico vs cuantitativo

◊ *Gráfico de barras*

En el siguiente gráfico se muestra el precio de los diamantes de una muestra del mejor (D) al peor color. Para evitar que aparezcan las escalas del eje *y* en notación científica, utilizamos la función `scale_y_continous` y agregamos la etiqueta "`dollar`".

```
ggplot (diamonds,
        aes(x = color,
            y = price)) +
  geom_bar(stat = "identity", fill="peru") +
  scale_y_continuous(label=dollar)
```

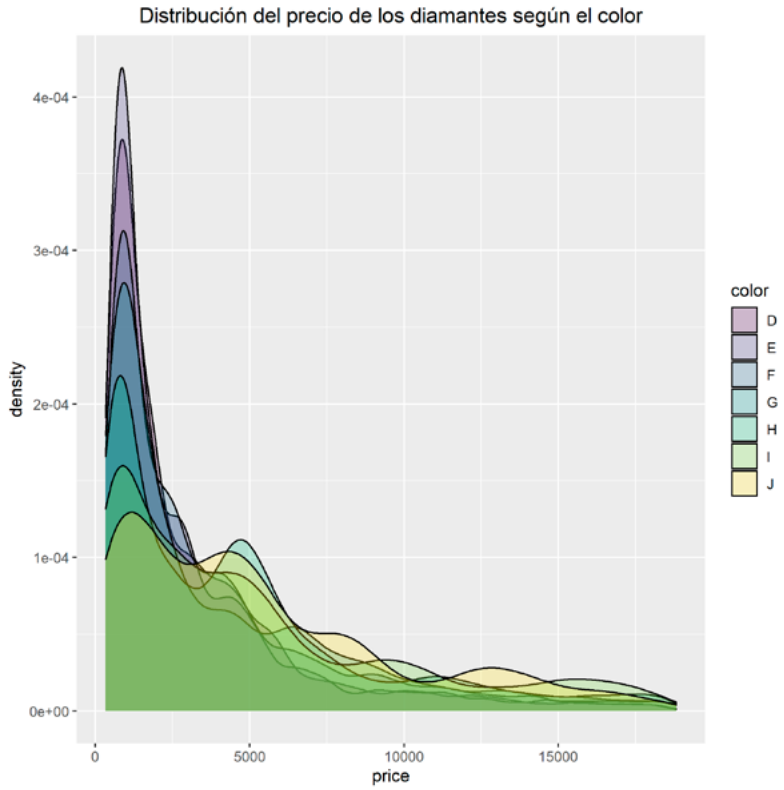


Vemos que los diamantes de color G tienen un precio superior a los de mejor color (D).

◊ *Densidad agrupada*

En este gráfico se superponen los diagramas de densidad en un solo gráfico.

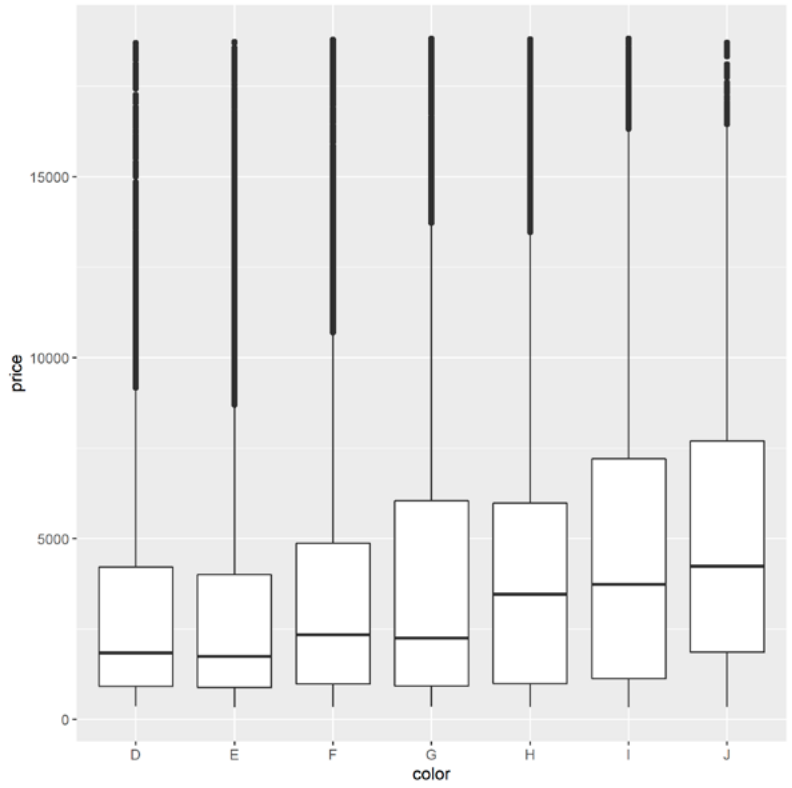
```
ggplot(diamonds,
       aes(x = price,
           fill = color)) +
  geom_density(alpha = 0.25) +
    labs(title = "Distribución del precio de los diamantes según
el color") +
  theme(plot.title = element_text(hjust = 0.5))
```



◊ *Gráficos de caja*

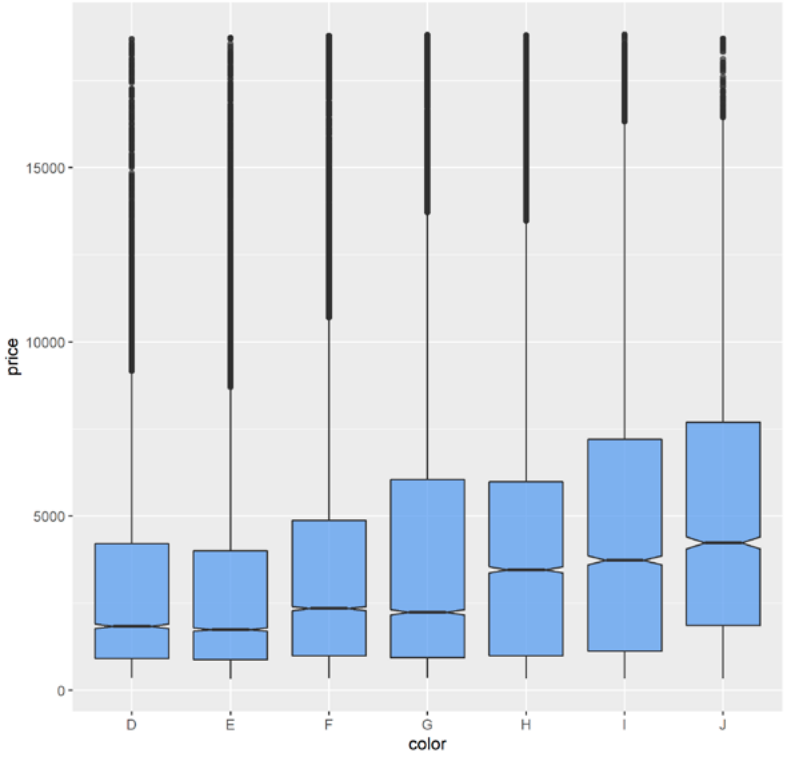
Como mencionamos en la sección anterior, los gráficos de caja nos brindan resúmenes de la distribución de los datos. Al representar estos diagramas juntos en un solo gráfico, nos permite realizar comparaciones útiles a los grupos de una variable categórica ("color") en una variable numérica ("price" o precio), tal como se muestra a continuación.

```
ggplot(diamonds,
       aes(x = color,
           y = price)) +
  geom_boxplot()
```



El corte del siguiente gráfico permite evaluar si las medianas difieren entre sí. Si el corte no se superpone, significa que las medianas son diferentes. El intervalo de confianza es de un 95% alrededor de la mediana. En la función `geom_boxplot` colocamos `notch=TRUE` para que nos aparezca el corte en los diagramas.

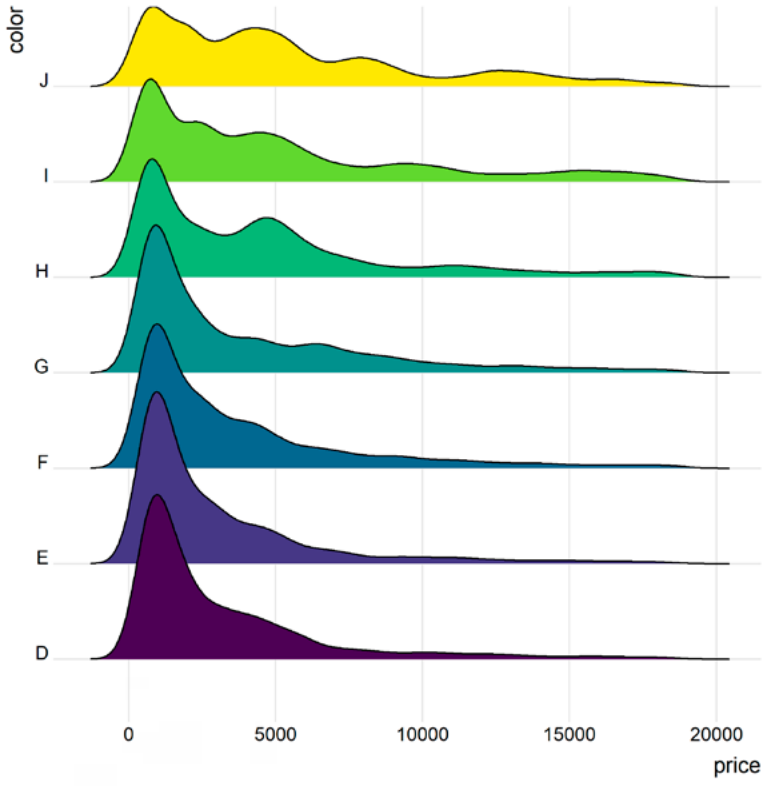
```
ggplot(diamonds, aes(x = color,
                     y = price)) +
  geom_boxplot(notch = TRUE,
               fill = "cornflowerblue",
               alpha = .7) +
  labs(title = "")
```



◊ *Gráfico "ridgelines"*

Este tipo de gráfico permite visualizar la distribución de una variable cuantitativa para varios grupos de una variable categórica. Asimismo, a diferencia del gráfico de densidad agrupada, tiene una mejor apreciación. Utilizamos el paquete `ggridges` para su construcción. Además, en vista de que aparecen las etiquetas de la variable "color" en el eje *y*, suprimimos la leyenda del gráfico con `legend.position="none"`.

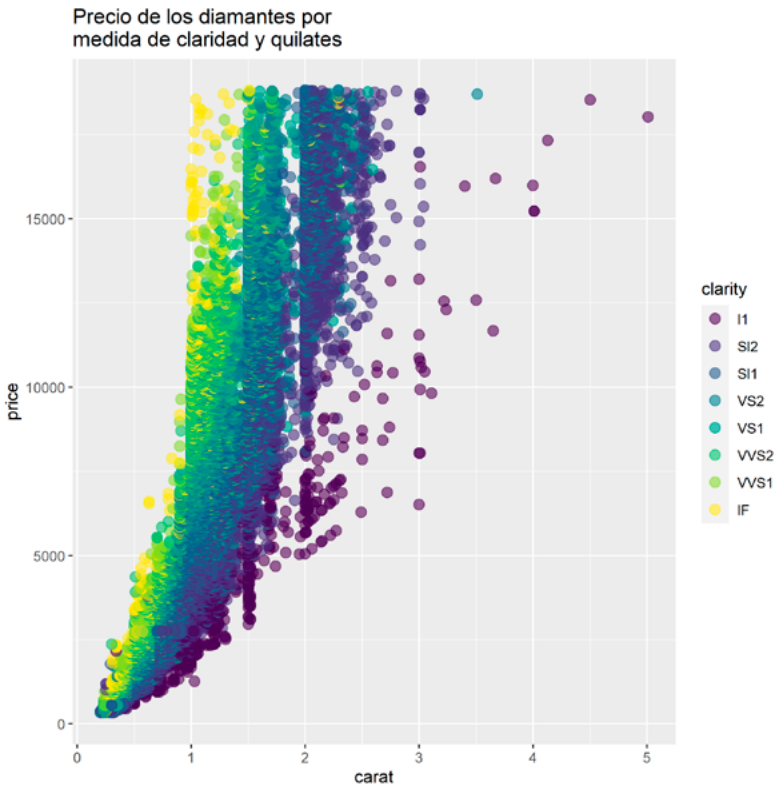
```
ggplot(diamonds,
       aes(x = price,
           y = color,
           fill = color)) +
  geom_density_ridges() +
  theme_ridges() +
  theme(legend.position = "none")
```



Gráficos multivariados

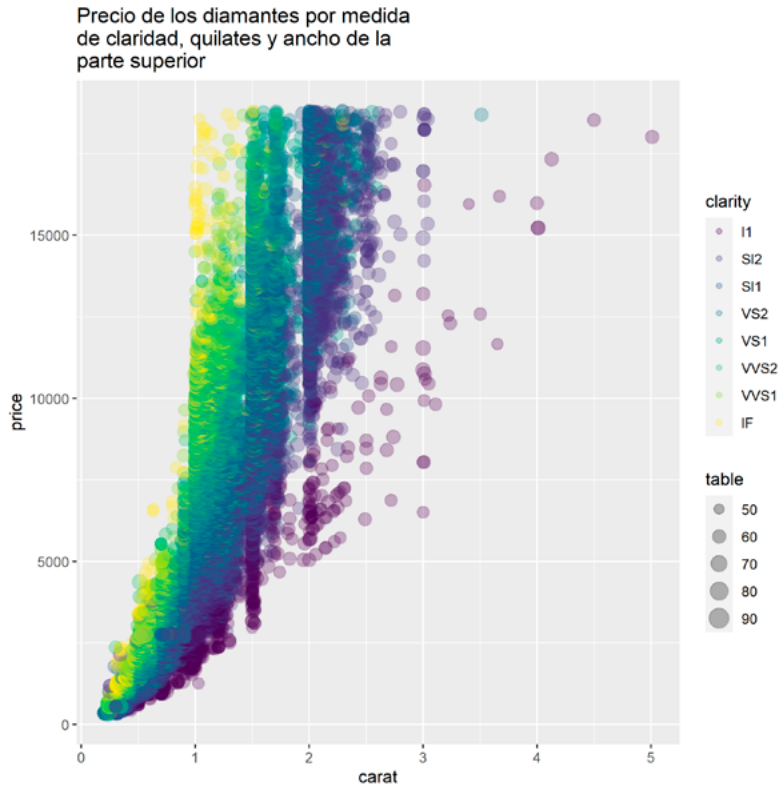
Los gráficos multivariados, como su mismo nombre lo indica, permiten visualizar las relaciones entre tres a más variables. Veamos un ejemplo.

```
ggplot(diamonds,
     aes(x = carat,
       y = price,
        color = clarity)) +
 geom_point( size = 3,
             alpha = .6) +
          labs(title = "Precio de los diamantes por \nmedida de
claridad y quilates")  +
  theme(plot.title = element_text(hjust = 0.5))
```



Agregamos una variable más utilizando el tamaño de los puntos para indicar la variable “table”.

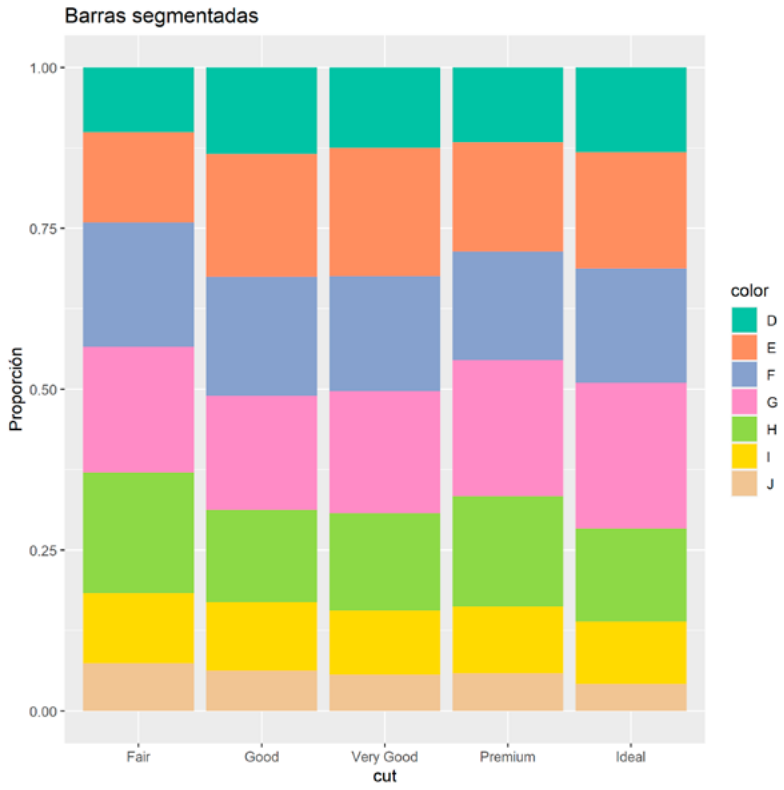
```
ggplot(diamonds,
     aes(x = carat,
       y = price,
        color = clarity, size = table)) +
 geom_point( alpha = .3) +
          labs(title = “Precio de los diamantes por medida \nde
claridad, quilates y ancho de la parte superior”) +
  theme(plot.title = element_text(hjust = 0.5))
```



Guardar gráficos

Una vez que estemos satisfechos con los resultados de los gráficos, podemos guardarlos. Existen diferentes extensiones en las que puedes guardar tus gráficos desde R, como jpg, png, tiff, pdf, entre otros. La función `ggsave` permite esto: solo tienes que llamar a la función justo después de trazar el gráfico. En este caso, deseamos guardar en nuestro directorio el gráfico de barras segmentadas en formato tiff y con una calidad de imagen de 300 dpi.

```
#Código del gráfico de barras segmentadas
##Lo guardamos en el objeto "barras_segmentadas"

barras_segmentadas <- ggplot(diamonds,
       aes(x = cut,
           fill = color)) +
  geom_bar(position = "fill") +
  labs(y = "Proporción", title = "Barras segmentadas") +
  scale_fill_brewer(palette="Set2")

barras_segmentadas
```




```
#guardamos el gráfico
ggsave("barras_segmentadas.tiff", dpi=300)
## Saving 5 x 4 in image
```

¡Listo! Ya tenemos el gráfico guardado en nuestro directorio de trabajo.

Resumen del capítulo

En este capítulo aprendimos a realizar gráficos con una, dos y más variables, tanto categóricas como numéricas. La visualización de datos es importante porque de manera sencilla uno puede comprender los datos, facilita la identificación de patrones, tendencias y asociaciones de una pequeña o gran cantidad de datos.

Además, brinda una representación gráfica de los datos. Es útil para explorar datos y comunicar resultados, así como para obtener una información rápida de los mismos.

CAPÍTULO 4

ANÁLISIS EXPLORATORIO DE DATOS

En este capítulo realizaremos un análisis exploratorio que nos permitirá realizar inferencias y extraer conclusiones a partir de los datos. Para ello, consideramos las bases de datos libre **cars**, **Titanic** y **birthwt**.

En primer lugar, realizaremos una exploración de datos y calcularemos las medidas de frecuencia, de tendencia central y de dispersión, para luego realizar pruebas de normalidad mediante gráficos y métodos estadísticos. Posteriormente, calcularemos las medidas de asociación según el tipo de diseño de estudio, ya sea transversal o de cohortes.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Identificar y calcular las medidas de frecuencia absoluta y relativa.
- Identificar y calcular las medidas de tendencia central y de dispersión.
- Realizar pruebas de normalidad mediante gráficos y métodos estadísticos.
- Identificar y calcular las medidas de asociación en diseños de estudio transversales y de cohortes.

Paquetes

Instalación

```
#install.packages("dplyr")
#install.packages("datasets")
#install.packages("modeest")
#install.packages("plotrix")
#install.packages("nortest")
```

```
#install.packages("dgof")
#install.packages("ggplot2")
#install.packages("RColorBrewer")
#install.packages("car")
#install.packages("MASS")
#install.packages("Hmisc")
#install.packages("epiR")
```

Carga

```
library(dplyr) #manipular datos
library(datasets) #base de datos libre
library(modeest) #moda
library(plotrix) #error estandar
library(nortest)#Kolmogorov
library(dgof) #ks.test
library(ggplot2) #gráficos
library(RColorBrewer)#Paleta de colores
library(car) #qqplot
library(MASS) #Base de datos birthwt
library(Hmisc)#función "describe"
library(epiR) #análisis de datos epidemiológicos
```

Estudio de caso

El conjunto de datos **cars** muestra la velocidad de los automóviles y las distancias que tardan en detenerse. Los datos, registrados en la década de 1920, incluyen dos variables:

Variable	Descripción	Tipo de variable
dist	Distancia de frenado (pies)	Cuantitativa
speed	Velocidad (millas por hora)	Cuantitativa

Esta base de datos la utilizaremos para calcular las medidas de frecuencia, de tendencia central y de dispersión, así como para realizar un análisis de normalidad gráfico y estadístico.

Medidas de frecuencia

Frecuencia absoluta

La frecuencia absoluta nos da información acerca de la cantidad de veces que se repite un suceso al realizar un número determinado de experimentos aleatorios. Utilizamos la función `table()` para conocer la frecuencia absoluta de la base de datos **cars**, además, examinamos si existen valores perdidos.

```
#Número de veces que se repite un valor en la base de datos
table(cars)
##      dist
## speed 2 4 10 14 16 17 18 20 22 24 26 28 32 34 36 40 42 46 48
50 52 54 56 60 64
##    4  1 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0
##    7  0 1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0
##    8  0 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0
##    9  0 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0
##    10 0 0  0  0  0  0  1  0  0  0  1  0  0  1  0  0  0  0  0
0  0  0  0  0  0
##    11 0 0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0
0  0  0  0  0  0
##    12 0 0  0  1  0  0  0  1  0  1  0  1  0  0  0  0  0  0  0
0  0  0  0  0  0
##    13 0 0  0  0  0  0  0  0  0  0  1  0  0  2  0  0  0  1  0
0  0  0  0  0  0
##    14 0 0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0
```

```
0  0  0  0  1  0
##    15 0 0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0
0  0  1  0  0  0
##    16 0 0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0
0  0  0  0  0  0
##    17 0 0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0
1  0  0  0  0  0
##    18 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
0  0  0  1  0  0
##    19 0 0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0
0  0  0  0  0  0
##    20 0 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1
0  1  0  1  0  1
##    22 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0
##    23 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  1  0  0  0
##    24 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0
##    25 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0
##      dist
## speed 66 68 70 76 80 84 85 92 93 120
##    4   0  0  0  0  0  0  0  0  0   0
##    7   0  0  0  0  0  0  0  0  0   0
##    8   0  0  0  0  0  0  0  0  0   0
##    9   0  0  0  0  0  0  0  0  0   0
##    10  0  0  0  0  0  0  0  0  0   0
##    11  0  0  0  0  0  0  0  0  0   0
##    12  0  0  0  0  0  0  0  0  0   0
##    13  0  0  0  0  0  0  0  0  0   0
##    14  0  0  0  0  1  0  0  0  0   0
##    15  0  0  0  0  0  0  0  0  0   0
##    16  0  0  0  0  0  0  0  0  0   0
```

```
##    17  0  0  0  0  0  0  0  0  0   0
##    18  0  0  0  1  0  1  0  0  0   0
##    19  0  1  0  0  0  0  0  0  0   0
##    20  0  0  0  0  0  0  0  0  0   0
##    22  1  0  0  0  0  0  0  0  0   0
##    23  0  0  0  0  0  0  0  0  0   0
##    24  0  0  1  0  0  0  0  1  1   1
##    25  0  0  0  0  0  0  1  0  0   0
#Para saber si hay valores perdidos
table(complete.cases(cars))
##
## TRUE
##   50
```

Si hubiera valores perdidos, estos aparecerían como `FALSE`. En este caso no hay valores perdidos.

Frecuencia relativa (porcentaje)

Es el cociente de la frecuencia absoluta de algún valor de la población/muestra entre el total de valores que componen la población/muestra. Utilizamos la función `prob.table()` para crear tablas de frecuencias relativas a partir de tablas de frecuencia absoluta.

```
#Nos muestra la proporción de cada frecuencia absoluta
prop.table(cars)
##          speed         dist
## 1  0.001370332 0.0006851662
## 2  0.001370332 0.0034258308
## 3  0.002398082 0.0013703323
## 4  0.002398082 0.0075368277
## 5  0.002740665 0.0054813292
## 6  0.003083248 0.0034258308
## 7  0.003425831 0.0061664954
```

```
## 8  0.003425831 0.0089071600
## 9  0.003425831 0.0116478246
## 10 0.003768414 0.0058239123
## 11 0.003768414 0.0095923261
## 12 0.004110997 0.0047961631
## 13 0.004110997 0.0068516615
## 14 0.004110997 0.0082219938
## 15 0.004110997 0.0095923261
## 16 0.004453580 0.0089071600
## 17 0.004453580 0.0116478246
## 18 0.004453580 0.0116478246
## 19 0.004453580 0.0157588215
## 20 0.004796163 0.0089071600
## 21 0.004796163 0.0123329908
## 22 0.004796163 0.0205549846
## 23 0.004796163 0.0274066461
## 24 0.005138746 0.0068516615
## 25 0.005138746 0.0089071600
## 26 0.005138746 0.0184994861
## 27 0.005481329 0.0109626584
## 28 0.005481329 0.0137033231
## 29 0.005823912 0.0109626584
## 30 0.005823912 0.0137033231
## 31 0.005823912 0.0171291538
## 32 0.006166495 0.0143884892
## 33 0.006166495 0.0191846523
## 34 0.006166495 0.0260363138
## 35 0.006166495 0.0287769784
## 36 0.006509078 0.0123329908
## 37 0.006509078 0.0157588215
## 38 0.006509078 0.0232956492
## 39 0.006851662 0.0109626584
## 40 0.006851662 0.0164439877
## 41 0.006851662 0.0178143200
```

```
## 42 0.006851662 0.0191846523
## 43 0.006851662 0.0219253169
## 44 0.007536828 0.0226104830
## 45 0.007879411 0.0184994861
## 46 0.008221994 0.0239808153
## 47 0.008221994 0.0315176430
## 48 0.008221994 0.0318602261
## 49 0.008221994 0.0411099692
## 50 0.008564577 0.0291195615
```

Medidas de tendencia central

Las medidas de tendencia central nos muestran un valor representativo de un conjunto de datos.

Media

Conocida también como promedio, es el valor que se obtiene al dividir la suma de un conglomerado de números entre la cantidad de ellos. Utilizamos la función `mean()` del paquete base de R; dentro del paréntesis primero se coloca el nombre de la base de datos y se escribe el signo de dólar `$`, este elemento permite extraer un solo elemento a la vez, en este caso se extrae la columna "dist" para obtener su media.

```
mean(cars$dist)
## [1] 42.98
```

La media, o el promedio aritmético de los datos, es 42.98.

Mediana

Es el valor que se encuentra en el lugar central de todos los datos de un estudio cuando estos están ordenados de menor a mayor. Utilizamos la función `median` del

paquete `stats` que viene incorporado en la versión principal de R, por lo que no es necesaria su instalación.

```
median(cars$dist)
## [1] 36
```

El valor medio de la distribución de datos de la variable "dist" es 36.

Moda

Es el valor del conjunto de datos que tiene una mayor frecuencia absoluta, es decir, la moda es el valor que más se repite. La función que se utiliza para obtener la moda es `mfv` del paquete `modeest`.

```
mfv(cars$dist)
## [1] 26
```

La moda, el valor más común de los datos de la variable, es 26.

Medidas de dispersión

Las medidas de dispersión nos muestran la variabilidad o en cuánto difieren los valores observados de una variable entre sí.

Desviación estándar

Es la raíz cuadrada del promedio de las distancias al cuadrado que van desde las observaciones a la media. Se calcula con la función `sd`.

```
sd(cars$dist)
## [1] 25.76938
```

La desviación estándar es de 25.76938.

Rango intercuartílico

Es la diferencia entre el primer y el tercer cuartil. Se calcula con la función `IQR`.

```
IQR(cars$dist)
## [1] 30
```

Varianza

Es el promedio de las distancias al cuadrado que van desde las observaciones a la media. Se calcula con la función `var`.

```
var(cars$dist)
## [1] 664.0608
```

La varianza o la medida de dispersión, que muestra la variabilidad de los datos respecto a su media, es 664.0608.

Rango

Es la diferencia existente entre el valor mayor y el menor de la distribución.

```
#Si deseamos saber solo el valor mínimo
min(cars$dist)
## [1] 2
#Si deseamos saber solo el valor máximo
max(cars$dist)
## [1] 120
#Si deseamos saber el valor minimo y máximo
#de una variable, el rango nos lo muestra
range(cars$dist)
## [1]   2 120
```

El valor mínimo es 2 y el máximo es 120.

Error estándar

Es la desviación estándar de la distribución muestral de un estadístico muestral.

```
std.error(cars$dist)
## [1] 3.64434
```

El error estándar o la desviación estándar de la distribución de los datos es 3.64434.

Cuantiles

Son medidas de localización, su función es informar del valor de la variable que ocupará la posición (en tanto por cien) que nos interese respecto de todo el conjunto de variables.

```
quantile(cars$dist)
##   0%  25%  50%  75% 100%
##    2   26   36   56  120
```

En la variable "dist", el caso equivalente al 0% es la distancia de frenado de 2 pies, el caso equivalente al 25% es cuando la distancia de frenado es de 26 pies y así sucesivamente.

Percentil

Dividen la sucesión de datos ordenados en cien partes porcentualmente iguales. Utilizamos `quantile` para este propósito.

```
quantile(cars$dist, c(.60))
## 60%
##  46
```

El caso equivalente al 60% es 46.

Resumen estadístico

Cada vez que trabajas con un conjunto de datos, necesitas conocer la descripción general de lo que estás tratando. La función `summary`, por ejemplo, te proporcionará un resumen estadístico para todas las variables numéricas de un conjunto de datos:

```
summary(cars)
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Con el resultado de esta función podemos ver los valores mínimos, medios y máximos de la velocidad y distancia de los automóviles, así como el primer y tercer cuartil de las dos variables.

Otra función útil es `describe`, la cual proporciona además el valor Gmd, es decir, información sobre la desviación y la varianza.

```
describe(cars)
## cars
##
##  2  Variables      50  Observations
## ---------------------------------------------------------------
-------------------
## speed
##        n  missing distinct     Info     Mean      Gmd      .05
.10
##       50        0       19    0.996     15.4    6.077      7.0
8.9
##      .25      .50      .75      .90      .95
##     12.0     15.0     19.0     23.1     24.0
```

```
##
## lowest :  4  7  8  9 10, highest: 20 22 23 24 25
##
## Value         4    7    8    9   10   11   12   13   14   15
16   17   18
## Frequency     2    2    1    1    3    2    4    4    4    3
2    3    4
## Proportion 0.04 0.04 0.02 0.02 0.06 0.04 0.08 0.08 0.08 0.06
0.04 0.06 0.08
##
## Value        19   20   22   23   24   25
## Frequency     3    5    1    1    4    1
## Proportion 0.06 0.10 0.02 0.02 0.08 0.02
## ------------------------------------------------------------
-------------------
## dist
##        n  missing distinct     Info     Mean      Gmd      .05
.10
##       50        0       35    0.999    42.98    28.85    10.00
15.80
##      .25      .50      .75      .90      .95
##    26.00    36.00    56.00    80.40    88.85
##
## lowest :   2   4  10  14  16, highest:  84  85  92  93 120
## ------------------------------------------------------------
-------------------
```

Análisis de normalidad

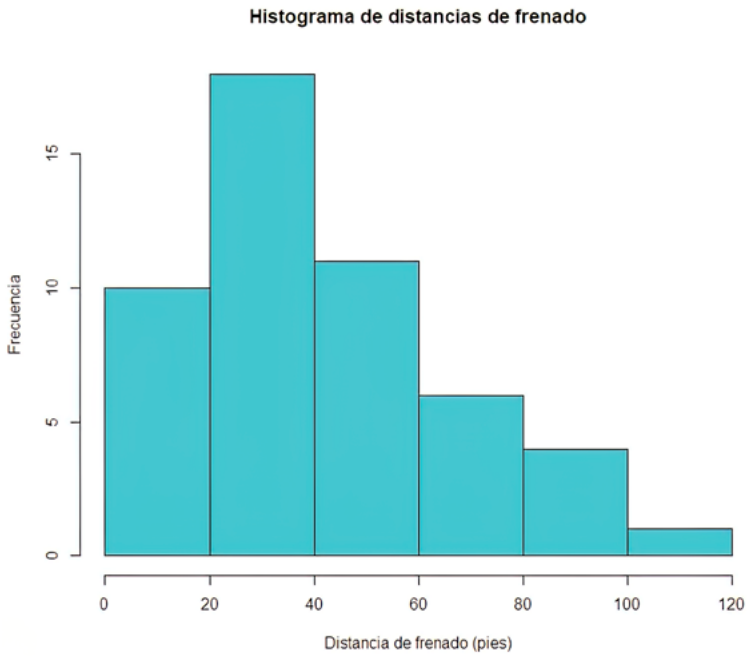
Es importante para los datos continuos porque permite decidir las medidas de tendencia central y los métodos estadísticos para el análisis de datos.

Análisis gráfico de normalidad

◊ *Histograma*

Una variable que se distribuye normalmente tiene un histograma con forma de campana y es simétrico alrededor de la media.

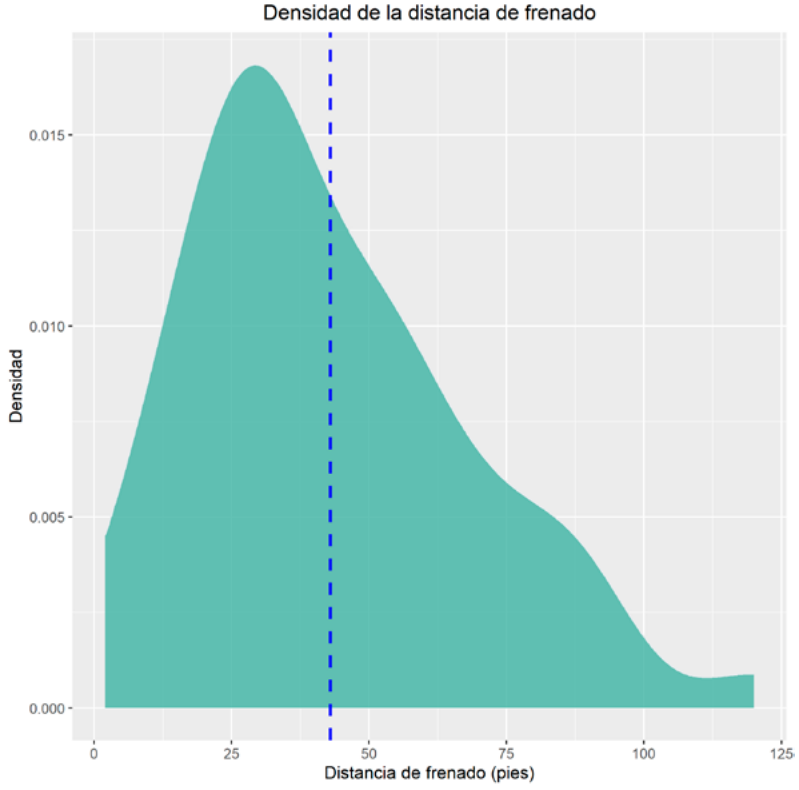
```
hist(cars$dist, main = "Histograma de distancias de frenado"
     ,xlab = "Distancia de frenado (pies)"
     ,ylab = "Frecuencia"
     ,col = "cadetblue3"
     ,border = "black")
```



◊ *Densidad*

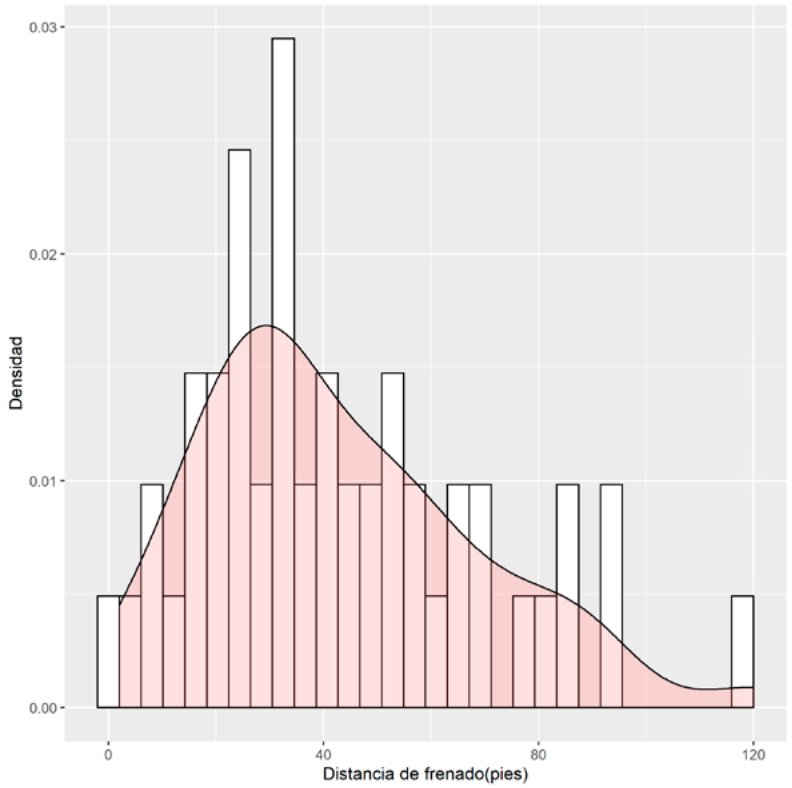
Permite visualizar la distribución de datos de forma más suave.

```
ggplot(cars,aes(x=dist))+
  geom_density(fill="#69b3a2", color="#e9ecef"
               #especificación de color hexadecimal
               , alpha=0.8) +
  labs (y= "Densidad", x="Distancia de frenado (pies)") +
  ggtitle("Densidad de la distancia de frenado")  +
  geom_vline(aes(xintercept=mean(dist)),
   color="blue", linetype="dashed", size=1) +
  theme(plot.title = element_text(hjust = 0.5))
```



◊ *Histograma y densidad en un solo gráfico*

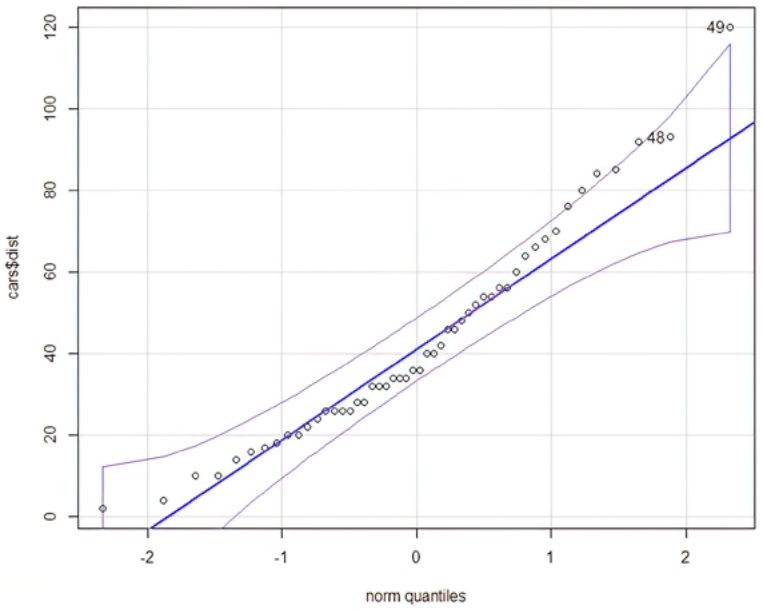
```
ggplot(cars, aes(x = dist)) +
  geom_histogram(aes(y = ..density..),
                 colour = 1, fill = "white") +
    geom_density(alpha=.2, fill="#FF6666") +
  labs (y= "Densidad", x="Distancia de frenado(pies)")
```



◊ *QQ plot*

Representa un diagrama de puntos que dibuja las funciones de distribución acumuladas. Si provienen de una misma distribución, los puntos aparecen alineados.

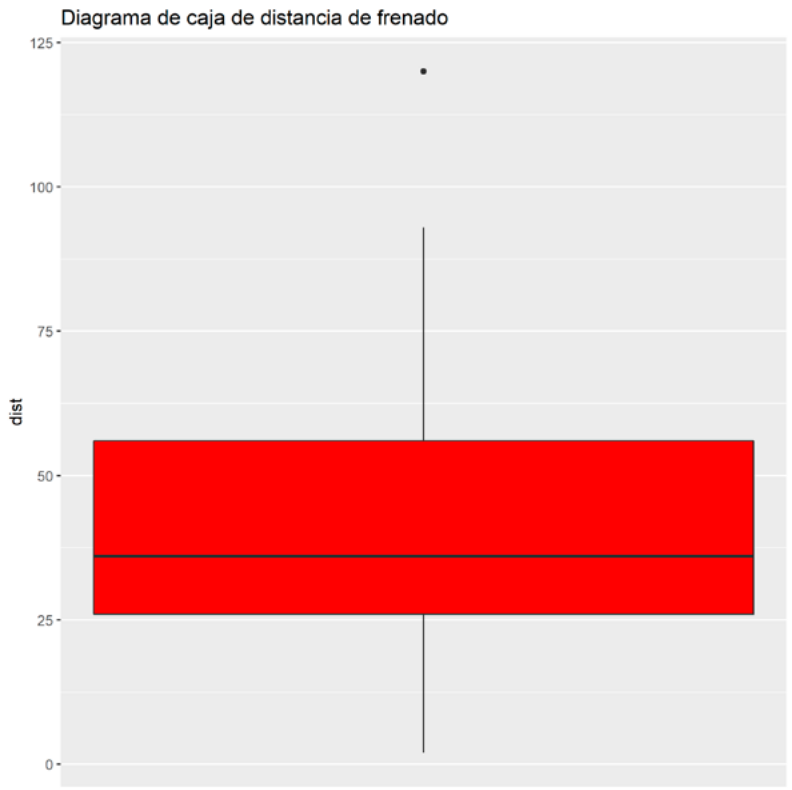
```
qqPlot(cars$dist)
```




```
## [1] 49 48
```

◊ *Boxplot o diagrama de cajas*

```
ggplot(cars, aes( y = dist)) +
  geom_boxplot(fill="red")  +
  scale_x_continuous( breaks=NULL) +
  labs(title="Diagrama de caja de distancia de frenado")
```



Análisis estadístico de normalidad

◊ *Prueba de Shapiro-Wilks*

El test de Shapiro-Wilks es aplicable solo para variables con hasta 50 observaciones. Esta prueba plantea la hipótesis nula de que una muestra proviene de una distribución normal. Si el *p-value* es menor a 0.05, se rechaza la hipótesis nula y se concluye que la variable no tiene una distribución paramétrica.

```
shapiro.test(cars$dist)
##
##  Shapiro-Wilk normality test
##
## data:  cars$dist
## W = 0.95144, p-value = 0.0391
```

El *p-value* es de 0.0391, en consecuencia, la distancia de frenado no tiene una distribución paramétrica.

◊ *Prueba de Shapiro-Francia*

El test de Shapiro-Francia plantea la hipótesis nula de que una muestra proviene de una distribución normal. Si el *p-value* es menor a 0.05 rechaza la hipótesis nula y se concluye que la variable no tiene una distribución normal o paramétrica.

```
sf.test(cars$dist)
##
##  Shapiro-Francia normality test
##
## data:  cars$dist
## W = 0.95206, p-value = 0.04179
```

El *p-value* es de 0.04179, en consecuencia, la distancia de frenado no tiene una distribución normal

◊ *Prueba de Kolmogorov-Smirnov con el ajuste de Lilliefors*

La prueba de Kolgomorov-Smirnov asume conocidas la media y varianza poblacional, lo que, en la mayoría de los casos, es imposible conocer. Para solucionar este problema se desarrolló una modificación conocida como test Lilliefors. Este asume que la media y varianza son desconocidas, por lo que está especialmente desarrollado para testear la normalidad.

```
lillie.test(cars$dist)
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  cars$dist
## D = 0.12675, p-value = 0.04335
```

El *p-value* es de 0.04335, es decir, no tiene una distribución paramétrica.

Medidas de asociación

Miden la fuerza de relación entre un determinado evento con un determinado factor.

Tabla de contingencia

La tabla de contingencia o tabla de doble entrada (tabla de 2x2) es importante para realizar este tipo de análisis. Está compuesta por dos filas y dos columnas. En las filas se coloca el nivel de exposición y en las columnas el de desenlace o resultado. Se ubican en la primera columna y fila los expuestos y resultados positivos respectivamente, tal como se muestra a continuación:

	Des+	**Des-**	**Total**
Exp+	a	b	a+b
Exp-	c	d	c+d
Total	a+c	b+d	a+b+c+d

Recordar este formato de tabla es esencial para continuar con los siguientes apartados.

Según diseño de estudio

◊ *Diseño transversal*

Razón de probabilidades (*odds ratios*)

El estado de exposición (+/-) es primero. En este diseño de estudio, se sigue a los sujetos a lo largo del tiempo para determinar su estado de resultado (+/-). Las probabilidades del resultado en la población expuesta se calculan de la siguiente manera:

```
OR= (a/c) / (b/d)
```

Diferencia de razón de probabilidades

```
(a/c) - (c/d)
```

Razón de prevalencia

Es la proporción de un grupo de sujetos con algún atributo o resultado en un período de tiempo determinado. En otras palabras, indica el grado de prevalencia de un resultado o evento en un grupo en relación con otro grupo (sin las mismas características o atributos).

```
(a/(a+b)) / (c/(c+d))
```

Diferencia de prevalencias

```
(a/(a+b) - (c/(c+d))
```

Estudio de caso

El Titanic fue un lujoso trasatlántico británico que se hundió tras chocar contra un iceberg la madrugada del 15 de abril de 1912. Este accidente provocó la muerte de más de 1500 personas, entre pasajeros y tripulantes.

La base de datos libre **Titanic** proporciona información sobre el destino de estas personas. En este caso, nos interesa conocer la asociación entre el género y la supervivencia: veremos si algunos grupos tenían más probabilidad de sobrevivir que otros, ya que aspectos como la priorización de las mujeres y niños y las proporciones de pasajeros de primera clase se reflejan en las tasas de supervivencia. Con esta base de datos calcularemos las medidas de asociación propias de los diseños de estudio transversales.

En primer lugar, exploraremos y manipularemos la base de datos de forma tal que se pueda crear una tabla de contingencia –o tabla de 2x2–. Luego, procederemos a explicar las diferentes medidas de asociación; finalmente, con el paquete `epiR` obtendremos de la base de datos los resultados de las distintas medidas para su posterior interpretación.

La base de datos contiene las siguientes variables:

Variable	Descripción	Tipo de variable
class	Status económico 1, 2 y 3 y tripulación	Cualitativa
sex	Sexo masculino y femenino	Cualitativa
age	Niño y adulto	Cualitativa
survival	Sí/No sobrevivió	Cualitativa

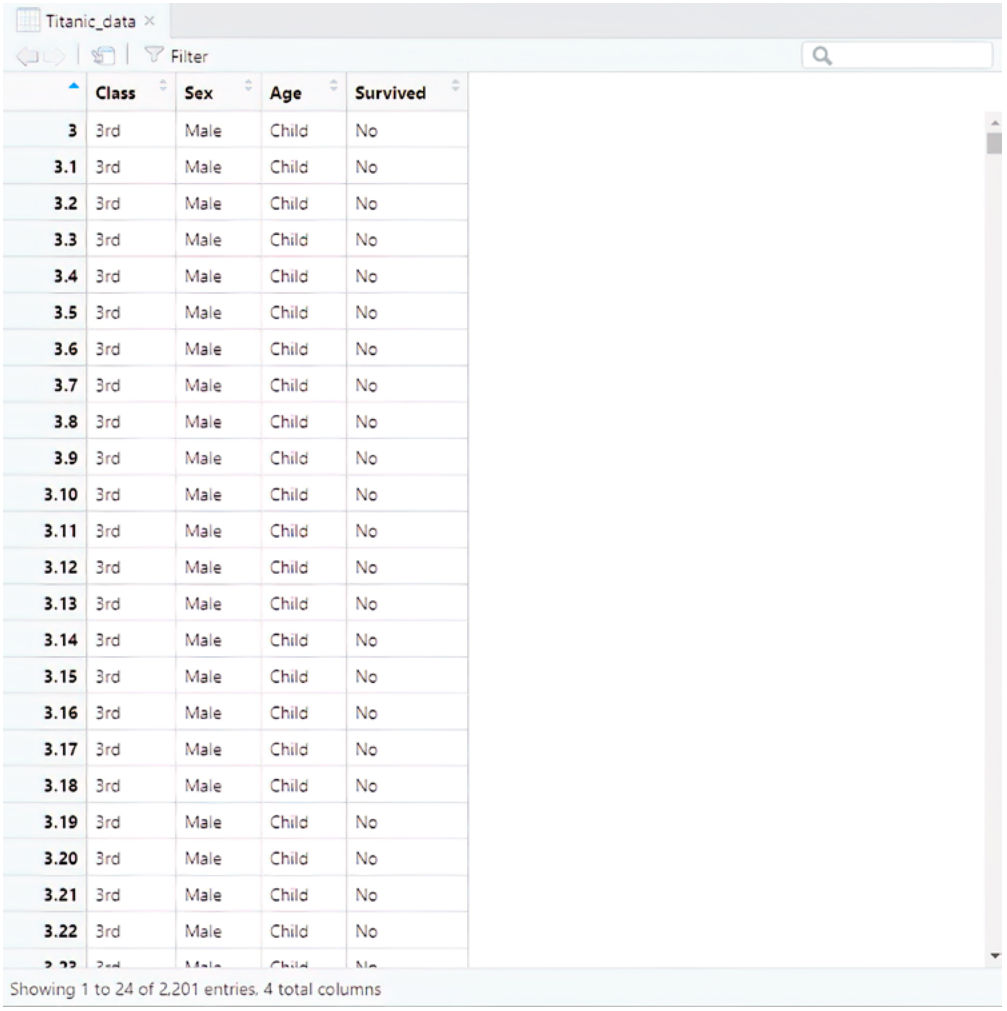
En principio, para poder visualizar los datos, convertimos **Titanic** en un data frame y lo guardamos en el objeto **Titanic_data**.

```
Titanic_data <- as.data.frame(Titanic)
```

Observamos que la base de datos de **Titanic** está agregada o combinada (tiene 32 filas), sin embargo, para convertir esa data en una tabla de contingencia se requiere que los datos estén desagregados o expandidos. Una forma sencilla de hacerlo es con la función `rep` que nos permite expandir los datos y convertirlos en observaciones o filas repetidas.

```
Titanic_data <- Titanic_data[rep(1:nrow(Titanic_data),
                                 Titanic_data[,5]),-5]
```

Observamos que contamos ahora con 2201 filas en nuestro marco de datos.


Titanic_data

Filter

	Class	Sex	Age	Survived
3	3rd	Male	Child	No
3.1	3rd	Male	Child	No
3.2	3rd	Male	Child	No
3.3	3rd	Male	Child	No
3.4	3rd	Male	Child	No
3.5	3rd	Male	Child	No
3.6	3rd	Male	Child	No
3.7	3rd	Male	Child	No
3.8	3rd	Male	Child	No
3.9	3rd	Male	Child	No
3.10	3rd	Male	Child	No
3.11	3rd	Male	Child	No
3.12	3rd	Male	Child	No
3.13	3rd	Male	Child	No
3.14	3rd	Male	Child	No
3.15	3rd	Male	Child	No
3.16	3rd	Male	Child	No
3.17	3rd	Male	Child	No
3.18	3rd	Male	Child	No
3.19	3rd	Male	Child	No
3.20	3rd	Male	Child	No
3.21	3rd	Male	Child	No
3.22	3rd	Male	Child	No

Showing 1 to 24 of 2,201 entries. 4 total columns

Sin embargo, nos quedaremos solo con las variables que nos interesan.

```
Titanic_data <- Titanic_data %>%
  dplyr::select(Sex, Survived )
```

Procederemos ahora sí a realizar nuestra tabla de 2x2 y mostraremos la asociación entre la supervivencia (variable de resultado) y el género (variable de exposición).

Con la función `table` creamos nuestra tabla de contingencia y la guardamos en el objeto **titanic_sex_superv**. Además, queremos obtener la suma de nuestros datos por columna, así que utilizamos la función `addmargins` y en el argumento `margin` colocamos el equivalente "1" para especificar la suma de columnas y "2" si quisiéramos especificar solo la suma de filas. Si omitimos dicho argumento nos arrojaría de manera predeterminada tanto la suma de columnas como de filas.

```
sex <- Titanic_data$Sex # a la fila
superv <- Titanic_data$Survived # a la columna

titanic_sex_superv <- table(sex,superv)
titanic_sex_superv
##         superv
## sex        No  Yes
##   Male   1364  367
##   Female  126  344
### suma por columna
addmargins(titanic_sex_superv,margin = 1)
##         superv
## sex        No  Yes
##   Male   1364  367
##   Female  126  344
##   Sum    1490  711
```

Lo reordenamos para que aparezca en la primera columna y fila los expuestos y resultados positivos, tal como el formato de tabla mostrado al inicio de esta sección.

```
titanic_sex_superv <- titanic_sex_superv[2:1,2:1];
titanic_sex_superv
##         superv
## sex       Yes   No
##   Female  344  126
##   Male    367 1364
```

A partir de la tabla de contingencia creada y ordenada correctamente, calculamos las medidas de asociación con la función `epi.2by2` del paquete `epiR`. Esta función nos muestra, particularmente, la razón de riesgo, la razón de probabilidades, el riesgo atribuible en los expuestos, la fracción atribuible en los expuestos, el riesgo atribuible en la población y la fracción atribuible en la población.

Especificamos en el argumento de la función el método `cross.sectional` ya que estamos tratando con datos de un estudio transversal. Asimismo, asignamos al argumento `conf.level 0.95` que representa el nivel de confianza. El argumento `units` representa las unidades de medida de resultado. Por último, en el argumento `outcome` especificamos `as.columns`, con esto referimos que la variable de resultado o desenlace está ubicada como columna en la tabla de contingencia.

```
epi.2by2(dat = titanic_sex_superv,
         method = "cross.sectional",
         conf.level = 0.95,
         units = 100, outcome = "as.columns")
##               Outcome +    Outcome -      Total
Prevalence *        Odds
## Exposed +           344          126        470
73.2       2.730
## Exposed -           367         1364       1731
21.2       0.269
## Total               711         1490       2201
32.3       0.477
##
## Point estimates and 95% CIs:
## -------------------------------------------------------------
------
## Prevalence ratio                                3.45 (3.10,
3.84)
## Odds ratio                                      10.15 (8.03,
12.83)
## Attrib prevalence in the exposed *              51.99 (47.55,
56.43)
```

```
## Attrib fraction in the exposed (%)            71.03 (67.79,
73.95)
## Attrib prevalence in the population *          11.10 (8.36,
13.84)
## Attrib fraction in the population (%)         34.37 (30.43,
38.08)
## ------------------------------------------------------------
------
## Uncorrected chi2 test that OR = 1: chi2(1) = 456.874 Pr>chi2
= <0.001
## Fisher exact test that OR = 1: Pr>chi2 = <0.001
##  Wald confidence limits
##  CI: confidence interval
##  * Outcomes per 100 population units
```

La prevalencia de sobrevivencia en mujeres expuestas fue 3.45 (95% IC 3.1 - 3.84) veces mayor que la prevalencia de sobrevivencia en los hombres. Además, la probabilidad de ser mujer y sobrevivir fue 10.15 (95% IC 8.03 - 12.83). El 43.6% de casos de sobrevivientes mujeres fue atribuido al hecho de ser mujer (95% IC 42% - 45.3%). Se puede conocer la interpretación correcta de los demás métodos de asociación estableciendo `interpret=TRUE` en el argumento:

```
epi.2by2(dat = titanic_sex_superv,
         method = "cross.sectional",
         interpret = TRUE #muestra la interpretación de los
resultados
         , conf.level = 0.95,
         units = 100,
         outcome = "as.columns")
##              Outcome +    Outcome -      Total
Prevalence *        Odds
## Exposed +          344          126        470
73.2       2.730
```

```
## Exposed -          367         1364       1731
21.2       0.269
## Total              711         1490       2201
32.3       0.477
##
## Point estimates and 95% CIs:
## -------------------------------------------------------------
------
## Prevalence ratio                                  3.45 (3.10,
3.84)
## Odds ratio                                        10.15 (8.03,
12.83)
## Attrib prevalence in the exposed *                51.99 (47.55,
56.43)
## Attrib fraction in the exposed (%)               71.03 (67.79,
73.95)
## Attrib prevalence in the population *             11.10 (8.36,
13.84)
## Attrib fraction in the population (%)            34.37 (30.43,
38.08)
## -------------------------------------------------------------
------
## Uncorrected chi2 test that OR = 1: chi2(1) = 456.874 Pr>chi2
= <0.001
## Fisher exact test that OR = 1: Pr>chi2 = <0.001
##  Wald confidence limits
##  CI: confidence interval
##  * Outcomes per 100 population units
##
##  Measures of association strength:
##  The outcome prevalence among the exposed was 3.45 (95% CI
3.1 to 3.84) times greater than the outcome prevalence among the
unexposed.
##
```

```
##  The outcome odds among the exposed was 10.15 (95% CI 8.03 to
12.83) times greater than the outcome odds among the unexposed.
##
##  Measures of effect in the exposed:
##  Exposure changed the outcome prevalence in the exposed by
246.11 (95% CI 217.13 to 275.09) per 100 population units. 90.1%
of outcomes in the exposed were attributable to exposure (95% CI
88.5% to 91.5%).
##
##  Number needed to treat for benefit (NNTB) and harm (NNTH):
##  The number needed to treat for one subject to benefit (NNTB)
is 2 (95% CI 2 to 2).
##
##  Measures of effect in the population:
##  Exposure changed the outcome prevalence in the population by
20.81 (95% CI 16.35 to 25.27) per 100 population units. 43.6% of
outcomes in the population were attributable to exposure (95% CI
42% to 45.3%).
```

◊ *Estudio de cohortes*

Razón de probabilidades

Posibilidades
OD+ = a/c
OD- = b/d
Total= (a+b) / (c+d)

Diferencia de probabilidades

```
a/c - b/d
```

Riesgo atribuible

La diferencia entre el riesgo de exposición positivo y el riesgo de exposición negativo da como resultado el riesgo atribuible.

```
RA = (a/(a+b)) - (c/(c+d))
```

Riesgo relativo

También conocido como razón de riesgo, es la relación entre la probabilidad de que ocurra un determinado evento en un grupo en comparación con el riesgo de que ocurra el mismo evento en otro grupo.

```
RR= (a/(a+b))  / (c /(c+d))
```

Estudio de caso

Para conocer los factores de riesgo asociados con el bajo peso al nacer del lactante, D. W. Hosmer y S. Lemeshow (1989) recopilaron datos del Baystate Medical Center, Springfield, Massachusetts, durante el año 1986. Cada fila de este conjunto de datos representa los datos de una madre. Nos interesa conocer la asociación entre el estado de fumadora o no de la madre durante el embarazo (variable "smoke") y el parto de un bebé de menos de 2.5 kg de peso corpora l(variable "low").

La base de datos libre `birthwt` que se encuentra en el paquete `MASS`, consta de 189 filas y 10 variables, las cuales se presentan y describen a continuación:

Variable	Descripción	Tipo de variable
low	Indicador de peso al nacer inferior a 2.5 kg	Cuantitativa
age	Edad de la madre en años	Cuantitativa
lwt	Peso de la madre en libras en el último período menstrual	Cuantitativo
race	raza de la madre (1 = blanca, 2 = negra, 3 = otra).	Cuantitativa
smoke	Tabaquismo durante el embarazo	Cuantitativa

Variable	Descripción	Tipo de variable
ptl	Número de partos prematuros previos	Cuantitativa
ht	Antecedentes de hipertensión	Cuantitativa
ui	Presencia de irritabilidad uterina	Cuantitativa
ftv	Número de visitas al médico durante el primer trimestre	Cuantitativo
bwt	Peso al nacer en gramos	Cuantitativa

Para visualizar los datos los convertimos en un data frame y les asignamos un nuevo nombre.

```
birthwt_data <- as.data.frame(birthwt)
```

Seleccionamos las variables de nuestro interés:

```
birthwt_data <- birthwt_data %>% dplyr::select(low, smoke)
```

Ahora crearemos la tabla de 2x2. Es importante que dicha tabla tenga el orden correcto, es decir, que los resultados y exposiciones positivas aparezcan en la primera columna y primera fila respectivamente y los resultados y exposiciones negativas aparezcan en la segunda columna y fila respectivamente. Si ejecutamos la función `table` observamos que tiene un formato incorrecto:

```
smoke <- birthwt_data$smoke # a la fila
low <- birthwt_data$low # a la columna

birthwt_smoke_low <- table(smoke,low)

### suma por columna
addmargins(birthwt_smoke_low,margin = 1)
##       low
## smoke   0   1
##   0    86  29
```

```
##   1    44  30
##   Sum 130  59
```

Para que nos aparezca de forma correcta, reordenamos las filas y columnas.

```
birthwt_smoke_low <- birthwt_smoke_low[2:1,2:1]; birthwt_smoke_
low
##      low
## smoke  1  0
##     1 30 44
##     0 29 86
```

Ahora sí, calculamos la razón de probabilidades para fumar y dar a luz a un bebé con bajo peso al nacer. Para los estudios de cohortes solo cambiamos el método y agregamos entre comillas `cohort.count`.

```
epi.2by2(dat = birthwt_smoke_low, method = "cohort.count",
   conf.level = 0.95, units = 100, interpret = TRUE,
   #agregamoms TRUE para que nos brinde la interpretación
   outcome = "as.columns")
##              Outcome +    Outcome -      Total        Inc risk
*        Odds
## Exposed +           30           44         74
40.5       0.682
## Exposed -           29           86        115
25.2       0.337
## Total               59          130        189
31.2       0.454
##
## Point estimates and 95% CIs:
## -------------------------------------------------------------
------
## Inc risk ratio                                 1.61 (1.06,
```

```
2.44)
## Odds ratio                                    2.02 (1.08,
3.78)
## Attrib risk in the exposed *                  15.32 (1.61,
29.04)
## Attrib fraction in the exposed (%)           37.80 (5.47,
59.07)
## Attrib risk in the population *               6.00 (-4.33,
16.33)
## Attrib fraction in the population (%)        19.22 (-0.21,
34.88)
## ---------------------------------------------------------
------
## Uncorrected chi2 test that OR = 1: chi2(1) = 4.924 Pr>chi2 =
0.026
## Fisher exact test that OR = 1: Pr>chi2 = 0.036
##  Wald confidence limits
##  CI: confidence interval
##  * Outcomes per 100 population units
##
##  Measures of association strength:
##  The outcome risk among the exposed was 1.61 (95% CI 1.06 to
2.44) times greater than the outcome risk among the unexposed.
##
##  The outcome odds among the exposed was 2.02 (95% CI 1.08 to
3.78) times greater than the outcome odds among the unexposed.
##
##  Measures of effect in the exposed:
##  Exposure changed outcome risk in the exposed by 15.32 (95%
CI 1.61 to 29.04) per 100 population units. 37.8% of outcomes in
the exposed were attributable to exposure (95% CI 5.5% to 59.1%).
##
##  Number needed to treat for benefit (NNTB) and harm (NNTH):
```

```
##  The number needed to treat for one subject to benefit (NNTB)
is 7 (95% CI 3 to 62).
##
##  Measures of effect in the population:
##  Exposure changed outcome risk in the population by 6 (95% CI
-4.33 to 16.33) per 100 population units. 19.2% of outcomes in
the population were attributable to exposure (95% CI -0.2% to
34.9%).
```

La probabilidad de tener un hijo con bajo peso al nacer para las mujeres fumadoras es 2.02 (95% CI 1.08 - 3.78) veces mayor que la probabilidad de tener un hijo con bajo peso al nacer para las embarazadas no fumadoras.

Resumen del capítulo

Se debe tener en cuenta que el análisis exploratorio de datos es un camino transitorio para el análisis final de los datos. Es una guía para el modelado posterior de sus datos y te informará directamente sobre las decisiones estadísticas. Calcular las medidas de frecuencia, de tendencia central, de dispersión y de asociación nos ha permitido presentar los datos de una forma más significativa en R, lo que permite una interpretación práctica y sencilla de los datos.

CAPÍTULO 5

ANÁLISIS BIVARIADO Y PRUEBA DE HIPÓTESIS

En el presente capítulo, realizaremos un análisis bivariado de un conjunto de datos con el fin de determinar el grado de asociación existente entre dos variables. Empezamos por comparar los diferentes métodos de análisis de correlación; seguidamente, se muestran las pruebas estadísticas para la comparación de dos medias independientes, dos muestras pareadas y más de dos muestras independientes. Este método de inferencia estadística nos permitirá decidir si los datos disponibles respaldan o sustentan de forma suficiente una hipótesis específica.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Comprender los diferentes métodos de correlación.
- Seleccionar la prueba de hipótesis correcta de acuerdo con tu pregunta de investigación.
- Conocer las diferentes pruebas de hipótesis y sus criterios de aplicación.

Paquetes

Instalamos los siguientes paquetes:

```
#install.packages("tidyverse")
#install.packages("infer")
#install.packages("broom")
#install.packages("knitr")
#install.packages("readxl")
#install.packages("cowplot")
#install.packages("nycflights13")
```

```
#install.packages("corrplot")
#install.packages("Hmisc")
#install.packages("psych")
#install.packages("skimr")
#install.packages("rlang")
#install.packages("estimatr")
#install.packages("margins")
#install.packages("stargazer")
#install.packages("lmtest")
#install.packages("foreign")
#install.packages("GGally")
#install.packages("AICcmodavg")
#install.package("gridExtra")
#install.packages("readxl")

library(tidyverse)
## -- Attaching packages ---------------------------------------
tidyverse 1.3.1 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## Warning: package 'readr' was built under R version 4.1.3
## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
library(infer)
library(broom)
library(knitr)
library(readxl)
## Warning: package 'readxl' was built under R version 4.1.3
library(cowplot)
library(nycflights13)
```

```
library(dplyr)
library(stringr)
library(infer)
library(datasets)
library(corrplot)
## Warning: package 'corrplot' was built under R version 4.1.3
## corrplot 0.92 loaded
library(Hmisc)
## Warning: package 'Hmisc' was built under R version 4.1.3
## Loading required package: lattice
## Loading required package: survival
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
## The following objects are masked from 'package:base':
##
##     format.pval, units
library(psych)
##
## Attaching package: 'psych'
## The following object is masked from 'package:Hmisc':
##
##     describe
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
library(skimr)
library(rlang)
## Warning: package 'rlang' was built under R version 4.1.3
##
## Attaching package: 'rlang'
```

```
## The following objects are masked from 'package:purrr':
##
##     %@%, as_function, flatten, flatten_chr, flatten_dbl, flatten_
int,
##     flatten_lgl, flatten_raw, invoke, splice
library(estimatr)
library(margins)
library(stargazer)
##
## Please cite as:
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression
and Summary Statistics Tables.
##  R package version 5.2.2. https://CRAN.R-project.org/
package=stargazer
library(lmtest)
## Loading required package: zoo
## Warning: package 'zoo' was built under R version 4.1.3
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
library(ggpubr)
## Warning: package 'ggpubr' was built under R version 4.1.3
##
## Attaching package: 'ggpubr'
## The following object is masked from 'package:cowplot':
##
##     get_legend
library(foreign)
library(GGally)
## Warning: package 'GGally' was built under R version 4.1.3
## Registered S3 method overwritten by 'GGally':
##   method from
```

```
##   +.gg   ggplot2
library(AICcmodavg)
## Warning: package 'AICcmodavg' was built under R version 4.1.3
library(gridExtra)
## Warning: package 'gridExtra' was built under R version 4.1.3
##
## Attaching package: 'gridExtra'
## The following object is masked from 'package:dplyr':
##
##     combine
```

Estudio de caso

Uno de los procesos quirúrgicos más practicados es el de las hernias inguinales, para el cual la técnica de Lichstenstein es la más empleada. No obstante, esta técnica genera complicaciones que pueden deberse a la inflamación producida por la sobremanipulación de uno de los tres nervios que atraviesan el conducto inguinal. En tal sentido, parece existir la necesidad de emplear otra técnica, que disminuya dichas complicaciones, como la técnica Nyhus.

Cargar data

```
granda <- read.csv("Granda.csv")

is.data.frame(granda) #Verificando si se ha cargado bien
## [1] TRUE
```

La función `read.csv` en R se usa para leer y cargar datos de un archivo en formato CSV (Comma Separated Values) en un marco de datos (*data frame*) de R, lo que facilita su posterior manipulación y análisis. En un archivo CSV, los datos se organizan en filas y columnas y cada celda en la tabla está separada por una coma o algún otro caracter separador.

Asimismo, la función `is.data.frame()` en R se utiliza para verificar si un objeto es un marco de datos de R o no, antes de realizar ciertas operaciones o funciones que solo pueden aplicarse a un marco de datos.

Visualización de la base de datos

La base de datos está compuesta por 30 variables que muestran la información de pacientes sometidos a cirugías para tratar sus hernias inguinales. La función `glimpse()` de la librería `dplyr` nos proporciona una visión general concisa y detallada de un *data frame*, favorable especialmente cuando se trabaja con conjuntos de datos grandes y complejos. De este modo, nos permite visualizar una lista de las columnas con información detallada sobre cada una de ellas, como su nombre, tipo de dato y una vista previa de los valores que contiene.

```
glimpse(granda) #Verificando si se ha cargado bien
## Rows: 202
Columns: 30
$ id          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16…
$ edad        <int> 81, 32, 25, 28, 41, 36, 39, 60, 64, 80, 40,
50, 60, 6…
$ sexo        <chr> “femenino”, “masculino”, “masculino”,
“masculino”, “f…
$ peso        <int> 68, 70, 68, 65, 68, 92, 76, 69, 66, 76, 90,
68, 62, 6…
$ talla       <dbl> 1.72, 1.69, 1.71, 1.65, 1.71, 1.72, 1.76,
1.74, 1.66,…
$ imc         <dbl> 25.1, 26.5, 25.3, 25.9, 25.4, 33.1, 26.5,
24.9, 26.0,…
$ hernia      <chr> “Izquieda”, “Izquieda”, “Izquieda”,
“Izquieda”, “Dere…
$ nihus       <chr> “Tipo III-A”, “Tipo II”, “Tipo II”, “Tipo
III-B”, “Ti…
$ t_enf       <int> 24, 12, 19, 24, 36, 20, 12, 24, 12, 24, 12,
```

```
6, 12, 24…
$ tecnica    <chr> “0”, “0”, “0”, “0”, “0”, “0”, “0”, “0”, “0”,
“0”, “0”…
$ t_qx       <int> 60, 40, 60, 55, 60, 50, 60, 80, 90, 40, 85,
40, 60, 4…
$ t_qx_2     <int> 70, 50, 70, 65, 70, 60, 70, 90, 100, 50, 95,
50, 70, …
$ anestesia  <chr> “Regional”, “Regional”, “Regional”,
“Regional”, “Regi…
$ t_hosp     <int> 6, 3, 3, 5, 4, 5, 4, 5, 5, 5, 4, 3, 4, 3, 4,
2, 3, 4,…
$ c_temp_d   <chr> “Negativo”, “Negativo”, “Negativo”,
“Negativo”, “Nega…
$ c_tardia_d <chr> “Negativo”, “Negativo”, “Negativo”,
“Negativo”, “Nega…
$ f_qx       <chr> “2/8/13”, “8/19/13”, “9/23/13”, “9/25/13”,
“4/1/12”, …
$ c_introp   <chr> “negativo”, “positivo”, “positivo”,
“positivo”, “nega…
$ c_temp     <chr> “negativo”, “negativo”, “negativo”,
“negativo”, “nega…
$ c_tardia   <chr> “negativo”, “negativo”, “negativo”,
“negativo”, “nega…
$ c_introp_d <chr> “Negativo”, “Lesion de vasos sanguiÂneos”,
“Lesion d…
$ complic    <chr> “negativo”, “negativo”, “negativo”,
“negativo”, “nega…
$ r_complic  <chr> “negativo”, “negativo”, “negativo”,
“negativo”, “nega…
$ seroma     <chr> “negativo”, “negativo”, “negativo”,
“negativo”, “nega…
$ neuralgia  <chr> “negativo”, “negativo”, “negativo”,
“negativo”, “nega…
$ orquitis   <chr> “negativo”, “negativo”, “negativo”,
```

```
"negativo", "nega…
$ dolor      <chr> "negativo", "negativo", "negativo",
"negativo", "nega…
$ recurrencia<chr> "negativo", "negativo", "negativo",
"negativo", "nega…
$ obeso      <chr> "No Obeso", "No Obeso", "No Obeso", "No
Obeso", "No O…
$ anciano    <int> 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1,…
```

Manejo de datos previo

Configuración de la base de datos

Para que no tenga que ingresar el nombre en cada variable, configure la base de datos con el paquete `dplyr`. Este paquete contiene la función `attach` que nos permite acceder a los nombres de las variables sin la necesidad de llamarlos.

```
attach(granda)
```

A continuación, se muestra un ejemplo que ilustra cómo funciona `attach` y cómo hacer lo mismo sin usar esta función.

Supongamos que queremos calcular la media de la columna "edad" y la desviación estándar de la columna "peso" en el marco de datos **granda**. Podemos hacerlo de la siguiente manera sin usar `attach`:

```
# Calcular la media de la columna "edad"
media_edad <- mean(granda$edad)

# Calcular la desviación estándar de la columna "peso"
desv_peso <- sd(granda$peso)
```

En este ejemplo, para referenciar una columna específica del marco de datos **granda**, debemos escribir el nombre completo del marco de datos y la columna correspondiente utilizando el operador `$`.

Ahora, veamos cómo se haría lo mismo utilizando `attach`:

```
# Adjuntar el marco de datos "granda" al espacio de nombres
global
attach(granda)
# Calcular la media de la columna "edad", sin especificar el
nombre del marco de datos
media_edad <- mean(edad)

# Calcular la desviación estándar de la columna "peso", sin
especificar el nombre del marco de datos
desv_peso <- sd(peso)
```

Después de adjuntar el marco de datos **granda** al espacio de nombres global con `attach`, podemos referenciar las columnas directamente por su nombre sin tener que escribir el nombre completo del marco de datos y el operador `$`.

Es recomendable que, después de terminar de trabajar con el marco de datos, se desadjunte el espacio de nombres global con `detach` para evitar conflictos de nombres.

```
detach(granda)
```

Análisis de correlación

El análisis de correlación determina el grado de asociación de dos variables. Se denomina coeficiente de correlación y se representa con la letra *r*. El coeficiente puede variar de -1 a +1, donde -1 significa una fuerte relación negativa y +1 significa una fuerte relación positiva. La hipótesis nula en una prueba de correlación es una correlación de 0, lo que significa que no hay ninguna relación.

Como primer paso, compararemos los diferentes métodos de correlación.

Índice de correlación de Pearson

Es una de las correlaciones más utilizadas en estadística. Es una medida de la fuerza y la dirección de una relación lineal entre dos variables. Se basa en cuatro suposiciones clave:

1. Los datos son de intervalo o razón. Estos tipos de datos continuos son importantes para la forma en que la correlación supone que se relacionarán los valores de las variables y, por lo tanto, la codificación de variables ordinales o categóricas no funcionará.
2. Sus dos factores correlacionados tienen que aproximarse a una línea y no a una forma curva o parabólica. Se puede usar la correlación de Pearson para observar si existe una relación lineal entre los datos, pero existen otras pruebas adecuadas para analizar esas diferentes estructuras de datos.
3. Los valores atípicos en tus datos realmente pueden descartar una correlación de Pearson.
4. Los datos que vas a analizar deben tener una distribución normal. Una forma rápida de confirmar si la tienen es a través de histogramas.

Índice de correlación de Spearman

Se basa en casi todos los mismos supuestos que la correlación de Pearson, pero no se basa en la normalidad y sus datos también pueden ser ordinales. Por lo tanto, es una prueba no paramétrica.

Índice de correlación de Kendall

Es similar a la correlación de Spearman en el sentido de que no es paramétrica. Se puede utilizar con datos ordinales o continuos. Es un estadístico de dependencia entre dos variables.

Comparación de correlaciones

```
cor(granda$peso,granda$talla,
    method = c("pearson"))
## [1] 0.7066369
cor(granda$peso, granda$talla,
  method = c("spearman"))
## [1] 0.7016775
cor(granda$peso,granda$talla,
    method = c("kendall"))
## [1] 0.5811741
```

Vemos que los resultados obtenidos con los índices de correlación de Pearson y Spearman son más o menos iguales, pero el obtenido con el índice de correlación de Kendall es muy diferente. Esto se debe a que el índice de Kendall es una prueba de fuerza de dependencia, mientras que los de Pearson y Spearman son casi equivalentes en la forma en que correlacionan los datos normalmente distribuidos. Todas estas correlaciones son correctas en sus resultados, solo que las de Pearson y Spearman están mirando los datos de una manera y la de Kendall de otra.

La correlación de Pearson se utiliza para medir la fuerza y la dirección de la relación lineal entre dos variables cuantitativas continuas. Por ejemplo, en este caso, la correlación de Pearson se está utilizando para medir la relación entre el peso y la altura de un grupo de pacientes. Si las dos variables están fuertemente relacionadas linealmente, la correlación de Pearson será alta.

A diferencia de la correlación de Pearson, la correlación de Spearman se utiliza para medir la relación entre dos variables cualitativas politómicas, ordinales o variables no paramétricas, es decir, no asume que las variables tienen una distribución normal o siguen una relación lineal. Por ejemplo, se podría utilizar la correlación de Spearman para medir la relación entre el nivel de satisfacción laboral en una escala ordinal y la antigüedad de los empleados en una empresa. Si las dos variables están fuertemente relacionadas, la correlación de Spearman será alta, cerca de 1 o -1.

Finalmente, la diferencia con la correlación de Kendall es que mide la asociación entre dos variables en términos de la concordancia y la discordancia entre los pares de observaciones. De ahí que este índice es una alternativa para analizar

correlaciones entre variables cualitativas dicotómicas o datos no paramétricos. Por ejemplo, se podría utilizar la correlación de Kendall para medir la relación entre las calificaciones dadas por dos jueces diferentes en un concurso de baile, donde las calificaciones son ordinales. Si los dos jueces están de acuerdo en las calificaciones, la correlación de Kendall será alta.

Una mejor situación para Spearman o Kendall (pero no para Pearson) es cuando los datos son ordinales, en el sentido de que están clasificados. Entonces, transformamos los datos de la variable "peso" en un rango.

```
#Creamos un rango para la variable "peso"
peso.Rank <-rank(granda$peso, na.last=NA,ties.method="first")
#Creamos un nuevo conjunto de datos con el rango de peso
#llamado "granda.Rank.1
granda.Rank.1<-cbind(peso.Rank=peso.Rank,granda)
#comprobamos
view(granda.Rank.1)
```

En el código proporcionado, la función `rank()` se aplica a la variable "granda$peso", que contiene los pesos de cada persona en el *data frame* **granda**. La opción `na.last=NA` indica que los valores perdidos se incluirán al final de la clasificación. Y la opción `ties.method="first"` indica que se debe utilizar el rango más bajo para los empates, lo que significa que se asigna el rango más bajo posible a los valores que tienen el mismo valor.

La función `cbind()` que proviene de la abreviatura de "column bind", toma uno o más vectores, matrices o *data frames* como argumentos y los une por columnas para formar una nueva matriz o *data frame*. La nueva matriz o *data frame* resultante denominada granda.Rank.1 tendrá el mismo número de filas que los objetos de entrada, la primera columna se llama "peso.Rank" y contiene los valores del objeto **peso.Rank**, y la segunda columna contiene todas la variables de **granda**.

Así que al aplicar `View(granda.Rank.1)` lo que R nos muestra es una nueva base de datos con 31 variables, debido a que se incluye la nueva variable llamada "peso. Rank" que contiene la posición o el rango asignado a cada valor de una variable numérica como "peso".

Revisemos las correlaciones nuevamente con los datos clasificados de la variable "peso" y los datos sin procesar de la variable "talla".

```
cor(granda.Rank.1$peso.Rank,granda.Rank.1$talla,
    method=c("pearson"))
## [1] 0.7264201
cor(granda.Rank.1$peso.Rank,granda.Rank.1$talla,
    method=c("spearman"))
## [1] 0.7136048
cor(granda.Rank.1$peso.Rank,granda.Rank.1$talla,
    method=c("kendall"))
## [1] 0.5670552
```

Nuevamente vemos que Pearson y Spearman son muy similares, aunque los tres han cambiado ligeramente. Esto probablemente se deba a la granularidad de una de las fuentes de datos que hemos modificado. Al usar `rank()`, se transformó la variable original "peso" en una variable ordinal o de rango, en la que los valores numéricos se sustituyeron por posiciones en la clasificación. En consecuencia, la información numérica original se pierde y solo se considera la relación de orden entre los valores.

De este modo, el uso de `rank` puede ser útil para reducir el impacto de valores extremos o de distribuciones no normales en los cálculos de correlación, lo cual explicaría el aumento en la correlación de Person y Spearman. Sin embargo, también puede ocultar patrones o relaciones no lineales en los datos y causar que disminuya la correlación por la pérdida de información.

```
#Transformemos la variable talla en un rango también,
#solo para ver cómo se ve:

talla.Rank <-rank(granda.Rank.1$talla, na.last=NA,ties.
method="first")
granda.Rank.2 <-cbind(talla.Rank=talla.Rank,granda.Rank.1)
#comprobamos
view(granda.Rank.2)
```

Volvemos a crear una nueva variable "talla.Rank" en la que cada observación se clasifica en función de su orden en la distribución de la variable original "talla". Asimismo, esta nueva variable ordinal se anexará con todas las variables de la base de datos **granda.Rank.1** en una nueva base de datos denominada **granda.Rank.2**, es decir, en total tendremos 32 variables.

Revisemos las correlaciones nuevamente con los datos modificados de las variables "peso" y "talla".

```
cor(granda.Rank.2$peso.Rank,granda.Rank.2$talla.Rank,
    method=c("pearson"))
## [1] 0.7181395
cor(granda.Rank.2$peso.Rank,granda.Rank.2$talla.Rank,
    method=c("spearman"))
## [1] 0.7181395
cor(granda.Rank.2$peso.Rank,granda.Rank.2$talla.Rank,
    method=c("kendall"))
## [1] 0.5711541
```

Vemos que Pearson coincide exactamente con Spearman, como era de esperar, ya que los números enteros ahora son enteros en todos los ámbitos, es decir, los datos ya no tienen un orden relativo ambiguo, ya que cada valor tiene un número entero asociado con él. Si bien estos datos son técnicamente ordinales, lo que realmente hemos hecho es una transformación de puntajes brutos a números enteros de rango. Deberíamos esperar que estos se correlacionen casi igual (o exactamente igual) que los puntajes brutos, ya que están inherentemente vinculados. En otras palabras, la transformación solo cambia la forma en que se representan los valores, pero no la relación que tienen entre sí.

Selección de la prueba estadística según la pregunta de investigación

Las pruebas estadísticas permiten evaluar hipótesis con el fin de determinar si una variable independiente (también denominada *de exposición* o *predictora*) tiene una relación estadísticamente significativa con una variable dependiente (*de resultado*).

La hipótesis nula asume que no existe diferencia entre el valor observado y el valor esperado. Mientras que la hipótesis alternativa propone que sí hay una diferencia.

Las pruebas de hipótesis determinarán si se rechaza la hipótesis nula bajo un parámetro de confianza. Rechazamos la hipótesis nula cuando *p* es menor a 0.05.

Ambas hipótesis ofrecen respuestas opuestas a una pregunta de investigación determinada. Además, para cada pregunta de investigación se elige una hipótesis estadística en particular, tal como se muestra en la siguiente tabla.

Pregunta de investigación -> hipótesis estadística

Pregunta	Paramétrica	No paramétrica
Una muestra		
• Ho: μ = vs. Ha: μ	t de Student	Man Whitney
• Ho: = vs. Ha:	NA	Chi2 o exacta de Fisher
Dos muestras independientes		
• Ho: = vs. Ha:	t de Student	Man Whitney
• Ho: = vs. Ha:	NA	Chi2 o exacta de Fisher
Dos muestras pareadas		
• Ho: = vs. Ha:	t pareada	Wilcoxon
• Ho: = vs. Ha:	NA	MC Nemar
Más de dos muestras independientes		
• Ho: vs. Ha:	ANOVA	Kruskal Wallis
• Ho: vs. Ha:	NA	Chi2 o exacta de Fisher

Comparación de dos muestras independientes

Es un supuesto en el cual se contrasta si existe una diferencia significativa en la media de una variable de resultado entre dos poblaciones diferentes e independientes.

T de Student para comparar medias independientes

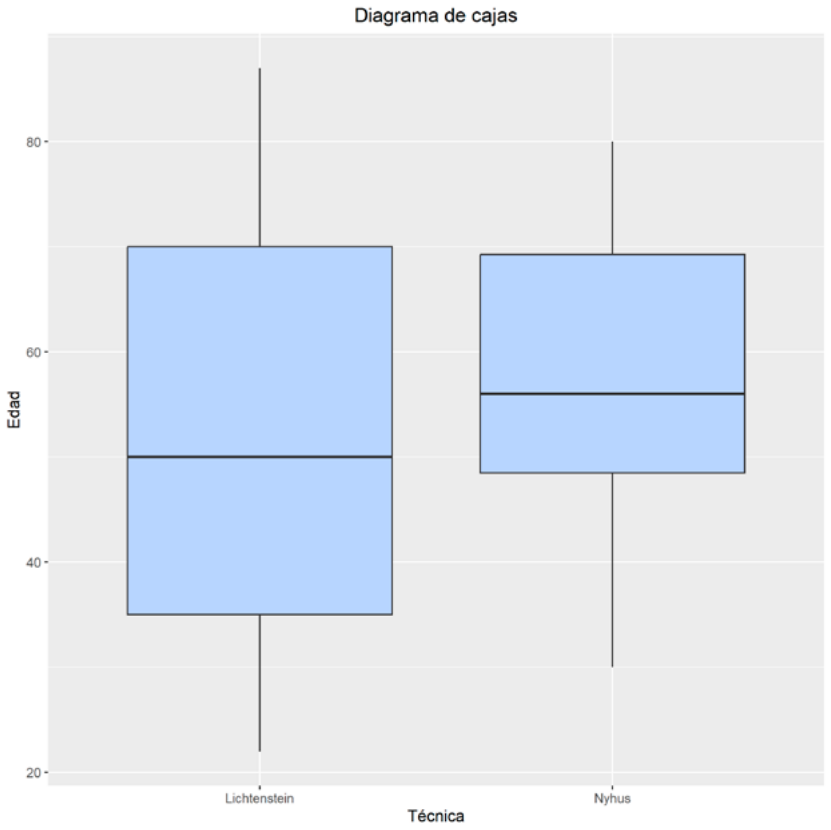
Para utilizar esta prueba es necesario cumplir 4 criterios: la muestra debe ser aleatoria, los grupos deben mostrar independencia entre las observaciones, la distribución debe ser normal o n > 30 para cada grupo y las varianzas de los grupos deben ser iguales.

La hipótesis nula para esta prueba es que la diferencia entre las medias o medianas de los grupos, respectivamente, es igual a 0 (μ2 = μ1), mientras que la hipótesis alterna indica lo contrario (μ2 ≠ μ1).

```
t.test(granda$edad ~ granda$tecnica) ### técnica vs edad
##
##  Welch Two Sample t-test
##
## data:  granda$edad by granda$tecnica
## t = -1.3911, df = 29.278, p-value = 0.1747
## alternative hypothesis: true difference in means between group
0 and group Nyhus is not equal to 0
## 95 percent confidence interval:
##  -11.770840   2.238517
## sample estimates:
##     mean in group 0 mean in group Nyhus
##            51.96111            56.72727
t.test(edad, mu = 50) ### Edad media observada vs. esperada
##
##  One Sample t-test
##
## data:  edad
## t = 1.9823, df = 201, p-value = 0.04881
## alternative hypothesis: true mean is not equal to 50
## 95 percent confidence interval:
##  50.01304 54.94736
## sample estimates:
## mean of x
##   52.4802
```

Notamos que no podemos rechazar la hipótesis nula al ser el valor *p* superior a nuestra alfa (valor *p* de 0.05 o un intervalo de confianza del 95%), por lo tanto, concluimos que tenemos pruebas estadísticas significativas para poder afirmar que las medias son iguales en las variables "técnica" y "edad". Podemos presentar estos resultados de manera gráfica mediante un diagrama de cajas:

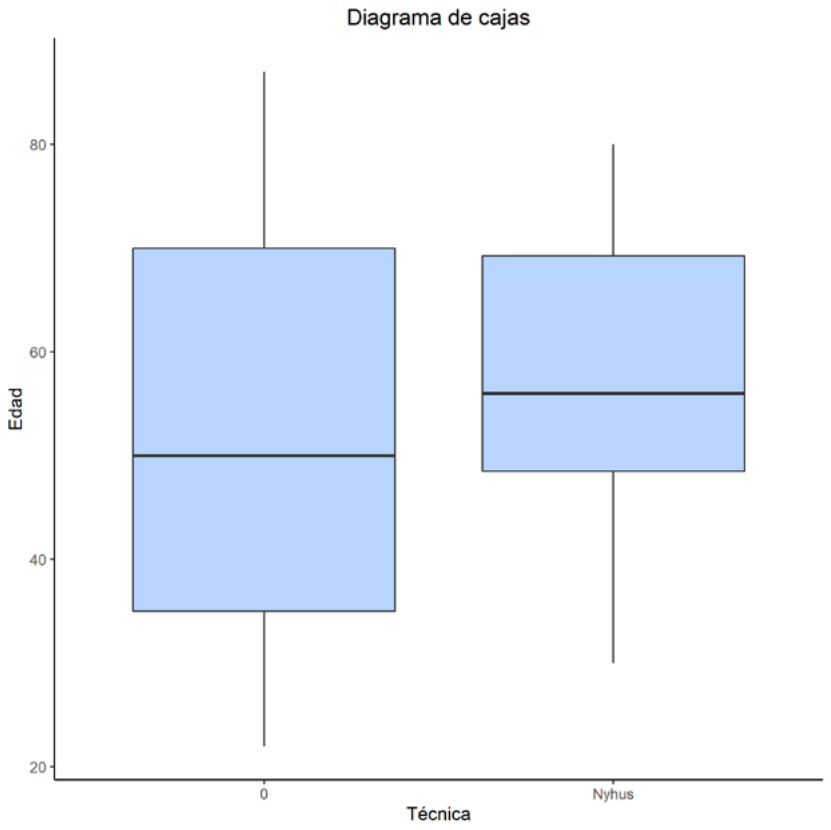
```
ggplot(granda, aes(x=tecnica, y=edad)) +
  geom_boxplot(fill='#BFD4FF') +
  labs (title="Diagrama de cajas", y= "Edad", x="Técnica")  +
  theme(plot.title = element_text(hjust = 0.5))
```



Para establecer el tema de un gráfico diferente a lo creado con `ggplot` se utiliza `theme_classic()`, una función de la librería `ggplot2` de R que permite eliminar algunos elementos estéticos del gráfico, como el fondo gris claro y las líneas de

cuadrícula. Asimismo, establece un estilo simple y minimalista que destaca los elementos más importantes del gráfico, como los datos y las etiquetas.

```
ggplot(granda, aes(x=tecnica, y=edad)) +
  geom_boxplot(fill='#BFD4FF') +
  theme_classic() +
  labs (title="Diagrama de cajas", y= "Edad", x="Técnica")  +
  theme(plot.title = element_text(hjust = 0.5))
```



Mann Whitney para comparación de medianas independientes

Para utilizar esta prueba es necesario tener en cuenta 4 supuestos: la muestra debe ser aleatoria, los grupos deben mostrar independencia entre las observaciones, la distribución debe ser normal o n > 30 para cada grupo y las varianzas de los grupos deben ser iguales.

La hipótesis nula para esta prueba es que la diferencia entre las medias o medianas de los grupos, respectivamente, es igual a 0 (μ2 = μ1), mientras que la hipótesis alterna indica lo contrario (μ2 ≠ μ1).

```
wilcox.test(granda$t_hosp ~ granda$r_complic) #Mediana de tiempo
de
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  granda$t_hosp by granda$r_complic
## W = 2034.5, p-value = 0.001561
## alternative hypothesis: true location shift is not equal to 0
#hospitalizacion x complicaciones

wilcox.test(edad, mu = 50) # Edad mediana observada vs. esperada
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  edad
## V = 9503.5, p-value = 0.05203
## alternative hypothesis: true location is not equal to 50
```

Se rechaza la hipótesis nula y se concluye que las distribuciones de la variable "tiempo de hospitalización (t_hosp)" y de la variable "complicaciones (r_complic)" son diferentes. Sin embargo, para la variable "edad", de media 50, no se rechaza la hipótesis nula.

Chi cuadrado para comparación de proporciones independientes

Es recomendable para variables nominales, categóricas o cuando deseamos trabajar con la frecuencia.

```
#SEXO vs. ANESTESIA
chisq.test(granda$sexo,granda$anestesia)
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  granda$sexo and granda$anestesia
## X-squared = 0.94005, df = 1, p-value = 0.3323
#SEXO vs. TECNICA
chisq.test(granda$sexo, granda$tecnica)
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  granda$sexo and granda$tecnica
## X-squared = 0.012115, df = 1, p-value = 0.9124
```

No se rechaza la hipótesis nula al ser el valor *p* superior a alfa, lo cual significa que las variables son independientes y no hay relación entre las dos variables categóricas.

Exacta de Fischer para comparación de proporciones independientes

Es recomendable para analizar tablas de contingencia con muestras pequeñas y frecuencias menores de 5.

```
#t_hosp vs. anciano
## Con este código podemos obtener el valor de Chi cuadrado
##para muestras pequeñas (frecuencias menores a 5)
simulacion <- granda %>% select(t_hosp, anciano) %>% head(n=20)
#reducimos la muestra
```

```
fisher.test(simulacion, simulate.p.value = TRUE)
##
##  Fisher's Exact Test for Count Data with simulated p-value
(based on
##  2000 replicates)
##
## data:  simulacion
## p-value = 0.989
## alternative hypothesis: two.sided
```

El código selecciona dos variables de interés "t_hosp" y "anciano" del marco de datos **granda** utilizando la función `select`. Luego, se reduce la muestra a las primeras 20 filas utilizando la función `head`. Esta muestra reducida se almacena en una nueva variable llamada "simulacion". El comando `head` es una función para mostrar las primeras n filas de un objeto, marco de datos o vector. El número de filas que se muestra se especifica mediante el argumento "n". En este caso, queremos mostrar las primeras 20 filas de la data seleccionada para reducirla, debido a que la prueba exacta de Fisher es considerada más precisa que otras pruebas estadísticas y permite calcular los valores p exactos, incluso en situaciones con muestras pequeñas y frecuencias menores que 5.

No se rechaza la hipótesis nula al ser *p* mayor a 0.05. Esto significa que la variable de "tiempo de hospitalización (t_hosp)" y "anciano" son independientes y por ende no existe una relación entre ambas.

Comparación de dos muestras pareadas

Normalmente se usa en investigaciones en las que se mide la variable dependiente antes y después de cambiar la variable independiente.

Por ejemplo, si se está investigando el efecto de una intervención médica en la presión arterial de los pacientes, la variable dependiente sería la presión arterial, que se mediría en una muestra de pacientes antes y después de la intervención. La variable independiente, en este caso, sería la intervención médica en sí misma. La hipótesis de investigación podría ser que al haber una intervención médica se reduce la

presión arterial en los pacientes. La prueba estadística utilizada para comparar las dos muestras pareadas se centraría en evaluar si hay una diferencia significativa entre las medidas de presión arterial antes y después de la intervención para evaluar la hipótesis de investigación.

T de Student para comparar medias pareadas

Para utilizar esta prueba es necesario tener en cuenta tres supuestos: la muestra debe ser aleatoria, las muestras deben ser pareadas y la distribución debe ser normal o haber un n > 30 en la diferencia de las muestras.

Se utiliza la prueba `T-test` agregando `paired=TRUE`; este parámetro adicional especifica que son datos pareados.

```
t.test(granda$peso, granda$t_enf, paired = TRUE)
##
##  Paired t-test
##
## data:  granda$peso and granda$t_enf
## t = 22.135, df = 201, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal
to 0
## 95 percent confidence interval:
##  33.59118 40.16130
## sample estimates:
## mean of the differences
##                36.87624
```

El valor *p* es inferior a alfa, por ende, es estadísticamente significativo y se rechaza la hipótesis nula.

Wilcoxon para la comparación de medianas pareadas

Para utilizar esta prueba es necesario tener en cuenta tres supuestos: las muestras deben ser aleatorias, la variable a contrastar debe ser continua y estar en, al menos,

una escala de intervalo tal que los valores se puedan restar, y la distribución de la diferencia debe ser simétrica alrededor de la media.

Como en el caso anterior, el parámetro adicional especifica que son datos pareados.

```
wilcox.test(granda$peso,granda$t_enf, paired = TRUE)
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  granda$peso and granda$t_enf
## V = 18524, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

El valor *p* es inferior a alfa (el nivel de significancia), por ende, podemos concluir que sí hay una diferencia significativa entre los dos conjuntos de datos pareados.

McNemar para la comparación de proporciones pareadas

Este test analiza si la probabilidad de un evento para una variable es igual en los dos niveles de otra variable. Se utiliza para variables dicotómicas. En este caso, la variable "sexo" y la variable "obeso" son variables dicotómicas que se evaluarán a través del test de McNemar para determinar si hay una diferencia significativa entre dos medidas repetidas en una muestra de individuos.

```
mcnemar.test(granda$sexo, granda$obeso)
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  granda$sexo and granda$obeso
## McNemar's chi-squared = 113.39, df = 1, p-value < 2.2e-16
```

La hipótesis nula es que no hay diferencia en la proporción de individuos que cambian de categoría (en este caso, ser obeso o no ser obeso) entre los niveles de la

variable "sexo". El valor *p* es inferior a alfa, lo quiere decir que se rechaza la hipótesis nula y se concluye que el sexo sí influye en la obesidad, por lo que la proporción de las personas de género masculino y femenino es diferente a los sujetos que pasan de ser a no ser obesos. Se puede interpretar que la proporción de personas de sexo masculino que pasan de ser obesas a no serlo es diferente de la proporción de personas de sexo femenino que pasan de ser obesas a no serlo.

Comparación de más de dos muestras independientes

ANOVA para la comparación de más de dos medias independientes

El análisis de varianza (ANOVA) se emplea para determinar si existen diferencias significativas entre las medias de dos o más poblaciones.

```
granda_anova <- aov(edad ~ sexo * tecnica,
                    data = granda)
summary(granda_anova)
##               Df Sum Sq Mean Sq F value Pr(>F)
## sexo           1    217   216.8   0.683  0.410
## tecnica        1    461   460.9   1.451  0.230
## sexo:tecnica   1      5     5.4   0.017  0.896
## Residuals    198  62879   317.6
```

El comando `aov()` se utiliza para realizar un análisis de varianza (ANOVA) en R de la variable continua "edad" en función de las variables explicativas "sexo" y "tecnica". El operador `*` indica que se desea incluir la interacción entre ambas variables explicativas. Esto significa que se desea evaluar si el efecto de la variable "sexo" sobre la variable "edad" varía en función de la variable "tecnica" y viceversa. Finalmente, con `summary()` se está obteniendo un resumen de los resultados del análisis de varianza almacenados en el objeto **granda_anova**.

El valor *p* es superior a alfa, por ende, no rechazamos la hipótesis nula de igualdad y concluimos que no hemos observado diferencias significativas entre las medias de edad según sexo y técnica.

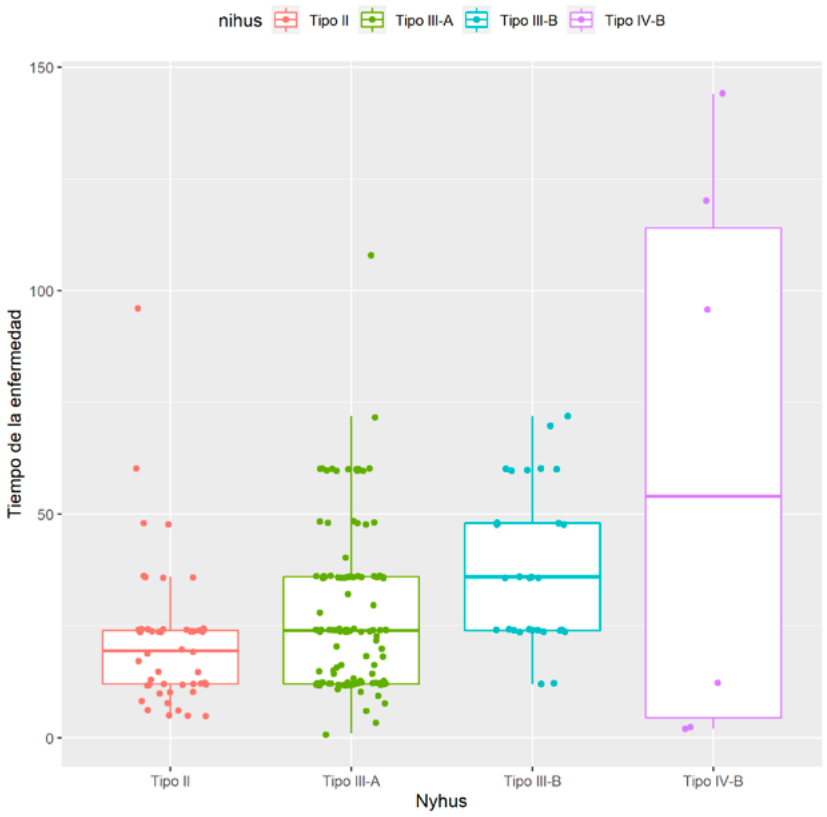
Kruskal Wallis para la comparación de más de dos medianas independientes

Emplea rangos para contrastar la hipótesis de que *k* muestras han sido obtenidas de una misma población.

A diferencia del ANOVA, en el que se comparan medias, el test de Kruskal-Wallis contrasta si las diferentes muestras están equidistribuidas y que, por lo tanto, pertenecen a una misma distribución (población). Bajo ciertas simplificaciones puede considerarse que el test de Kruskal-Wallis compara las medianas.

La hipótesis nula es que todas las muestras provienen de la misma población (distribución). La hipótesis alternativa es que al menos una muestra proviene de una población con una distribución distinta.

```
#Prueba de Kruskal-Wallis
kruskal.test(t_enf ~ nihus, data = granda)
##
##  Kruskal-Wallis rank sum test
##
## data:  t_enf by nihus
## Kruskal-Wallis chi-squared = 20.61, df = 3, p-value =
0.0001269
#diagrama de caja para verificar la distribucion
#de los datos
ggplot(granda, aes(x = nihus, y = t_enf, col = nihus)) +
geom_boxplot(outlier.shape = NA) + geom_jitter(width = 0.2)
+ theme(legend.position="top") + labs (y= "Tiempo de la
enfermedad", x="Nyhus")
```



Como el valor de p obtenido de la prueba de Kruskal-Wallis es significativo (20.61, $p < 0.05$), concluimos que existen diferencias significativas en el tiempo de la enfermedad entre los tipos de Nyhus debido a que se rechaza la hipótesis nula que establece que no hay diferencias significativas entre los tipos de Nyhus en términos de la variable de interés.

Sin embargo, esta conclusión no comprueba directamente la hipótesis alternativa que establece que al menos un tipo de Nyhus es significativamente diferente de los demás en términos del tiempo de la enfermedad, ya que la prueba de Kruskal-Wallis solo nos dice que hay diferencias significativas en el tiempo de la enfermedad entre los grupos, pero no nos dice específicamente qué grupos son diferentes entre sí.

Resumen del capítulo

Para analizar la asociación entre dos variables podemos optar por diferentes pruebas estadísticas. Como una primera alternativa, podemos analizar la asociación entre dos variables calculando su nivel de correlación. Entre las diferentes alternativas que existen, es importante conocer las fortalezas y limitaciones de las tres más utilizadas: Pearson, Spearman y Kendall. Si bien la correlación de Pearson es una de las más utilizadas, es importante conocer los otros dos métodos de correlación para seleccionar el que mejor se ajusta a nuestros datos. En segunda instancia, para analizar la magnitud de asociación entre dos variables podemos utilizar una prueba de hipótesis estadística. Entre las muchas descritas en la literatura, es altamente recomendable conocer las fortalezas y limitaciones de las pruebas de hipótesis más utilizadas. Estas pruebas incluyen las de chi cuadrado, exacta de Fisher, t de Student, ANOVA, Man Whitney, Wilcoxon, Mac Nemar y Kruskal Wallis. Y para seleccionar la prueba de hipótesis idónea para cada comparación, debemos conocer a priori la distribución de nuestras variables, si las muestras son independientes o no, así como el número de muestras de nuestra investigación. Finalmente, es importante reconocer que todo análisis bivariado, independientemente de si resulta significativo o no, debe ser complementado con un análisis multivariado a fin de controlar potenciales sesgos de confusión.

CAPÍTULO 6

ANÁLISIS DE REGRESIÓN

Este capítulo es exclusivamente teórico y será de gran utilidad para avanzar y comprender mejor los siguientes capítulos. En primer lugar, se explica qué es un análisis de regresión y la importancia de la regresión multivariable para controlar los sesgos de confusión. Luego, se presentan los modelos lineales generalizados y sus funciones de enlace y de soporte en R. Posteriormente, se presentan los análisis de regresión simple y múltiple y los métodos forward y stepwise. Por último, y como paso final de un análisis de regresión multivariado, se presentan supuestos post regresión.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Comprender la importancia del análisis de regresión en el control de sesgo de confusión.
- Conocer los modelos lineales generalizados y las funciones para su aplicación en R.
- Comprender los métodos de regresión multivariable y las diferentes formas de comparar los modelos.
- Conocer los principales supuestos para un análisis post regresión.

Análisis de regresión

El análisis de regresión es una herramienta estadística que busca predecir el comportamiento de la variable dependiente (denominada también *de resultado* o *desenlace*) en función de una o más variables independientes (denominadas también *predictoras* o *de exposición*) de un conjunto de datos. La forma en que se realiza un análisis de regresión dependerá del tipo de datos de la variable dependiente; por ejemplo, si la variable es continua (como la edad) se realiza una regresión lineal, pero si es una variable dependiente binaria (como "con educación secundaria" / "sin educación

secundaria") o de tiempo hasta el evento (como el tiempo hasta la muerte) se aplicarían una regresión logística y una regresión de Cox, respectivamente.

Es importante tener en cuenta que la regresión de Cox y la regresión lineal comparten algunas similitudes, ya que ambas se usan para variables continuas. Sin embargo, la principal diferencia es que la regresión de Cox se enfoca en modelar la tasa de riesgo o fracaso, es decir, un análisis de supervivencia, mientras que la regresión lineal se utiliza para variables continuas más generales. En la regresión de Cox, se estima la función de riesgo instantáneo, que indica cómo cambia la tasa de fracaso a lo largo del tiempo. En cambio, la regresión lineal se enfoca en estimar la relación entre la variable dependiente y las variables explicativas.

Sin embargo, realizar este tipo de análisis complejo podría desembocar en conclusiones falsas si no se lo lleva a cabo de manera correcta. Por ello, se deben tener en cuenta ciertos parámetros, como la elección adecuada de las variables explicativas, la inclusión de interacciones relevantes, la validación del modelo y la comprobación de supuestos clave. Realizar un análisis de regresión de manera incorrecta puede conducir a conclusiones falsas y a errores de interpretación. Por ejemplo, el sesgo de confusión puede surgir si no se controlan adecuadamente las variables que pueden influir en la relación entre la variable independiente y la dependiente. Por lo tanto, es importante realizar un análisis cuidadoso y riguroso para garantizar resultados precisos y confiables.

Análisis de regresión como alternativa para controlar el sesgo de confusión

El análisis de regresión puede ser útil para controlar el sesgo de confusión, que es uno de los problemas más frecuentes cuando hay una asociación irrelevante debido a la presencia de variables o factores (denominados *de confusión*) que hacen que los resultados no reflejen la relación real entre las variables que se estudian. Existen diferentes métodos para controlar o excluir las variables de confusión, como la aleatorización, la restricción y el pareamiento.

Sin embargo, estos métodos se aplican en el momento del diseño de estudio y antes de la recopilación de los datos; cuando esto ya no es posible, el modelo de regresión multivariado es una alternativa, pues permite ajustar los posibles efectos del sesgo de confusión durante la etapa del análisis. El análisis de regresión se

presenta, así, como una estrategia para controlar el sesgo de confusión al incluir el confusor en un modelo de múltiples variables a fin de analizar cómo varía la asociación de interés en presencia de todos los elementos.

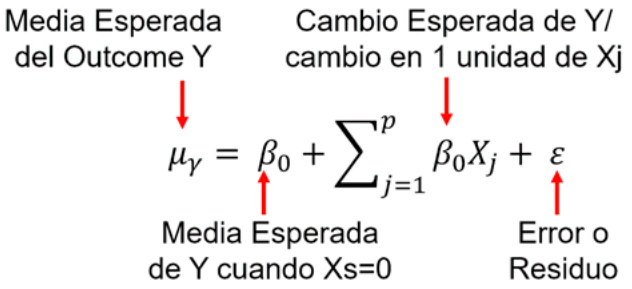
El análisis de regresión permite determinar cómo varían la direccionalidad y la magnitud de la asociación entre las variables de exposición y desenlace antes y después de realizar el ajuste por la variable de confusión. Sin embargo, es importante tener en consideración que se pierde precisión mientras más variables se introduzcan en un modelo multivariable. A continuación, se presentan las ventajas y desventajas de utilizar un modelo de regresión multivariado para controlar el sesgo de confusión:

Ventajas	Desventajas
Es adecuado para ajustar por muchos factores de confusión medidos	Requiere un gran tamaño de muestra si hay una alta cantidad de covariables
Estima en el resultado los efectos de todos los factores de confusión medidos	Solo para factores de confusión conocidos y para datos disponibles

Modelos lineales generalizados (GLM)

El modelo lineal asume que la variable dependiente *Y* está condicionada por una combinación lineal de las variables explicativas *X*. No obstante, el modelo de regresión lineal puede fallar cuando *Y* es binaria o de conteo. En ese sentido, los modelos lineales generalizados (GLM por las siglas en inglés de *generalized linear model*) resultan útiles, pues proporcionan un enfoque común para una amplia gama de problemas de modelado de respuestas. Las respuestas de Poisson y binomial son las más utilizadas, pero también se pueden utilizar otras distribuciones. Asimismo, además de requerir que se especifique la distribución de probabilidad de la variable dependiente, los GLM también necesitan establecer una función de enlace que permita una mayor flexibilidad en el modelado.

La función esperada es:



Los modelos lineales generalizados son los siguientes:

Modelo	Respuesta	Distribución	Interpretación del coeficiente de regresión
Lineal	Continua	Gausiana	Cambio en la media de *Y* por cambio en una unidad de *X*
Logística	Binaria	Binomial	Log *odds* ratio
Log-lineal	Tiempos para evento/conteo	Poisson	Log riesgo relativo
Hazards proporcionales	Tiempo para evento	Semi-paramétrico (Cox)	Log *hazard*

◊ *Modelos lineales generalizados con R*

En R, el comando para producir un GLM es:

glm(formula, **family=**family(**link=**function), **data=**)

Donde `formula` especifica la relación entre las variables explicativas y la variable respuesta, `family` describe la distribución de errores que se ajustará y `link` especifica la función de enlace utilizada en el modelo. Por ejemplo, supongamos que queremos ajustar un modelo de regresión logística para predecir si un estudiante

aprueba o no un examen basado en el número de horas de estudio y el nivel de educación. La sintaxis del comando sería:

```
glm(aprobado ~ horas_estudio + nivel_educacion,
family=binomial(link=logit), data=datos)
```

Donde "aprobado" es la variable binaria de respuesta, "horas_estudio" y "nivel_educacion" son las variables explicativas y **datos** es el conjunto de datos que contiene estas variables. La familia binomial con enlace logit se utiliza para el modelo de regresión logística.

A continuación, se presentan las funciones de enlace en R según la función familiar:

Familia	Función de enlace predeterminado
binomial	(link="logit")
gaussian	(link="identity")
gamma	(link="inverse")
inverse gaussian	(link= "1/mu^2")
poisson	(link="log")
quasi	(link="identity", variance= "constant")
quasibinomial	(link="logit")
quasipoisson	(link="log")

Funciones de soporte de GLM

Hay funciones en R que facilitan y brindan soporte al momento de trabajar con los GLM:

Funciones	Descripción
`summary`	Muestra resultados detallados para el modelo ajustado
`coefficients, coef`	Enumera el parámetro del modelo (intersección y pendientes) para el modelo ajustado
`confint`	Proporciona intervalos de confianza para los parámetros del modelo (95% por defecto)

Funciones	Descripción
`residuals`	Enumera los valores residuales para un modelo ajustado
`anova`	genera una tabla ANOVA (análisis de varianza, por sus siglas en inglés) que permite comparar dos modelos ajustados
`plot`	Generar un diagnóstico gráfico para evaluar el ajuste de un modelo
`predict`	Utiliza un modelo ajustado para predecir la respuesta para un nuevo conjunto de datos

Regresión lineal

Regresión lineal simple

La regresión lineal simple predice el comportamiento de la variable dependiente *Y* en función de una variable independiente o predictora *X*:

```
`Y = β0 + β1*X1 + ε`
```

Regresión lineal multivariada con método forward

La regresión lineal multivariada predice el comportamiento de la variable dependiente en función de más de dos variables independientes o predictoras:

```
`Y = β0 + β1*X1 +…+ βp*Xp + ε`
```

Además, brinda mayor precisión y consistencia en la inferencia estadística, puesto que identifica patrones y correlaciones entre múltiples variables a la vez y permite tener mayor control sobre las mismas. Al incluir varias variables en el modelo, es posible controlar su efecto sobre la variable dependiente y evaluar su importancia relativa en la predicción. De esta manera, se puede determinar cómo cambia la variable dependiente ante variaciones específicas en las variables predictoras, lo que facilita la interpretación y la toma de decisiones informadas.

El método forward, también denominado "selección hacia adelante", parte de un modelo muy sencillo al que se le van agregando términos siguiendo un criterio determinado, hasta que no procede añadir ningún término más. Es decir, en cada etapa se introduce la variable más significativa hasta una cierta regla de parada. Se añadirá el término que al añadirlo al modelo resulte más significativo. Veamos el siguiente ejemplo:

Supongamos que queremos predecir el precio de un coche usado. Tenemos acceso a datos que incluyen información sobre la marca, el modelo, el año de fabricación, la distancia recorrida y el tipo de combustible utilizado. Inicialmente, podemos comenzar con un modelo nulo que solo incluye la variable dependiente, el precio del coche.

Modelo 0: Desenlace Y como función nula

Precio = $B_0 + \varepsilon_0$

Luego, podemos utilizar el método forward para ir añadiendo variables independientes al modelo. En cada etapa, seleccionaremos la variable que mejore el ajuste general del modelo de acuerdo con un criterio determinado. Por ejemplo, podemos utilizar el criterio de R^2 ajustado, que en este caso mide cuánta varianza del precio del coche se explica por las variables incluidas en el modelo, ajustando por el número de variables. Se puede comparar el valor de R^2 ajustado antes y después de la inclusión de la variable. Si el valor es mayor después de agregar la variable, entonces la predicción ha mejorado.

Supongamos que después de agregar la variable marca, obtenemos el siguiente modelo:

Modelo 1: Desenlace Y como función de "Marca"

Precio = $B_0 + (B_1 * \text{Marca} + \varepsilon_1)$

Podemos seguir agregando variables independientes, como el modelo del coche y la distancia recorrida, y utilizando el mismo criterio de R^2 ajustado para evaluar la calidad del modelo. Supongamos que después de agregar estas variables, obtenemos el siguiente modelo:

Modelo 3: Desenlace Y como función de "Marca", "Modelo" y "Distancia"

$Precio = B_0 + (B_1*Marca + B_2*Modelo + B_3*Distancia + \varepsilon_3)$

La construcción de los modelos multivariados es el proceso de decidir qué variables independientes incluir de forma que se mejore el ajuste general del modelo. La decisión de incluir variables adicionales no debe ser arbitraria, se debe tomar en cuenta que, al añadir una variable adicional al modelo, se deberá verificar la predicción mejorada, los coeficientes estimados significativos y la estabilidad de los coeficientes del modelo. Si estos criterios no se cumplen, puede ser necesario eliminar la variable del modelo.

Para verificar la mejora de la predicción se puede emplear el R^2, también denominado coeficiente de determinación. Es una medida de bondad de ajuste para modelos de regresión lineal que muestra qué tan cerca están los puntos de datos de la línea de regresión. Para evaluar si los coeficientes estimados son significativos, se puede calcular el valor del estadístico *t* para cada coeficiente y compararlo con el valor crítico de *t* correspondiente al nivel de significancia deseado. Si el valor del estadístico *t* es mayor que el valor crítico de *t*, entonces el coeficiente es significativo.

Para verificar si los coeficientes del modelo son estables, se puede realizar un análisis de sensibilidad en el que se comparan los coeficientes estimados utilizando diferentes subconjuntos de los datos. Si los coeficientes estimados no varían significativamente cuando se utilizan diferentes subconjuntos de los datos, entonces se puede concluir que los coeficientes son estables.

Adicionalmente, para seleccionar el modelo óptimo, se pueden utilizar varios criterios, como el criterio de información Akaike (AIC) y el estadístico *F*. El AIC es un criterio que toma en cuenta la calidad del ajuste y la complejidad del modelo y se utiliza para comparar modelos con diferentes números de variables. Un modelo con un valor de AIC más bajo se considera mejor. Este criterio no pretende identificar el modelo verdadero, sino el mejor modelo entre los modelos candidatos. El estadístico *F*, por otro lado, indica qué tan lejos están los datos de la media (relación de varianzas). Permite comparar ajustes de diferentes modelos lineales de regresión y ayuda a decidir si existe relación significativa entre las variables. Si el valor de *F* es mayor que el valor crítico, entonces se puede concluir que el modelo es significativo.

Por ejemplo, en el caso de los datos de los autos, se pueden utilizar el AIC y el estadístico *F* para seleccionar el modelo óptimo. Supongamos que se han ajustado dos modelos, uno con solo la variable de cilindrada y otro con la variable de cilindrada y la variable de peso. Se puede comparar el valor del AIC y el valor del estadístico *F* de ambos modelos para seleccionar el mejor modelo.

Comando de R	Descripción	Ejemplo
`summary(lm(Y ~ X1, data = autos))$r.squared`	Cálculo del R-cuadrado	`summary(lm(mpg ~ hp, data = mtcars))$r.squared`
`summary(lm(Y ~ X1, data = autos))$coefficients`	Cálculo del AIC	`summary(lm(mpg ~ hp, data = mtcars))$coefficients`
`AIC(lm(Y ~ X1, data = autos))`	Cálculo del AIC	`AIC(lm(mpg ~ hp, data = mtcars))`
`summary(lm(Y ~ X1 + X2, data = autos))$fstatistic`	Cálculo del estadístico F	`summary(lm(mpg ~ hp + wt, data = mtcars))$fstatistic`

Regresión lineal multivariada con método stepwise

El método stepwise, también conocido como "paso a paso", es un enfoque que comienza con un modelo que incluye todas las variables independientes y luego elimina variables de forma iterativa según un criterio determinado, hasta que no hay variables restantes que cumplan el criterio. En cada paso, se consideran tanto las variables que ya están en el modelo como aquellas que aún no se han agregado.

La construcción de modelos multivariados mediante el método stepwise implica decidir qué variables independientes incluir o eliminar del modelo para mejorar su ajuste general. La selección de variables se basa en una combinación de dos criterios: el valor *p* y el criterio de información de Akaike (AIC). El valor *p* se utiliza para evaluar la significación estadística de cada variable, mientras que el AIC se utiliza para comparar diferentes modelos y seleccionar el que mejor se ajuste a los datos.

Por ejemplo, supongamos que queremos construir un modelo para predecir el precio de una casa en función de varias variables independientes, como el tamaño de la casa, el número de habitaciones, el año de construcción y la ubicación. Podemos aplicar el método stepwise de la siguiente manera:

Comenzamos con un modelo que incluye todas las variables independientes:

$$\text{Precio} = \beta 0 + \beta 1\ \text{Tamaño} + \beta 2\ \text{Habitaciones} + \beta 3\ \text{Año} + \beta 4\ \text{Ubicación} + \varepsilon$$

Luego, eliminamos iterativamente las variables que no cumplan con los criterios de selección. Supongamos que la primera variable que se elimina es la ubicación debido a su valor *p* elevado:

$$\text{Precio} = \beta 0 + \beta 1\ \text{Tamaño} + \beta 2\ \text{Habitaciones} + \beta 3\ \text{Año} + \varepsilon$$

A continuación, eliminamos la variable de año debido a su valor *p* elevado:

$$\text{Precio} = \beta 0 + \beta 1\ \text{Tamaño} + \beta 2\ \text{Habitaciones} + \varepsilon$$

Finalmente, llegamos a un modelo que incluye solo las variables de tamaño y habitaciones, las cuales tienen valores *p* bajos y una mejora en el AIC:

$$\text{Precio} = \beta 0 + \beta 1\ \text{Tamaño} + \beta 2\ \text{Habitaciones} + \varepsilon$$

Es importante destacar que la decisión de incluir o excluir variables adicionales en el modelo debe ser cuidadosamente considerada, ya que agregar demasiadas variables puede conducir a un sobreajuste del modelo y disminuir su capacidad de generalización.

Análisis post regresión

◊ *Linealidad*

Esta prueba asume que existe una relación lineal o correlativa entre cada variable independiente y la variable dependiente. Un gráfico que se utiliza para representar visualmente esta prueba es el gráfico de dispersión. Si los puntos forman una línea diagonal recta aproximada, quiere decir que existe una relación lineal entre las variables. Por ejemplo, si se desea estudiar la relación entre el tiempo de estudio y el rendimiento académico, se pueden recolectar datos de un grupo de estudiantes, para los que se registra el tiempo de estudio en horas y el rendimiento académico en una escala de 1 a 10. Se puede presentar los datos en un gráfico de dispersión, en que el tiempo de estudio se coloca en el eje *x* y el rendimiento académico en el eje *y*. Si los puntos en el gráfico forman una línea diagonal recta aproximada, esto sugiere que existe una relación lineal positiva entre el tiempo de estudio y el rendimiento académico.

Además del gráfico de dispersión, también se pueden realizar pruebas estadísticas para evaluar la suposición de linealidad. Por ejemplo, se puede utilizar el coeficiente de correlación de Pearson para medir la fuerza y la dirección de la relación lineal entre dos variables. Un valor de 1 indica una correlación lineal positiva perfecta, un valor de -1 indica una correlación lineal negativa perfecta y un valor de 0 indica ausencia de correlación lineal.

Otra prueba estadística útil para evaluar la suposición de linealidad es el análisis de residuos, con el que se examinan los residuos (diferencia entre los valores observados y los valores predichos por el modelo) en función de los valores ajustados (valores predichos por el modelo). Si los residuos están aleatorizados alrededor de cero, esto sugiere que el modelo se ajusta bien a la suposición de linealidad. Si, por el contrario, hay patrones sistemáticos en los residuos, esto sugiere que la suposición de linealidad no se cumple y se debe considerar otro modelo o una transformación de las variables.

En caso de no cumplirse esta suposición, se puede aplicar una transformación no lineal a las variables independientes, es decir, obtener sus logaritmos o raíces cuadradas. Al hacer esto, puede que la relación entre las variables se convierta en una más lineal. Otra opción es añadir una variable predictora más al modelo o, en una situación extrema, se podría eliminar la variable independiente que no es útil incluir en el modelo.

◊ *Independencia de los errores*

La independencia de los errores es una prueba estadística que se utiliza para comprobar si los errores del modelo son independientes entre sí. Esta prueba es importante porque si los errores no son independientes, entonces los resultados del modelo pueden ser sesgados y poco fiables. Se verifica mediante la inspección de los residuos del modelo, estos son la diferencia entre los valores observados y los valores predichos por el modelo. Si los residuos no están correlacionados entre sí, entonces se acepta la hipótesis nula asumiendo que los errores son independientes. Rechazar esta hipótesis supone un gran problema para el modelo, ya que:

- Los estimadores dejan de ser eficientes, es decir, de varianza mínima.
- Los contrastes de significación dejan de ser válidos, tendiendo a detectar modelos inexistentes.
- Los predictores son ineficientes.

La falta de independencia se suele dar en situaciones donde las observaciones son recogidas secuencialmente en el tiempo. La independencia de los errores se puede evaluar estadísticamente (mediante la prueba de Durbin-Watson) y gráficamente con un diagrama de dispersión de los residuos (eje *y*) vs. el orden en que se tomaron las observaciones (eje *x*). De no cumplirse este supuesto se puede revisar la diferenciación, agregar retrasos, revisar los factores estacionales o estandarizar los datos.

◊ *Normalidad*

Asume que los residuos del modelo se distribuyen normalmente. Una de las formas de verificarlo es con una prueba de normalidad de Kolmogorov. Esta prueba diagnostica la normalidad de la variable a nivel poblacional. Visualmente, un gráfico QQ (cuantil-cuantil) también resulta útil para determinar si los residuos de un modelo siguen o no una distribución normal. Si los puntos en el gráfico forman aproximadamente una línea diagonal recta, entonces se cumple la suposición de normalidad.

De no cumplirse la suposición, se deberá verificar, en primer lugar, que no haya valores atípicos en los datos. Otra alternativa es aplicar una transformación no lineal a la variable de respuesta, como sacar la raíz cuadrada, el logaritmo o la raíz cúbica

de todos los valores de la variable de respuesta. Esto, por lo general, permite que los residuos del modelo se distribuyan de forma más normal.

◊ *Homocedasticidad*

Asume que los residuos de la regresión lineal múltiple tienen una varianza constante en cada punto del modelo. Caso contrario, estamos en una situación de heterocedasticidad, lo que hace que el estudio sea poco confiable ya que un término del modelo puede parecer significativo cuando realmente no lo es.

Para comprobar si cumple con la suposición de homocedasticidad, la forma más práctica es crear un gráfico de residuos estandarizados frente a los valores predichos. Cuando se ajusta un modelo de regresión a un conjunto de datos, se puede crear un gráfico de dispersión que muestre los valores pronosticados. Para saber si estamos en un caso de homocedasticidad, los puntos no deberán presentar un patrón. Si se observa un patrón, estamos frente a un problema heterocedasticidad.

De no cumplirse la suposición, una opción es transformar la variable dependiente considerando el algoritmo y la raíz cuadrada de todos los valores hasta que no aparezca la heterocedasticidad. Otra opción es redefinir la variable dependiente con el uso de una tasa, en lugar del valor bruto.

◊ *Análisis de sensibilidad de valores influyentes*

Un valor influyente es un valor cuya inclusión o exclusión puede alterar los resultados del análisis de regresión. Tal valor está asociado con un gran residuo. La métrica llamada *distancia de Cook* permite determinar la influencia de un valor. Esta métrica define la influencia como una combinación de apalancamiento y tamaño residual. Una regla general es que una observación tiene una gran influencia si la distancia de Cook supera 4/(n - p - 1), donde *n* es el número de observaciones y *p* el número de variables predictoras. Si bien las observaciones influyentes no transgreden necesariamente ningún supuesto de regresión, pueden generar dudas sobre las conclusiones extraídas de la muestra.

◊ *Modificación de efecto*

La modificación de efecto implica que la asociación entre una variable dependiente y una independiente puede ser diferente dependiendo de una tercera variable. A diferencia de los factores de confusión que distorsionan la asociación entre el predictor y el resultado, los modificadores de efecto diferencian la asociación entre el predictor y el resultado. Si la inclusión de un modificador de efecto ajusta mejor el modelo, podríamos elegirlo como nuestro modelo final.

Confusor

- X_2 es un confusor de la asociación entre Y y X_1 si β_1 deja de ser significativo

$Y = \beta_0 + \varepsilon$

$Y = \beta_0 + \beta_1 * X_1 + \varepsilon$

$Y = \beta_0 + \beta_2 * X_2 + \varepsilon$

$Y = \beta_0 + \cancel{\beta_1 * X_1} + \beta_2 * X_2 + \varepsilon$ (X_2 = Es un confusor)

$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \varepsilon$ (X_2 = No es un confusor)

Modificador de Efecto

- X_2 es un ME de la asociación entre Y y X_1 si β_3 es significativo

$Y = \beta_0 + \varepsilon$

$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \varepsilon$

Creamos un $X_1 * X_2 = (X_1) * (X_2)$

$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \beta_3 X_1 * X_2 + \varepsilon$ (X_2 = Es un ME)

$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \cancel{\beta_3 X_1 * X_2} + \varepsilon$ (X_2 = No es un ME)

Resumen del capítulo

El análisis de regresión multivariable es una de las estrategias más importantes para controlar el sesgo de confusión. Sin embargo, se tienen que considerar las desventajas tales como es estrictamente dependiente del poder del estudio, la calidad de los datos y qué tan completa haya sido la medición de las variables confusoras en el estudio. Así mismo hay que considerar que mientras más complejo el modelo, más difícil será explicar sus resultados al público en general. Otro aspecto muy importante a considerar es el propósito del mismo, toda vez que cuando lo que se busca es determinar qué factores están asociados a un desenlace de interés, lo importante es conseguir el mejor ajuste posible del modelo, mientras que si lo que se busca es analizar una asociación en específico, lo importante es ajustar el modelo por las principales variables confusoras independientemente de si estas son significativas o no. Esto último cobra mayor relevancia cuando la muestra es pequeña y muchas veces variables predictoras deben ser incluidas en el modelo por su relevancia

teórica más que por su significancia estadística. Finalmente, es importante destacar que un análisis de regresión termina con el análisis post regresión. De ahí que en R se hayan desarrollado una serie de funciones de soporte que nos permiten realizar este análisis y comprar los principales supuestos de cada modelo de regresión. Entre estos vale la pena destacar la importancia de comprobar los supuestos del modelo, evaluar los valores extremos y los modificadores de efecto, como mínimo.

CAPÍTULO 7

REGRESIÓN LINEAL

En este capítulo se llevará a cabo una regresión lineal, pero antes de proceder, se seguirán algunos pasos previos. En ese sentido, y con el fin de familiarizarnos con los datos y conocer los mejores predictores para el modelo, realizamos un análisis exploratorio y un análisis de correlación múltiple. Luego, nos introducimos en el modelo de regresión propiamente dicho aplicando los métodos forward y stepwise; para finalizar, realizamos el análisis post regresión y la selección del modelo óptimo.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Conocer las aplicaciones de una regresión lineal.
- Aprender a manejar los métodos forward y stepwise, que son algunos de los métodos de selección de variables de una regresión lineal.
- Interpretar los resultados de una regresión lineal y seleccionar el modelo óptimo.
- Realizar un análisis post regresión que valide el modelo.

Paquetes

```
#install.packages("dplyr")
#install.packages("psych")
#install.packages("GGally")
#install.packages("AICcmodavg")
#install.packages("stargazer")
#install.packages("ggplot2")
#install.packages("gridExtra")
#install.packages("car")
```

```
#install.packages("nortest")

library(dplyr) #manipular datos
library(psych) #múltiples histogramas
library(GGally) #correlación
library(AICcmodavg) #comparación de modelos
library(stargazer) #comparación R2
library(ggplot2) #gráficos
library (gridExtra) #organiza gráficos
library(car) #durbinWatsonTest
library(nortest) #kolgomorov
```

Estudio de caso

Para analizar la relación entre dos o más variables cuantitativas se utilizan, por lo general, las técnicas de correlación y regresión lineal. La primera mide la fuerza de la relación y la segunda predice una variable a partir de la suma lineal ponderada de múltiples variables.

En este capítulo examinamos el conjunto de datos libre **mtcars** extraído de la revista *Motor Trend US* de 1974. En ella se analiza el consumo de combustible y 10 aspectos del diseño y el rendimiento de 32 modelos de automóviles (1973-74). Utilizaremos 6 de estas 10 variables numéricas para construir un modelo de regresión lineal aplicando los métodos forward y stepwise.

Variable	Descripción	Tipo de variable
mpg	Millas por galón	Cuantitativa
disp	Desplazamiento	Cuantitativa
hp	Potencia bruta (caballos de fuerza)	Cuantitativa
drat	engranaje del eje trasero	Cuantitativa
wt	Peso en libras	Cuantitativa
qsec	Segundos que toma recorrer ¼ de milla	Cuantitativa

Llamamos primero a la base de datos y la guardamos en el nuevo objeto **data** para luego seleccionar las variables de interés.

```
#llamamos a la base de datos
data <- mtcars

#seleccionamos las variables de interés
data <- data %>% dplyr::select(mpg, disp, hp, drat, wt, qsec)
```

Análisis exploratorio

Observamos las primeras 10 filas de la base de datos.

```
head(data, n=10)
##                    mpg  disp  hp drat    wt  qsec
## Mazda RX4         21.0 160.0 110 3.90 2.620 16.46
## Mazda RX4 Wag     21.0 160.0 110 3.90 2.875 17.02
## Datsun 710        22.8 108.0  93 3.85 2.320 18.61
## Hornet 4 Drive    21.4 258.0 110 3.08 3.215 19.44
## Hornet Sportabout 18.7 360.0 175 3.15 3.440 17.02
## Valiant           18.1 225.0 105 2.76 3.460 20.22
## Duster 360        14.3 360.0 245 3.21 3.570 15.84
## Merc 240D         24.4 146.7  62 3.69 3.190 20.00
## Merc 230          22.8 140.8  95 3.92 3.150 22.90
## Merc 280          19.2 167.6 123 3.92 3.440 18.30
```

Valores perdidos

```
table(complete.cases(data))
##
## TRUE
##   32
```

No hay valores perdidos. Todos los datos son verdaderos. La conclusión de que no hay valores perdidos se puede respaldar con el resultado del comando `table(complete.cases(data))` aplicado en Rstudio, el cual muestra el número de casos completos en la base de datos. Si el número de casos completos coincide con el número total de observaciones en la base de datos, entonces se confirma que no hay valores perdidos y que todos los datos son verdaderos.

Descripción de los datos

El resumen estadístico corresponde a las 6 variables seleccionadas de la base de datos **data**. El resumen incluye el mínimo y máximo valor, los cuartiles (Q1, Q3), la media y la mediana.

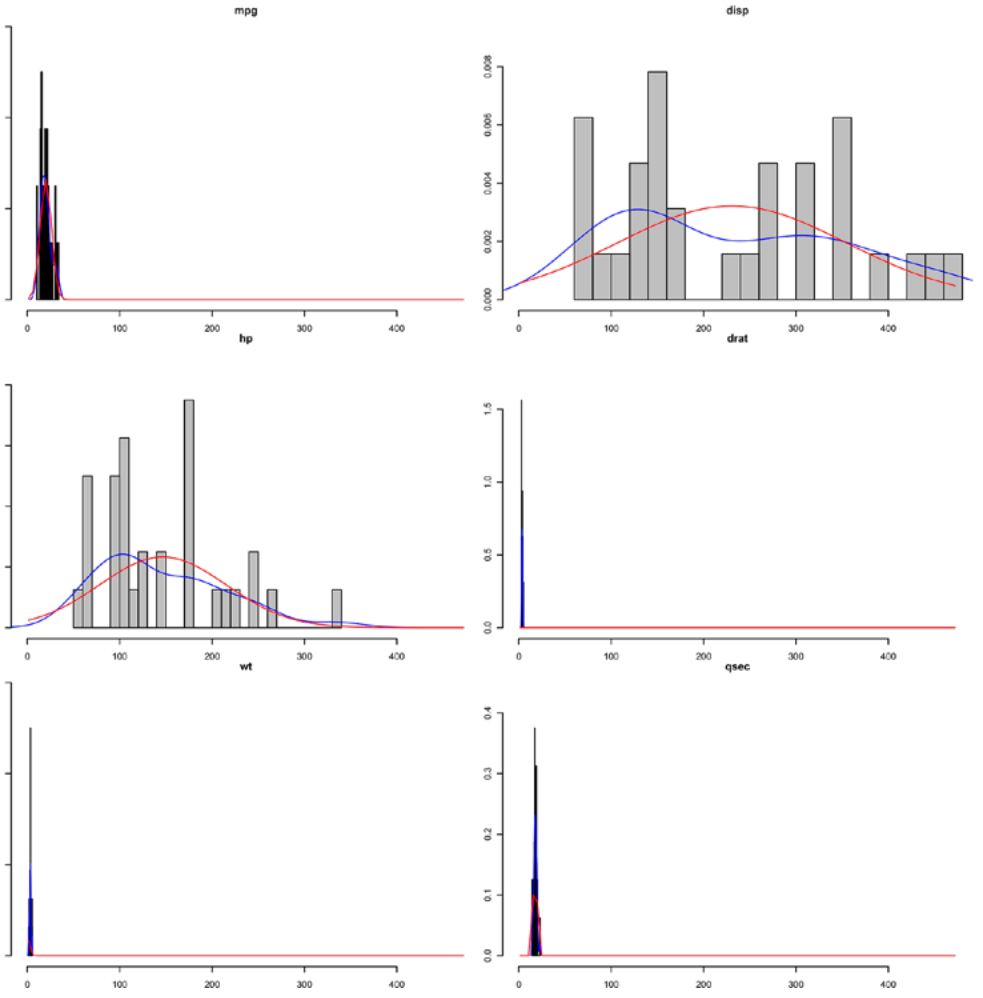
```
summary(data)

##       mpg             disp             hp             drat
##  Min.   :10.40   Min.   : 71.1   Min.   : 52.0   Min.   :2.760
##  1st Qu.:15.43   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080
##  Median :19.20   Median :196.3   Median :123.0   Median :3.695
##  Mean   :20.09   Mean   :230.7   Mean   :146.7   Mean   :3.597
##  3rd Qu.:22.80   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920
##  Max.   :33.90   Max.   :472.0   Max.   :335.0   Max.   :4.930
##        wt             qsec
##  Min.   :1.513   Min.   :14.50
##  1st Qu.:2.581   1st Qu.:16.89
##  Median :3.325   Median :17.71
##  Mean   :3.217   Mean   :17.85
##  3rd Qu.:3.610   3rd Qu.:18.90
##  Max.   :5.424   Max.   :22.90
```

Distribución de las variables

Con la función `multi.hisp` del paquete `psych` vamos a producir histogramas para cada variable del conjunto de datos. Estos gráficos incluyen ajustes normales y

distribuciones de densidad. En la siguiente línea de comandos, `dcol` establece los colores para los ajustes normales y de densidad, respectivamente, y `dlty` define el tipo de línea con que se representa cada uno de estos ajustes. Le asignaremos el color azul y una línea punteada al ajuste de normalidad y el rojo y una línea sólida al de densidad. Adicionalmente, le asignamos el color gris a las barras con el comando `bcol`.

```
multi.hist(data,dcol = c("blue", "red"),
           dlty = c("dotted", "solid"), bcol= "grey")
```



Observamos que los valores de las variables "mpg", "drat", "qsec" y "wt" se ubican en el extremo izquierdo, sin embargo, esto no quiere decir que no estén distribuidos normalmente.

El propósito del análisis exploratorio es descubrir patrones y relaciones en los datos y comparar las distribuciones de varias variables, lo que puede ayudar en la comprensión de los datos y en la selección de técnicas de análisis estadístico adecuadas para la interpretación de los resultados.

Análisis de correlación

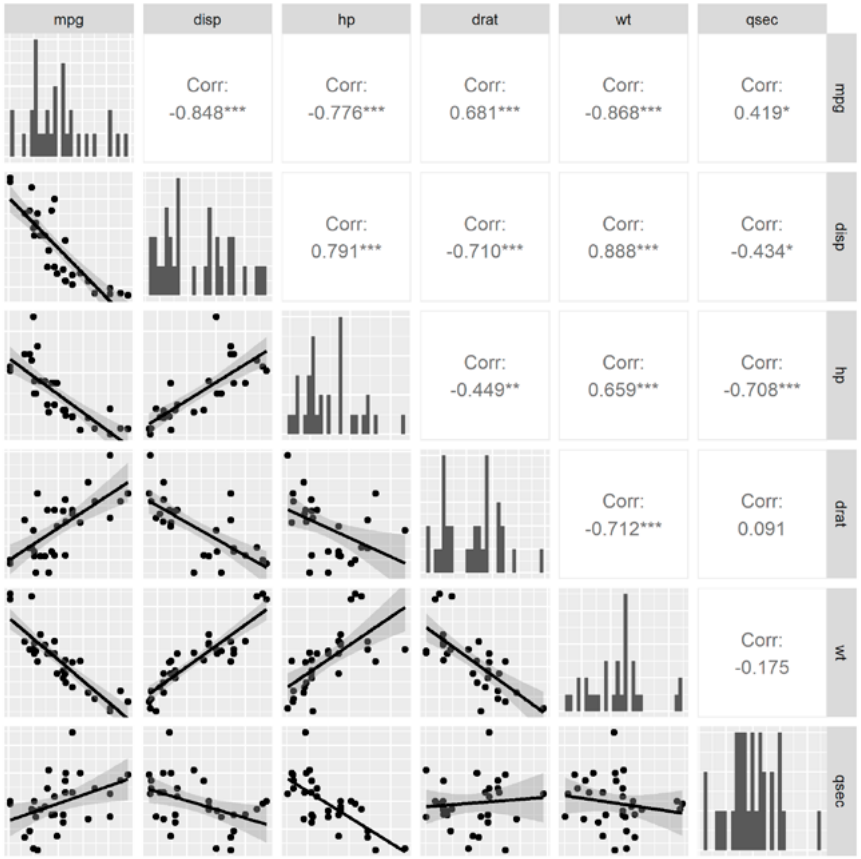
El coeficiente de correlación, también conocido como factor "r", indica relaciones lineales en las que a un mayor valor en *X* corresponde un mayor valor de *Y* y expresa en qué grado los sujetos tienen el mismo orden en ambas variables. Si la correlación es perfecta (es decir, si r=1), el diagrama de dispersión coincidirá con una línea recta, la cual traza una perpendicular desde el eje *x* hasta la recta y trazando desde la recta otra perpendicular hasta el eje *y* (variable dependiente o predicha). En este caso, el orden de los sujetos en ambas variables es el mismo y la recta de regresión será la misma que la línea recta del diagrama de dispersión.

R^2, también conocido como el coeficiente de determinación, expresa la proporción de variación conjunta, es decir, la varianza común entre las dos variables.

Para identificar cuáles pueden ser los mejores predictores para el modelo de regresión, realizamos una correlación múltiple con la función `ggpairs` del paquete `Ggally`, que permite observar los diagramas de dispersión en el lado izquierdo y los valores de correlación de Pearson y la distribución de las variables al lado derecho, en un solo gráfico. Esta función toma varios parámetros, como el conjunto de datos a visualizar, el tipo de gráfico que se desea mostrar en la diagonal y la parte inferior de la matriz, y la ubicación de las etiquetas de los ejes.

Específicamente, el parámetro `lower` se utiliza para definir el tipo de gráfico que se mostrará en la parte inferior de la matriz de gráficos, mientras que el parámetro `diag` se utiliza para definir el tipo de gráfico que se mostrará en la diagonal. Además, el parámetro `axisLabels` se utiliza para especificar si se desean etiquetas de ejes en los gráficos. En este caso, la sintaxis de la función `ggpairs` utiliza un gráfico suave para las variables continuas en la parte inferior, un gráfico de barras para las variables continuas en la diagonal y no muestra etiquetas de ejes en los gráficos.

```
ggpairs(data, lower = list(continuous = "smooth"),
        diag = list(continuous = "bar"), axisLabels = "none")
```



Observamos que existen 3 variables con niveles altos de correlación (r >0.75) con las millas por galon ("mpg"): peso ("wt"), volumen del motor ("disp") y caballos de fuerza ("hp").

El número de asteriscos (*) que aparecen en el gráfico de dispersión depende del nivel de significancia estadística de la correlación entre dos variables. Por lo general, un asterisco (*) se utiliza para indicar una correlación significativa a un nivel de significancia del 0.05, dos asteriscos (**) se utilizan para indicar una correlación significativa a un nivel de significancia del 0.01 y tres asteriscos (***) se utilizan para indicar una correlación significativa a un nivel de significancia del 0.001. Es

importante tener en cuenta que la presencia de correlación significativa no implica necesariamente causalidad entre las dos variables. Además, otros factores pueden influir en la relación entre las variables.

A partir de estos resultados que reflejan que hay varias correlaciones fuertes entre las diferentes variables, nos centraremos en las correlaciones vinculadas con la variable "mpg" porque, como sabemos, el consumo de combustible es una de las principales preocupaciones de los consumidores de automóviles y también es una medida importante de la eficiencia del vehículo.

La correlación entre "mpg" y "disp" (desplazamiento) también es fuerte y negativa (-0.85). Esto sugiere que los automóviles con motores más grandes tienen una menor eficiencia de combustible. Así que, para mejorar la eficiencia de combustible de un automóvil, puede ser útil considerar motores más pequeños. La correlación entre "mpg" y "wt" (peso del automóvil) es una correlación fuerte y negativa (-0.87). Esto indica que a medida que aumenta el peso del automóvil, disminuye su eficiencia de combustible. Por lo tanto, si el objetivo es mejorar la eficiencia de combustible de un automóvil, puede ser útil reducir su peso.

A continuación, vamos a realizar análisis de regresión lineal para comprender mejor cómo la variable "mpg" está relacionada con otras variables. Esto puede ayudar a los fabricantes de automóviles a diseñar vehículos más eficientes en términos de consumo de combustible y también puede ayudar a los consumidores a tomar decisiones informadas al comprar un automóvil.

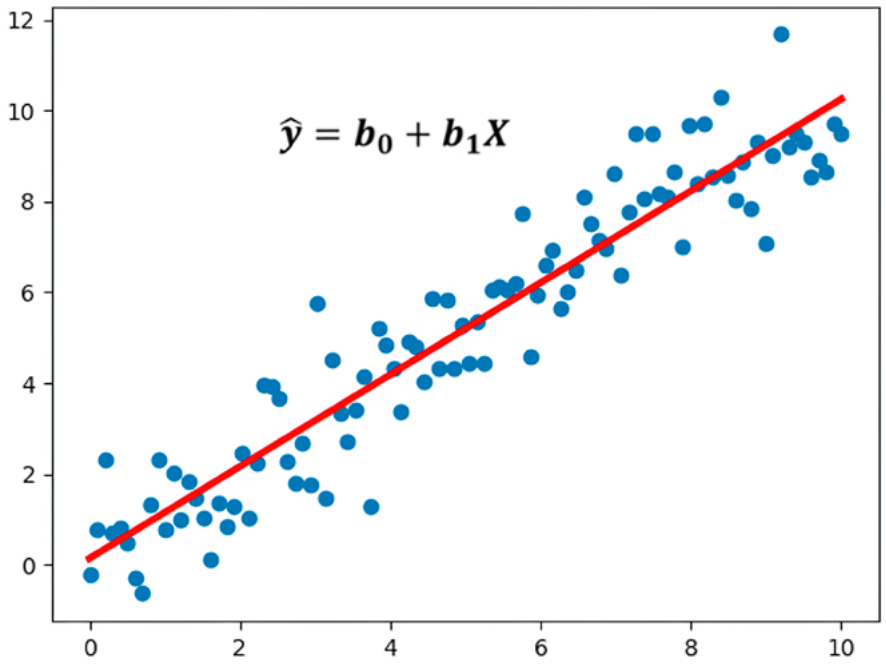
Regresión lineal

La regresión lineal puede ser simple (dos variables) o múltiple (más de dos variables). La regresión lineal simple o univariable estudia la relación lineal entre la variable dependiente *Y* y una única variable independiente *X*. El modelo de regresión lineal describe la variable dependiente con una línea recta definida por la ecuación $Y = a + b \times X$, donde "a" es la "y" –intercepto de la recta– y "b" es su pendiente.

Como se aprecia en la siguiente imagen, la línea de regresión permite predecir el valor de la variable dependiente *Y* a partir de la variable independiente *X*.

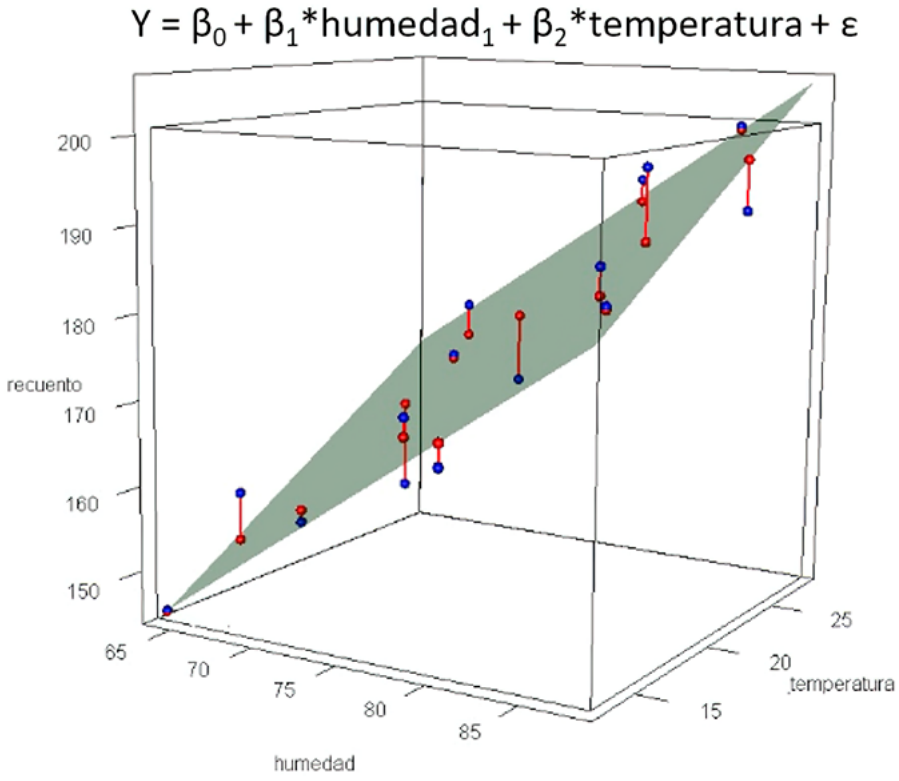
La pendiente "b" de la línea de regresión se llama coeficiente de regresión. Proporciona una medida de la contribución de la variable independiente *X* para

explicar la variable dependiente *Y*.



Por otro lado, la regresión multivariada permite el estudio de múltiples variables independientes al mismo tiempo, con ajuste de sus coeficientes de regresión para posibles efectos de confusión entre variables. Este tipo de regresión se aplica ya que en muchos casos la contribución de una sola variable independiente no basta para explicar la variable dependiente.

En el modelo de regresión multivariable, la variable dependiente se describe como una función lineal de las variables independientes X_i , como sigue: $Y = a + b_1 \times X_1 + b_2 \times X_2 + ... + b_n \times X_n$. El modelo permite calcular un coeficiente de regresión b_i para cada variable independiente X_i, tal como se muestra en la siguiente imagen.



Por lo anterior, una regresión lineal se aplica para predecir o pronosticar, es decir, para crear un modelo de pronóstico para un conjunto de datos específico. A partir de la moda, puedes usar la regresión para predecir valores de respuesta donde solo se conocen los predictores. Así, también, para determinar si existe una relación entre una variable y un predictor y cuán estrecha es esta relación. A continuación, presentaremos los dos métodos más usados de selección de variables para elegir el mejor modelo de regresión.

Por ejemplo, la ecuación $Y = b_0 + b_1$ humedad $+ b_2$ temperatura $+ \varepsilon$ es un modelo de regresión lineal que relaciona la variable de respuesta *Y* con las variables explicativas o independientes humedad y temperatura.

En esta ecuación, *Y* es la variable dependiente, b_0 representa el término de intercepción o constante, que indica el valor de *Y* cuando ambas variables explicativas son cero. Los coeficientes b_1 y b_2 representan la contribución de cada variable explicativa al valor de *Y*, lo que significa que b_1 indica cuánto varía *Y* por cada unidad de

cambio en la variable humedad, manteniendo constante la variable temperatura, y b_2 indica cuánto varía *Y* por cada unidad de cambio en la variable temperatura, manteniendo constante la variable humedad. Mientras que **ε** (epsilon) representa el término de error o residuo, que es la variabilidad no explicada por el modelo y que puede deberse a factores no considerados o a la presencia de ruido aleatorio en los datos.

Al graficar la variable de respuesta *Y* en función de la humedad y la temperatura para visualizar la relación entre las variables, observamos una dispersión de los puntos con una tendencia lineal creciente, lo cual indica que existe una relación lineal positiva entre la humedad o temperatura y la variable respuesta *Y*. Esto implica que a medida que la humedad o temperatura aumentan, la variable de respuesta también tiende a aumentar.

Método forward

Para comparar modelos en estadística inferencial se recomienda utilizar el Criterio de Información de Akaike (AIC), el mismo que en R se calcular con la función `aictab` del paquete `AICcmodavg`.

Veamos cómo se procede con el método forward.

◊ *Modelo nulo*

Solo consideramos la variable dependiente.

```
modelo_0 <-lm(mpg ~1, data = data )
summary(modelo_0)

##
## Call:
## lm(formula = mpg ~ 1, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6906 -4.6656 -0.8906  2.7094 13.8094
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   20.091      1.065   18.86   <2e-16 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 6.027 on 31 degrees of freedom
```

Este modelo asume que todos los posibles *X* o variables dependientes tienen un valor de 0, de modo que el valor ajustado para cada conjunto de valores *X* es la media de la variable de respuesta *Y*. La intersección de la pendiente correspondiente es la media de *Y* y la desviación estándar de los residuos es la desviación estándar de *Y*. Este modelo nulo se constituye entonces en nuestra línea de base de comparación para los siguientes modelos, de ahí que a continuación lo que debemos hacer es calcular nuestro AIC de partida.

```
#Definir el modelo
models0 <- list(modelo_0)

#Especificar el nombre del modelo
mod.names0 <- c('mpg')

#Calcular el AIC del modelo nulo
aictab(cand.set = models0, modnames = mod.names0)
##
## Model selection based on AICc:
##
##     K   AICc Delta_AICc AICcWt Cum.Wt      LL
## mpg 2 209.17          0      1      1 -102.38
```

En nuestro ejemplo, el AIC del modelo nulo es 209.17

◊ *Modelos de primer orden*

El mejor modelo con una variable.

```
modelo_1.1 <- lm(mpg ~ wt, data = data )
summary(modelo_1.1)

##
## Call:
## lm(formula = mpg ~ wt, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

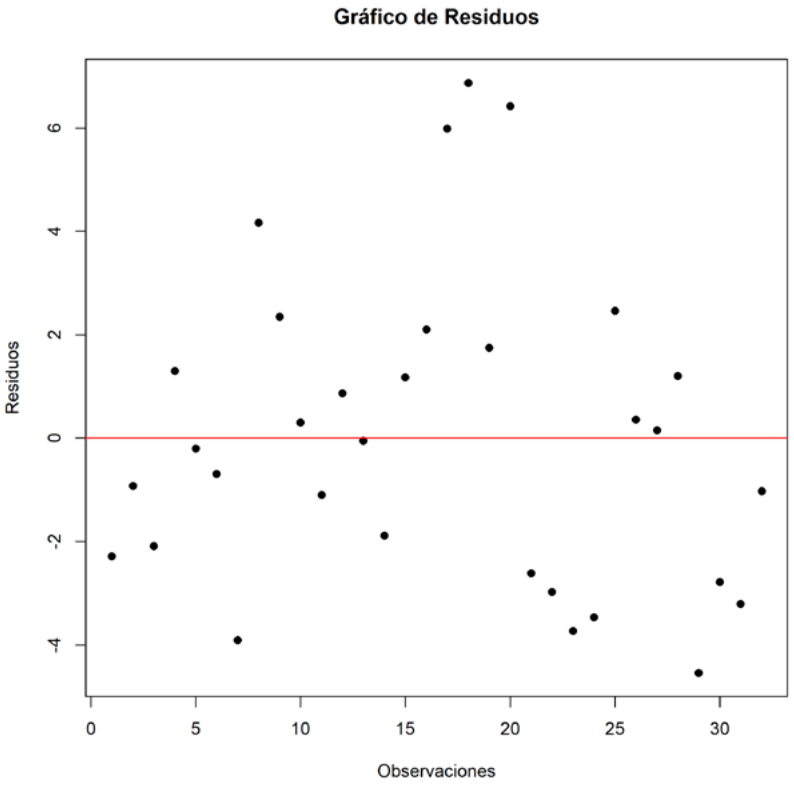
Interpretación del output de un modelo de regresión lineal:

- El modelo (call): lm(formula = mpg ~ wt, data = data) mpg = b_0 + b_1(wt)
- Los residuos (*residuals*): son la diferencia entre los valores reales y los valores predichos. En el *output* se resumen el valor mínimo (Min =-4.5432), el cuartil 1 o percentil 25 (1Q =-2.3647), la mediana (Median =-0.1252), el cuartil 3 o percentil 75 (3Q =1.4096) y el valor máximo (Max =6.8727) de los residuos. Para interpretar esto hay que tomar en consideración que, si el modelo tuviese un ajuste perfecto, la mediana tendría un valor de cero y los valores máximo

y mínimo serían simétricos. Según nuestros resultados, podemos notar que la distribución de los residuos del modelo no es simétrica y está sesgada hacia la izquierda, porque la mayoría de los residuos están más cerca del primer cuartil y se extienden más hacia valores más bajos que hacia los valores más altos. Esto significa que nuestro modelo no predice bien los valores más bajos de la variable "mpg" como sí lo hace con los valores altos de "mpg", es decir, tiende a subestimar los valores de "mpg" para observaciones con valores bajos de "wt". Esto sugiere que el modelo podría no ser tan preciso para predecir los valores de "mpg" para automóviles más livianos.

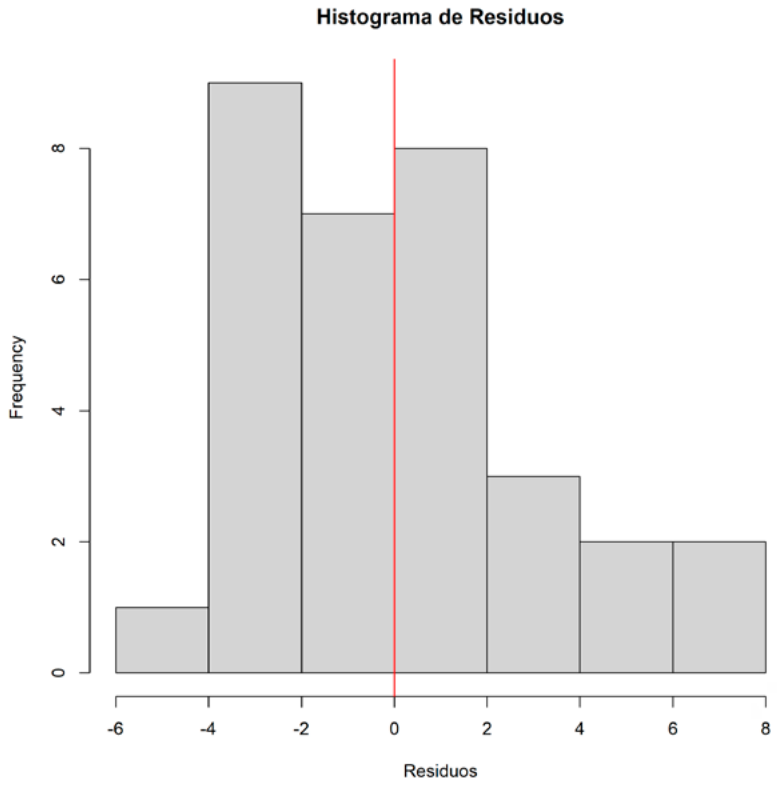
Sin embargo, también es importante tener en cuenta la forma en que los residuos se distribuyen visualmente. Si realizamos un gráfico de dispersión de los residuos del modelo en el eje *y* y las observaciones en el eje *x*, podremos reafirmar que los residuos no están distribuidos de manera uniforme alrededor de cero porque hay algún patrón específico en su distribución, lo que puede indicar algún tipo de sesgo en el modelo. También agregar una línea horizontal en cero nos ayudará a visualizar si los residuos están distribuidos uniformemente alrededor de cero o no, en este caso se observa que no están distribuidos simétricamente.

```
# Graficar residuos
plot(modelo_1.1$residuals, pch = 16, main = "Gráfico de Residuos",
xlab = "Observaciones", ylab = "Residuos")
# Añadir línea horizontal en cero
abline(h = 0, col = "red")
```




```
# Crear histograma de residuos
hist(modelo_1.1$residuals, breaks = 8, main = “Histograma de
Residuos”, xlab = “Residuos”)
# Añadir línea vertical en cero
abline(v = 0, col = “red”)
```

También creamos un histograma de los residuos del modelo en el eje *x* y la frecuencia en el eje *y* para evaluar visualmente su forma de distribución. Observamos que la distribución de los residuos está sesgada hacia la derecha, debido a que el histograma tendrá una cola larga hacia la derecha, lo que demuestra que la distribución de los residuos es asimétrica. Además, se añadió una línea vertical en cero para ver mejor cómo los residuos no se distribuyen uniformemente alrededor de cero.



Si bien los valores del primer cuartil y tercer cuartil pueden indicar que la mayoría de los residuos están más cerca del primer cuartil, el histograma y el diagrama de dispersión sugieren un sesgo hacia la derecha. Por lo tanto, es posible que los residuos estén distribuidos de manera asimétrica en ambas direcciones. En este caso, la conclusión que se podría obtener es que el modelo no está prediciendo de manera precisa los valores extremos tanto en el extremo inferior como en el superior de la variable "mpg".

- El estimado de los coeficientes b_0 (intercepto) y b_1 (X_1) mpg = 37.2851 + -5.3445(wt). Dicho en términos simples, si el valor de "wt" es 0 los valores promedio de "mpg" serían 37.2851 y por cada incremento en una unidad de "wt" los valores de "mpg" disminuyen en 5.3445 unidades.
- El error estándar del coeficiente b_1 es base para calcular el intervalo de confianza al 95% de b_0 y b_1 estimados.

En este caso, se está calculando un IC como una medida estadística que indica el rango de valores en el que se espera que se encuentre el coeficiente b_1 del modelo lineal, que indica la relación entre la variable independiente y la variable dependiente. Se espera que el verdadero valor del coeficiente b_1 se encuentre en el intervalo (-6.440336, -4.248664) en el 95% de las veces. Para calcular se intervalo aplicamos lo siguiente: $ICb_1 = b_1$ +/- Z x Error Estándar. Donde, Z es el z-score que se utiliza para calcular la amplitud del intervalo porque indica la distancia entre un valor observado y la media en términos de desviaciones estándar, siendo su valor 1.96 para un nivel de confianza del 95%. Reemplazando, obtenemos el IC:

IC 95% de b_1 = -5.3445 +/- 1.96 x 0.5591
IC 95% : (-6.440336 a -4.248664)

Como el intervalo no incluye el valor cero, podemos concluir que el coeficiente de peso es significativamente diferente de cero al nivel de significancia del 5%. Esto sugiere que el peso del automóvil tiene un efecto significativo en el consumo de combustible (mpg) y que el modelo es bueno para predecir el consumo de combustible en función del peso del automóvil.

- El estadístico *t* y su valor *p*: el valor *p*, en asociación con el estadístico *t* prueba la hipótesis nula (H0) de b_1=0. De ahí que cualquier valor de *p* por debajo de 0.05 suele considerarse significativo y nos permite concluir que X1 explica una fracción significativa de la varianza o variabilidad de *Y*. En nuestro modelo, podemos ver que los valores *p* para el intercepto y "wt" son significativos y tanto b0 como b1 son diferentes de cero.
- El error estándar residual (residual standard error) es una medida de qué tan bien se ajusta el modelo a los datos. El error estándar residual nos dice la cantidad promedio en que los valores reales de *Y* (valores observados) difieren de las predicciones (valores predichos) en unidades de *Y*. Idealmente, lo que queremos es el error estándar residual más pequeño posible, porque eso significa que los valores predichos por nuestro modelo están muy cercanos a los valores observados, en promedio. En nuestro modelo, los valores predichos difieren en promedio 3.046 unidades de los valores observados de "mpg" (con 30 grados de libertad, es decir, la base de datos tiene 31 observaciones).
- El valor de R-cuadrado múltiple o R^2 (multiple R-squared) es un estadístico que cuantifica qué porcentaje de la variabilidad de *Y* está explicada por el modelo.

En otras palabras, es otro método para determinar qué tan bien se ajusta nuestro modelo a los datos. En nuestro ejemplo, nuestros datos explican ~75.28% de la variabilidad de "mpg", nuestra variable dependiente. No solo es una variabilidad relativamente alta, sino que el modelo tiene en general una precisión bastante alta para ser un modelo de primer orden. Por lo tanto, el modelo es capaz de explicar una cantidad considerable de la variabilidad en la variable dependiente "mpg" a partir de la variable independiente "wt".

- El estadístico *F* y su valor *p* nos permiten probar la hipótesis nula de que no hay asociación entre la variable dependiente y la(s) variable(s) independiente(s), mientras que la hipótesis alternativa es que sí hay asociación. Dicho de otra manera, la hipótesis nula es que los coeficientes de todas las variables del modelo son cero. La hipótesis alternativa es que al menos uno de ellos no es cero. En nuestro ejemplo, el estadístico *F* tiene un valor *p* significativo debido a que se rechaza la hipótesis nula al ser el valor *p,* 1.294e-10, menos que 0.05, por lo que concluimos que existe asociación entre "mpg" y "wt".

  El nivel de significancia depende del contexto del problema y la interpretación del resultado. En algunos casos, puede ser necesario utilizar un nivel de significación más estricto, como 0.01 o 0.001, para reducir el riesgo de cometer un error tipo I (rechazar la hipótesis nula cuando es verdadera).

En conclusión: "wt" nos permite predecir los valores de "mpg" de los carros con una alta precisión por sí solo.

```
modelo_1.2 <- lm(mpg ~ disp, data = data )
summary(modelo_1.2)

##
## Call:
## lm(formula = mpg ~ disp, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8922 -2.2022 -0.9631  1.6272  7.2305
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.599855   1.229720  24.070  < 2e-16 ***
## disp        -0.041215   0.004712  -8.747 9.38e-10 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 3.251 on 30 degrees of freedom
## Multiple R-squared:  0.7183, Adjusted R-squared:  0.709
## F-statistic: 76.51 on 1 and 30 DF,  p-value: 9.38e-10
```

```
modelo_1.3 <- lm(mpg ~ hp, data = data )
summary(modelo_1.3)
```

```
##
## Call:
## lm(formula = mpg ~ hp, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7121 -2.1122 -0.8854  1.5819  8.2360
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
## hp          -0.06823    0.01012  -6.742 1.79e-07 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 3.863 on 30 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5892
## F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
```

Para el modelo 1.2, podemos decir que hay una fuerte relación negativa entre "mpg" y "disp", es decir, a medida que el "disp" aumenta, el "mpg" disminuye. El coeficiente de regresión para "disp" es significativo (valor $p < 0.05$) lo que indica que hay una asociación significativa entre ambas variables. El R^2 de 0.7183 sugiere que el modelo explica aproximadamente el 72% de la variabilidad en "mpg", es decir, predice los valores de "mpg" con una alta precisión

Para el modelo 1.3 podemos decir que también hay una relación negativa entre "mpg" y "hp". En otras palabras, a medida que el "hp" aumenta, el "mpg" disminuye. El coeficiente de regresión para "hp" también es significativo porque su valor *p* es menor a 0.05, lo que indica que hay una asociación significativa entre ambas variables. El R^2 de 0.602 sugiere que el modelo explica aproximadamente el 60% de la variabilidad en "mpg", por lo que no se ajusta mejor que los modelos anteriores debido a su menor precisión.

El R^2 es una forma más directa de medir la variabilidad del *outcome* o desenlace de interés explicada por tu modelo. Sin embargo, detrás del valor *p* hay una prueba estadística denominada prueba F que establece como prueba de hipótesis que el R^2 es igual a 0. Como el valor *p* de todos los modelos evaluados es menor a 0.05, rechazamos la hipótesis nula y concluimos que el R^2 es estadísticamente diferente que 0. De este modo, comprobamos que las magnitudes son significativas.

En conclusión, podemos afirmar que el modelo de "hp" es moderadamente preciso ya que explica alrededor del 60% de la variabilidad en "mpg", seguido del modelo de "disp" que explica aproximadamente el 72%. Es importante destacar que, aunque estos modelos son útiles para predecir el consumo de combustible de un automóvil, el modelo de "wt" sigue siendo el predictor más preciso, ya que explica alrededor del 75% de la variabilidad en "mpg".

Comparando los R^2 con stargazer

Utilizamos la función `stargazer` del paquete del mismo nombre para comparar los R^2. La comparación de los coeficientes de determinación (R^2) de diferentes modelos puede proporcionar información útil sobre qué modelo se ajusta mejor a los datos; se espera que un modelo con un coeficiente de determinación más alto se ajuste mejor a los datos que uno con un coeficiente de determinación más bajo.

Asimismo, al utilizar esta función, se puede crear una tabla que muestre los coeficientes, errores estándar, valores *t* y valores *p* de cada variable independiente en los diferentes modelos, así como los valores de R^2 y R^2ajustados de cada modelo.

```
stargazer(modelo_0, modelo_1.1, type = "text")

##
## ===============================================================
##                               Dependent variable:
##                     ------------------------------------------
##                                          mpg
##                             (1)                 (2)
## ---------------------------------------------------------------
## wt                                          -5.344***
##                                              (0.559)
##
## Constant                  20.091***          37.285***
##                            (1.065)            (1.878)
##
## ---------------------------------------------------------------
## Observations                32                  32
## R2                         0.000               0.753
## Adjusted R2                0.000               0.745
## Residual Std. Error 6.027 (df = 31)    3.046 (df = 30)
## F Statistic                           91.375*** (df = 1; 30)
## ===============================================================
## Note:                              *p<0.1; **p<0.05; ***p<0.01
```

Aplicando `stargazer()` estamos comparando los resultados de dos modelos, modelo_0 y modelo_1.1. y la opción `type = "text"` especifica que la salida deseada es una tabla en formato de texto, como se observa.

La comparación de los R^2 con la función `stargazer()` muestra una mejora significativa en la capacidad explicativa del modelo 1.1 en comparación con el modelo 0. El modelo 1.1 tiene un R^2 de 0.753, lo que indica que el 75.3% de la variabilidad en la variable dependiente "mpg" puede ser explicada por la variable independiente "wt", mientras que el modelo 0 no tiene ninguna capacidad explicativa. Además,

el valor ajustado del R^2 también es mayor en el modelo 1.1 en comparación con el modelo 0, lo que sugiere que el modelo 1.1 proporciona un ajuste mejorado que tiene en cuenta el número de variables explicativas en el modelo.

Comparando los modelos de primer orden con la prueba F

Esta prueba se obtiene al ejecutar un ANOVA. En el contexto del análisis de regresión, técnica estadística se utiliza para realizar un análisis de varianza para comparar diferentes modelos de regresión. En particular, al proporcionar varios modelos de regresión a la función `anova()`, podemos determinar si hay diferencias significativas entre los modelos y si alguno de ellos proporciona un mejor ajuste a los datos que los demás.

Asimismo, debemos recordar que un modelo de primer orden es aquel en que una variable se puede predecir en función de una sola variable explicada. A medida que se incrementa el orden del modelo, se incluyen más términos de variables independientes.

```
anova( modelo_1.1, modelo_1.2, modelo_1.3)

## Analysis of Variance Table
##
## Model 1: mpg ~ wt
## Model 2: mpg ~ disp
## Model 3: mpg ~ hp
##   Res.Df    RSS Df Sum of Sq F Pr(>F)
## 1     30 278.32
## 2     30 317.16  0   -38.837
## 3     30 447.67  0  -130.516
```

Comparando los modelos de primer orden con AIC

```
#Definir la lista de modelos
models1 <- list(modelo_1.1, modelo_1.2, modelo_1.3)
```

```
#Especificar los nombres del modelo
mod.names1 <- c('mpg.wt', 'mpg.disp', 'mpg.hp')

#Calcular el AIC de cada modelo
#Usamos el paquete AICcmodavg el cual contiene la función aictab

aictab(cand.set = models1, modnames = mod.names1)

##
## Model selection based on AICc:
##
##          K   AICc Delta_AICc AICcWt Cum.Wt     LL
## mpg.wt   3 166.89       0.00   0.89   0.89 -80.01
## mpg.disp 3 171.07       4.18   0.11   1.00 -82.10
## mpg.hp   3 182.10      15.21   0.00   1.00 -87.62
```

A simple vista se ve que el modelo 1.1 es el que tiene el menor AIC por lo que la variable peso ("wt") entraría primero al modelo. Aquí es importante recordar que el criterio de consenso para decidir si vale la pena incluir o no una variable más al modelo se basa en evaluar si al incluirla nuestro AIC se reduce en más del 4% con respecto al AIC del modelo previamente ajusta. Esto se interpreta como que esta nueva variable contribuye a explicar una fracción adicional estadísticamente significativa de la variabilidad de nuestro *outcome* o desenlace de interés.

El método forward, también denominado "selección hacia adelante", parte de un modelo muy sencillo al que se le van agregando términos con algún criterio, hasta que no procede añadir ningún término más. En cada etapa se introduce la variable más significativa hasta una cierta regla de parada para que el modelo resulte más significativo.

◊ *Modelos de segundo orden*

```
modelo_2.1 <- lm(mpg ~ wt + disp, data = data )
summary(modelo_2.1 )
```

```
##
## Call:
## lm(formula = mpg ~ wt + disp, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4087 -2.3243 -0.7683  1.7721  6.3484
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.96055    2.16454  16.151 4.91e-16 ***
## wt          -3.35082    1.16413  -2.878  0.00743 **
## disp        -0.01773    0.00919  -1.929  0.06362 .
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 2.917 on 29 degrees of freedom
## Multiple R-squared:  0.7809, Adjusted R-squared:  0.7658
## F-statistic: 51.69 on 2 and 29 DF,  p-value: 2.744e-10

modelo_2.2 <- lm(mpg ~ wt + hp , data = data )
summary(modelo_2.2 )

##
## Call:
## lm(formula = mpg ~ wt + hp, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.941 -1.600 -0.182  1.050  5.854
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
## wt          -3.87783    0.63273  -6.129 1.12e-06 ***
## hp          -0.03177    0.00903  -3.519  0.00145 **
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 2.593 on 29 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.8148
## F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

Comparando los R2 con stargazer

```
stargazer(modelo_2.1, modelo_2.2, type = “text”)

##
## ===============================================================
##                                   Dependent variable:
##                               ----------------------------
##                                           mpg
##                                     (1)            (2)
## ---------------------------------------------------------------
## wt                                -3.351***      -3.878***
##                                    (1.164)        (0.633)
##
## disp                               -0.018*
##                                    (0.009)
##
## hp                                                -0.032***
##                                                    (0.009)
##
## Constant                          34.961***      37.227***
##                                    (2.165)        (1.599)
##
## ---------------------------------------------------------------
```

```
## Observations                            32            32
## R2                                    0.781         0.827
## Adjusted R2                           0.766         0.815
## Residual Std. Error (df = 29)         2.917         2.593
## F Statistic (df = 2; 29)           51.689***     69.211***
## ============================================================
## Note:                             *p<0.1; **p<0.05; ***p<0.01
```

Comparando los AIC

```
#Definir la lista de modelos
models2 <- list(modelo_2.1, modelo_2.2)

#Especificar los nombres del modelo
mod.names2 <- c('mpg.wt.disp', 'mpg.wt.hp')

#Calcular el AIC de cada modelo
aictab(cand.set = models2, modnames = mod.names2)
##
## Model selection based on AICc:
##
##             K   AICc Delta_AICc AICcWt Cum.Wt     LL
## mpg.wt.hp   4 158.13       0.00   0.98   0.98 -74.33
## mpg.wt.disp 4 165.65       7.52   0.02   1.00 -78.08
```

El modelo 2.2 es el que tiene el menor AIC por lo que peso ("wt") y caballos de fuerza ("hp") se quedan en el modelo.

Modelo de tercer orden

```
modelo_3.1 <- lm(mpg ~ wt + hp + disp , data = data )
# Observamos
summary(modelo_3.1 )

##
```

```
## Call:
## lm(formula = mpg ~ wt + hp + disp, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.891 -1.640 -0.172  1.061  5.861
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.105505   2.110815  17.579  < 2e-16 ***
## wt          -3.800891   1.066191  -3.565  0.00133 **
## hp          -0.031157   0.011436  -2.724  0.01097 *
## disp        -0.000937   0.010350  -0.091  0.92851
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## Residual standard error: 2.639 on 28 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.8083
## F-statistic: 44.57 on 3 and 28 DF,  p-value: 8.65e-11
```

Comparando los AIC

```
#Definir la lista de modelos
models3 <- list(modelo_3.1)

#Especificar los nombres del modelo
mod.names3 <- c('mpg.wt.hp.disp')

#Calcular el AIC de cada modelo
aictab(cand.set = models3, modnames = mod.names3)
##
## Model selection based on AICc:
##
##                K   AICc Delta_AICc AICcWt Cum.Wt     LL
```

```
## mpg.wt.hp.disp 5 160.95          0      1      1 -74.32
```

El modelo 3.1 tiene mayor AIC que el modelo 2.2. En conclusión, nos quedamos con el modelo 2.2 como modelo óptimo y descartamos de nuestro análisis la variable "disp".

Método stepwise

A igual que el anterior método, la regresión stepwise se inicia con un modelo sin variables independientes y en cada paso se agrega una variable independiente significativa; sin embargo, si en un paso hay una variable no significativa, se la retira del modelo y se repite el paso. En cada etapa se plantea si todas las variables introducidas deben de permanecer en el modelo.

◊ *Modelo inicial*

Agregamos las tres variables independientes al modelo inicial.

```
modelo_inicial <- lm(mpg ~ wt + disp + hp, data = data )

# Observamos
summary(modelo_inicial)
##
## Call:
## lm(formula = mpg ~ wt + disp + hp, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.891 -1.640 -0.172  1.061  5.861
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.105505   2.110815  17.579  < 2e-16 ***
## wt          -3.800891   1.066191  -3.565  0.00133 **
## disp        -0.000937   0.010350  -0.091  0.92851
```

```
## hp          -0.031157   0.011436  -2.724  0.01097 *
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## Residual standard error: 2.639 on 28 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.8083
## F-statistic: 44.57 on 3 and 28 DF,  p-value: 8.65e-11
```

Las variables de significancia en el modelo son peso (“wt”) y caballos de fuerza (“hp”), basándonos en los valores de los coeficientes de regresión y sus valores *p*.

El coeficiente de “wt” tiene un valor de -3.80 con un valor *p* de 0.00133, lo que indica que es altamente significativo en el modelo, porque al ser menor que 0.05 se rechaza la hipótesis nula que establece que el coeficiente de regresión es igual a cero. Además, el coeficiente de “hp” tiene un valor de -0.03 con un valor *p* de 0.01097, lo que también indica que es significativo en el modelo.

Por otro lado, el coeficiente de “disp” tiene un valor de -0.000937 y un valor *p* de 0.92851, lo que sugiere que no es significativo en el modelo debido a que no se rechaza la prueba de hipótesis nula de que el coeficiente de regresión es igual a cero, lo que significa que no hay una relación significativa entre la variable independiente y la variable dependiente en el modelo de regresión lineal, considerando una confianza al 95%.

Comprobar los modelos de significancia

La función `step` nos permite seleccionar un modelo a partir de la selección de predictores por diferentes criterios. Básicamente, realiza un proceso iterativo empezando con un modelo inicial y luego agregando o eliminando predictores uno por uno, evaluando cada modelo para seleccionar el mejor. El argumento `direction` especificado como `both` (ambos) significa que la función considerará tanto agregar como eliminar predictores en cada paso de la iteración. El argumento `trace` se utiliza para controlar si se imprimirán los detalles de la iteración de selección de modelo; agregamos `FALSE` en el argumento para que no se imprima la información detallada al momento de ejecutar la función.

```
step(modelo_inicial, direction = "both", trace=FALSE)

##
## Call:
## lm(formula = mpg ~ wt + hp, data = data)
##
## Coefficients:
## (Intercept)           wt           hp
##    37.22727     -3.87783     -0.03177
```

El resultado nos confirma que las variables de significancia del modelo son peso ("wt") y caballos de fuerza ("hp").

Modelo final con las variables de significancia

```
modelo_final <- lm(mpg  ~ wt + hp, data = data )

# Observamos
summary(modelo_final)

##
## Call:
## lm(formula = mpg ~ wt + hp, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.941 -1.600 -0.182  1.050  5.854
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
## wt          -3.87783    0.63273  -6.129 1.12e-06 ***
## hp          -0.03177    0.00903  -3.519  0.00145 **
## ---
```

```
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 2.593 on 29 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.8148
## F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

Al haber solo seleccionado las variables de significancia para el modelo, se puede observar que cuenta con un alto coeficiente de determinación (0.8268) y un valor *p* significativo (9.109e-12), con lo que R nos dice que este es el mejor modelo para usar.

Análisis post regresión

Linealidad

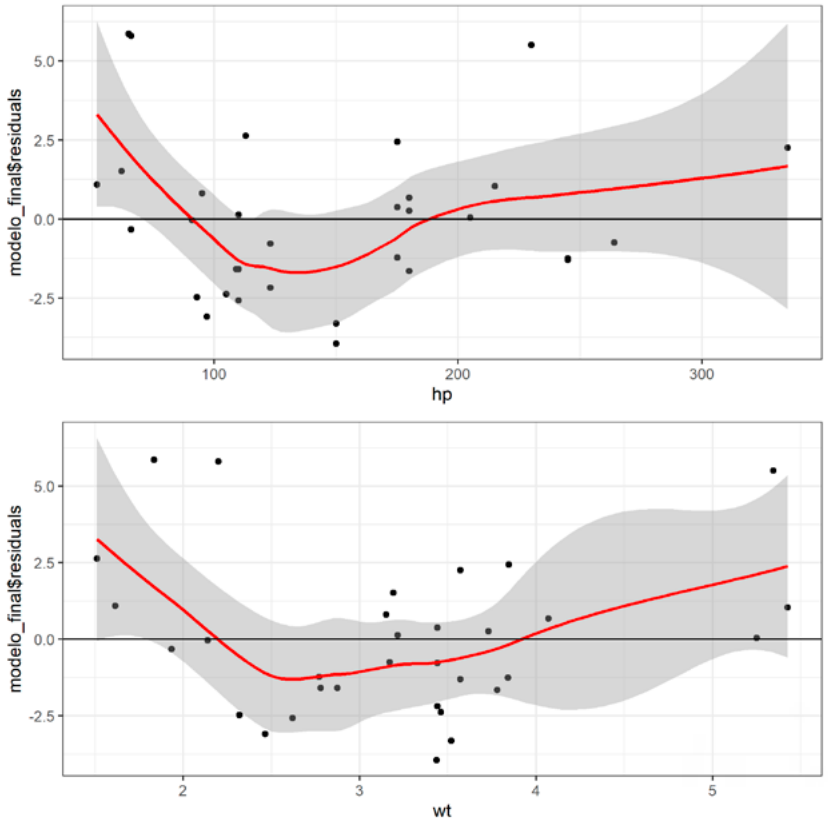
En RStudio es posible realizar pruebas de especificación de modelos de regresión lineal a partir de la prueba RESET (Regression Equation Specification Error Test), que evalúa la adecuación de un modelo de regresión lineal al incluir términos polinómicos de las variables independientes. Al evaluar **modelo_final**, modelo de regresión lineal que se ha ajustado previamente, se busca obtener una medida de la adecuación del modelo, lo que nos permitirá tomar decisiones sobre su validez y ajuste a los datos. Asimismo, `lmtest` se refiere al paquete de R que contiene la función `resettest` que realiza la prueba de RESET.

```
lmtest::resettest(modelo_final)

##
##  RESET test
##
## data:  modelo_final
## RESET = 7.2384, df1 = 2, df2 = 27, p-value = 0.003041
```

Dado que el valor *p* es significativo, rechazamos la hipótesis de linealidad y concluimos que este supuesto no se cumple, por lo que tendríamos que optar por un modelo de regresión no lineal.

Veamos ahora gráficamente los residuos del modelo final. En el siguiente comando, con la función `geom_hline` se intercepta una línea horizontal en el punto 0 y se agregan las funciones `geom_point` y `geom_smooth` para que aparezcan las observaciones y un sobretrazado, respectivamente. Por último, `theme_bw` nos permite cambiar el fondo del gráfico y la función `grid.arrange` del paquete `gridExtra` organiza los gráficos de forma que aparezcan en una sola imagen.

```
# Caballos de fuerza (hp)
Linealidad1 <- ggplot(data = data, aes(hp, modelo_
final$residuals)) +
  geom_point() +
  geom_smooth (color = "red") +
  geom_hline (yintercept = 0) +
  theme_bw ()

# Peso (wt)
Linealidad2 <- ggplot(data = data, aes(wt, modelo_
final$residuals)) +
  geom_point() +
  geom_smooth (color = "red") +
  geom_hline (yintercept = 0) +
  theme_bw ()

# Graficamos
grid.arrange(Linealidad1, Linealidad2)
```



En las gráficas podemos observar diferentes puntos o casos atípicos que retan la hipótesis de linealidad entre las variables independientes y nuestra variable dependiente.

Independencia de los errores

Para evaluar la independencia de los errores de forma estadística, utilizamos la prueba de Durbin-Watson del paquete `car`.

```
durbinWatsonTest(modelo_final)

##  lag Autocorrelation D-W Statistic p-value
##    1       0.2954091      1.362399   0.044
```

```
##  Alternative hypothesis: rho != 0
```

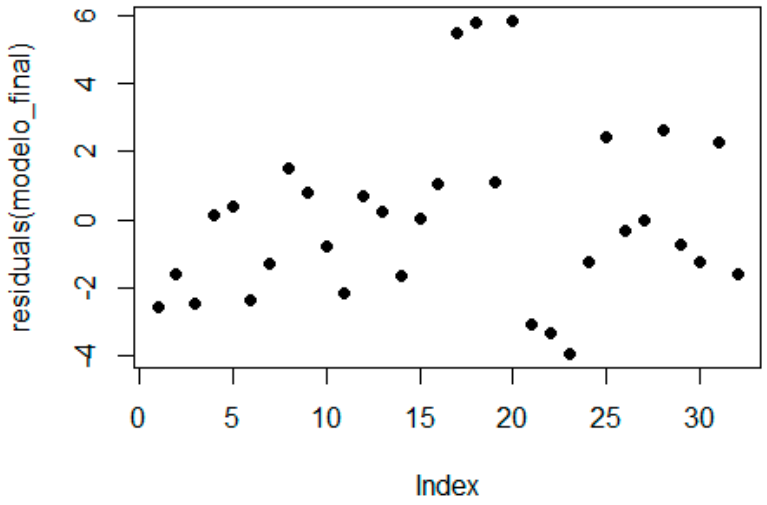
No se rechaza la hipótesis nula, es decir, no hay una autocorrelación y, por ende, se comprueba la independencia de los errores.

Realizamos un gráfico simple de dispersión de residuos:

```
plot(residuals(modelo_final), pch=19)
```

Un gráfico simple de dispersión de residuos se utiliza para evaluar si los residuos (diferencias entre los valores ajustados y observados) de un modelo de regresión lineal tienen un patrón aleatorio y si se distribuyen normalmente alrededor de cero. Si los residuos se distribuyen uniformemente alrededor de cero, sin patrones aparentes, se considera que el modelo es adecuado y que los supuestos del modelo de regresión lineal se cumplen. Si se observan patrones o agrupaciones de residuos, se deben realizar análisis adicionales para determinar la causa del comportamiento y mejorar el ajuste del modelo.

`pch` es un parámetro utilizado en la función `plot()` para especificar el símbolo utilizado para representar los puntos en un gráfico de dispersión. En este caso, `pch=19` representa un círculo sólido, aunque el valor por defecto de `pch` es 1, que representa un círculo. Por lo tanto, los puntos en el gráfico de dispersión serán representados por círculos sólidos en lugar de los círculos huecos predeterminados, lo que hace que sean más fáciles de distinguir entre sí.



Normalidad

Para comprobar la normalidad realizamos la prueba de Lilliefors (Kolgomorov-Smirnov). La función `lillie.test` se encuentra dentro del paquete `nortest`.

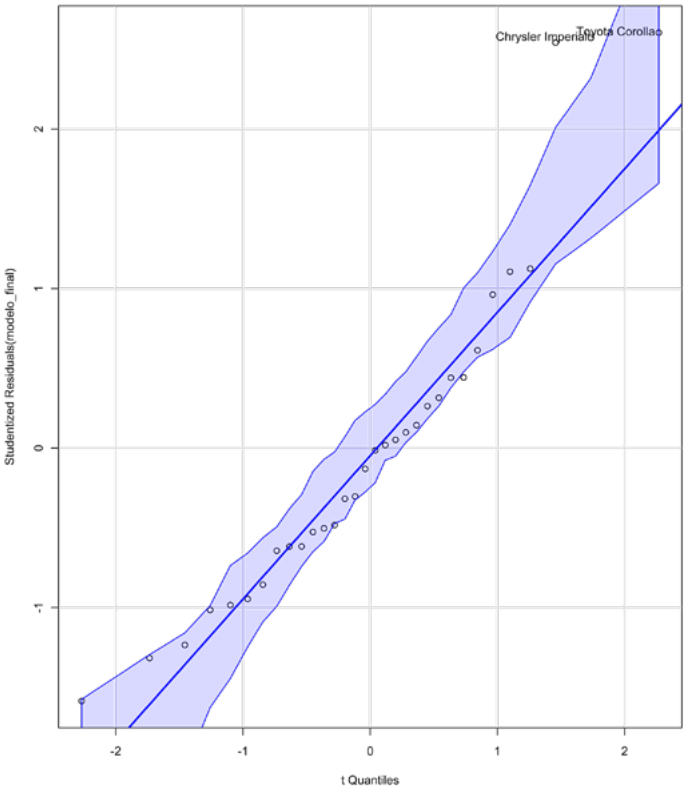
```
lillie.test(modelo_final$residuals)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  modelo_final$residuals
## D = 0.11354, p-value = 0.3675
```

El resultado del test de normalidad de Lilliefors se presenta en dos partes: el valor de D, que es una medida de la discrepancia entre la distribución empírica de los datos y la distribución teórica normal, y el valor *p*, que es la probabilidad de obtener un valor de D igual o mayor que el valor observado si la distribución subyacente de los datos

no es normal. En este caso, el valor de D es pequeño y el valor *p* es mayor que 0.05, lo que indica que no hay suficiente evidencia para rechazar la hipótesis nula de que los datos están distribuidos normalmente. Por lo tanto, se concluye que los residuos del modelo están distribuidos normalmente y que el supuesto de normalidad en los residuos del modelo se cumple.

Realizamos un gráfico QQ para observar la distribución de los residuos. Este gráfico nos muestra la distribución de los residuos del modelo de regresión lineal y los compara con una distribución normal teórica. Si hay observaciones que se destacan en el gráfico, como en este caso, significa que estas observaciones tienen residuos que son significativamente diferentes de lo que se esperaría bajo la distribución normal.

```
qqPlot(modelo_final)
```



```
## Chrysler Imperial    Toyota Corolla
##                17                20
```

Por lo tanto, Chrysler Imperial y Toyota Corolla son observaciones que pueden ser consideradas como valores atípicos o inusuales en términos de sus residuos en el modelo. Por lo tanto, si hay observaciones destacadas en el gráfico QQ-plot, puede ser útil examinarlas más detenidamente y determinar si es necesario tomar medidas para manejarlas.

Homocedasticidad

```
lmtest::bgtest(modelo_final)

##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  modelo_final
## LM test = 3.4777, df = 1, p-value = 0.0622
```

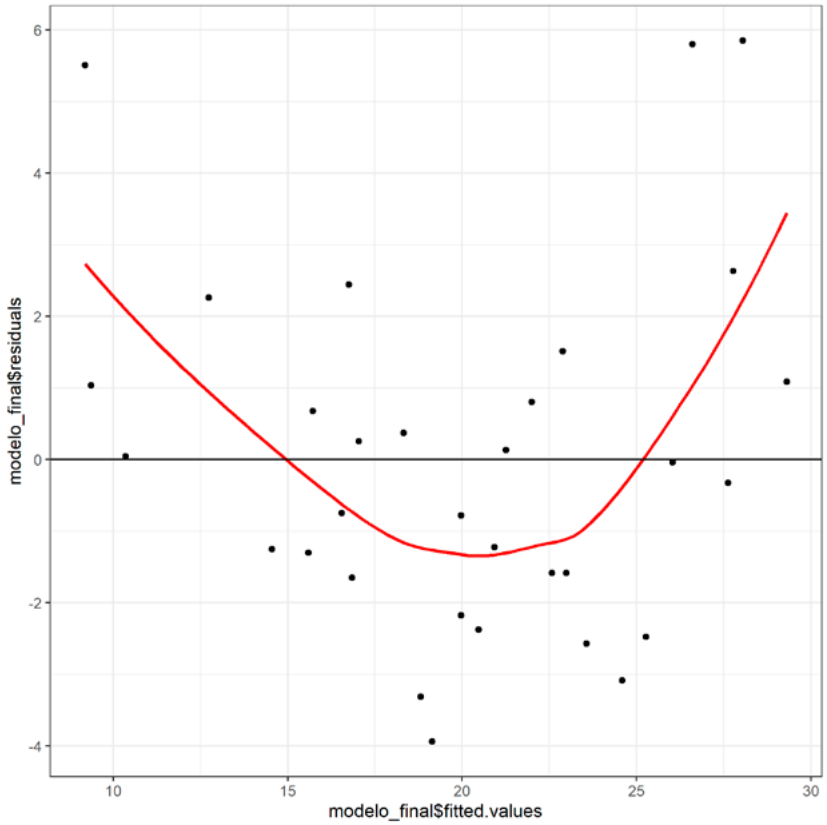
No se rechaza la hipótesis de homocedasticidad.

El test de Breusch-Godfrey para autocorrelación de orden hasta 1 indica que el valor *p* es mayor que el nivel de significancia de 0.05, lo que significa que no hay suficiente evidencia estadística para rechazar la hipótesis nula de que no hay autocorrelación de orden 1 en los residuos del modelo (es decir, los residuos son independientes entre sí). La hipótesis alternativa es que sí hay autocorrelación. Por lo tanto, se puede asumir que los residuos son independientes y el supuesto de homocedasticidad se cumple.

Graficamos:

```
ggplot(data = data, aes(modelo_final$fitted.values, modelo_
final$residuals)) +
  geom_point() +
  geom_smooth(color = "red", se = FALSE)  +
```

```
# FALSE para que no aparezca el suavizado de fondo
geom_hline(yintercept = 0) + theme_bw()
```



El gráfico de residuos vs. valores ajustados se utiliza para evaluar la calidad del ajuste de un modelo de regresión lineal. La línea horizontal en el valor 0 representa el valor esperado de los residuos en caso de que no haya ninguna relación entre las variables predictoras y la variable de respuesta. Por otro lado, la línea roja convexa hacia arriba en el gráfico no necesariamente indica heterocedasticidad, sino que podría ser interpretada como la línea de ajuste suavizado (`geom_smooth`) que se traza para resaltar la tendencia general en los datos.

Asimismo, se observa una distribución aleatoria y relativamente homogénea de los residuos alrededor de la línea horizontal, lo cual indica que los supuestos del

modelo de regresión lineal se cumplen y que el modelo es adecuado para explicar la variabilidad en los datos.

Modificador de efecto

La presencia o ausencia de un modificador de efecto puede cambiar la asociación de una exposición con el resultado de interés.

◊ *Modelos 4 y 5*

En la regresión lineal múltiple, el operador `*` indica la interacción entre las variables predictoras "wt" y "hp". La inclusión de términos de interacción se utiliza para modelar cómo el efecto de una variable predictora en la variable respuesta puede depender del valor de otra variable predictora.

Si se utilizara el operador `+` en lugar de `*`, se estarían incluyendo solo los efectos principales de ambas variables predictoras y no la interacción entre ellas. Por lo tanto, el uso del operador `*` es necesario para modelar la posible interacción entre las variables predictoras y su efecto combinado en la variable respuesta.

```
mod4 <-lm(mpg ~ wt*hp, data =data)

summary(mod4)

##
## Call:
## lm(formula = mpg ~ wt * hp, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0632 -1.6491 -0.7362  1.4211  4.5513
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.80842    3.60516  13.816 5.01e-14 ***
```

```
## wt          -8.21662    1.26971  -6.471 5.20e-07 ***
## hp          -0.12010    0.02470  -4.863 4.04e-05 ***
## wt:hp        0.02785    0.00742   3.753 0.000811 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 2.153 on 28 degrees of freedom
## Multiple R-squared:  0.8848, Adjusted R-squared:  0.8724
## F-statistic: 71.66 on 3 and 28 DF,  p-value: 2.981e-13

mod5 <- lm(mpg ~  wt*hp + wt*disp, data =data)
summary(mod5)

##
## Call:
## lm(formula = mpg ~ wt * hp + wt * disp, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0872 -1.4986 -0.7324  1.5193  4.4707
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.988882   4.483253  11.150 2.10e-11 ***
## wt          -8.143227   1.616487  -5.038 3.04e-05 ***
## hp          -0.131901   0.067149  -1.964   0.0603 .
## disp         0.002847   0.028831   0.099   0.9221
## wt:hp        0.031830   0.019414   1.640   0.1132
## wt:disp     -0.001644   0.007698  -0.214   0.8325
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 2.227 on 26 degrees of freedom
## Multiple R-squared:  0.8855, Adjusted R-squared:  0.8635
```

```
## F-statistic: 40.21 on 5 and 26 DF,  p-value: 1.989e-11
```

El modelo 4, presenta dos variables predictoras ("wt" y "hp") y su interacción ("wt:hp") para predecir el valor de la variable dependiente "mpg" (millas por galón). Se encontró que todas las variables predictoras son significativas ($p<0.05$), lo que indica que tanto el peso como la potencia del motor influyen en el consumo de combustible y que existe una interacción significativa entre ambas variables. Asimismo, el modelo explica el 88.48% de la variabilidad en el consumo de combustible.

El modelo 5 de regresión lineal múltiple se ajustó con tres predictores: "wt" (peso del vehículo), "hp" (caballos de fuerza) y "disp." (desplazamiento), considerando las interacciones entre ellos ("wt:hp" y "wt:disp."). Los coeficientes asociados al intercepto y la variable "wt" son significativos al nivel de significancia del 0.05, indicado por los símbolos "***". Sin embargo, el coeficiente asociado a la variable "disp" y "hp" no son significativos, indicado por un valor *p* mayor a 0.05. Tampoco los coeficientes asociados a las interacciones "wt:hp" y "wt:disp" son significativos al nivel de 0.05, aunque el coeficiente "wt:hp" es cercano a ser significativo, con un valor de p de 0.1132.

◊ *Comparando el modelo 4 con el modelo final*

Se eligió el modelo 4 para la comparación con el modelo final porque todos sus coeficientes son significativos con valores *p* menores a 0.05, lo que indica que son importantes para explicar la variabilidad en la variable de respuesta "mpg". En cambio, en el modelo 5, el coeficiente para "hp", "disp." y sus interacciones no eran significativos, lo que indica que no contribuyen significativamente a la variabilidad de la variable de respuesta. Por lo tanto, el modelo 4 es mejor que el modelo 5 en términos de ajuste a los datos y explicación de la variabilidad del "mpg".

```
#Definir la lista de modelos
models <- list(modelo_final, mod4)

#Especificar los nombres del modelo
mod.names_effect <- c('mpg.wt+hp','mpg.hp.wt')
```

```
#Calcular el AIC de cada modelo
aictab(cand.set = models, modnames = mod.names_effect)
##
## Model selection based on AICc:
##
##           K   AICc Delta_AICc AICcWt Cum.Wt     LL
## mpg.hp.wt 5 147.92       0.00   0.99   0.99 -67.81
## mpg.wt+hp 4 158.13      10.22   0.01   1.00 -74.33
```

Interpretación: el nuevo modelo final sería el modelo 4 porque tiene un AICc significativamente menor (reducción >4%) que el AICc del modelo final, lo que significa que tiene una mejor bondad de ajuste.

Resumen del capítulo

Antes de realizar un análisis de regresión es importante saber qué variables podrían ingresar al modelo. Por eso, un paso importante es realizar un análisis de correlación para observar qué variables pueden tener un mayor coeficiente de correlación. Así como un análisis previo es importante, en una regresión lineal también se tiene que comprobar que el modelo final es el óptimo. Esto se puede verificar a partir de los supuestos de linealidad, independencia de los errores, normalidad y homocedasticidad, asimismo, un modificador de efecto podría también ajustar mejor nuestro modelo y convertirse en nuestro modelo final, como sucedió en el ejemplo aquí presentado.

CAPÍTULO 8

REGRESIÓN LOGÍSTICA

En este capítulo deseamos conocer cómo la presencia o ausencia de aves introducidas en Nueva Zelanda puede ser predicha por más de una variable explicativa. La regresión logística nos permite responder a este tipo de pregunta, es por ello que en primer lugar realizaremos un análisis previo para conocer los mejores posibles predictores del modelo. Luego, ajustaremos el modelo de regresión logística paso a paso y realizaremos un diagnóstico que confirme nuestro modelo final.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Realizar un análisis previo a una regresión logística para identificar los mejores predictores para el modelo.
- Aplicar los métodos de selección de variables forward y stepwise en modelos de regresión logística.
- Realizar un análisis post regresión.

Paquetes

```
# Instalación
#install.packages("readr")
#install.packages("dplyr")
#install.packages("Amelia")
#install.packages("VIM")
#install.packages("ggplot2")
#install.packages("mlogit")
#install.packages("gridExtra")
#install.packages("AICcmodavg")
#install.packages("PerformanceAnalytics")
```

```
#install.packages("corrplot")
#install.packages("aod")
#install.packages("car")


# Llamamos a los paquetes
library(readr) #leer archivos
library(dplyr) #tratamiento de datos
library(Amelia) #valores perdidos
library (VIM) #visualización de valores perdidos
library(mlogit) # regresión logistica
library(ggplot2) #graficos
library(gridExtra)#validacion de condiciones para la regresion
library(AICcmodavg) # Método Forward
library(PerformanceAnalytics) #Analisis de correlación
library(corrplot) #exploración visual en la matriz de correlación
library(car) #multicolinealidad
```

Estudio de caso

En el siglo XIX, muchas personas intentaron llevar sus especies de aves a Nueva Zelanda, liberarlas y esperar a que se establecieran en la naturaleza. Veltman et al. (1996) querían saber qué determinaba el éxito o el fracaso de estas especies introducidas. Determinaron la presencia de 79 especies de aves introducidas artificialmente (variable dependiente "Status") y 14 variables independientes. La regresión logistíca se empleará para predecir la probabilidad de éxito de la introducción de una nueva especie de ave en ese país. Este tipo de regresión se caracteriza por su método para modelar una variable dependiente binaria, es decir, aquella que toma los valores 1 y 0. Las variables independientes o explicativas pueden ser continuas o categóricas.

Creamos una tabla con los valores a utilizar:

```
aves <- ("Species   Status Length Mass Range Migr Insect Diet
Clutch Broods Wood Upland Water Release Indiv
 Cyg_olor  1      1520  9600 1.21  1    12     2     6     1
0    0      1      6        29
 Cyg_atra  1      1250  5000 0.56  1     0     1     6     1
0    0      1     10        85
 Cer_nova  1       870  3360 0.07  1     0     1     4     1
0    0      1      3         8
 Ans_caer  0       720  2517 1.1   3    12     2     3.8   1
0    0      1      1        10
 Ans_anse  0       820  3170 3.45  3     0     1     5.9   1
0    0      1      2         7
 Bra_cana  1       770  4390 2.96  2     0     1     5.9   1
0    0      1     10        60
 Bra_sand  0        50  1930 0.01  1     0     1     4     2
0    0      0      1         2
 Alo_aegy  0       680  2040 2.71  1    NA     2     8.5   1
0    0      1      1         8
 Ana_plat  1       570  1020 9.01  2     6     2    12.6   1
0    0      1     17      1539
 Ana_acut  0       580   910 7.9   3     6     2     8.3   1
0    0      1      3       102
 Ana_pene  0       480   590 4.33  3     0     1     8.7   1
0    0      1      5        32
 Aix_spon  0       470   539 1.04  3    12     2    13.5   2
1    0      1      5        10
 Ayt_feri  0       450   940 2.17  3    12     2     9.5   1
0    0      1      3         9
 Ayt_fuli  0       435   684 4.81  3    12     2    10.1   1
0    0      1      2         5
 Ore_pict  0       275   230 0.31  1     3     1     9.5   1
1    1      0      9       398
 Lop_cali  1       256   162 0.24  1     3     1    14.2   2
0    0      0     15      1420
```

```
 Col_virg  1       230   170 0.77  1     3     1    13.7   1
0    0      0     17      1156
 Ale_grae  1       330   501 2.23  1     3     1    15.5   1
0    1      0     15       362
 Ale_rufa  0       330   439 0.22  1     3     2    11.2   2
0    0      0      2        20
 Per_perd  0       300   386 2.4   1     3     1    14.6   1
0    1      0     24       676
 Cot_pect  0       182    95 0.33  3    NA     2     7.5   1
0    0      0      3       NA
 Cot_aust  1       180    95 0.69  2    12     2    11     1
0    0      1     11       601
 Lop_nyct  0       800  1150 0.28  1    12     2     5     1
1    1      0      4         6
 Pha_colc  1       710   850 1.25  1    12     2    11.8   1
1    0      0     27       244
 Syr_reev  0       750   949 0.2   1    12     2     9.5   1
1    1      0      2         9
 Tet_tetr  0       470   900 4.17  1     3     1     7.9   1
1    1      0      2        13
 Lag_lago  0       390   517 7.29  1     0     1     7.5   1
1    1      0      2         4
 Ped_phas  0       440   815 1.83  1     3     1    12.3   1
1    0      0      1        22
 Tym_cupi  0       435   770 0.26  1     4     1    12     1
0    0      0      3        57
 Van_vane  0       300   226 3.93  2    12     3     3.8   1
0    0      0      8       124
 Plu_squa  0       285   318 1.67  3    12     3     4     1
0    0      1      2         3
 Pte_alch  0       350   225 1.21  2     0     1     2.5   2
0    0      0      1         8
 Pha_chal  0       320   350 0.6   1    12     2     2     2
1    0      0      8        42
```

```
 Ocy_loph  0       330   205 0.76  1     0     1     2     7
1    0      1      4        23
 Leu_mela  0       372  NA   0.07  1    12     2     2     1
1    0      0      6        34
 Ath_noct  1       220   176 4.84  1    12     3     3.6   1
1    0      0      7       221
 Tyt_alba  0       340   298 8.9   2     0     3     5.7   2
1    0      0      1         7
 Dac_nova  1       460   382 0.34  1    12     3     2     1
1    0      0      7        21
 Lul_arbo  0       150  32.1 1.78  2     4     2     3.9   2
1    0      0      1         5
 Ala_arve  1       185  38.9 5.19  2    12     2     3.7   3
0    0      0     11       391
 Pru_modu  1       145  20.5 1.95  2    12     2     3.4   2
1    0      0     14       245
 Eri_rebe  0       140  15.8 2.31  2    12     2     5     2
1    0      0     11       123
 Lus_mega  0       161  19.4 1.88  3    12     2     4.7   2
1    0      0      4         7
 Tur_meru  1       255  82.6 3.3   2    12     2     3.8   3
1    0      0     16       596
 Tur_phil  1       230  67.3 4.84  2    12     2     4.7   2
1    0      0     12       343
 Syl_comm  0       140  12.8 3.39  3    12     2     4.6   2
1    0      0      1         2
 Syl_atri  0       142  17.5 2.43  2     5     2     4.6   1
1    0      0      1         5
 Man_mela  0       180  NA   0.04  1    12     3     1.9   5
1    0      0      1         2
 Man_mela  0       265    59 0.25  1    12     2     2.6   NA
1    0      0      1        80
 Gra_cyan  0       275   128 0.83  1    12     3     3     2
1    0      1      1      NA
```

```
 Gym_tibi  1       400   380 0.82  1    12     3     4     1
1    0      0     15       448
 Cor_mone  0       335   203 3.4   2    12     2     4.5   1
1    0      0     2          3
 Cor_frug  1       400   425 3.73  1    12     2     3.6   1
1    0      0     10       182
 Stu_vulg  1       222  79.8 3.33  2     6     2     4.8   2
1    0      0     14       653
 Acr_tris  1       230 111.3 0.56  1    12     2     3.7   1
1    0      0     5         88
 Pas_dome  1       149  28.8 6.5   1     6     2     3.9   3
1    0      0     12       416
 Pas_mont  0       133    22 6.8   1     6     2     4.7   3
1    0      0     3         14
 Aeg_temp  0       120  NA   0.17  1     6     2     4.7   3
1    0      0     3         14
 Emb_gutt  0       120    19 0.15  1     4     1     5     3
0    0      0     4        112
 Poe_gutt  0       100  12.4 0.75  1     4     1     4.7   3
0    0      0     1         12
 Lon_punc  0       110  13.5 1.06  1     0     1     5     3
0    0      0     1          8
 Lon_cast  0       100  NA   0.13  1     4     1     5     NA
0    0      1     4         45
 Pad_oryz  0       160  NA   0.09  1     0     1     5     NA
0    0      0     2          6
 Fri_coel  1       160  23.5 2.61  2    12     2     4.9   2
1    0      0     17       449
 Fri_mont  0       146  21.4 3.09  3    10     2     6     NA
1    0      0     7        121
 Car_chlo  1       147  29   2.09  2     7     2     4.8   2
1    0      0     6         65
 Car_spin  0       117  12   2.09  3     3     1     4     2
1    0      0     3         54
```

```
 Car_card  1       120  15.5 2.85  2     4     1     4.4   3
1    0      0     14       626
 Aca_flam  1       115  11.5 5.54  2     6     1     5     2
1    0      0     10       607
 Aca_flavi 0       133  17   1.67  2     0     1     5     3
0    1      0      3        61
 Aca_cann  0       136  18.5 2.52  2     6     1     4.7   2
1    0      0     12       209
 Pyr_pyrr  0       142  23.5 3.57  1     4     1     4     3
1    0      0      2       NA
 Emb_citr  1       160  28.2 4.11  2     8     2     3.3   3
1    0      0     14       656
 Emb_hort  0       163  21.6 2.75  3    12     2     5     1
0    0      0      1         6
 Emb_cirl  1       160  23.6 0.62  1    12     2     3.5   2
1    0      0      3        29
 Emb_scho  0       150  20.7 5.42  1    12     2     5.1   2
0    0      1      2         9
 Pir_rubr  0       170  31   0.55  3    12     2     4     NA
1    0      0      1         2
 Age_phoe  0       210  36.9 2     2     8     2     3.7   1
0    0      1      1         2
 Stu_negl  0       225 106.5 1.2   2    12     2     4.8   2
0    0      0      1         2")
```

Debido a la forma de ingresar los datos, la variable "aves" es de tipo caracter, pero es necesario convertirla en un *data frame* para almacenar y manipular datos de manera eficiente. Para ello se empleará las funciones `textConnection` y `read.table`. La primera se utiliza para crear un objeto de conexión de texto que permite leer o escribir datos desde o hacia una cadena de caracteres. Mientras que la segunda se utiliza para leer datos de un archivo de texto y generar un *data frame*. En particular, `read.table` es útil para leer archivos de texto que contienen datos tabulares con valores separados por espacios en blanco o tabuladores, como en nuestro caso. Asimismo,

especificamos en el argumento `header` la palabra `TRUE` que indica que la primera fila corresponde a los nombres de las variables.

```
Data <- read.table(textConnection(aves),header=TRUE)
```

Análisis exploratorio

```
#nombres de las variables
names(Data)

##  [1] "Species" "Status"  "Length"  "Mass"    "Range"   "Migr"
"Insect"
##  [8] "Diet"    "Clutch"  "Broods"  "Wood"    "Upland"  "Water"
"Release"
## [15] "Indiv"

#estructura de los datos
str(Data)

## 'data.frame':    79 obs. of  15 variables:
##  $ Species: chr  "Cyg_olor" "Cyg_atra" "Cer_nova" "Ans_caer"
...
##  $ Status : int  1 1 1 0 0 1 0 0 1 0 ...
##  $ Length : int  1520 1250 870 720 820 770 50 680 570 580 ...
##  $ Mass   : num  9600 5000 3360 2517 3170 ...
##  $ Range  : num  1.21 0.56 0.07 1.1 3.45 2.96 0.01 2.71 9.01
7.9 ...
##  $ Migr   : int  1 1 1 3 3 2 1 1 2 3 ...
##  $ Insect : int  12 0 0 12 0 0 0 NA 6 6 ...
##  $ Diet   : int  2 1 1 2 1 1 1 2 2 2 ...
##  $ Clutch : num  6 6 4 3.8 5.9 5.9 4 8.5 12.6 8.3 ...
##  $ Broods : int  1 1 1 1 1 1 2 1 1 1 ...
##  $ Wood   : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ Upland : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Water  : int  1 1 1 1 1 1 0 1 1 1 ...
##  $ Release: int  6 10 3 1 2 10 1 1 17 3 ...
##  $ Indiv  : int  29 85 8 10 7 60 2 8 1539 102 ...
```

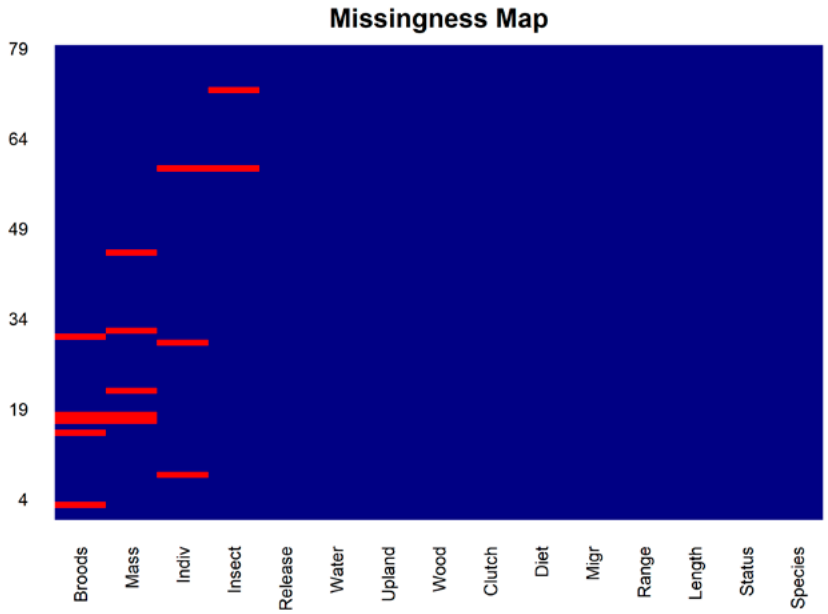
La variable dependiente "Status" presenta valores enteros numéricos al igual que la mayoría de las variables. La variable "Species" es la única de tipo categórico.

Valores perdidos

```
table(complete.cases(Data)) # Explorar

##
## FALSE  TRUE
##    12    67
```

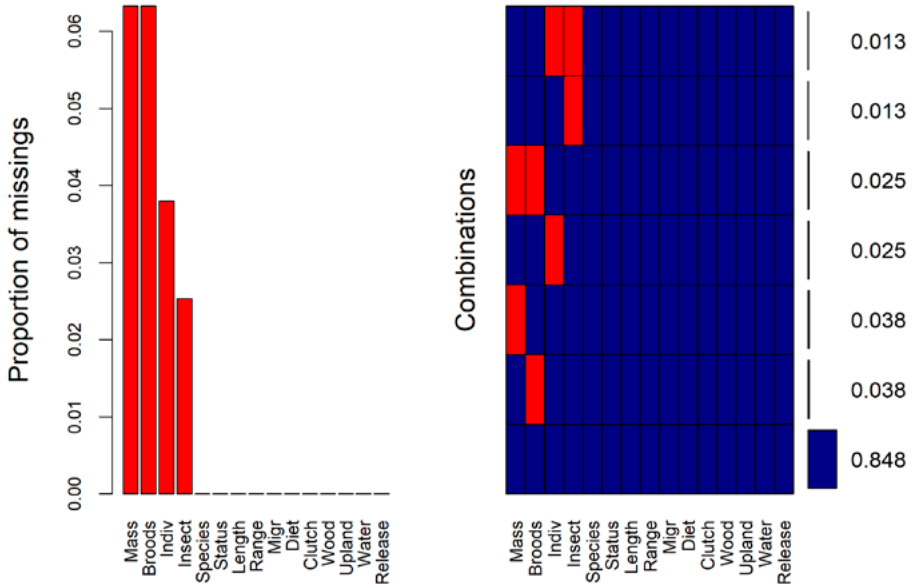
Observamos que hay 12 valores perdidos. Vamos a graficar los datos perdidos con la función `missmap` del paquete `Amelia` y le asignaremos el color rojo a los valores faltantes. El eje *x* muestra atributos y el eje *y* muestra instancias. Las líneas horizontales indican datos faltantes para una instancia, los bloques verticales representan datos faltantes para un atributo o variable.

```
missmap(Data, col=c("red", "navyblue"), legend=FALSE)
```



Vemos que hay valores perdidos. Otra opción para graficar los valores perdidos es utilizando la función `aggr` del paquete `VIM` que nos permite visualizar la estructura de los datos faltantes. El argumento `sortVars=TRUE` indica que las variables deben ordenarse según el número de datos faltantes y `cex.axis` nos permite incrementar el tamaño de las etiquetas.

```
aggr_plot <- aggr(Data, col=c('navyblue','red'),
                  numbers=TRUE, sortVars=TRUE,
                   cex.axis=.8)

## Warning in plot.aggr(res, ...): not enough horizontal space
to display
## frequencies
```




```
##
##  Variables sorted by number of missings:
##  Variable      Count
##      Mass 0.06329114
##    Broods 0.06329114
##     Indiv 0.03797468
##    Insect 0.02531646
##   Species 0.00000000
##    Status 0.00000000
##    Length 0.00000000
##     Range 0.00000000
##      Migr 0.00000000
##      Diet 0.00000000
##    Clutch 0.00000000
##      Wood 0.00000000
##    Upland 0.00000000
```

```
##     Water 0.00000000
##   Release 0.00000000
```

Son 4 las variables que tienen valores perdidos: "Mass", "Broods", "Indiv" e "Insect".

Eliminamos los valores perdidos y nos quedamos solo con los casos completos.

```
Data <-Data[complete.cases(Data), ]
table(complete.cases(Data)) #comprobamos

##
## TRUE
##   67
```

Observamos que ya no figura `FALSE`.

Descripción de los datos

Veamos la variable dependiente:

```
table(Data$Status)

##
##  0  1
## 40 27
```

Solo toma los valores "0" y "1".

Veamos, ahora, todas las variables.

```
summary(Data)

##    Species              Status          Length            Mass
##  Length:67          Min.   :0.000   Min.   :  50.0   Min.   :
11.50
##  Class :character   1st Qu.:0.000   1st Qu.: 150.0   1st Qu.:
```

```
23.55
##  Mode  :character   Median :0.000   Median : 256.0   Median :
203.00
##                     Mean   :0.403   Mean   : 346.7   Mean   :
697.47
##                     3rd Qu.:1.000   3rd Qu.: 437.5   3rd Qu.:
637.00
##                     Max.   :1.000   Max.   :1520.0   Max.
:9600.00
##      Range             Migr           Insect            Diet
##  Min.   :0.010   Min.   :1.000   Min.   : 0.000   Min.
:1.000
##  1st Qu.:0.765   1st Qu.:1.000   1st Qu.: 3.000   1st
Qu.:1.000
##  Median :2.090   Median :2.000   Median : 6.000   Median
:2.000
##  Mean   :2.577   Mean   :1.716   Mean   : 7.119   Mean
:1.716
##  3rd Qu.:3.590   3rd Qu.:2.000   3rd Qu.:12.000   3rd
Qu.:2.000
##  Max.   :9.010   Max.   :3.000   Max.   :12.000   Max.
:3.000
##      Clutch          Broods           Wood            Upland
##  Min.   : 2.00   Min.   :1.000   Min.   :0.0000   Min.
:0.0000
##  1st Qu.: 3.95   1st Qu.:1.000   1st Qu.:0.0000   1st
Qu.:0.0000
##  Median : 4.80   Median :1.000   Median :1.0000   Median
:0.0000
##  Mean   : 6.27   Mean   :1.716   Mean   :0.5373   Mean
:0.1194
##  3rd Qu.: 8.10   3rd Qu.:2.000   3rd Qu.:1.0000   3rd
Qu.:0.0000
##  Max.   :15.50   Max.   :7.000   Max.   :1.0000   Max.
```

```
:1.0000
##      Water           Release          Indiv
##  Min.   :0.0000   Min.   : 1.00   Min.   :   2.0
##  1st Qu.:0.0000   1st Qu.: 2.00   1st Qu.:   8.0
##  Median :0.0000   Median : 4.00   Median :  42.0
##  Mean   :0.2537   Mean   : 6.94   Mean   : 205.4
##  3rd Qu.:0.5000   3rd Qu.:11.00   3rd Qu.: 294.0
##  Max.   :1.0000   Max.   :27.00   Max.   :1539.0
```

Conversión de variables

Trabajaremos solo con variables numéricas a fin de realizar el análisis de correlación múltiple e identificar los mejores predictores para el modelo. Primero eliminamos la variable "Species" porque no es numérica.

```
Data$Species <- NULL
```

Ahora, convertiremos a numéricas solo aquellas variables que no lo sean, en este caso son 11 de las 14 del *data frame*.

```
#Convertimos las variables en númericas

Data$Status  = as.numeric(Data$Status)
Data$Length  = as.numeric(Data$Length)
Data$Migr    = as.numeric(Data$Migr)
Data$Insect  = as.numeric(Data$Insect)
Data$Diet    = as.numeric(Data$Diet)
Data$Broods  = as.numeric(Data$Broods)
Data$Wood    = as.numeric(Data$Wood)
Data$Upland  = as.numeric(Data$Upland)
Data$Water   = as.numeric(Data$Water)
Data$Release = as.numeric(Data$Release)
Data$Indiv   = as.numeric(Data$Indiv)
```

Analisis de correlación

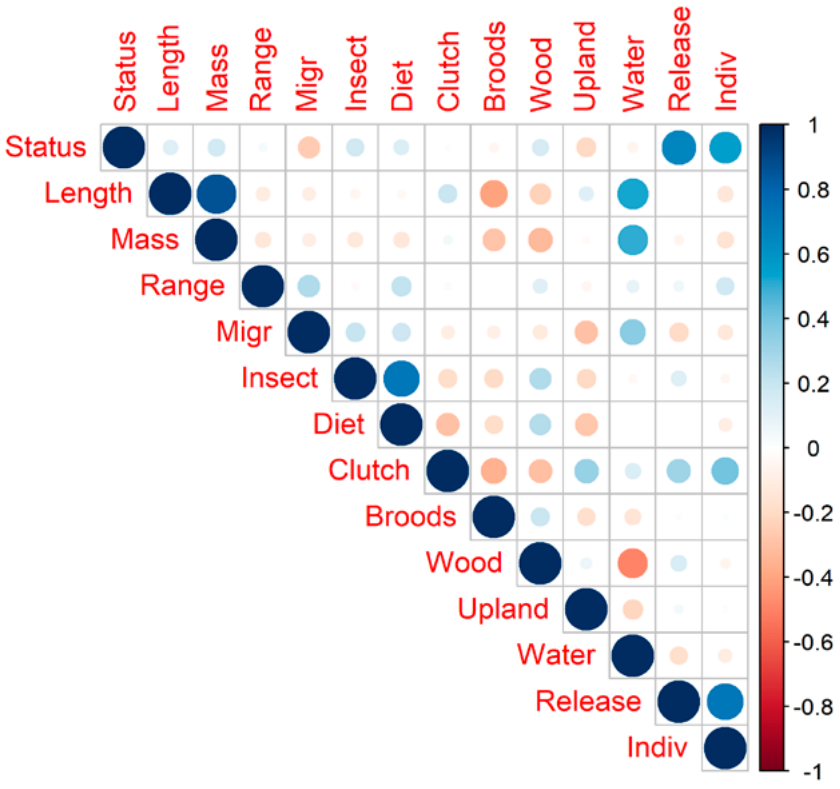
Con el paquete Corrplot

El paquete `corrplot` proporciona una herramienta de exploración visual en la matriz de correlación que admite el reordenamiento automático de variables para ayudar a detectar patrones ocultos entre ellas.

Hay siete métodos de visualización en el paquete: `circle`, `square`, `ellipse`, `number`, `shade`, `color`, `pie`. Para este caso utilizaremos el método circle. La intensidad del color es proporcional a los coeficientes de correlación. El azul representa una correlación positiva y el rojo una negativa. A mayor tamaño del círculo que se dibuja en el gráfico, mayor será la correlación.

Asimismo, el argumento `type=”upper”` nos muestra solo el triángulo superior de la matriz de correlación. Antes de realizar el gráfico, obtenemos la correlación de las 14 variables con la función `cor` y la guardamos en el objeto **correlations**.

```
correlations <- cor(Data[,1:14])

corrplot(correlations, type=”upper”, method = “circle”)
```

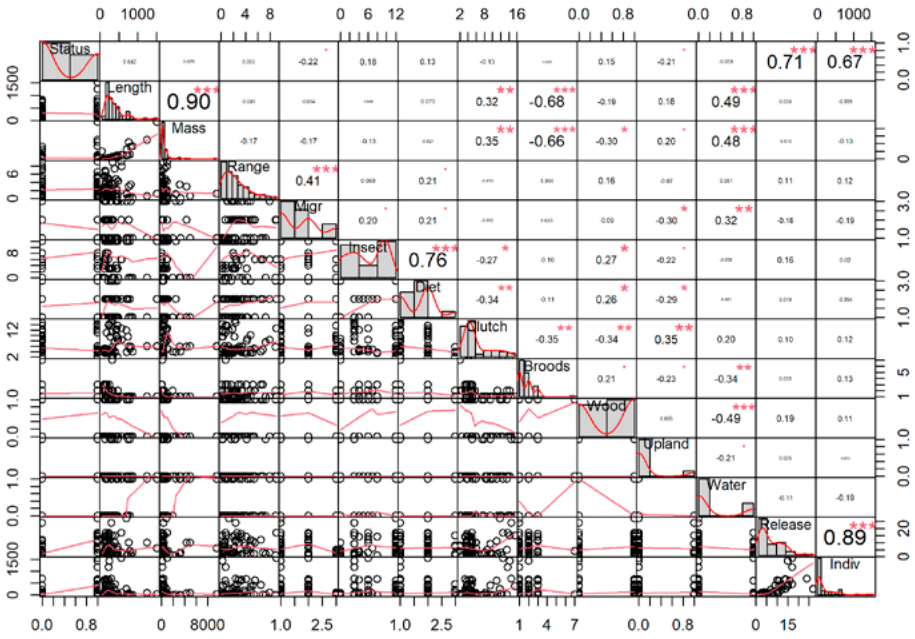


Observamos que la variable dependiente "Status" tiene una fuerte correlación con las variables "Release" e "Indiv" y, en menor medida, con las variables "Migr", "Insect" y "Mass".

Con el paquete PerformanceAnalytics

La función `chart.Correlation` del paquete `PerformanceAnalytics` nos permite visualizar la distribución de cada variable en diagonal. Asimismo, en la parte superior de la diagonal se muestra el valor de la correlación y las estrellas que indican su nivel de significancia, los valores más grandes son los más relevantes. En la parte inferior se muestran los gráficos de dispersión bivariada con una línea ajustada.

```
chart.Correlation(Data,
                  method=”spearman”,
                  histogram=TRUE)
```



Regresión logística

Los modelos de regresión logística expresan la asociación entre una variable de desenlace *Y* dicotómica y una o más variables independientes *X*. Este tipo de regresión nos ayuda a predecir la probabilidad de que una observación pertenezca a una categoría particular, según sus características, o si un evento ocurrirá o no. Por ejemplo, en el caso de estimar la probabilidad de la ocurrencia de un evento, si tenemos información sobre una especie de ave que tiene una masa de 50 g, vive en zonas de montaña y tiene una dieta de insectos, podemos utilizar la regresión logística para predecir la probabilidad de que esta especie tenga éxito en su introducción en Nueva Zelanda. Y en el caso de pertenencia a una categoría particular, si queremos saber si la especie de ave X que tiene una longitud de 30 cm, se alimenta de

insectos y vive en zonas boscosas tiene éxito o fracaso en su introducción en el país, podemos utilizar la regresión logística para determinar a qué categoría pertenece, es decir, si se considera una especie introducida exitosa o no.

La regresión logística es una alternativa de solución no paramétrica altamente recomendable cuando la prevalencia del evento de interés es rara (<10%). Para este ejemplo, realizaremos un análisis de regresión logística con `glm`, usando el argumento `family=binomial`. La probabilidad es la relación entre el número de eventos favorables a algún resultado y el número total de eventos. Sin embargo, es importante distinguir la probabilidad de los *odds*, que son la razón de la probabilidad de que el evento suceda dividido por la probabilidad de que el evento no suceda. Mientras que la probabilidad se expresa mediante un valor entre cero y uno, los *odds* se expresan con valores que están entre cero y el infinito.

Utilizaremos solo 6 variables para la regresión logística:

Variable	**Descripción**
Status (variable dependiente)	Ausencia o presencia de la especie en Nueva Zelanda
Release	Número de liberaciones
Migr	1: sedentarias, 2: mixtas, 3: migratorias
Mass	Masa corporal
Indiv	Número de individuos liberados
Insect	Dieta en base a insectos

Guardamos las 6 variables en el objeto **Data_Final**

```
Data_Final <- Data %>% select(Status, Release, Migr, Mass, Indiv,
Insect)
```

Método forward

Modelo nulo

```
modelo_nulo = glm(Status ~ 1,
                  data=Data_Final,
                  family = binomial)

summary(modelo_nulo)

##
## Call:
## glm(formula = Status ~ 1, family = binomial, data = Data_
Final)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.016  -1.016  -1.016   1.348   1.348
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.3930     0.2491  -1.578    0.115
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 90.343  on 66  degrees of freedom
## AIC: 92.343
##
## Number of Fisher Scoring iterations: 4

#Definir el modelo
modelo_0 <- list(modelo_nulo)
```

```
#Especificar el nombre del modelo
mod.names0 <- c('status')

#Calcular el AIC del modelo nulo
aictab(cand.set = modelo_0, modnames = mod.names0)

##
## Model selection based on AICc:
##
##        K AICc Delta_AICc AICcWt Cum.Wt     LL
## status 1 92.4          0      1      1 -45.17
```

El AIC del modelo nulo es 92.4.

Modelos de primer orden

```
modelo_1.1 <- glm(Status ~ Release, data = Data_Final, family =
binomial )
summary(modelo_1.1)

##
## Call:
## glm(formula = Status ~ Release, family = binomial, data =
Data_Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2454  -0.5745  -0.4202   0.5188   1.9407
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.71511    0.61118  -4.442 8.90e-06 ***
## Release      0.33234    0.07554   4.400 1.08e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 56.130  on 65  degrees of freedom
## AIC: 60.13
##
## Number of Fisher Scoring iterations: 5
```

Interpretación del *output* de un modelo de regresión logística:

- El modelo (call): glm(formula=Status ~ Release, family = binomial, data = Data_Final)
- Los residuos de desviación miden la diferencia entre las probabilidades estimadas del modelo y las proporciones observadas de éxitos. En términos simples, los residuos de desviación nos dan una idea de la dispersión de los errores para los casos individuales utilizados en el modelo.
- El coeficiente estimado para el intercepto, -2.71511, es la probabilidad de presencia de aves ("Status") cuando el número de liberaciones ("Release") es igual a cero. Por otro lado, la presencia de aves está positivamente relacionada con el número de liberaciones. Esto quiere decir que por un aumento de la variable "Release" se espera que el logaritmo de *odds ratio* de la presencia de aves se incremente en 0.33234 unidades, en comparación con la ausencia de aves. La salida también nos muestra la estadística z de Wald, así como el valor *p* Pr(>|z|), la cual nos dice la probabilidad asociada con un valor *z* particular. Básicamente, la variable "Release" es estadísticamente significativa.
- La desviación nula (90.343 con 66 grados de libertad) se basa en el modelo solo de la intercepción. Mientras que la desviación residual (56.130 con 65 grados de libertad) nos dice qué tan bien se puede predecir la variable de respuesta mediante un modelo con *p* variables predictoras. Cuanto menor sea el valor, mejor podrá predecir el valor de la variable de respuesta. La diferencia entre la desviación nula y la residual con *p* grados de libertad nos da el chi cuadrado y este nos permite conocer que tan útil es el modelo. En este ejemplo, el chi cuadrado es igual a 34.213 con 1 grado de libertad y tiene un valor *p* de

0.000000, al ser menor que 0.05 concluimos que la variable “Release” predice la probabilidad de presencia de aves.

```
modelo_1.2 <- glm(Status ~ Migr, family = binomial, data  =
Data_Final )
summary(modelo_1.2)

##
## Call:
## glm(formula = Status ~ Migr, family = binomial, data = Data_
Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2255  -1.0720  -0.6634   1.1301   1.4606
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.8696     0.6475   1.343   0.1793
## Migr         -0.7572     0.3683  -2.056   0.0398 *
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 85.704  on 65  degrees of freedom
## AIC: 89.704
##
## Number of Fisher Scoring iterations: 4

modelo_1.3 <- glm(Status ~ Mass, family = binomial, data = Data_
Final )
summary(modelo_1.3)
```

```
##
## Call:
## glm(formula = Status ~ Mass, family = binomial, data = Data_
Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2792  -0.9803  -0.9493   1.3946   1.4248
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.5678837  0.2833974  -2.004   0.0451 *
## Mass         0.0002536  0.0002011   1.261   0.2071
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 88.380  on 65  degrees of freedom
## AIC: 92.38
##
## Number of Fisher Scoring iterations: 4

modelo_1.4 <- glm(Status ~ Indiv,family = binomial, data = Data_
Final )
summary(modelo_1.4)

##
## Call:
## glm(formula = Status ~ Indiv, family = binomial, data = Data_
Final)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6743  -0.6268  -0.6085   0.5513   1.8721
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.623197   0.400882  -4.049 5.14e-05 ***
## Indiv        0.007649   0.002095   3.651 0.000261 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 60.692  on 65  degrees of freedom
## AIC: 64.692
##
## Number of Fisher Scoring iterations: 6

modelo_1.5 <- glm(Status ~ Insect,family = binomial, data =
Data_Final )
summary(modelo_1.5)

##
## Call:
## glm(formula = Status ~ Insect, family = binomial, data =
Data_Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1639  -1.0068  -0.8071   1.1909   1.6001
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -0.95441    0.47546  -2.007   0.0447 *
## Insect       0.07689    0.05372   1.431   0.1523
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 88.231  on 65  degrees of freedom
## AIC: 92.231
##
## Number of Fisher Scoring iterations: 4
```

Comparación de AIC

```
#Definir la lista de modelos
models1 <- list(modelo_1.1, modelo_1.2, modelo_1.3, modelo_1.4,
modelo_1.5)

#Especificar los nombres del modelo
mod.names1 <- c('st.release', 'st.migr', 'st.mass', 'st.indiv',
'st.insect')

#Calcular el AIC de cada modelo
aictab(cand.set = models1, modnames = mod.names1)

##
## Model selection based on AICc:
##
##            K  AICc Delta_AICc AICcWt Cum.Wt     LL
## st.release 2 60.32       0.00   0.91   0.91 -28.07
## st.indiv   2 64.88       4.56   0.09   1.00 -30.35
## st.migr    2 89.89      29.57   0.00   1.00 -42.85
```

```
## st.insect  2 92.42      32.10   0.00   1.00 -44.12
## st.mass    2 92.57      32.25   0.00   1.00 -44.19
```

Observamos que el modelo 1.1 es el que tiene el menor AIC, por lo que "Release" entraría primero al modelo.

Modelos de segundo orden

```
modelo_2.1 <- glm(Status ~ Release + Indiv,
                  family = binomial,data = Data_Final)
summary(modelo_2.1)

##
## Call:
## glm(formula = Status ~ Release + Indiv, family = binomial,
data = Data_Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1480  -0.5689  -0.4483   0.4782   1.9534
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.508031   0.632974  -3.962 7.42e-05 ***
## Release      0.247626   0.117321   2.111   0.0348 *
## Indiv        0.002238   0.002668   0.839   0.4015
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 55.273  on 64  degrees of freedom
## AIC: 61.273
```

```
##
## Number of Fisher Scoring iterations: 5

modelo_2.2 <- glm(Status ~ Release + Migr, family = binomial,data
= Data_Final)
summary(modelo_2.2)

##
## Call:
## glm(formula = Status ~ Release + Migr, family = binomial, data
= Data_Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4360  -0.5294  -0.3154   0.5977   1.7181
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.35350    0.97902  -1.383    0.167
## Release      0.33890    0.07915   4.282 1.85e-05 ***
## Migr        -0.87970    0.57719  -1.524    0.127
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 53.309  on 64  degrees of freedom
## AIC: 59.309
##
## Number of Fisher Scoring iterations: 5

modelo_2.3 <- glm(Status ~ Release + Mass,
                  family = binomial, data  = Data_Final)
```

```
summary(modelo_2.3)

##
## Call:
## glm(formula = Status ~ Release + Mass, family = binomial, data
= Data_Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2982  -0.5390  -0.3344   0.4583   2.1290
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.2271265  0.7304220  -4.418 9.95e-06 ***
## Release      0.3528479  0.0794506   4.441 8.95e-06 ***
## Mass         0.0005014  0.0003416   1.468    0.142
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 52.052  on 64  degrees of freedom
## AIC: 58.052
##
## Number of Fisher Scoring iterations: 5

modelo_2.4 <- glm(Status ~ Release + Insect, family = binomial,
data  = Data_Final)
summary(modelo_2.4)

##
## Call:
## glm(formula = Status ~ Release + Insect, family = binomial,
```

```
data = Data_Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1264  -0.5582  -0.3730   0.5394   2.1704
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.24459    0.86902  -3.734 0.000189 ***
## Release      0.32962    0.07585   4.346 1.39e-05 ***
## Insect       0.07110    0.07329   0.970 0.331997
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 55.157  on 64  degrees of freedom
## AIC: 61.157
##
## Number of Fisher Scoring iterations: 5
```

Comparación de AIC

```
#Definir la lista de modelos
models2 <- list(modelo_2.1, modelo_2.2, modelo_2.3, modelo_2.4)

#Especificar los nombres del modelo
mod.names2 <- c('st.release.indiv','st.release.migr',
                'st.release.mass','st.release.insect')

#Calcular el AIC de cada modelo
aictab(cand.set = models2, modnames = mod.names2)
```

```
##
## Model selection based on AICc:
##
##                    K  AICc Delta_AICc AICcWt Cum.Wt     LL
## st.release.mass    3 58.43       0.00   0.51   0.51 -26.03
## st.release.migr    3 59.69       1.26   0.27   0.79 -26.65
## st.release.insect 3 61.54       3.10   0.11   0.90 -27.58
## st.release.indiv   3 61.65       3.22   0.10   1.00 -27.64
```

Observamos que el modelo 2.3 es el que tiene el menor AIC por lo que "Release" y "Mass" se quedarían en el modelo si disminuye el AIC del modelo_2.1 en más del 4%.

Comparación de AIC entre los modelos de primer y segundo orden

```
#Definir la lista de modelos
models3 <- list(modelo_1.1, modelo_2.3)

#Especificar los nombres del modelo
mod.names3 <- c('st.release', 'st.release.mass')

#Calcular el AIC de cada modelo
aictab(cand.set = models3, modnames = mod.names3)

##
## Model selection based on AICc:
##
##                  K  AICc Delta_AICc AICcWt Cum.Wt     LL
## st.release.mass  3 58.43       0.00   0.72   0.72 -26.03
## st.release       2 60.32       1.88   0.28   1.00 -28.07
```

Dado que la variable "Mass" no disminuye el AIC en más del 4% la única variable que entra al modelo es "Release".

Odds ratio del modelo final

Para estimar el *odds ratio*, debemos exponenciar cada coeficiente del modelo de regresión logística final.

```
exp(coef(modelo_1.1))
## (Intercept)     Release
##   0.0661979   1.3942287
```

El *odds ratio* exponenciado del intercepto es 0.0661979, lo que significa que la presencia de aves es 0.0661979 veces menos probable que su ausencia cuando todas las variables predictoras tienen un valor de cero. Por otro lado, el *odds ratio* exponenciado de la variable "Release" es 1.3942287, lo que significa que un aumento de una unidad en la variable "Release" se asocia con un aumento proporcional del 1.3942287 en la *odds ratio* de la presencia de aves en comparación con la ausencia de aves, manteniendo constante el efecto de otras variables predictoras en el modelo.

Intervalo de confianza del *odds ratio*

También calculamos los intervalos de confianza para el modelo final o ajustado.

```
exp(cbind(coef(modelo_1.1), confint(modelo_1.1)))

##                             2.5 %    97.5 %
## (Intercept) 0.0661979 0.01715994 0.1956924
## Release     1.3942287 1.22088717 1.6479282
```

En dicha línea se realizan los siguientes pasos:

1. `modelo_1.1` es el nombre del modelo de regresión que se ha ajustado previamente.
2. `confint(modelo_1.1)` calcula los intervalos de confianza para los coeficientes del modelo. Estos intervalos indican el rango de valores que es probable que contenga el verdadero valor del coeficiente con un cierto nivel de confianza. Por defecto, se utiliza un nivel de confianza del 95%.

3. `coef(modelo_1.1)` extrae los coeficientes estimados del modelo de regresión.
4. `cbind(coef(modelo_1.1), confint(modelo_1.1))` combina los coeficientes estimados y los intervalos de confianza en una matriz, donde cada columna corresponde a un coeficiente y sus intervalos de confianza asociados.
5. `exp(cbind(coef(modelo_1.1), confint(modelo_1.1)))` toma la exponencial de cada elemento en la matriz anterior. Esto es útil en el contexto de modelos de regresión lineal logarítmica, en los que los coeficientes estimados se interpretan en términos de la relación log-lineal entre las variables.

Además, la función `cbind()` se utiliza para combinar vectores o matrices en una matriz, donde los vectores o matrices se concatenan por columnas. En este caso, se utiliza para combinar los coeficientes del modelo y sus intervalos de confianza en una única matriz. Mientras que la función `confint()` se utiliza para calcular los intervalos de confianza de los coeficientes de un modelo de regresión ajustado. Retorna una matriz donde cada fila corresponde a un coeficiente y sus intervalos de confianza asociados.

Respecto al resultado se infiere que, cuando se repite el análisis muchas veces, el valor del parámetro estará por debajo del intervalo de confianza en el 2.5% de los casos y por encima en otro 2.5% de los casos. Asimismo, el intervalo de confianza lo cubrirá en el 95% de los casos.

Análisis post regresión

Evaluación del modelo ajustado

Luego de seleccionar el mejor modelo es importante evaluar su ajuste. Primero lo evaluamos mediante pruebas de significación de los coeficientes y luego mediante pruebas de bondad de ajuste.

◊ *Residuos estandarizados*

Es una medida de fuerza de la diferencia entre los valores observados y esperados. Permite identificar posibles valores atípicos que pueden repercutir negativamente en la calidad de la predicción. Por lo general se considera un residuo estandarizado como atípico cuando su valor absoluto es superior a 3.

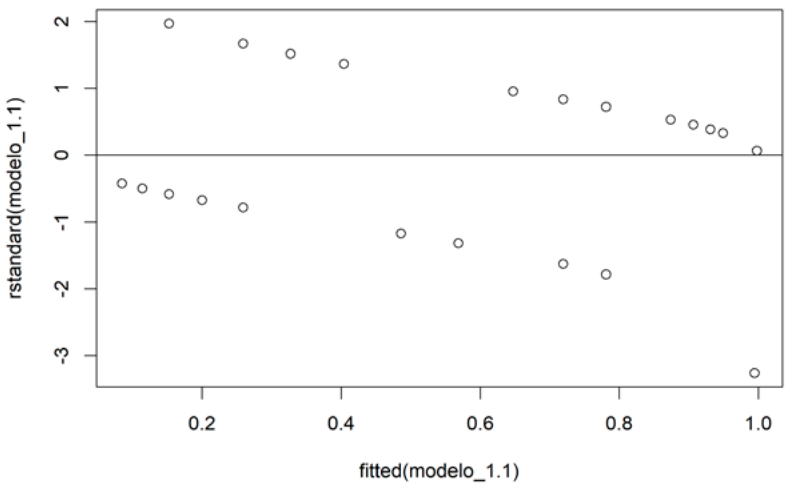
```
rstandard(modelo_1.1)

##           1           2           3           4           5
6
##  1.51452366  0.95139566  1.96541511 -0.42519366 -0.49817277
0.95139566
##           7           9          10          11          12
13
## -0.42519366  0.32746744 -0.58180663 -0.78346517 -0.78346517
-0.58180663
##          14          15          16          17          18
19
## -0.49817277 -1.31999379  0.45320786  0.32746744  0.45320786
-0.49817277
##          20          22          23          24          25
26
## -3.26060160  0.83030527 -0.67676980  0.06200048 -0.49817277
-0.49817277
##          27          28          29          30          31
32
## -0.49817277 -0.42519366 -0.58180663 -1.17171014 -0.49817277
-0.42519366
##          33          34          36          37          38
39
## -1.17171014 -0.67676980  1.36524499 -0.42519366  1.36524499
-0.42519366
##          40          41          42          43          44
45
##  0.83030527  0.53102721 -1.63005287 -0.67676980  0.38564516
0.71954391
##          46          47          51          52          53
54
## -0.42519366 -0.42519366  0.45320786 -0.49817277  0.95139566
0.53102721
```

```
##          55          56          57          59          60
61
##  1.66584949  0.71954391 -0.58180663 -0.67676980 -0.42519366
-0.42519366
##          64          66          67          68          69
70
##  0.32746744  1.51452366 -0.58180663  0.53102721  0.95139566
-0.58180663
##          71          73          74          75          76
78
## -1.78540278  0.53102721 -0.42519366  1.96541511 -0.49817277
-0.42519366
##          79
## -0.42519366
```

Del resultado podemos observar que ninguno de los residuales estandarizados excede el valor absoluto de 3, por ende, no parece haber observaciones atípicas.

Gráficamente se aprecia así:

```
plot(fitted(modelo_1.1),
     rstandard(modelo_1.1), abline(0, 0))
```



La sintaxis del comando anterior es:

1. `modelo_1.1` es el nombre del modelo de regresión lineal que se ha ajustado previamente.
2. `fitted(modelo_1.1)` retorna los valores ajustados del modelo de regresión para cada valor de la variable independiente en el conjunto de datos de entrenamiento. Los valores ajustados son las estimaciones de la variable dependiente para cada valor de la variable independiente, utilizando los coeficientes del modelo.
3. `rstandard(modelo_1.1)` retorna los residuos estandarizados del modelo de regresión para cada observación en el conjunto de datos de entrenamiento. Los residuos estandarizados son los residuos divididos por la desviación estándar de los residuos. Los residuos son la diferencia entre los valores observados de la variable dependiente y los valores ajustados por el modelo.
4. `abline(0, 0)` traza una línea recta horizontal en el gráfico en el punto y=0 e x=0. Esta línea es una referencia útil para identificar puntos que se desvían mucho de lo que se espera en un modelo de regresión lineal.
5. `plot(fitted(modelo_1.1), rstandard(modelo_1.1), abline(0, 0))` traza un gráfico de dispersión donde se representa en el eje *x* los valores ajustados del modelo (fitted values) y en el eje *y* los residuos estandarizados (standardized residuals) para cada observación. El argumento `abline(0, 0)` agrega una línea de referencia horizontal en el punto y=0 e x=0.

◊ *Revisando la dispersión*

Si la relación entre la desviación residual y los grados de libertad residuales excede 1.5, el modelo está sobredispersado. La dispersión excesiva indica que el modelo no se ajusta bien a los datos: es posible que las variables explicativas no describan bien la variable dependiente o que el modelo no se especifique correctamente para estos datos. Si hay sobredispersión, una posible solución es utilizar la opción de familia Poisson, quasipoisson, binomial negativa, zero inflado, quasibinomial, etc.

◊ *Desviación residual*

La desviación residual nos dice qué tan bien se puede predecir la variable de respuesta mediante un modelo con *p* variables predictoras. Cuanto menor sea el valor,

mejor podrá el modelo predecir el valor de la variable de respuesta.

```
#desviación residual
summary(modelo_1.1)

##
## Call:
## glm(formula = Status ~ Release, family = binomial, data =
Data_Final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2454  -0.5745  -0.4202   0.5188   1.9407
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.71511    0.61118  -4.442 8.90e-06 ***
## Release      0.33234    0.07554   4.400 1.08e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.343  on 66  degrees of freedom
## Residual deviance: 56.130  on 65  degrees of freedom
## AIC: 60.13
##
## Number of Fisher Scoring iterations: 5
```

La desviación residual es 56.130, la cual debe ser comparada con los demás modelos para ver cuál tiene mejores predicciones, por ejemplo el 1.2 obtuvo un 85.704 (este valor fue hallado anteriormente en los modelos de primer orden) por ello se puede decir que el modelo 1.1 es mejor que el 1.2 según este criterio.

```
# Calculando la libertad residual
summary(modelo_1.1)$deviance / summary(modelo_1.1)$df.residual

## [1] 0.8635435
```

El valor obtenido de 0.8635435 es el cociente de la desviación residual por los grados de libertad residual. Se utiliza para evaluar la bondad de ajuste del modelo. Un valor bajo del cociente indica un buen ajuste del modelo, ya que la cantidad de varianza no explicada en relación con la cantidad de información disponible es baja. En general, se puede considerar un valor bajo cuando el cociente es cercano a 1 o menor que 1.

◊ *Prueba de Chi cuadrado*

Prueba los coeficientes del modelo logístico y determina si un modelo es útil.

```
    Chi2 = Desviación nula – Desviación residual

modelChi1=modelo_1.1$null.deviance - modelo_1.1$deviance
modelChi1

## [1] 34.21294
```

El valor obtenido de 34.21294 representa la diferencia entre la desviación nula y la desviación residual del modelo de regresión logística, lo que se conoce como prueba de bondad de ajuste de la regresión. Esta medida se utiliza para evaluar la utilidad del modelo, es decir, si el modelo es capaz de explicar adecuadamente la variabilidad de la variable dependiente.

◊ *Bondad de ajuste del modelo*

Una prueba de bondad de ajuste dice qué tan bien el modelo describe la variable dependiente (de respuesta). Hay diferentes pruebas para hacerlo, entre ellas la prueba de Hosmer-Lemeshow y el coeficiente de determinación R-cuadrado para regresión logística.

Prueba de Hosmer-Lemeshow

Es una prueba de bondad de ajuste para la regresión logística, especialmente para los modelos de predicción de riesgo. Específicamente, esta prueba calcula si las tasas de eventos observadas coinciden con las tasas de eventos esperados en los subgrupos de población. Obtener un resultado significativo en la prueba indicaría que el modelo no está bien estimado, por lo que el ajuste no es bueno.

```
Hosmer1 <- modelChi1/modelo_1.1$null.deviance
Hosmer1

## [1] 0.3786994
```

El modelo arroja un valor *p* superior a 0.05, por lo que no se requiere refinar el modelo. A esta prueba se le pueden hacer verificaciones adicionales como la inspección de residuos, como las realizadas anteriormente, a fin de identificar valores atípicos y su posible influencia en el modelo ajustado.

R-cuadrado para regresión logística

Se emplean para medir la utilidad del modelo, son similares al coeficiente de determinación (R-cuadrado) en la regresión lineal. A continuación veremos dos de esas estadísticas.

Residuos de Cox & Snell

Estima la proporción de varianza de la variable dependiente que es explicada por las independientes. Está basada en la comparación del logaritmo de la verosimilitud para el modelo con respecto al logaritmo del modelo de línea base. Sus valores oscilan entre el 0 y 1. Se divide entre el número de observaciones que, en este caso, es 67.

```
Cox_Sme <- 1 - exp((modelo_1.1$deviance - modelo_1.1$null.
deviance)/67)
Cox_Sme
```

```
## [1] 0.3998891
```

En este caso, el valor de 0.3998891 indica que alrededor del 40% de la variabilidad en el éxito o fracaso de la introducción de especies de aves en Nueva Zelanda puede ser explicada por las variables independientes del modelo.

El R-cuadrado de Nagelkerke

El R cuadrado de Nagelkerke nos permite incrementar el coeficiente de Cox y Snell para cubrir todo el rango de 0 a 1.

```
Nagel <- Cox_Sme/(1-(exp(-(modelo_1.1$null.deviance/67))))
Nagel
## [1] 0.5401379
```

El R cuadrado de Nagelkerke es otra medida para evaluar la bondad de ajuste del modelo y determinar su capacidad de explicar la varianza en la variable dependiente. Esta prueba se basa en la comparación de la verosimilitud del modelo de regresión logística con respecto a un modelo de referencia o modelo nulo (modelo que no incluye ninguna variable independiente).

En este caso, el valor obtenido para el R cuadrado de Nagelkerke es de 0.5401379. Esto indica que el modelo de regresión logística es capaz de explicar el 54% de la varianza en la variable dependiente, lo que es una mejora significativa con respecto al coeficiente de Cox y Snell obtenido anteriormente.

Multicolinealidad

En un análisis post regresión logística no se verifica el supuesto de distribución de las variables independientes, no obstante, es importante que dichas variables no estén altamente correlacionadas unas con otras porque podría causar problemas de multicolinealidad. La multicolinealidad corresponde a una situación en la que los datos contienen variables predictoras altamente correlacionadas. Esto es un problema en el análisis de regresión logística que debe solucionarse eliminando las variables en cuestión. Una forma de evaluarlo es utilizando la función `vif` (siglas en inglés del factor de inflación de la varianza) del paquete `car`. Cuando el valor del `vif`

supera 5, indica una cantidad problemática de multicolinealidad. Para este ejemplo, este supuesto no aplica por tratarse de una sola variable predictora.

Resumen del capítulo

El análisis exploratorio y de correlación son pasos importantes previos a un análisis de regresión. Si bien en el análisis de correlación observamos cinco variables como posibles predictoras, la que tuvo alta correlación ("Release") finalmente fue la única que se ajustó al modelo al tener el menor AIC en comparación con los otros modelos de primer y segundo orden. Ello, además, se verificó al realizar diferentes pruebas post regresión, con lo cual, se confirmó que el modelo elegido era el óptimo.

Referencias

Veltman, C. J., Nee, S. & Crawley, M. J. (1996). Correlates of introduction success in exotic New Zealand birds. *The American Naturalist*, *147* (4), 542-557. https://www.jstor.org/stable/2463234

CAPÍTULO 9

REGRESIÓN DE POISSON

Los modelos de regresión de Poisson expresan la asociación entre una variable de conteo y una o más variables independientes. Un conteo es el número de veces que ocurre un evento. En este capítulo aprenderemos a realizar un análisis de regresión de Poisson y a estimar el *incidence rate ratio* o razón de tasas de incidencia como magnitud de asociación de interés. Adicionalmente aprenderemos a utilizar la regresión de Poisson con varianzas robustas para estimar el *prevalence ratio* o razón de prevalencia como magnitud de asociación de interés. Esta razón de prevalencias representa una mejor alternativa para estimar el riesgo, comparado con el *odds ratio* o razón de momios, cuando la prevalencia del evento de interés es superior al 10%. En ambos casos, aprenderemos a construir modelos de regresión multivariable con el objetivo de encontrar el modelo que mejor caracteriza la variabilidad de nuestro desenlace de interés o *outcome*. Finalmente, aprenderemos cómo verificar los supuestos post-regresión para verificar la validez del modelo. De esta manera este capítulo nos permitirá aprender cuales son las principales aplicaciones del análisis de regresión de Poisson y cómo nos permite obtener resultados valiosos para la comprensión y predicción de conteos de eventos.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Realizar un análisis y manejo previo de datos.
- Estimar la razón de prevalencia e intervalo de confianza.
- Aplicar modelos de regresión de Poisson.
- Seleccionar el modelo óptimo para una regresión.
- Realizar un análisis post regresión.

Paquetes

```
#instalamos paquetes
#install.packages("dplyr")
#install.packages("tableone")
#install.packages("Epi")
#install.packages("foreign")
#install.packages("sandwich")
#install.packages("lmtest")
#install.packages("AICcmodavg")
#install.packages("tidyverse")
#install.packages("coefplot")
#install.packages("broom")

#llamamos a los paquetes
library(dplyr) #manejo de datos
library(tableone) #resumen descriptivo de datos
library(Epi) #análisis demográfico y epidemiológico en el diagrama
Lexis
library(foreign) #lee archivos
library(sandwich)#estimadores de matrices de covarianza robustos
a modelos.
library(lmtest) #prueba de modelos
library(AICcmodavg) #selección de modelos
library(tidyverse) #manipulación de datos
library(broom) #diagnóstico
library(coefplot) #coeficientes
```

Estudio de caso

Durante la pandemia de COVID-19, los casos de violencia contra la mujer se incrementaron debido a que se potenciaron factores de riesgo tras las drásticas medidas de emergencia sanitaria que dictaron la mayoría de gobiernos en el mundo. En este

estudio, analizaremos la violencia física contra las mujeres en el Perú durante el año 2021, a partir de la base de datos del programa Aurora del Ministerio de la Mujer.

Al momento de cargar la base de datos "violencia_2021", usamos el comando `sep` para especificar que la separación se haga mediante comas y el comando `encoding` con el parámetro `utf-8` para que R reconozca las tildes de la base de datos. Puede encontrar la base de datos de violencia durante el año 2021 en el siguiente enlace Bases de datos Cap9

```
violencia_2021 <- read.csv("violencia_2021.csv",
                           sep=",", encoding = "utf-8")
```

Análisis exploratorio

Seleccionamos las variables de interés y las guardamos en un nuevo objeto al que denominaremos **violencia**.

```
violencia <- violencia_2021 %>%
  dplyr::select(`crime_violence_type`,`agressor_sec_education`,
                `agressor_age_group_0_39`,`agressor_job_status`,
               `victim_filesacomplaint`,`victim_age_group_0_39`,
               `agressor_foreign`)
```

Variable	Descripción	Tipo de variable
crime_violence_type (variable dependiente)	Tipo de violencia ejercida contra la víctima	Cualitativa
agressor_sec_education	Agresor con educación secundaria (positivo/negativo)	Cualitativa
agressor_age_ group_0_39	Edad del agresor se encuentra dentro del rango 0-39 (positivo/negativo)	Cualitativa
agressor_job_status	Agresor trabaja (positivo/negativo)	Cualitativa

Variable	Descripción	Tipo de variable
victim_filesacomplaint	Víctima realizó la denuncia (sí/no)	Cualitativa
victim_age_group_0_39	Edad de la víctima se encuentra dentro del rango 0-39 (positivo/negativo)	Cualitativa
agressor_foreign	Agresor es extranjero (positivo/ negativo)	Cualitativa

Estructura de las variables

```
str(violencia)

## 'data.frame':    140833 obs. of  7 variables:
##  $ crime_violence_type    : chr  "Violencia Sexual" "Violencia
Psicológica" "Violencia Física" "Violencia Física" ...
##  $ agressor_sec_education : chr  "" "Negative" "Positive"
"Negative" ...
##  $ agressor_age_group_0_39: chr  "" "Negative" "Positive"
"Negative" ...
##  $ agressor_job_status    : chr  "" "Negative" "" "Positive"
...
##  $ victim_filesacomplaint : chr  "Si" "Si" "Si" "Si" ...
##  $ victim_age_group_0_39  : chr  "Positive" "Positive"
"Positive" "Positive" ...
##  $ agressor_foreign       : chr  "Negative" "Negative"
"Negative" "Negative" ...
```

Las variables son cualitativas dicotómicas, es decir, solo pueden tomar dos valores (sí/no o positivo/negativo), con excepción de la variable "crime_violence_type" que es nominal y donde cada tipo de violencia comprende un comportamiento y actitud distintos.

Características de la violencia contra la mujer peruana

La función `CreateTableOne`, del paquete `tableone`, nos permite resumir todas las variables.

```
CreateTableOne(data=violencia)
```

```
##
##                                        Overall
##   n                                    140833
##   crime_violence_type (%)
##      Violencia Económica - Patrimonial    432 ( 0.3)
##      Violencia Física                   55767 (39.6)
##      Violencia Psicológica              63402 (45.0)
##      Violencia Sexual                   21232 (15.1)
##   agressor_sec_education (%)
##                                          3999 ( 2.8)
##      Negative                           49338 (35.0)
##      Positive                           87496 (62.1)
##   agressor_age_group_0_39 (%)
##                                          1396 ( 1.0)
##      Negative                           59726 (42.4)
##      Positive                           79711 (56.6)
##   agressor_job_status (%)
##                                          2730 ( 1.9)
##      Negative                           33506 (23.8)
##      Positive                          104597 (74.3)
##   victim_filesacomplaint = Si (%)      108200 (76.8)
##   victim_age_group_0_39 = Positive (%) 103107 (73.2)
##   agressor_foreign (%)
##                                           479 ( 0.3)
##      Negative                          138529 (98.4)
##      Positive                            1825 ( 1.3)
```

Como podemos ver en el resultado anterior, esta función nos brinda un resumen de todos nuestros datos en una tabla. Nos muestra el número de registros (n) así como el porcentaje de nuestras variables. Con respecto al tipo de violencia, la violencia psicológica es la preponderante (45%), seguida de la violencia física (39.6%) y la sexual (15.1%).

Violencia física

El objetivo de este estudio de caso es analizar la violencia física contra las mujeres en el Perú durante el 2021, así que nos quedamos solo con los casos de violencia física. Creamos una nueva variable denominada "tipo_violencia" que contiene los mismos datos que "crime_violence_type". Esta nueva columna servirá de respaldo ante los cambios que haremos a la variable "crime_violence_type".

```
violencia$tipo_violencia=
  violencia$crime_violence_type
```

Ahora, vamos a representar los casos de violencia física con el valor "1" y los otros casos de violencia con el valor "0". Para ello, utilizamos la función `ifelse`. Esta función nos permite especificar el dato que queremos reemplazar. Para el caso de la variable "crime_violence_type", especificamos "Violencia Física" y agregamos en el argumento "1,0". De esta forma se reemplazará "Violencia Física" por "1" y los otros valores, por "0".

```
violencia$crime_violence_type <-ifelse(violencia$crime_violence_
type==
                                        "Violencia Física" ,1,0)

str(violencia$crime_violence_type) #comprobamos

##  num [1:140833] 0 0 1 1 0 1 1 0 1 0 ...
```

Podemos verificar que que el reemplazo se realizó adecuadamente al comparar la columna "crime_violence_type" con la columna, anteriormente creada, "tipo_violencia". Ahora que dicotomizamos la variable dependiente en los valores violencia

física (“1”) y otro tipo de violencia (“0”), cambiamos el nombre de la variable “crime_violence_type” a “victimas_vfisica” y eliminamos la variable “tipo_violencia”.

```
#cambiamos de nombre a la variable

names(violencia)[names(violencia) ==
                  “crime_violence_type”] <- “victimas_vfisica”

names(violencia) #comprobamos

## [1] “victimas_vfisica”        “agressor_sec_education”
## [3] “agressor_age_group_0_39” “agressor_job_status”
## [5] “victim_filesacomplaint”  “victim_age_group_0_39”
## [7] “agressor_foreign”         “tipo_violencia”

#eliminamos la variable “tipo_violencia”
violencia$tipo_violencia <-NULL
```

Asimismo, cambiamos el nombre del objeto a **violencia_fisica**.

```
violencia_fisica  <- violencia
```

Análisis bivariado

El análisis bivariado nos permitirá explorar cómo una variable afecta a otra variable. Por esta razón, utilizaremos la prueba de Chi^2 o Chi-cuadrado para conocer la asociación entre dos variables cualitativas. Esta prueba estadística se basa en comparar la distribución observada de frecuencias de las categorías de ambas variables con la distribución que se esperaría si no hubiera ninguna asociación entre ellas.

Víctimas de violencia física vs. educación del agresor

```
chisq.test(violencia_fisica$victimas_vfisica,
```

```
           violencia_fisica$agressor_sec_education)

##
##  Pearson's Chi-squared test
##
## data:  violencia_fisica$victimas_vfisica and violencia_
fisica$agressor_sec_education
## X-squared = 583.82, df = 2, p-value < 2.2e-16
```

El valor *p* (*p value*) es inferior a 0.05, por lo tanto, rechazamos la hipótesis nula y concluimos que hay una asociación significativa entre las víctimas de violencia física y los agresores con educación secundaria.

Víctimas de violencia física vs. edad del agresor <40

```
chisq.test(violencia_fisica$victimas_vfisica,
           violencia_fisica$agressor_age_group_0_39)

##
##  Pearson's Chi-squared test
##
## data:  violencia_fisica$victimas_vfisica and violencia_
fisica$agressor_age_group_0_39
## X-squared = 2412.8, df = 2, p-value < 2.2e-16
```

Hay una fuerte asociación entre las víctimas de violencia física y los agresores menores de 40 años.

Víctimas de violencia física vs. agresor trabaja

```
chisq.test(violencia_fisica$victimas_vfisica,
           violencia_fisica$agressor_job_status)
```

```
##
##  Pearson's Chi-squared test
##
## data:  violencia_fisica$victimas_vfisica and violencia_
fisica$agressor_job_status
## X-squared = 695.21, df = 2, p-value < 2.2e-16
```

Existe una asociación entre las víctimas de violencia física y los agresores que trabajan.

Víctimas de violencia física vs. víctima denuncia

```
chisq.test(violencia_fisica$victimas_vfisica,
           violencia_fisica$victim_filesacomplaint)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  violencia_fisica$victimas_vfisica and violencia_
fisica$victim_filesacomplaint
## X-squared = 636.39, df = 1, p-value < 2.2e-16
```

Existe una asociación entre las víctimas de violencia física y las denuncias que éstas realizan. Aunque la asociación entre las víctimas de violencia física y las denuncias que realizan parece ser obvia. Sin embargo, el análisis bivariado utilizando la prueba de Chi^2 no busca demostrar algo obvio o tautológico, sino que se utiliza para cuantificar la fuerza de la asociación entre dos variables cualitativas y determinar si esta asociación es estadísticamente significativa.

Víctimas de violencia física vs. edad de la víctima <40

```
chisq.test(violencia_fisica$victim_age_group_0_39,
           violencia_fisica$victimas_vfisica)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  violencia_fisica$victim_age_group_0_39 and violencia_
fisica$victimas_vfisica
## X-squared = 279.66, df = 1, p-value < 2.2e-16
```

Existe una asociación entre las víctimas de violencia física y su edad inferior a 40 años.

Víctimas de violencia física vs. agresor extranjero

```
chisq.test(violencia_fisica$victimas_vfisica,
           violencia_fisica$agressor_foreign)

##
##  Pearson's Chi-squared test
##
## data:  violencia_fisica$victimas_vfisica and violencia_
fisica$agressor_foreign
## X-squared = 313.89, df = 2, p-value < 2.2e-16
```

Como hemos podido comprobar, en todos los casos se rechaza la hipótesis nula al ser el valor *p* inferior a 0.05. Esto quiere decir que existe una fuerte asociación entre las variables independientes y la variable dependiente.

Análisis de regresión de Poisson multivariable

Es similar a la regresión logística en el sentido de que las variables dependientes son discretas. Sin embargo, en la regresión logística las variables dependientes son proporciones, mientras que la regresión de Poisson trabaja con datos de conteo.

De esa manera, los modelos de regresión de Poisson expresan la asociación entre una variable de desenlace de tipo conteo (Y = 0,1,2,3…) y una o más variables

independientes (*X*'s) –el conteo hace referencia al número de veces en que ocurre un evento en una misma unidad de observación durante un determinado periodo de tiempo o en un determinado espacio–. Resultan especialmente útiles para modelar valores enteros no negativos y cuando la frecuencia de ocurrencia es baja.

Ejemplos de aplicaciones

Conteos en el tiempo
Número de visitas a un sitio web en una hora determinada.
Número de horas que los trabajadores de una empresa han dedicado a un proyecto en una semana.
Número de partidos de fútbol jugados en una temporada de la Champions League.
Número de mensajes enviados por Whatsapp en un día.

Conteos en el espacio
Número de departamentos en un edificio residencial de determinada ciudad.
Número de tiendas Tambo que están distribuidas por los distritos Lima.
Número de vehículos que pasan por un peaje de una autopista en un día.
Número de negocios que ofrecen servicios de delivery con PedidosYa en una zona urbana.

Al igual que en el capítulo anterior, utilizaremos la función `glm`, que no solo nos permite ejecutar modelos de regresión logística, sino también, dada su flexibilidad, modelos con datos de conteo.

Conversión de variables

Para nuestro análisis, las variables dicotómicas se representarán mediante valores numéricos, donde se asignó el valor "1" a una de las dos categorías posibles y el valor "0" a la otra categoría que queda. Es importante considerar que esta conversión no significa que la variable dicotómica se convierte en una variable continua, ya que seguirá teniendo solamente dos valores posibles. En otras palabras, la

variable sigue siendo dicotómica, pero se ha representado en términos numéricos para poder realizar los siguientes análisis estadísticos:

```
#educación del agresor
violencia_fisica$agressor_sec_education<-
  ifelse(violencia_fisica$agressor_sec_education==
                                                  “Positive”,1,0)

#agresor de 0-39 años
violencia_fisica$agressor_age_group_0_39<-
  ifelse(violencia_fisica$agressor_age_group_0_39==

“Positive”,1,0)

# agresor trabaja
violencia_fisica$agressor_job_status<-
  ifelse(violencia_fisica$agressor_job_status==
                                              “Positive”,1,0)

#víctima denuncia
violencia_fisica$victim_filesacomplaint<-
  ifelse(violencia_fisica$victim_filesacomplaint
                                            ==”Si”,1,0)

#víctima edad 0-39
violencia_fisica$victim_age_group_0_39<-
  ifelse(violencia_fisica$victim_age_group_0_39==
                                                “Positive”,1,0)

#agresor extranjero
violencia_fisica$agressor_foreign<-
  ifelse(violencia_fisica$agressor_foreign==
                                            “Positive”,1,0)
```

Renombramos variables

```
names(violencia_fisica)[names(violencia_fisica) ==
                        "agressor_sec_education"] <- "agresor_
edu"
names(violencia_fisica)[names(violencia_fisica) ==
                        "agressor_age_group_0_39"] <-
"agresor_0_39"
names(violencia_fisica)[names(violencia_fisica) ==
                        "agressor_job_status"] <- "agresor_
trabaja"
names(violencia_fisica)[names(violencia_fisica) ==
                        "victim_filesacomplaint"] <- "victima_
denuncia"
names(violencia_fisica)[names(violencia_fisica) ==
                        "victim_age_group_0_39"] <-
"victima_0_39"
names(violencia_fisica)[names(violencia_fisica) ==
                        "agressor_foreign"] <- "agresor_
extranjero"

names(violencia_fisica)#comprobamos

## [1] "victimas_vfisica"   "agresor_edu"        "agresor_0_39"
## [4] "agresor_trabaja"    "victima_denuncia"   "victima_0_39"
## [7] "agresor_extranjero"
```

Modelo de primer orden

◊ *agresor_edu*

```
modelo_1.1 <-glm(victimas_vfisica ~ agresor_edu, data=violencia_
fisica,
                 family=poisson(link=log))
```

```
summary(modelo_1.1)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_edu, family =
poisson(link = log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8937  -0.8876  -0.8876   0.7965   0.8069
##
## Coefficients:
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -0.917829   0.006851 -133.960   <2e-16 ***
## agresor_edu -0.013819   0.008715   -1.586    0.113
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 103322  on 140831  degrees of freedom
## AIC: 214860
##
## Number of Fisher Scoring iterations: 5
```

Interpretación del *output* de un modelo de regresión de Poisson:

- El modelo (call): glm(formula = victimas_vfisica ~ agresor_edu, family = poisson(link = log), data = violencia_fisica). El *output* del ajuste del modelo, especifica la variable dependiente (victimas_vfisica) y la variable independiente (agresor_edu) en la fórmula, así como el tipo de modelo (familia Poisson) y

la función de enlace utilizando `log`. Finalmente, indica la fuente de los datos utilizados para ajustar el modelo (data = violencia_fisica).

- Lo ideal es que los residuos de desviación se distribuyan simétricamente, sin embargo, no parece ser el caso en este ejemplo. Esto probablemente se deba a la sobredispersión de los datos. Se puede observar, en la sección "Deviance Residuals", que los valores mínimos y máximos de los residuos son asimétricos, lo que indica que los residuos no se distribuyen simétricamente.
- El coeficiente estimado para el intercepto,-0.917829, representa el logaritmo de la tasa media de ocurrencia de víctimas de violencia física cuando la educación secundaria del agresor es igual a 0. El coeficiente estimado para la variable "agresor_edu" indica que un aumento en la educación secundaria del agresor se asocia con una disminución del logaritmo de la tasa media de ocurrencia de víctimas de violencia física en un valor de -0.013819. Observamos, también, que la asociación no es estadísticamente significativa (valor p = 0.113) entre la educación secundaria del agresor y la tasa media de ocurrencia de víctimas de violencia física en la población de interés.

Razón de prevalencia e intervalo de confianza

Los modelos de regresión de Poisson con varianza robusta permiten estimar el *prevalence ratio* o razón de prevalencias como magnitud de asociación de interés, la misma que representa una mejor alternativa al *odds ratio* o razón de momios cuando lo que se quiere es estimar el parámetro riesgo y el *outcome* o desenlace de interés no es raro (considerado así cuando al prevalencia un *outcome* o desenlace de interés dicotómico es >10%). A continuación, aprenderemos cómo estimar el *prevalence ratio* o razón de prevalencias utilizando un modelo glm de Poisson con varianzas robustas. Para ello utilizaremos el paquete `sandwich` y la función `coeftest` del paquete `lmtest` para aplicarlo a las pruebas z o de Wald.

```
#Coeficientes
coef<-coeftest(modelo_1.1,vcov=sandwich)
coef

##
## z test of coefficients:
```

```
##
##                Estimate Std. Error   z value Pr(>|z|)
## (Intercept) -0.9178293  0.0053099 -172.8519  < 2e-16 ***
## agresor_edu -0.0138195  0.0067662   -2.0424  0.04111 *
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
```

En este análisis se observa una asociación significativa entre la educación del agresor y las víctimas de violencia física. Así como cambios en el valor *z* y el error estimado. Podemos notar que tanto el intercepto como la variable “agresor_edu” son estadísticamente significativos, ya que sus valores *p* son más bajos que 0.05. Además, el coeficiente para la variable “agresor_edu” indica que un aumento de una unidad en la educación secundaria del agresor se asocia con una disminución del 1,4% en el número de víctimas de violencia física.

Cabe resaltar que la función coeftest() se utiliza para calcular diferentes tipos de errores estándar y estadísticas de prueba para los coeficientes de un modelo estimado con la función glm(). De este modo, calculamos los errores estándar robustos para los coeficientes del modelo modelo_1.1 utilizando la matriz de varianza-covarianza robusta de tipo sandwich.

La matriz de varianza-covarianza robusta de tipo sandwich se llama así porque está compuesta por 3 rebanadas, la primera es la matriz de varianza-covarianza de los coeficientes obtenida bajo la suposición de que los errores son homocedásticos e independientes; la segunda es una medida de la influencia de cada observación en los coeficientes del modelo; y la tercera es una versión corregida de la matriz de varianza-covarianza de los coeficientes que tiene en cuenta la información de las dos “rebanadas” anteriores.

Veamos ahora el intervalo de confianza.

```
B <-coef[“agresor_edu”, “Estimate”]
SE<-coef[“agresor_edu”,”Std. Error”]

#Razón de prevalencia: punto de estimación
exp(B)
```

```
## [1] 0.9862756

#Intervalo inferior
exp(B+ qnorm(0.05/ 2)*SE)

## [1] 0.9732825

#Intervalo superior
exp(B + qnorm(1-0.05/2)*SE)

## [1] 0.9994422

#Unión de los intervalos + redondeo
Li=exp(B+qnorm(1-0.05/2)*SE)
Ls=exp(B+qnorm(0.05/2)*SE)

PR=exp(B)
print(paste(round(PR,3),” (“, round(Li,3),”-”, round(Ls,3),”)”))

## [1] “0.986  ( 0.999 - 0.973 )”
```

- De ese modo, se calculó los intervalos de confianza para la razón de prevalencia correspondiente a la variable “agresor_edu”, utilizando la aproximación normal para la distribución de los coeficientes de regresión, ya que se asume que la muestra es lo suficientemente grande para que esta aproximación sea válida. Por esa razón, se utiliza la función `qnorm` para obtener los cuantiles de la distribución normal estándar en los niveles de significancia correspondientes. La función `print(paste(round(PR,3),...))`, redondea los valores de la razón de prevalencia y los límites inferior y superior de los intervalos de confianza a tres decimales, imprimiéndolos como un único string.
- La razón de prevalencia puntual estimada es 0.986, lo que significa que por cada aumento de una unidad en la educación secundaria del agresor, la razón de prevalencia de la violencia física disminuye en un 1.4%. La disminución

porcentual se calculó restando la razón de prevalencia de 1 y luego multiplicando por 100, es decir, 1 - 0.986 = 0.014, lo que indica la reducción porcentual. Además, con un nivel de confianza del 95%, se espera que el valor real de la razón de prevalencia de la población esté entre 0.973 y 0.999.

◊ *agresor 0-39 años*

```
modelo_1.2 <-glm(victimas_vfisica ~ agresor_0_39, data=violencia_
fisica, family=poisson(link=log))

summary(modelo_1.2)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39, family =
poisson(link = log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9467  -0.9467  -0.8098   0.7082   0.9412
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.114992   0.007063  -157.9   <2e-16 ***
## agresor_0_39  0.312407   0.008825    35.4   <2e-16 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 102039  on 140831  degrees of freedom
## AIC: 213577
```

```
##
## Number of Fisher Scoring iterations: 5
```

Razón de prevalencia e intervalo de confianza

```
#Coeficientes
coef<-coeftest(modelo_1.2,vcov=sandwich)
coef

##
## z test of coefficients:
##
##                Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)  -1.1149919  0.0057907 -192.549 < 2.2e-16 ***
## agresor_0_39  0.3124070  0.0069985   44.639 < 2.2e-16 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1

B <-coef[“agresor_0_39”, “Estimate”]
SE<-coef[“agresor_0_39”,”Std. Error”]

#Razón de prevalencia: punto de estimación
exp(B)

## [1] 1.366711

#Intérvalo inferior
exp(B+ qnorm(0.05/ 2)*SE)

## [1] 1.348092

#Intervalo superior
exp(B + qnorm(1-0.05/2)*SE)
```

```
## [1] 1.385587

#Unión de los intervalos + redondeo
Li=exp(B+qnorm(1-0.05/2)*SE)
Ls=exp(B+qnorm(0.05/2)*SE)

PR=exp(B)
print(paste(round(PR,3),” (“, round(Li,3),”-”, round(Ls,3),”)”))

## [1] “1.367  ( 1.386 - 1.348 )”
```

◊ *agresor trabaja*

```
modelo_1.3 <-glm(victimas_vfisica ~ agresor_trabaja,
data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_1.3)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_trabaja, family =
poisson(link = log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9078  -0.9078  -0.8363   0.7729   0.8949
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.050737   0.008884 -118.28   <2e-16 ***
```

```
## agresor_trabaja  0.164052   0.010105   16.23   <2e-16 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 103053  on 140831  degrees of freedom
## AIC: 214591
##
## Number of Fisher Scoring iterations: 5
```

Razón de prevalencia e intervalo de confianza

```
#Coeficientes
coef<-coeftest(modelo_1.3,vcov=sandwich)
coef

##
## z test of coefficients:
##
##                   Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)     -1.0507372  0.0071640 -146.668 < 2.2e-16 ***
## agresor_trabaja  0.1640525  0.0080602   20.353 < 2.2e-16 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1

B <-coef[“agresor_trabaja”, “Estimate”]
SE<-coef[“agresor_trabaja”,”Std. Error”]

#Razón de prevalencia: punto de estimación
exp(B)
```

```
## [1] 1.178276

#Intervalo inferior
exp(B+ qnorm(0.05/ 2)*SE)

## [1] 1.159808

#Intervalo superior
exp(B + qnorm(1-0.05/2)*SE)

## [1] 1.197038

#Unión de los intervalos + redondeo
Li=exp(B+qnorm(1-0.05/2)*SE)
Ls=exp(B+qnorm(0.05/2)*SE)

PR=exp(B)
print(paste(round(PR,3),” (“, round(Li,3),”-”, round(Ls,3),”)”))

## [1] “1.178  ( 1.197 - 1.16 )”
```

◊ *víctima denuncia*

```
modelo_1.4 <-glm(victimas_vfisica ~ victima_denuncia,
data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_1.4)

##
## Call:
## glm(formula = victimas_vfisica ~ victima_denuncia, family =
poisson(link = log),
```

```
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9100  -0.9100  -0.8199   0.7692   0.9235
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.090342   0.009548 -114.19   <2e-16 ***
## victima_denuncia  0.208547   0.010653   19.58   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 102925  on 140831  degrees of freedom
## AIC: 214463
##
## Number of Fisher Scoring iterations: 5
```

Razón de prevalencia e intervalo de confianza

```
#Coeficientes
coef<-coeftest(modelo_1.4,vcov=sandwich)
coef

##
## z test of coefficients:
##
##                   Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)      -1.0903421  0.0077801 -140.144 < 2.2e-16 ***
## victima_denuncia  0.2085466  0.0085796   24.307 < 2.2e-16 ***
```

```
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1

B <-coef[“victima_denuncia”, “Estimate”]
SE<-coef[“victima_denuncia”,”Std. Error”]

#Razón de prevalencia: punto de estimación
exp(B)

## [1] 1.231886

#Intervalo inferior
exp(B+ qnorm(0.05/ 2)*SE)

## [1] 1.211344

#Intervalo superior
exp(B + qnorm(1-0.05/2)*SE)

## [1] 1.252777

#Unión de los intervalos + redondeo
Li=exp(B+qnorm(1-0.05/2)*SE)
Ls=exp(B+qnorm(0.05/2)*SE)

PR=exp(B)
print(paste(round(PR,3),” (“, round(Li,3),”-”, round(Ls,3),”)”))

## [1] “1.232  ( 1.253 - 1.211 )”
```

◊ *agresor extranjero*

```
modelo_1.5 <-glm(victimas_vfisica ~ agresor_extranjero,
data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_1.5)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_extranjero, family =
poisson(link = log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9719  -0.8888  -0.8888   0.8049   0.8049
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.928927   0.004268 -217.67  < 2e-16 ***
## agresor_extranjero  0.178847   0.034326    5.21 1.89e-07 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 103299  on 140831  degrees of freedom
## AIC: 214837
##
## Number of Fisher Scoring iterations: 5
```

Razón de prevalencia e intervalo de confianza

```
#Coeficientes
coef<-coeftest(modelo_1.5,vcov=sandwich)
coef

##
## z test of coefficients:
##
##                       Estimate Std. Error   z value  Pr(>|z|)
## (Intercept)        -0.9289271  0.0033196 -279.8348 < 2.2e-16
***
## agresor_extranjero  0.1788471  0.0249633    7.1644 7.813e-13
***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1

B <-coef[“agresor_extranjero”, “Estimate”]
SE<-coef[“agresor_extranjero”,”Std. Error”]

#Razón de prevalencia: punto de estimación
exp(B)

## [1] 1.195838

#Intervalo inferior
exp(B+ qnorm(0.05/ 2)*SE)

## [1] 1.138737

#Intervalo superior
exp(B + qnorm(1-0.05/2)*SE)

## [1] 1.255802
```

```
#Unión de los intervalos + redondeo
Li=exp(B+qnorm(1-0.05/2)*SE)
Ls=exp(B+qnorm(0.05/2)*SE)

PR=exp(B)
print(paste(round(PR,3),” (“, round(Li,3),”-”, round(Ls,3),”)”))

## [1] “1.196  ( 1.256 - 1.139 )”
```

◊ *victima 0-39*

```
modelo_1.6 <-glm(victimas_vfisica ~ victima_0_39, data=violencia_
fisica,
                 family=poisson(link=log))

summary(modelo_1.6)

##
## Call:
## glm(formula = victimas_vfisica ~ victima_0_39, family =
poisson(link = log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9046  -0.9046  -0.8485   0.7782   0.8738
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.021825   0.008581 -119.07   <2e-16 ***
## victima_0_39  0.128194   0.009866   12.99   <2e-16 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
```

```
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 103152  on 140831  degrees of freedom
## AIC: 214690
##
## Number of Fisher Scoring iterations: 5

#Coeficientes
coef<-coeftest(modelo_1.6,vcov=sandwich)
coef

##
## z test of coefficients:
##
##                Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)  -1.0218250  0.0068656 -148.833 < 2.2e-16 ***
## victima_0_39  0.1281936  0.0078193   16.395 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

B <-coef["victima_0_39", "Estimate"]
SE<-coef["victima_0_39","Std. Error"]

#Razón de prevalencia: punto de estimación
exp(B)

## [1] 1.136773

#Intervalo inferior
exp(B+ qnorm(0.05/ 2)*SE)

## [1] 1.119484
```

```
#Intervalo superior
exp(B + qnorm(1-0.05/2)*SE)

## [1] 1.154329

#Unión de los intervalos + redondeo
Li=exp(B+qnorm(1-0.05/2)*SE)
Ls=exp(B+qnorm(0.05/2)*SE)

PR=exp(B)
print(paste(round(PR,3),” (“, round(Li,3),”-”, round(Ls,3),”)”))
## [1] “1.137  ( 1.154 - 1.119 )”
```

◊ Comparando los modelos de primer orden

```
#Definir la lista de modelos
models1 <- list(modelo_1.1, modelo_1.2 ,
modelo_1.3,modelo_1.4,modelo_1.5,modelo_1.6)

#Especificar los nombres del modelo
mod.names <- c(‘agresor_edu’,’agresor_0_39’, ‘agresor_trabaja’,
‘victima_denuncia’, ‘agresor_extranjero’,’victima_0_39’)

#Calcular el AIC de cada modelo
aictab(cand.set = models1, modnames = mod.names)

##
## Model selection based on AICc:
##
##                    K     AICc Delta_AICc AICcWt Cum.Wt
LL
## agresor_0_39       2 213576.6       0.00      1      1
```

```
-106786.3
## victima_denuncia    2 214462.9     886.21      0      1
-107229.4
## agresor_trabaja     2 214590.9    1014.26      0      1
-107293.4
## victima_0_39        2 214689.8    1113.12      0      1
-107342.9
## agresor_extranjero 2 214836.6    1259.97      0      1
-107416.3
## agresor_edu         2 214859.8    1283.10      0      1
-107427.9
```

El mejor modelo es el modelo 1.2.

Modelos de segundo orden

```
modelo_2.1 <-glm(victimas_vfisica ~ agresor_0_39 + agresor_edu,
data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_2.1)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + agresor_edu,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9553  -0.9419  -0.8052   0.7161   0.9494
##
## Coefficients:
##               Estimate Std. Error  z value Pr(>|z|)
```

```
## (Intercept)  -1.098321   0.008735 -125.739  < 2e-16 ***
## agresor_0_39  0.313675   0.008834   35.508  < 2e-16 ***
## agresor_edu  -0.028138   0.008724   -3.225  0.00126 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 102028  on 140830  degrees of freedom
## AIC: 213568
##
## Number of Fisher Scoring iterations: 5

modelo_2.2 <-glm(victimas_vfisica ~ agresor_0_39 + agresor_trabaja
, data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_2.2)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + agresor_
trabaja,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9669  -0.8881  -0.8261   0.6752   1.0326
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.245235   0.010596 -117.52   <2e-16 ***
## agresor_0_39     0.314833   0.008826   35.67   <2e-16 ***
```

```
## agresor_trabaja  0.169905   0.010107   16.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101747  on 140830  degrees of freedom
## AIC: 213287
##
## Number of Fisher Scoring iterations: 5

modelo_2.3 <-glm(victimas_vfisica ~ agresor_0_39 + victima_
denuncia, data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_2.3)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9669  -0.8739  -0.8283   0.6751   1.0512
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.272392   0.011028 -115.38   <2e-16 ***
## agresor_0_39      0.309574   0.008826   35.07   <2e-16 ***
## victima_denuncia  0.202387   0.010655   19.00   <2e-16 ***
## ---
```

```
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101663  on 140830  degrees of freedom
## AIC: 213203
##
## Number of Fisher Scoring iterations: 5

modelo_2.4 <-glm(victimas_vfisica ~ agresor_0_39 + victima_0_39,
data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_2.4)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_0_39,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9562  -0.9456  -0.8055   0.7100   0.9488
##
## Coefficients:
##                Estimate Std. Error  z value Pr(>|z|)
## (Intercept)  -1.103409   0.009019 -122.339   <2e-16 ***
## agresor_0_39  0.320755   0.009724   32.987   <2e-16 ***
## victima_0_39 -0.022330   0.010871   -2.054     0.04 *
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## (Dispersion parameter for poisson family taken to be 1)
```

```
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 102034  on 140830  degrees of freedom
## AIC: 213574
##
## Number of Fisher Scoring iterations: 5

modelo_2.5 <-glm(victimas_vfisica ~ agresor_0_39 + agresor_
extranjero, data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_2.5)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + agresor_
extranjero,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9965  -0.9458  -0.8096   0.7098   0.9416
##
## Coefficients:
##                     Estimate Std. Error  z value Pr(>|z|)
## (Intercept)        -1.115589   0.007066 -157.874  < 2e-16 ***
## agresor_0_39        0.310946   0.008840   35.176  < 2e-16 ***
## agresor_extranjero  0.104402   0.034383    3.036  0.00239 **
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
```

```
## Residual deviance: 102030  on 140830  degrees of freedom
## AIC: 213570
##
## Number of Fisher Scoring iterations: 5
```

◊ *Comparando modelos de segundo orden*

```
#Definir la lista de modelos
models2 <- list(modelo_2.1, modelo_2.2 ,
modelo_2.3,modelo_2.4,modelo_2.5)

#Especificar los nombres del modelo
mod.names2 <- c('agresor_edu','agresor_trabaja', 'victima_
denuncia', 'victima_0_39', 'agresor_extranjero')

#Calcular el AIC de cada modelo
aictab(cand.set = models2, modnames = mod.names2)

##
## Model selection based on AICc:
##
##                    K     AICc Delta_AICc AICcWt Cum.Wt
LL
## victima_denuncia   3 213203.0       0.00      1      1
-106598.5
## agresor_trabaja    3 213287.4      84.32      0      1
-106640.7
## agresor_edu        3 213568.3     365.22      0      1
-106781.1
## agresor_extranjero 3 213569.7     366.68      0      1
-106781.9
## victima_0_39       3 213574.4     371.39      0      1
-106784.2
```

El mejor modelo es el 2.3.

Modelos de tercer orden

```
modelo_3.1 <-glm(victimas_vfisica ~  agresor_0_39 + victima_
denuncia + agresor_edu, data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_3.1)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_edu, family = poisson(link = log), data =
violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9777  -0.8829  -0.8226   0.6846   1.0615
##
## Coefficients:
##                   Estimate Std. Error  z value Pr(>|z|)
## (Intercept)      -1.253238   0.012049 -104.009  < 2e-16 ***
## agresor_0_39      0.311075   0.008835   35.211  < 2e-16 ***
## victima_denuncia  0.203803   0.010661   19.117  < 2e-16 ***
## agresor_edu      -0.034168   0.008729   -3.914 9.06e-05 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101648  on 140829  degrees of freedom
```

```
## AIC: 213190
##
## Number of Fisher Scoring iterations: 5

modelo_3.2 <-glm(victimas_vfisica ~  agresor_0_39 + victima_
denuncia + agresor_trabaja, data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_3.2)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_trabaja, family = poisson(link = log), data =
violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9868  -0.8931  -0.7767   0.6428   1.1359
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.398124   0.013490 -103.64   <2e-16 ***
## agresor_0_39      0.311979   0.008827   35.34   <2e-16 ***
## victima_denuncia  0.199600   0.010656   18.73   <2e-16 ***
## agresor_trabaja   0.166860   0.010108   16.51   <2e-16 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101382  on 140829  degrees of freedom
```

```
## AIC: 212924
##
## Number of Fisher Scoring iterations: 5

modelo_3.3 <-glm(victimas_vfisica ~  agresor_0_39 + victima_
denuncia + victima_0_39, data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_3.3)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     victima_0_39, family = poisson(link = log), data =
violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9730  -0.8734  -0.8254   0.6764   1.0557
##
## Coefficients:
##                   Estimate Std. Error  z value Pr(>|z|)
## (Intercept)      -1.264609   0.012519 -101.013   <2e-16 ***
## agresor_0_39      0.314916   0.009729   32.370   <2e-16 ***
## victima_denuncia  0.201872   0.010662   18.934   <2e-16 ***
## victima_0_39     -0.014244   0.010879   -1.309     0.19
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101661  on 140829  degrees of freedom
```

```
## AIC: 213203
##
## Number of Fisher Scoring iterations: 5

modelo_3.4 <-glm(victimas_vfisica ~  agresor_0_39 + victima_
denuncia + agresor_extranjero, data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_3.4)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_extranjero, family = poisson(link = log), data =
violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0140  -0.8731  -0.8280   0.6767   1.0514
##
## Coefficients:
##                     Estimate Std. Error  z value Pr(>|z|)
## (Intercept)        -1.272703   0.011028 -115.406  < 2e-16 ***
## agresor_0_39        0.308210   0.008841   34.862  < 2e-16 ***
## victima_denuncia    0.202082   0.010655   18.966  < 2e-16 ***
## agresor_extranjero  0.097012   0.034385    2.821  0.00478 **
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101655  on 140829  degrees of freedom
```

```
## AIC: 213197
##
## Number of Fisher Scoring iterations: 5
```

◊ *Comparando modelos de tercer orden*

```
#Definir la lista de modelos
models3 <- list(modelo_3.1, modelo_3.2 , modelo_3.3,modelo_3.4)

#Especificar los nombres del modelo
mod.names3 <- c('vic_denun.agr_edu','vic_denun.agr_trabaja',
'vic_denun.vic_0_39', 'vic_denun.agr_ext')

#Calcular el AIC de cada modelo
aictab(cand.set = models3, modnames = mod.names3)

##
## Model selection based on AICc:
##
##                       K     AICc Delta_AICc AICcWt Cum.Wt
LL
## vic_denun.agr_trabaja 4 212924.3       0.00      1      1
-106458.2
## vic_denun.agr_edu     4 213189.8     265.44      0      1
-106590.9
## vic_denun.agr_ext     4 213197.3     273.00      0      1
-106594.7
## vic_denun.vic_0_39    4 213203.3     279.00      0      1
-106597.7
```

El mejor modelo es el 3.2.

Modelos de cuarto orden

```
modelo_4.1 <-glm(victimas_vfisica ~  agresor_0_39 + victima_
denuncia + agresor_trabaja + agresor_edu ,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_4.1)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_trabaja + agresor_edu, family = poisson(link =
log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0017  -0.8994  -0.7742   0.6548   1.1509
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.375623   0.014163 -97.128  < 2e-16 ***
## agresor_0_39      0.314051   0.008837  35.540  < 2e-16 ***
## victima_denuncia  0.201372   0.010661  18.888  < 2e-16 ***
## agresor_trabaja   0.170551   0.010133  16.830  < 2e-16 ***
## agresor_edu      -0.045092   0.008751  -5.153 2.57e-07 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
```

```
## Residual deviance: 101356  on 140828  degrees of freedom
## AIC: 212900
##
## Number of Fisher Scoring iterations: 5

modelo_4.2 <-glm(victimas_vfisica ~  agresor_0_39 + victima_
denuncia + agresor_trabaja + victima_0_39,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_4.2)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_trabaja + victima_0_39, family = poisson(link =
log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9965  -0.8924  -0.7723   0.6446   1.1429
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.386848   0.014610  -94.92   <2e-16 ***
## agresor_0_39      0.320239   0.009754   32.83   <2e-16 ***
## victima_denuncia  0.198793   0.010663   18.64   <2e-16 ***
## agresor_trabaja   0.167669   0.010116   16.57   <2e-16 ***
## victima_0_39     -0.021822   0.010910   -2.00   0.0455 *
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
```

```
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101378  on 140828  degrees of freedom
## AIC: 212922
##
## Number of Fisher Scoring iterations: 5

modelo_4.3 <-glm(victimas_vfisica ~  agresor_0_39 + victima_
denuncia + agresor_trabaja + agresor_extranjero ,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_4.3)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_trabaja + agresor_extranjero, family =
poisson(link = log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0353  -0.8923  -0.7765   0.6444   1.1362
##
## Coefficients:
##                     Estimate Std. Error  z value Pr(>|z|)
## (Intercept)        -1.398466   0.013490 -103.664  < 2e-16 ***
## agresor_0_39        0.310597   0.008842   35.127  < 2e-16 ***
## victima_denuncia    0.199295   0.010656   18.702  < 2e-16 ***
## agresor_trabaja     0.166899   0.010108   16.512  < 2e-16 ***
## agresor_extranjero  0.097861   0.034386    2.846  0.00443 **
```

```
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101374  on 140828  degrees of freedom
## AIC: 212918
##
## Number of Fisher Scoring iterations: 5
```

◊ *Comparamos los modelos de cuarto orden*

```
#Definir la lista de modelos
models4 <- list(modelo_4.1, modelo_4.2 , modelo_4.3)

#Especificar los nombres del modelo
mod.names4 <- c(‘agresor_edu’, ‘victima_0_39’,’agresor_
extranjero’ )

#Calcular el AIC de cada modelo
aictab(cand.set = models4, modnames = mod.names4)

##
## Model selection based on AICc:
##
##                    K     AICc Delta_AICc AICcWt Cum.Wt
LL
## agresor_edu        5 212899.9       0.00      1      1
-106444.9
## agresor_extranjero 5 212918.5      18.60      0      1
-106454.2
## victima_0_39       5 212922.3      22.46      0      1
```

```
-106456.2
```

El mejor modelo es el 4.1.

Modelos de quinto orden

```
modelo_5.1 <-glm(victimas_vfisica ~  agresor_0_39 + victima_
denuncia + agresor_trabaja + agresor_edu + victima_0_39,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_5.1)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_trabaja + agresor_edu + victima_0_39, family =
poisson(link = log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0115  -0.8980  -0.7788   0.6566   1.1577
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.364471   0.015231 -89.585  < 2e-16 ***
## agresor_0_39      0.322228   0.009761  33.013  < 2e-16 ***
## victima_denuncia  0.200569   0.010669  18.799  < 2e-16 ***
## agresor_trabaja   0.171331   0.010141  16.895  < 2e-16 ***
## agresor_edu      -0.045027   0.008751  -5.145 2.67e-07 ***
## victima_0_39     -0.021615   0.010909  -1.981   0.0476 *
```

```
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101352  on 140827  degrees of freedom
## AIC: 212898
##
## Number of Fisher Scoring iterations: 5

modelo_5.2 <-glm(victimas_vfisica ~ agresor_0_39 + victima_
denuncia + agresor_trabaja + agresor_edu + agresor_extranjero,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_5.2)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_trabaja + agresor_edu + agresor_extranjero, family
= poisson(link = log),
##     data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0523  -0.8984  -0.7741   0.6565   1.1512
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.375801   0.014163 -97.140  < 2e-16 ***
## agresor_0_39        0.312648   0.008851  35.323  < 2e-16 ***
```

```
## victima_denuncia    0.201070   0.010662  18.859  < 2e-16 ***
## agresor_trabaja     0.170617   0.010133  16.837  < 2e-16 ***
## agresor_edu        -0.045429   0.008752  -5.191 2.09e-07 ***
## agresor_extranjero  0.100320   0.034389   2.917  0.00353 **
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101348  on 140827  degrees of freedom
## AIC: 212894
##
## Number of Fisher Scoring iterations: 5
```

◊ *Comparamos los modelos de quinto orden*

```
#Definir la lista de modelos
models5 <- list(modelo_5.1, modelo_5.2 )

#Especificar los nombres del modelo
mod.names5 <- c('victima_0_39', 'agresor_extranjero')


#Calcular el AIC de cada modelo
aictab(cand.set = models5, modnames = mod.names5)

##
## Model selection based on AICc:
##
##                    K     AICc Delta_AICc AICcWt Cum.Wt
LL
## agresor_extranjero 6 212893.6       0.00    0.9    0.9
```

```
-106440.8
## victima_0_39       6 212898.0       4.33    0.1    1.0
-106443.0
```

Nos quedamos con el modelo 5.2.

Modelos de sexto orden

```
modelo_6 <-glm(victimas_vfisica ~ agresor_0_39 + victima_
denuncia + agresor_trabaja + agresor_edu + agresor_extranjero +
victima_0_39,
               data=violencia_fisica,
               family=poisson(link=log))

summary(modelo_6)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 + victima_
denuncia +
##     agresor_trabaja + agresor_edu + agresor_extranjero +
victima_0_39,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0633  -0.8970  -0.7789   0.6583   1.1582
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.364421   0.015231 -89.582  < 2e-16 ***
## agresor_0_39        0.320979   0.009771  32.849  < 2e-16 ***
## victima_denuncia    0.200248   0.010670  18.768  < 2e-16 ***
```

```
## agresor_trabaja     0.171412   0.010141  16.903  < 2e-16 ***
## agresor_edu        -0.045367   0.008752  -5.184 2.18e-07 ***
## agresor_extranjero  0.101322   0.034393   2.946  0.00322 **
## victima_0_39       -0.022059   0.010911  -2.022  0.04320 *
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101344  on 140826  degrees of freedom
## AIC: 212892
##
## Number of Fisher Scoring iterations: 5
```

El mejor modelo es el 6, ya que tiene menor AIC que los modelos anteriores como el modelo 5.2 que tenía un AIC de 212893.6, modelo 4.1 correspondiente a 212899.9, modelo 3.2 igual 213189.8 y del modelo 2.3 igual 213203.0.

El mejor AIC debe ser menor porque indica que el modelo ajusta mejor los datos y tiene un equilibrio adecuado entre el ajuste y la complejidad.

Análisis post-regresión

Modificador de efecto

◊ *Modelo 7*

```
modelo_7.1 <-glm(victimas_vfisica ~ agresor_0_39 * victima_
denuncia + agresor_trabaja + agresor_edu + agresor_extranjero +
victima_0_39,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_7.1)
```

```
##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 * victima_
denuncia +
##     agresor_trabaja + agresor_edu + agresor_extranjero +
victima_0_39,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0634  -0.8971  -0.7796   0.6581   1.1570
##
## Coefficients:
##                                Estimate Std. Error z value
Pr(>|z|)
## (Intercept)                   -1.362519   0.018847 -72.296  <
2e-16 ***
## agresor_0_39                   0.317975   0.020088  15.829  <
2e-16 ***
## victima_denuncia               0.197873   0.017496  11.310  <
2e-16 ***
## agresor_trabaja                0.171410   0.010141  16.903  <
2e-16 ***
## agresor_edu                   -0.045357   0.008752  -5.183
2.19e-07 ***
## agresor_extranjero             0.101277   0.034394   2.945
0.00323 **
## victima_0_39                  -0.022095   0.010913  -2.025
0.04290 *
## agresor_0_39:victima_denuncia  0.003775   0.022053   0.171
0.86410
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
```

```
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101344  on 140825  degrees of freedom
## AIC: 212894
##
## Number of Fisher Scoring iterations: 5

modelo_7.2 <-glm(victimas_vfisica ~ agresor_0_39 * agresor_trabaja
+ victima_denuncia + agresor_trabaja + agresor_edu + agresor_
extranjero + victima_0_39,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_7.2)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 * agresor_trabaja
+
##     victima_denuncia + agresor_trabaja + agresor_edu +
agresor_extranjero +
##     victima_0_39, family = poisson(link = log), data =
violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0643  -0.8946  -0.7779   0.6571   1.1519
##
## Coefficients:
##                               Estimate Std. Error z value
Pr(>|z|)
## (Intercept)                  -1.354391   0.018769 -72.160  <
```

```
2e-16 ***
## agresor_0_39                   0.306165   0.018971  16.139  <
2e-16 ***
## agresor_trabaja                0.158908   0.017052   9.319  <
2e-16 ***
## victima_denuncia               0.200227   0.010670  18.766  <
2e-16 ***
## agresor_edu                   -0.045443   0.008752  -5.192
2.08e-07 ***
## agresor_extranjero             0.101229   0.034393   2.943
0.00325 **
## victima_0_39                  -0.022461   0.010920  -2.057
0.03971 *
## agresor_0_39:agresor_trabaja  0.019285   0.021183   0.910
0.36261
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101343  on 140825  degrees of freedom
## AIC: 212893
##
## Number of Fisher Scoring iterations: 5

modelo_7.3 <-glm(victimas_vfisica ~ agresor_0_39 * agresor_edu
+ victima_denuncia + agresor_trabaja  + agresor_extranjero +
victima_0_39,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_7.3)
```

```
##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 * agresor_edu +
##     victima_denuncia + agresor_trabaja + agresor_extranjero +
##     victima_0_39, family = poisson(link = log), data =
violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0593  -0.8990  -0.7834   0.6550   1.1636
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.35258    0.01664 -81.308  < 2e-16
***
## agresor_0_39              0.30186    0.01463  20.635  < 2e-16
***
## agresor_edu              -0.06531    0.01434  -4.553 5.28e-06
***
## victima_denuncia          0.20036    0.01067  18.778  < 2e-16
***
## agresor_trabaja           0.17119    0.01014  16.880  < 2e-16
***
## agresor_extranjero        0.10129    0.03439   2.945  0.00323
**
## victima_0_39             -0.02218    0.01091  -2.033  0.04205
*
## agresor_0_39:agresor_edu  0.03168    0.01807   1.753  0.07955
.
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
```

```
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101340  on 140825  degrees of freedom
## AIC: 212890
##
## Number of Fisher Scoring iterations: 5

modelo_7.4 <-glm(victimas_vfisica ~ agresor_0_39 *  agresor_
extranjero + victima_denuncia + agresor_trabaja + agresor_edu +
victima_0_39,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_7.4)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 * agresor_
extranjero +
##     victima_denuncia + agresor_trabaja + agresor_edu +
victima_0_39,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0741  -0.8968  -0.7791   0.6587   1.1829
##
## Coefficients:
##                                 Estimate Std. Error z value
Pr(>|z|)
## (Intercept)                    -1.363674   0.015237 -89.495
< 2e-16 ***
## agresor_0_39                    0.319698   0.009806  32.603
< 2e-16 ***
## agresor_extranjero             -0.038879   0.098802  -0.394
```

```
0.6939
## victima_denuncia                 0.200188   0.010670  18.762
< 2e-16 ***
## agresor_trabaja                  0.171307   0.010141  16.892
< 2e-16 ***
## agresor_edu                     -0.045245   0.008752  -5.170
2.35e-07 ***
## victima_0_39                    -0.021896   0.010912  -2.007
0.0448 *
## agresor_0_39:agresor_extranjero  0.161140   0.105387   1.529
0.1263
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101341  on 140825  degrees of freedom
## AIC: 212891
##
## Number of Fisher Scoring iterations: 5

modelo_7.5 <-glm(victimas_vfisica ~ agresor_0_39 * victima_0_39
+ victima_denuncia + agresor_trabaja +  agresor_edu + agresor_
extranjero ,
                 data=violencia_fisica,
                 family=poisson(link=log))

summary(modelo_7.5)

##
## Call:
## glm(formula = victimas_vfisica ~ agresor_0_39 * victima_0_39 +
##     victima_denuncia + agresor_trabaja + agresor_edu +
```

```
agresor_extranjero,
##     family = poisson(link = log), data = violencia_fisica)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0557  -0.8882  -0.7863   0.6512   1.1769
##
## Coefficients:
##                            Estimate Std. Error z value
Pr(>|z|)
## (Intercept)               -1.328331   0.015961 -83.223  < 2e-
16 ***
## agresor_0_39               0.200157   0.019653  10.185  < 2e-
16 ***
## victima_0_39              -0.086381   0.014134  -6.112 9.86e-
10 ***
## victima_denuncia           0.200166   0.010669  18.761  < 2e-
16 ***
## agresor_trabaja            0.167184   0.010156  16.461  < 2e-
16 ***
## agresor_edu               -0.045694   0.008752  -5.221 1.78e-
07 ***
## agresor_extranjero         0.099791   0.034394   2.901
0.00371 **
## agresor_0_39:victima_0_39  0.162705   0.022746   7.153 8.49e-
13 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103324  on 140832  degrees of freedom
## Residual deviance: 101292  on 140825  degrees of freedom
## AIC: 212842
```

```
##
## Number of Fisher Scoring iterations: 5
```

◊ *Comparando los modelos con modificador de efecto*

```
#Definir la lista de modelos
models7 <- list(modelo_7.1, modelo_7.2 , modelo_7.3,modelo_7.4,
modelo_7.5)

#Especificar los nombres del modelo
mod.names7 <- c('me_denuncia','me_agresor_edu', 'me_agresor_
trabaja',  'me_agresor_extranjero', 'me_victima_0_39')

#Calcular el AIC de cada modelo
aictab(cand.set = models7, modnames = mod.names7)

##
## Model selection based on AICc:
##
##                       K     AICc Delta_AICc AICcWt Cum.Wt
LL
## me_victima_0_39       8 212841.6       0.00      1      1
-106412.8
## me_agresor_trabaja    8 212890.5      48.91      0      1
-106437.2
## me_agresor_extranjero 8 212891.1      49.55      0      1
-106437.6
## me_agresor_edu        8 212892.7      51.16      0      1
-106438.4
## me_denuncia           8 212893.5      51.96      0      1
-106438.8
```

El modelo final es el 7.5.

Análisis de sensibilidad de valores influyentes

◊ *Diagnóstico de métricas del modelo final 7.5*

Puedes aumentar fácilmente tus datos para agregar valores ajustados y residuales usando la función `augment` del paquete `broom`. Denominamos al objeto **model.diag.metrics** porque contiene varias métricas útiles para el diagnóstico de regresión.

```
model.diag.metrics <- augment(modelo_7.5)

head(model.diag.metrics)

## # A tibble: 6 x 13
##   victimas_vfisica agresor_0_39 victima_0_39 victima_denuncia
agresor_trabaja
##              <dbl>        <dbl>        <dbl>             <dbl>
<dbl>
## 1                0            0            1                 1
0
## 2                0            0            1                 1
0
## 3                1            1            1                 1
0
## 4                1            0            1                 1
1
## 5                0            0            1                 1
1
## 6                1            1            1                 1
0
## # ... with 8 more variables: agresor_edu <dbl>, agresor_
extranjero <dbl>,
## #   .fitted <dbl>, .resid <dbl>, .std.resid <dbl>, .hat <dbl>,
.sigma <dbl>,
## #   .cooksd <dbl>
```

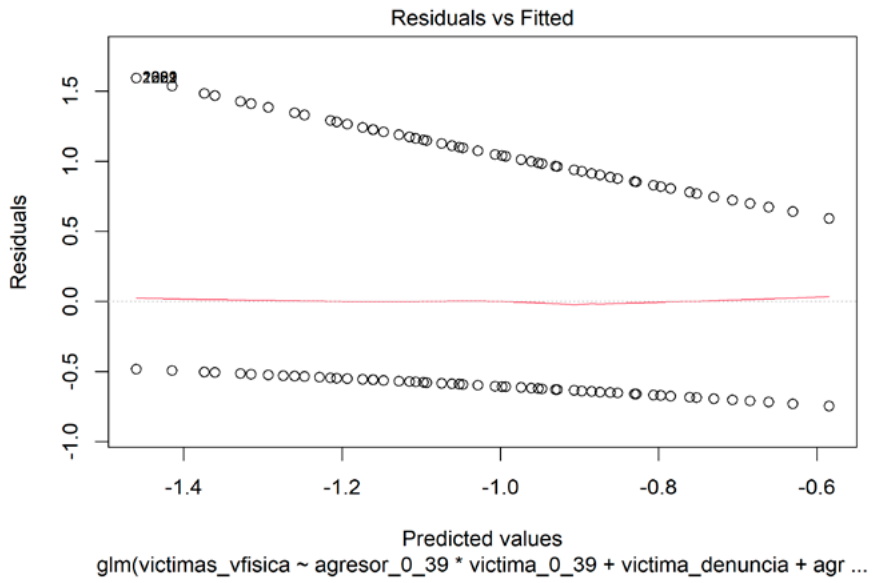
Diagnóstico del modelo

Las gráficas de diagnóstico muestran residuos de cuatro maneras diferentes:

◊ *Residuales vs. ajustados*

Se utiliza para comprobar los supuestos de asociación lineal. Una línea horizontal, sin patrones definidos, es una indicación de una asociación lineal, lo que es bueno.

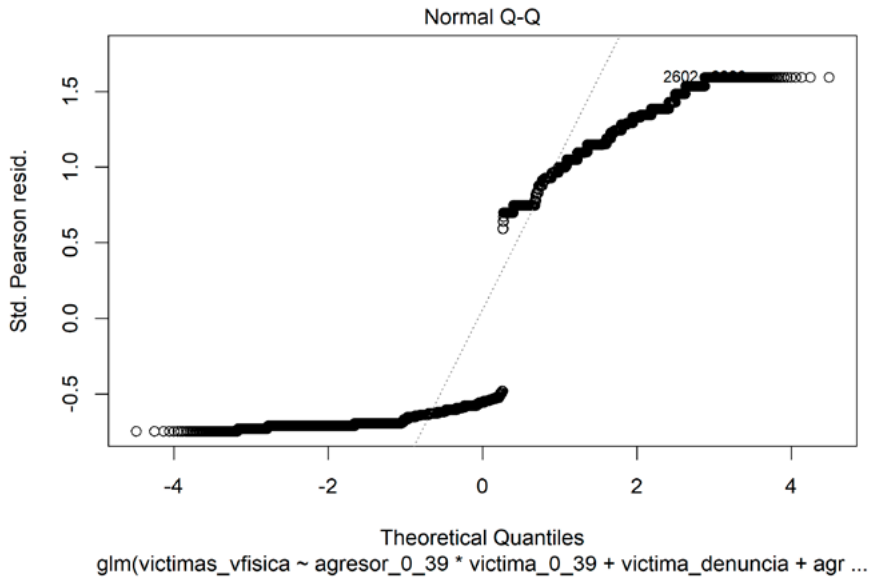
```
plot(modelo_7.5, 1)
```



◊ *Q-Q Normal*

Se utiliza para examinar si los residuos se distribuyen normalmente. Es bueno si los puntos residuales siguen la línea recta discontinua.

```
plot(modelo_7.5, 2)
```

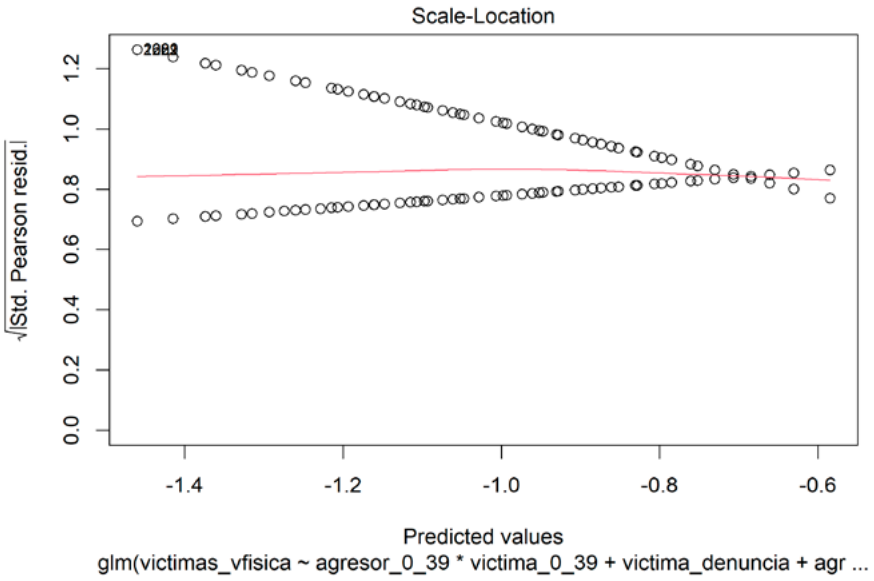


En el gráfico Q-Q normal, se compara la distribución de los residuos del modelo con la distribución normal. Si los residuos se distribuyen normalmente, los puntos en el gráfico seguirán aproximadamente la línea recta discontinua que representa la distribución normal. En este caso, los puntos se desvían de la línea, esto indica que los residuos no se distribuyen normalmente, es decir, hay una desviación de la normalidad y esto puede afectar la fiabilidad del modelo.

◊ *Scale-location* (o *spread-location*)

Se utiliza para comprobar la homogeneidad de varianza de los residuos (homocedasticidad). La línea horizontal con puntos igualmente separados es una buena indicación de homocedasticidad. Este no es el caso en nuestro ejemplo, donde tenemos un problema de heterocedasticidad. La presencia de heterocedasticidad en un modelo de regresión puede tener varias consecuencias como reducir la fiabilidad y precisión del modelo de regresión, afectar la validez de las inferencias estadísticas y reducir su capacidad para explicar la variabilidad de la variable dependiente.

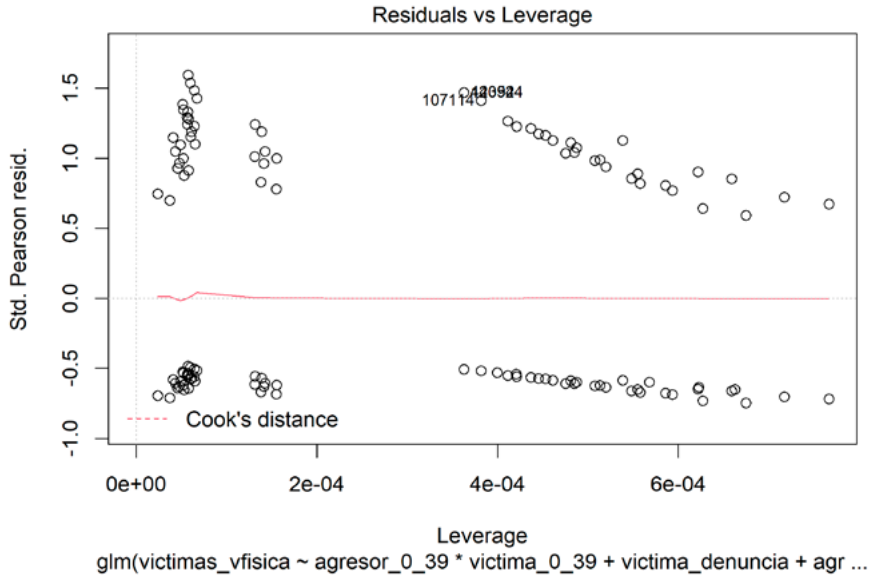
```
plot(modelo_7.5,3)
```



◊ *Residuales vs. apalancamiento.*

Se utiliza para identificar casos influyentes, es decir, valores extremos que pueden influir en los resultados de la regresión cuando se incluyen o excluyen del análisis.

```
plot(modelo_7.5, 5)
```



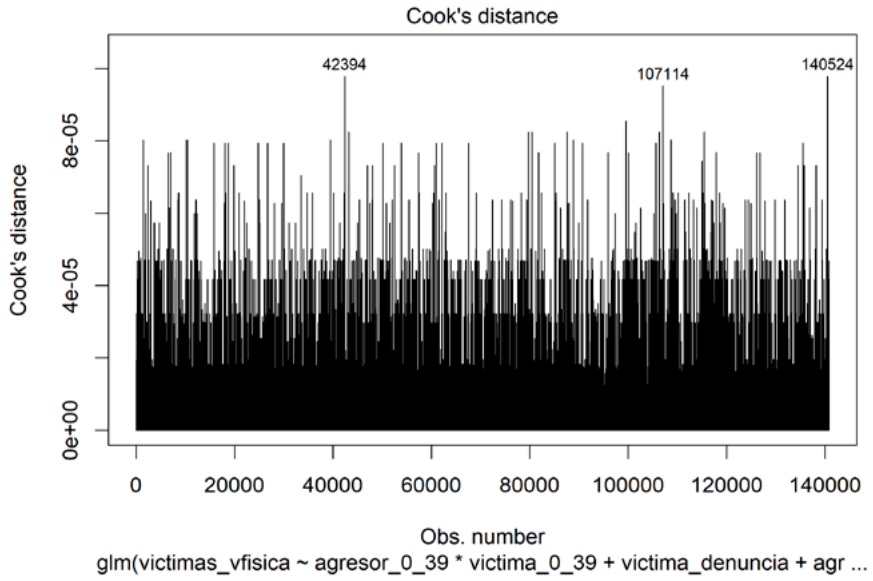
El eje horizontal de este gráfico representa los valores de apalancamiento, que miden la influencia que tiene cada observación en la estimación de los coeficientes del modelo. Las observaciones con un valor de apalancamiento cercano a 1 tienen un mayor efecto en la estimación de los coeficientes del modelo. El eje vertical representa los residuos estandarizados, que se utilizan para identificar valores extremos que se alejan significativamente de la línea de regresión.

Por lo tanto, cada punto representa una observación en su conjunto de datos. La ausencia de puntos ubicados en la línea roja del gráfico sugiere que no hay casos influyentes en su modelo de regresión. En otras palabras, no hay observaciones que tengan un alto valor de apalancamiento y residuos estandarizados muy altos. Sin embargo, es importante tener en cuenta que esto no garantiza que el modelo sea perfecto.

◊ *Distancia de Cook*

El gráfico "Distancia de Cook" es otra herramienta comúnmente utilizada para identificar casos influyentes en un modelo de regresión.

```
plot (modelo_7.5,4)
```



En este gráfico, las barras representan la distancia de Cook de cada observación en su conjunto de datos. Si las barras son cortas y están cerca de cero, esto sugiere que la eliminación de esa observación no tendría un impacto significativo en los resultados del modelo. Sin embargo, tenemos presencia de barras largas, que podrían indicar la presencia de valores extremos o casos influyentes que están afectando significativamente los resultados de su modelo.

Si deseas ver estas 3 observaciones principales (las que tienen la distancia de Cook más alta) a fin de evaluarlas más a fondo, para determinar si son casos influyentes que están afectando significativamente los resultados del modelo, puedes hacerlo escribiendo el siguiente código:

```
model.diag.metrics %>%
  top_n(3, wt = .cooksd)
```

```
## # A tibble: 3 x 13
##   victimas_vfisica agresor_0_39 victima_0_39 victima_denuncia
agresor_trabaja
##              <dbl>        <dbl>        <dbl>            <dbl>
<dbl>
## 1                1            0            1                0
0
## 2                1            0            1                0
0
## 3                1            0            1                0
0
## # ... with 8 more variables: agresor_edu <dbl>, agresor_
extranjero <dbl>,
## #   .fitted <dbl>, .resid <dbl>, .std.resid <dbl>, .hat <dbl>,
.sigma <dbl>,
## #   .cooksd <dbl>
```

El código utiliza la función `top_n` del paquete `dplyr` para seleccionar las tres observaciones principales que tienen la distancia de Cook más alta en el modelo de regresión (`wt = .cooksd`). Asimismo, `model.diag.metrics` especifica el modelo de regresión que se ajustó previamente. Finalmente, el resultado se presenta en una tabla (`tibble`) que muestra las variables para las tres observaciones seleccionadas. El número y las variables que aparecen en la tabla dependen de las variables que se incluyeron en el modelo de regresión.

En nuestro ejemplo, los datos no presentan ningún punto influyente. Las líneas de distancia de Cook (una línea discontinua roja) no se muestran en el gráfico de residuales vs. apalancamiento porque todos los puntos están bien dentro de las líneas de distancia de Cook.

Intervalo de confianza

La función `confint` calcula los intervalos de confianza para los parámetros de un modelo ajustado.

```
confint(modelo_7.5)

##                                 2.5 %      97.5 %
## (Intercept)                -1.35967832 -1.29711044
## agresor_0_39                0.16151757  0.23855968
## victima_0_39               -0.11408452 -0.05867811
## victima_denuncia            0.17929785  0.22112255
## agresor_trabaja             0.14731311  0.18712575
## agresor_edu                -0.06283629 -0.02852929
## agresor_extranjero          0.03164198  0.16648191
## agresor_0_39:victima_0_39   0.11822110  0.20738877
```

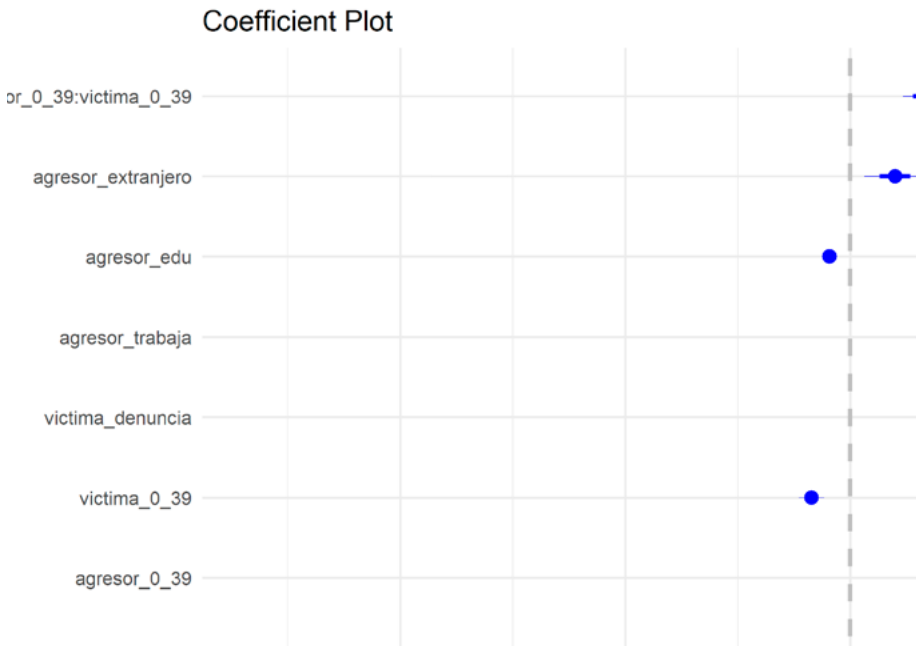
◊ *Intervalo de confianza del* odds ratio *o de las razones de probabilidad*

```
exp(confint(modelo_7.5, level=0.99))

## Waiting for profiling to be done...

##                                    0.5 %    99.5 %
## (Intercept)                    0.2542202 0.2760078
## agresor_0_39                   1.1610514 1.2847690
## victima_0_39                   0.8844490 0.9512549
## victima_denuncia               1.1885790 1.2557411
## agresor_trabaja                1.1515210 1.2133762
## agresor_edu                    0.9340567 0.9771346
## agresor_extranjero             1.0099685 1.2058016
## agresor_0_39:victima_0_39      1.1099151 1.2479116
```

◊ *Grado de coeficientes*

A partir de los intervalos de confianza de los coeficientes se puede obtener una salida gráfica de la estimación del modelo, esto nos permite ubicar las pendientes relativas de cada predictor.

```
coefplot(modelo_7.5)  +
  theme_minimal()
```



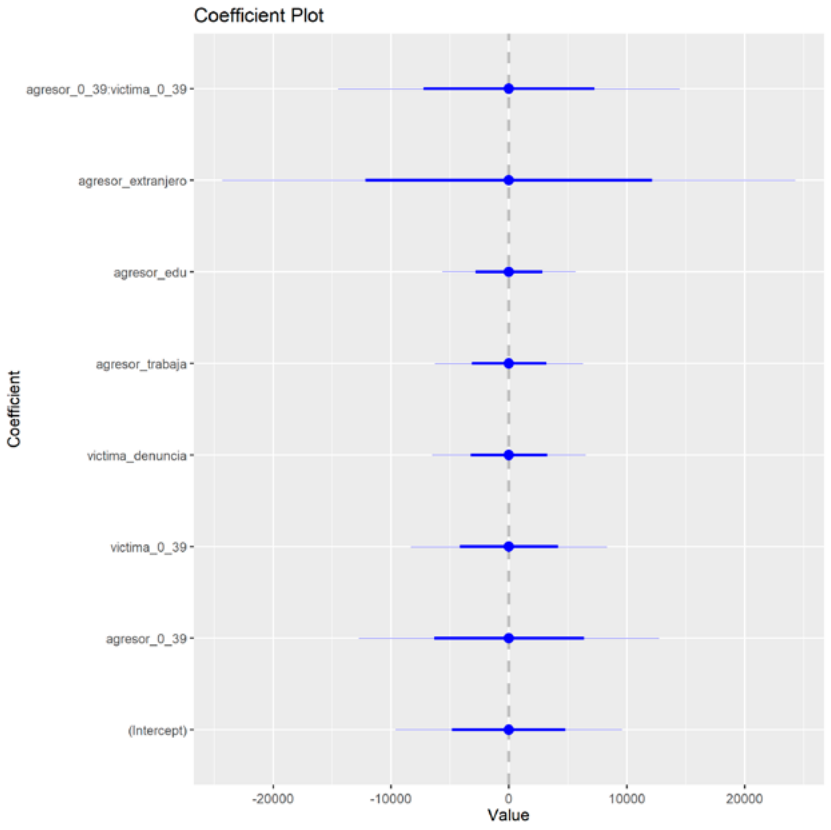
Las pendientes relativas se refieren a la magnitud de la asociación entre cada predictor y la variable de respuesta en comparación con los otros predictores incluidos en el modelo.

En el gráfico obtenido con la función `coefplot()` se pueden leer los coeficientes estimados para cada predictor y sus respectivos intervalos de confianza, representados por las líneas horizontales que se extienden desde cada punto. Los valores para cada predictor indican el cambio en la *log-odds* o razón de probabilidad asociado a un cambio de una unidad en el predictor, manteniendo todos los demás

predictores constantes. Asimismo, la dirección de la asociación (positiva o negativa) se puede inferir a partir del signo del coeficiente estimado.

La función `theme_minimal()` define un tema minimalista que elimina los elementos visuales no esenciales, como los bordes del panel y los ejes secundarios, lo que resulta en una visualización más limpia y simple. De esta manera, personalizamos la apariencia del gráfico que muestra los coeficientes de un modelo de regresión para mejorar su comprensión. A continuación, se mostrará el gráfico de coeficientes por default que solo toma como entrada un modelo de regresión.

```
coefplot(modelo_7.5)
```



Resumen del capítulo

En este capítulo se revisaron las principales aplicaciones de la regresión de Poisson, tanto para estimar el *incidence rate ratio* o razón de tasas de incidencia como para estimar el *prevalence ratio* o razón de prevelancencias. En el ejemplo utilizado, el mismo exigió un manejo de datos previo a fin de que la base de datos estuviera preparada para utilizar la razón de prevalencia como magnitud de asociación de interés. Esto es importante porque para poder modelar variables dicotómicas en R primero tenemos que convertir la variable cualitativa en una variable cuantitativa. Una vez ajustado el modelo final de nuestra análisis de regresión multivariables, ya sea con el método forward o el método stepwise, es importante realizar un buen análisis post-regresión, incluyendo como mínimo la verificación de los supuestos y un análisis de sensibilidad. Para ello debemos diagnosticar la bondad de ajuste del modelo, determinar si hay patrones sistemáticos en los residuos, puntos influyentes o valores atípicos que podrían afectar la bondad de ajuste del modelo.

CAPÍTULO 10

REGRESIÓN BINOMIAL

En una regresión binomial, la variable dependiente es categórica y dicotómica. Este tipo de regresión suele utilizarse para predecir la probabilidad de un determinado evento. Al igual que en capítulos anteriores, realizaremos un análisis exploratorio de datos y un análisis de correlación antes de iniciar la regresión binomial. Una vez seleccionado el mejor modelo, realizamos un diagnóstico del mismo a fin de confirmar su ajuste.

Sin embargo, en la regresión binomial, se asume que la relación entre las variables predictoras y la variable de respuesta sigue una distribución binomial, lo que puede no ser cierto si las variables predictoras son continuas. Por lo tanto, se requiere un enfoque diferente para modelar relaciones no lineales en variables continuas, tales como la regresión cuantílica o la regresión polinómica.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Realizar un análisis exploratorio y de correlación previo a una regresión binomial.
- Calcular los AIC, comparar modelos y aplicar la prueba de Wald.
- Realizar un análisis post-regresión binomial.

Paquetes

Instalación

```
#instalamos paquetes

#install.packages("readr")
#install.packages("dplyr")
#install.packages("VIM")
```

```
#install.packages("ggplot2")
#install.packages("corrplot")
#install.packages("AICcmodavg")
#install.packages("coefplot")
#install.packages("effects")
```

Carga

```
#llamamos a los paquetes
library(haven) #lee archivos .dta
library(dplyr)#manipular data
library(scales)
library(VIM) #valores perdidos
library(ggplot2) # gráficos
library(corrplot) #correlación
library(AICcmodavg) #comparar modelos
library(coefplot)#coeficientes
library(effects) #efectos
```

Estudio de caso

Para realizar la regresión binomial utilizaremos la base de datos ubicada en la web de la Universidad de California que aprendimos a importar en el capítulo 1. Dicha base de datos contiene información sobre la admisión a programas de posgrado universitario. Son 4 las variables que contiene esta la base de datos:

Variable	Descripción
admit (variable dependiente)	0 = no admitido, 1 = admitido
gre	Nota del examen GRE (puntuación en el examen de registro de graduados, oscila entre 200 y 800).

Variable	Descripción
gpa	El promedio de calificaciones acumulado, puede tomar valores de entre 0 y 4.
rank	Ranking (prestigio) de la universidad de pregrado

Se busca, entonces, evaluar si dichas variables afectan la admisión a una universidad de posgrado, donde la variable "admit" es la variable de respuesta binaria.

Cargamos la base de datos y la guardamos en el objeto **data**.

```
data <-read_dta("http://stats.idre.ucla.edu/stat/stata/dae/
binary.dta")
```

Análisis exploratorio

```
str(data)

## tibble [400 x 4] (S3: tbl_df/tbl/data.frame)
##  $ admit: num [1:400] 0 1 1 1 0 1 1 0 1 0 ...
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ gre  : num [1:400] 380 660 800 640 520 760 560 400 540 700
...
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ gpa  : num [1:400] 3.61 3.67 4 3.19 2.93 ...
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ rank : num [1:400] 3 3 1 4 4 2 1 2 3 2 ...
##   ..- attr(*, "format.stata")= chr "%9.0g"
```

Todas las variables de la base de datos son numéricas.

```
#Dimensión del data frame (número de filas y columnas)
dim(data)
```

```
## [1] 400   4
```

Hay 400 filas, es decir, se tienen los datos de 400 estudiantes.

Valores perdidos

```
table(complete.cases(data)) # Explorar

##
## TRUE
##  400
```

No hay ningún valor perdido.

Descripción de los datos

Si bien "admit" y "rank" aparecen en la base de datos como variables con valores numéricos, en realidad son variables categóricas porque, por ejemplo, en el caso de "admit", esta solo puede tomar los valores 0 y 1, que significan "no admitidos" y "admitidos". Entonces, para explorar los datos de nuestro universo, especificamos las columnas 2 y 3 ("gre" y "gpa", respectivamente), que son numéricas, y utilizamos la función `summary`, que es apropiada para ellas.

```
summary(data[,2:3])

##       gre             gpa
##  Min.   :220.0   Min.   :2.260
##  1st Qu.:520.0   1st Qu.:3.130
##  Median :580.0   Median :3.395
##  Mean   :587.7   Mean   :3.390
##  3rd Qu.:660.0   3rd Qu.:3.670
##  Max.   :800.0   Max.   :4.000
```

Del resultado, observamos que el promedio de calificaciones ("gpa") es 3.390. y que la nota del examen GRE es 587.7. Veamos, ahora, las variables categóricas con la función `table`.

```
table(data$admit)

##
##   0   1
## 273 127
```

273 alumnos no fueron admitidos.

```
table(data$rank)

##
##   1   2   3   4
##  61 151 121  67
```

En cuanto al nivel de prestigio de la universidad, la mayoría de los estudiantes provienen de universidades de un prestigio regular.

Exploración gráfica

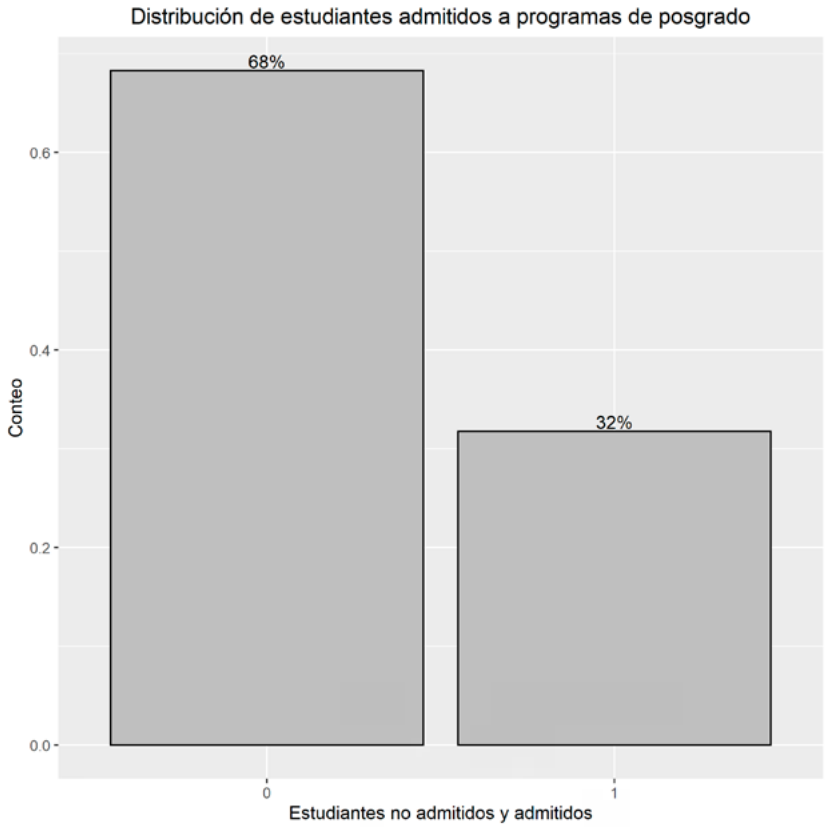
Antes de realizar los gráficos, convertimos en factores las variables "admit" y "rank" para poder visualizarlas correctamente. Antes, guardamos la base de datos en un nuevo objeto.

```
data1 <- data
data1$admit <-as.factor(data1$admit)
data1$rank <-as.factor(data1$rank)

str(data1) #comprobamos

## tibble [400 x 4] (S3: tbl_df/tbl/data.frame)
```

```
##  $ admit: Factor w/ 2 levels “0”,”1”: 1 2 2 2 1 2 2 1 2 1 ...
##  $ gre  : num [1:400] 380 660 800 640 520 760 560 400 540 700
...
##   ..- attr(*, “format.stata”)= chr “%9.0g”
##  $ gpa  : num [1:400] 3.61 3.67 4 3.19 2.93 ...
##   ..- attr(*, “format.stata”)= chr “%9.0g”
##  $ rank : Factor w/ 4 levels “1”,”2”,”3”,”4”: 3 3 1 4 4 2 1 2
3 2 ...
plot <- data1 %>% count(admit) %>%
  mutate(count=n/sum(n), countlabel= paste0(round(count*100),
“%”))
```

El comando anterior tiene como objetivo crear una tabla resumen de la variable “admit” que muestra la frecuencia y proporción de cada nivel en la muestra. El operador `%>%` se utiliza para pasar la base de datos a la función `count`, que cuenta el número de observaciones para cada valor de “admit”. Después, el operador `%>%` se utiliza nuevamente para pasar los resultados anteriores a la función `mutate`, que crea dos nuevas variables: “count”, que es la proporción de observaciones en cada nivel de “admit” y “countlabel”, que es una etiqueta que indica la proporción de cada nivel en términos porcentuales. Por último, el resultado se almacena en un objeto llamado **plot**, que se puede utilizar para crear gráficos o realizar análisis posteriores.

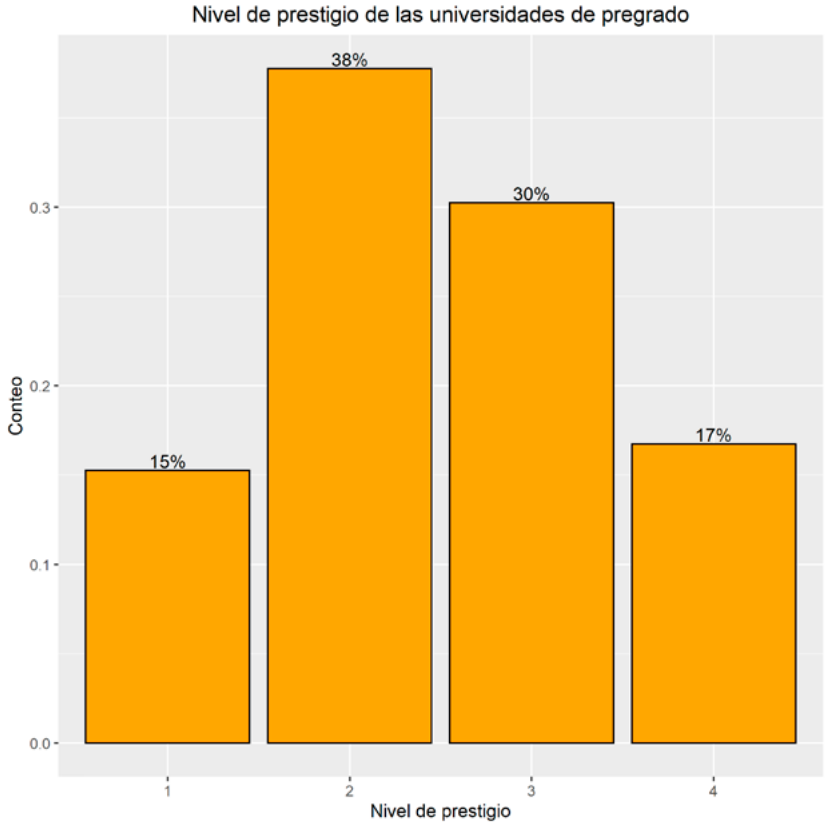
```
ggplot(plot, aes(x = admit, y=count)) +
  geom_bar(stat=”identity”, fill= “gray”,
           color= “black”) +
    labs(x = “Estudiantes no admitidos y admitidos”,
       y = “Conteo”,
       title = “Distribución de estudiantes \n a programas de
postgrado”) +
  scale_y_continuous(labels = comma) +
geom_text(aes(label=countlabel), vjust=-0.2)  +
  theme(plot.title = element_text(hjust = 0.5)) #centrar título
```



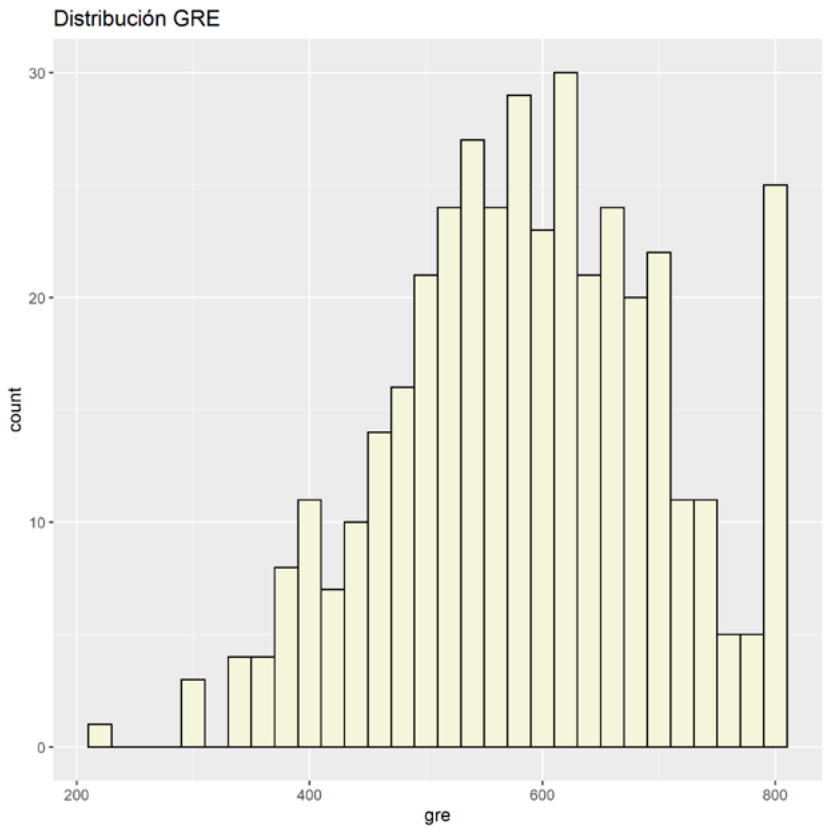
El 68% de estudiantes no fueron admitidos, mientras que el 32% sí lo fue.

```
plot <- data1 %>% count(rank) %>%
  mutate(count=n/sum(n), countlabel= paste0(round(count*100),
"%"))


ggplot(plot, aes(x = rank, y=count)) +
  geom_bar(stat="identity", fill= "orange",
           color= "black") +
    labs(x = "Nivel de prestigio",
       y = "Conteo",
       title = "Nivel de prestigio de las universidades de
```

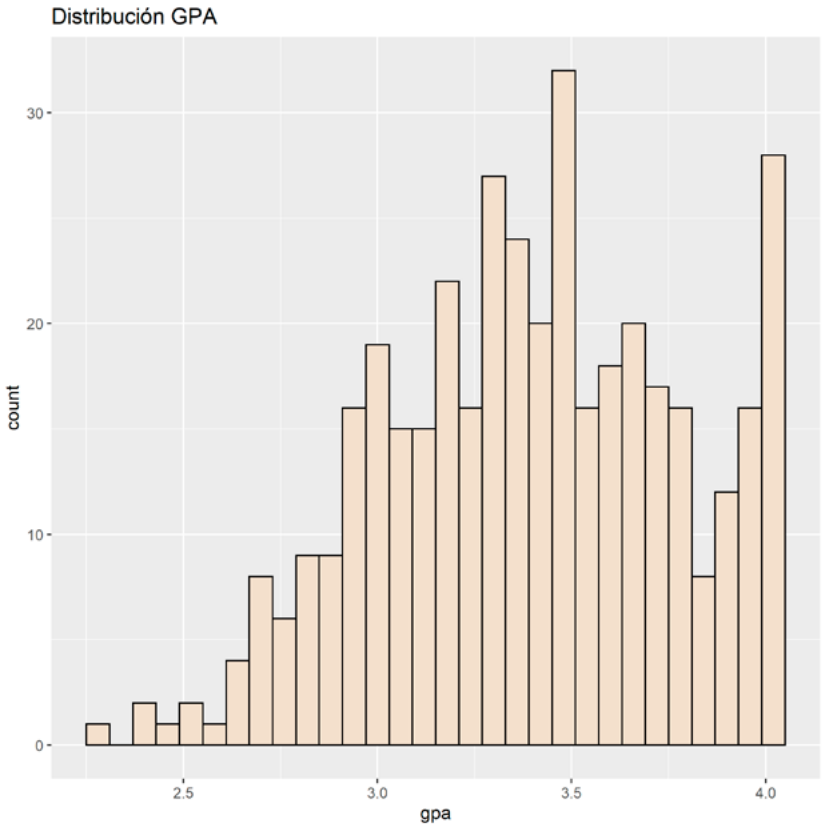
```
pregrado \n de las que provienen los candidatos a postgrado") +
  scale_y_continuous(labels = function(x) x*nrow(data1)) +
geom_text(aes(label=countlabel), vjust=-0.2)  +
  theme(plot.title = element_text(hjust = 0.5))
```




```
ggplot(data1, aes(x = gre)) +
  geom_histogram(fill= "beige", color="black") +
  labs(x = "Puntaje obtenido en el gre",y = "Conteo",title =
"Distribución GRE")
```




```
ggplot(data1, aes(x = gpa)) +
  geom_histogram(fill= “antiquewhite2”, color=”black”) +
  labs(x = “Promedio de calificaciones acumulado GPA”,y =
“Conteo”, title = “Distribución GPA”)
```

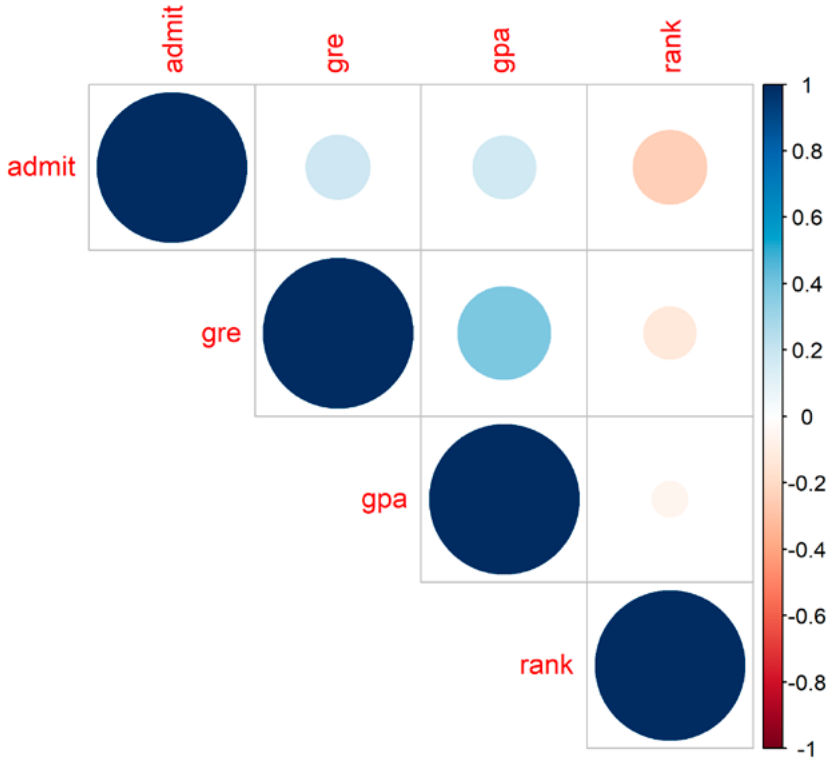


En los dos últimos gráficos, notamos una distribución con tendencia a la derecha.

Análisis de correlación

Para realizar la correlación, utilizamos la primera base de datos denominada “data”, ya que ahí se encuentran todas las variables numéricas sin modificar.

```
correlacion <-cor(data)
corrplot(correlacion, type=”upper”)
```



La variable "admit" tiene una fuerte correlación con todas las variables independientes. Con la variable "rank", la correlación es fuerte, pero negativa. Esto quiere decir que existe una asociación entre los estudiantes admitidos y las universidades de pregrado de alto prestigio, pues un número menor de "rank" significa mayor prestigio y mientras más bajo sea "rank" más es la probabilidad de ser admitido.

Regresión binomial

Son modelos que expresan la asociación entre una variable de desenlace (*Y*) dicotómica y una o más variables independientes (*X*'s). La regresión binomial suele tener problemas de convergencia: aparecen errores en el modelo producto de una falla propia de la regresión binomial. En ese caso, una alternativa es comprobar los modelos con Poisson.

A continuación, usamos la familia binomial.

Método forward

◊ *Modelo nulo*

```
model0 <- glm(admit ~ gre, family = binomial (link="log"), data
= data)

summary(model0)

##
## Call:
## glm(formula = admit ~ gre, family = binomial(link = "log"),
data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1829  -0.8951  -0.7570   1.3572   1.9279
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.5612565  0.4094800  -6.255 3.98e-10 ***
## gre          0.0023432  0.0006365   3.681 0.000232 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 486.34  on 398  degrees of freedom
## AIC: 490.34
##
## Number of Fisher Scoring iterations: 6
```

◊ *Modelos de primer orden*

```
modelo_1.1 <- glm(admit ~ gre, family=binomial(link = "log"),
data=data)

summary (modelo_1.1)

##
## Call:
## glm(formula = admit ~ gre, family = binomial(link = "log"),
data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1829  -0.8951  -0.7570   1.3572   1.9279
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.5612565  0.4094800  -6.255 3.98e-10 ***
## gre          0.0023432  0.0006365   3.681 0.000232 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 486.34  on 398  degrees of freedom
## AIC: 490.34
##
## Number of Fisher Scoring iterations: 6

modelo_1.2 <- glm( admit ~ gpa, family=binomial(link = "log"),
data=data)
```

```
summary (modelo_1.2)

##
## Call:
## glm(formula = admit ~ gpa, family = binomial(link = "log"),
data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1302  -0.8788  -0.7562   1.3306   1.9338
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.5836     0.7054  -5.080 3.78e-07 ***
## gpa           0.7082     0.1985   3.567 0.000361 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 486.93  on 398  degrees of freedom
## AIC: 490.93
##
## Number of Fisher Scoring iterations: 6

modelo_1.3 <- glm( admit ~ rank, family=binomial(link = "log"),
data=data)

summary (modelo_1.3)

##
## Call:
## glm(formula = admit ~ rank, family = binomial(link = "log"),
```

```
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2362  -0.9441  -0.7445   1.1197   1.9049
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.2311     0.1706  -1.354    0.176
## rank         -0.3958     0.0789  -5.017 5.25e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 475.18  on 398  degrees of freedom
## AIC: 479.18
##
## Number of Fisher Scoring iterations: 6
```

Comparando los modelos de primer orden

```
CalculoAIC <-list(modelo_1.1, modelo_1.2, modelo_1.3)
# especificar nombres
NamesAIC <- c('modelo1.1', 'modelo1.2', 'modelo1.3')

#calcular AIC
aictab(cand.set=CalculoAIC, modnames = NamesAIC)

##
## Model selection based on AICc:
##
```

```
##           K   AICc Delta_AICc AICcWt Cum.Wt      LL
## modelo1.3 2 479.21       0.00   0.99   0.99 -237.59
## modelo1.1 2 490.37      11.16   0.00   1.00 -243.17
## modelo1.2 2 490.96      11.75   0.00   1.00 -243.46
```

El modelo 1.3 es el que tiene el AIC menor, en consecuencia es el mejor modelo.

Análisis post regresión

Prueba de Wald

En modelos logit (un tipo de modelo de regresión logística utilizado para analizar datos binarios o dicotómicos, como el empleado en este caso) la prueba de hipótesis que se lleva a cabo para contrastar la hipótesis de que los coeficientes son diferentes de 0 es la prueba de Wald, que estima un estadístico □ o *z* de cada coeficiente, a diferencia del estadístico *t* que utilizamos en los modelos lineales con distribución gaussiana y enlace directo.

Para llevar a cabo la prueba de Wald, primero se deben extraer los coeficientes y errores estándar del modelo final y luego calcular los valores *z*. En este sentido, se puede obtener un subconjunto de la lista de coeficientes del modelo, en el que la columna 1 corresponde a los estimados y la columna 2 a los errores estándar.

```
#Subconjunto de la lista de coeficientes
##columna 1: estimados
summary(modelo_1.3)$coefficients[,1]

##  (Intercept)        rank
##  -0.2311016  -0.3958029

#Subconjunto de la lista de coeficientes
##columna 2: errores estándar
summary(modelo_1.3)$coefficients[,2]

## (Intercept)        rank
```

```
##  0.1706290   0.0788957
```

Estimar el valor *z*

Una vez se tienen los coeficientes y errores estándar, se puede calcular el valor *z*, que se obtiene dividiendo el estimado entre el error estándar. Este valor *z* nos indica el valor del coeficiente normalizado, es decir, dentro de una distribución de probabilidad normal con media 0 y desviación estándar 1.

```
ValorZ <- summary(modelo_1.1)$coefficients[,1]/
summary(modelo_1.1)$coefficients[,2]

ValorZ

## (Intercept)         rank
##   -1.354410   -5.016787
```

Estos valores *z* se utilizaron para calcular la probabilidad de que los coeficientes fueran significativamente diferentes de cero, utilizando una distribución normal estándar.

◊ *Probabilidad de z en una distribución normal con media 0 y sigma 1*

La probabilidad se calculará usando la función de probabilidad acumulada `pnorm()` y se multiplicará por dos para obtener la probabilidad bilateral. Utilizamos `pnorm` para estimar la probabilidad. Dado que es una función de probabilidad acumulada usamos `1-pnorm`.

```
ProbabilidadZ <- (1 - pnorm(abs(ValorZ), 0, 1)) * 2
ProbabilidadZ

##  (Intercept)          rank
```

```
## 1.756056e-01 5.254273e-07
```

El valor obtenido para "Intercept" fue de 0.1756056. Como esta probabilidad es mayor que el nivel de significancia comúnmente empleado de 0.05, no hay suficiente evidencia estadística para concluir que el coeficiente del intercepto es significativamente diferente de cero, lo que sugiere que el valor de "Intercept" no tiene un efecto significativo en la variable dependiente.

Por otro lado, para "rank" fue de 5.254273e-07. Como esta probabilidad es mucho menor que el nivel de significancia de 0.05, hay evidencia estadística para concluir que el coeficiente de "rank" es significativamente diferente de cero, lo que sugiere que la variable "rank" tiene un efecto significativo en la variable dependiente.

Intervalo de confianza

```
confint(modelo_1.3)

##                   2.5 %       97.5 %
## (Intercept)   -0.5881182  0.08429077
## rank          -0.5514152 -0.24036747
```

El intervalo de confianza del modelo 1.3 muestra los posibles valores de los coeficientes del intercepto y de "rank" que son consistentes con los datos y con un nivel de confianza del 95%. En este caso, el intervalo de confianza del intercepto va de -0.588 a 0.084, lo que sugiere que este coeficiente podría ser negativo o positivo y no se puede afirmar con confianza que sea diferente de cero. Por otro lado, el intervalo de confianza para el coeficiente de "rank" va de -0.551 a -0.240, lo que sugiere que este coeficiente es negativo y que la variable "rank" tiene un efecto negativo significativo en la variable dependiente.

Intervalo de confianza de las razones de probabilidad

```
exp(confint(modelo_1.3, level=0.99))
```

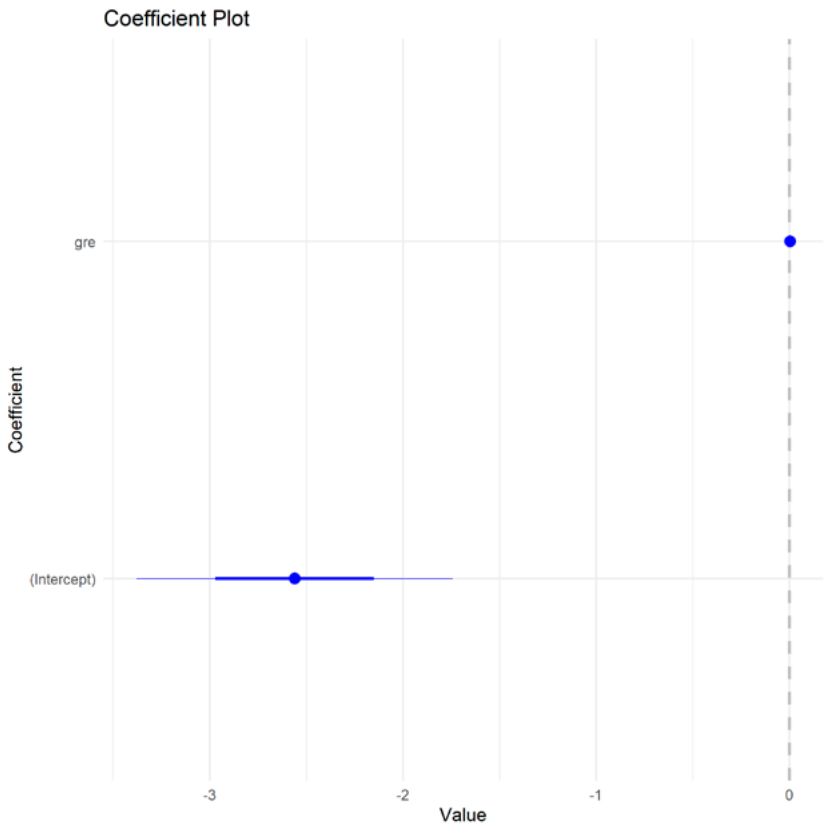
```
##                  0.5 %    99.5 %
## (Intercept)  0.4923448  1.190911
## rank         0.5484743  0.825772
```

Este valor se utiliza para expresar los coeficientes en términos de las razones de probabilidad. Los valores de este intervalo indican que la probabilidad de que la variable "rank" tenga un efecto significativo en la variable dependiente es entre 0.55 y 0.83 veces la probabilidad de que no tenga efecto. De manera similar, el intervalo de confianza para el intercepto sugiere que la probabilidad de que tenga un efecto significativo en la variable dependiente es entre 0.49 y 1.19 veces la probabilidad de que no tenga efecto.

Grado de coeficientes

A partir de los intervalos de confianza de los coeficientes se puede obtener una salida gráfica de la estimación del modelo, que es especialmente útil para ubicar las pendientes relativas de cada predictor.

```
coefplot(modelo_1.3)  +
  theme_minimal()
```



Graficando los efectos

```
allEffects(modelo_1.3)

##  model: admit ~ rank
##
##  rank effect
## rank
##       1       1.8       2.5       3.2         4
## 0.5342430 0.3892448 0.2950505 0.2236504 0.1629498
```

La función `allEffects` en el modelo 1.3 muestra los efectos de la variable independiente "rank" en la variable dependiente "admit". Los valores en la columna "rank" representan los diferentes niveles de "rank". La columna siguiente, "1", representa el efecto de tener un "rank" de 1 en la probabilidad de ser admitido. Los resultados indican que a medida que el nivel de "rank" aumenta, la probabilidad de ser admitido disminuye. Por ejemplo, si un estudiante tiene un "rank" de 1, la probabilidad de ser admitido es del 53.4%, mientras que si el "rank" es de 4, la probabilidad de ser admitido es solo del 16.3%.

Resumen del capítulo

El análisis previo a la regresión nos permitió comprender mejor la distribución de nuestras variables. Asimismo, aquí es importante conocer que la regresión binomial suele tener problemas de ajuste cuando alguna de las variables predictores es continua, de ahí que la recomendación es usar como alternativa un modelo de regresión de Poisson con varianzas robustas. Por último, siempre es importante recordar la importancia de complementar el análisis de regresión con un buen análisis post-regresión. Este debe considerar como mínimo un análisis del ajuste del modelo, el uso de la prueba de Wald, la estimación *z*, el gráfico de efectos, entre otras técnicas para comprobar los supuestos del modelo.

CAPÍTULO 11

REGRESIÓN DE COX

La regresión de Cox se emplea para investigar los efectos de diversas variables sobre el tiempo que tarda en producirse un evento específico. Como uno de los desenlaces más utilizados en la regresión de Cox es el evento muerte, este suele conocerse como el análisis de supervivencia. En este capítulo realizaremos una regresión de Cox, para lo cual primero aprenderemos como configurar nuestros datos para un análisis de supervivencia y utilizaremos diferentes métodos para realizar la regresión de Cox. Por último, se culminará con el análisis post-regresión.

Objetivos de aprendizaje

Después de finalizar este capítulo deberías ser capaz de:

- Conocer los conceptos básicos de análisis de supervivencia.
- Realizar gráficos de Kaplan-Meier para visualizar las curvas de regresión de Cox.
- Aplicar regresiones simples y múltiples de Cox con los métodos forward y stepwise.
- Realizar un análisis post-regresión y estimar la proporcionalidad de los Hazards.
- Graficar e interpretar los resultados.

Paquetes

```
# Instalación

#install.packages("dplyr")
#install.packages("corrplot")
#install.packages("VIM")
#install.packages("ggplot2")
#install.packages("ggfortify")
```

```
#install.packages("AICcmodavg")
#install.packages("survival")
#install.packages("survminer")
#install.packages("rms")

# Llamamos a los paquetes
library(dplyr) #manipulación de datos
library(corrplot) #correlación
library (VIM) #visualización de valores perdidos
library(ggplot2) #gráficos
library(ggfortify)#Herramientas de trazado unificado
library(AICcmodavg) # Comparación de modelos
library(survival)#análisis de supervivencia
library(survminer) #dibuja curvas de supervivencia
library(rms)#estrategias modelado de regresión
```

Estudio de caso

Los factores pronósticos proporcionan información relevante a los pacientes con cáncer y al médico para tomar decisiones adecuadas. Asimismo, pueden ser valiosos para estratificar a los pacientes en los ensayos clínicos e interpretar los datos generados por las pruebas.

Este estudio busca determinar si la información descriptiva de un cuestionario completado por pacientes con cáncer de pulmón del North Central Cancer Treatment Group podría proporcionar información pronóstica que fuera independiente de la ya obtenida por el médico del paciente. Las puntuaciones de rendimiento permiten evaluar qué tan bien el paciente puede realizar las actividades diarias habituales.

La base de datos del estudio se encuentra incorporada en el paquete `survival` y contiene 10 variables de estudio:

Variable	Descripción	Tipo
inst	Código de la institución	Cuantitativa
time	tiempo de supervivencia en días	Cuantitativa
status	Estado de censura (1 = censurado, 2 = muerto)	Cuantitativa
age	Edad en años	Cuantitativa
sex	Sexo masculino = 1 , femenino = 2	Cuantitativa
ph.ecog	Puntuación de desempeño ECOG según la calificación del médico (0 = asintomático, 1= sintomático pero completamente ambulatorio, 2= en cama <50% del día, 3= en cama>50% del día pero no postrado en cama, 4= postrado en cama)	Cuantitativa
ph.karno	Puntuación de desempeño de Karnofsky (malo = 0 - bueno = 100) calificado por el médico	Cuantitativa
pat.karno	Puntuación de desempeño de Karnofsky según la calificación del paciente	Cuantitativa
meal.cal	Calorías consumidas en las comidas	Cuantitativa
wt.loss	Pérdida de peso en los últimos seis meses (libras)	Cuantitativa

Para visualizar con mayor detalle los datos, convertimos en un *data frame* la base de datos "lung" del paquete `survival`.

```
lung <- as.data.frame(lung)
```

Análisis exploratorio

Veamos las primeras 10 filas de la base de datos "lung".

```
head(lung, n=10)

##    inst time status age sex ph.ecog ph.karno pat.karno meal.
```

```
cal wt.loss
## 1     3  306      2  74   1       1       90       100
1175      NA
## 2     3  455      2  68   1       0       90        90
1225      15
## 3     3 1010      1  56   1       0       90        90
NA       15
## 4     5  210      2  57   1       1       90        60
1150      11
## 5     1  883      2  60   1       0      100        90
NA        0
## 6    12 1022      1  74   1       1       50        80
513       0
## 7     7  310      2  68   2       2       70        60
384       10
## 8    11  361      2  71   2       2       60        80
538       1
## 9     1  218      2  53   1       1       70        80
825       16
## 10    7  166      2  61   1       2       70        70
271       34
```

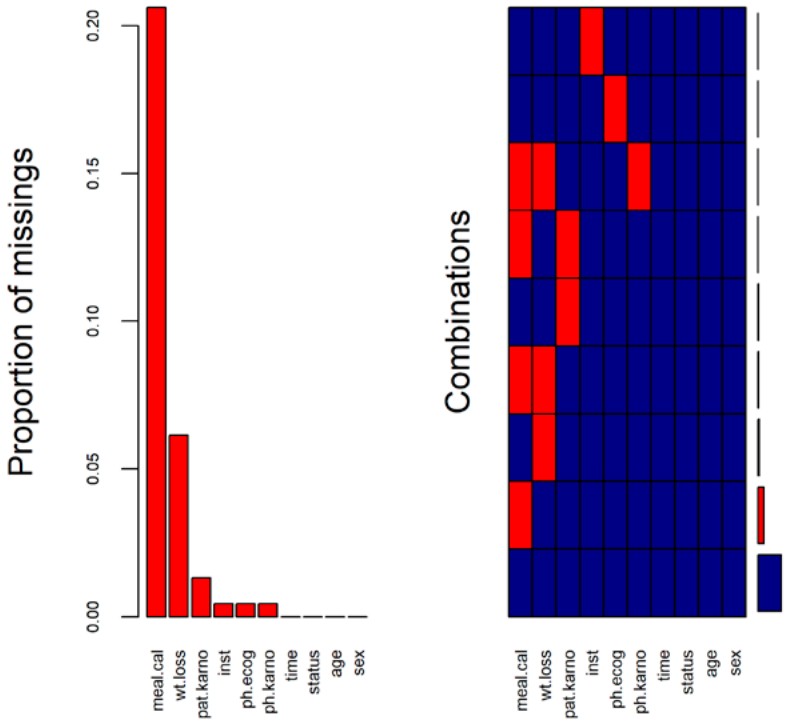
Notamos que todas las variables son numéricas y que hay valores faltantes en la base de datos. Vamos a comprobar esto último.

Valores perdidos

```
table(complete.cases(lung)) # Explorar

##
## FALSE  TRUE
##    61   167
```

En la base de datos hay 61 valores perdidos. A través del siguiente gráfico podremos conocer en qué variables hay más valores perdidos.

```
aggr_plot <- aggr(lung,
                  col=c('navyblue','red'),
                  numbers=TRUE,
                  sortVars=TRUE,
                  labels=names(data),
                  cex.axis=.6,
                  gap=4)

## Warning in plot.aggr(res, ...): not enough horizontal space
to display
## frequencies
```



```
##
##  Variables sorted by number of missings:
##   Variable       Count
##   meal.cal 0.206140351
##    wt.loss 0.061403509
##  pat.karno 0.013157895
##       inst 0.004385965
##    ph.ecog 0.004385965
##   ph.karno 0.004385965
##       time 0.000000000
##     status 0.000000000
##        age 0.000000000
##        sex 0.000000000
```

Como se puede observar, las variables "meal.cal" y "wt.loss" tienen más valores perdidos, seguidas de las variables "pat.karno", "inst", "ph.ecog" y "ph.karno". Nos quedaremos solo con los casos completos.

```
lung <-lung[complete.cases(lung), ] #Quedarnos solo con casos
completos

table(complete.cases(lung)) #comprobamos que ya no figura FALSE

##
## TRUE
##  167
```

Estructura de los datos

```
str(lung)

## 'data.frame':    167 obs. of  10 variables:
##  $ inst     : num  3 5 12 7 11 1 7 6 12 22 ...
```

```
##  $ time     : num  455 210 1022 310 361 ...
##  $ status   : num  2 2 1 2 2 2 2 2 2 ...
##  $ age      : num  68 57 74 68 71 53 61 57 57 70 ...
##  $ sex      : num  1 1 1 2 2 1 1 1 1 1 ...
##  $ ph.ecog  : num  0 1 1 2 2 1 2 1 1 1 ...
##  $ ph.karno : num  90 90 50 70 60 70 70 80 80 90 ...
##  $ pat.karno: num  90 60 80 60 80 80 70 80 70 100 ...
##  $ meal.cal : num  1225 1150 513 384 538 ...
##  $ wt.loss  : num  15 11 0 10 1 16 34 27 60 -5 ...
```

Efectivamente, todas las variables son tratadas como numéricas continuas. Así que vamos a convertir a las variables “sex”, “status” y “ph.ecog” en factores.

```
lung$sex <-as.factor(lung$sex)

str(lung) #comprobamos

## 'data.frame':    167 obs. of  10 variables:
##  $ inst     : num  3 5 12 7 11 1 7 6 12 22 ...
##  $ time     : num  455 210 1022 310 361 ...
##  $ status   : num  2 2 1 2 2 2 2 2 2 2 ...
##  $ age      : num  68 57 74 68 71 53 61 57 57 70 ...
##  $ sex      : Factor w/ 2 levels "1","2": 1 1 1 2 2 1 1 1 1 1
...
##  $ ph.ecog  : num  0 1 1 2 2 1 2 1 1 1 ...
##  $ ph.karno : num  90 90 50 70 60 70 70 80 80 90 ...
##  $ pat.karno: num  90 60 80 60 80 80 70 80 70 100 ...
##  $ meal.cal : num  1225 1150 513 384 538 ...
##  $ wt.loss  : num  15 11 0 10 1 16 34 27 60 -5 ...
```

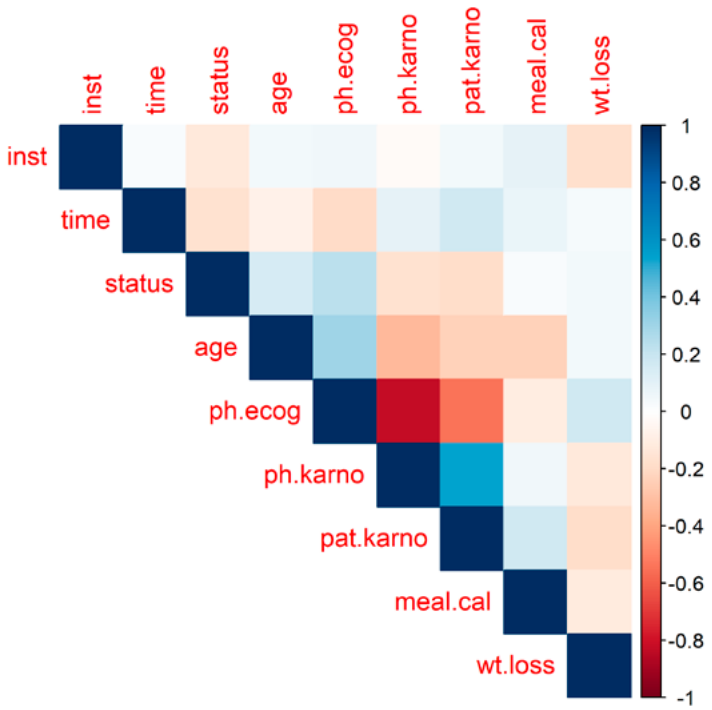
Descripción de los datos

```
summary(lung)

##       inst             time            status           age
sex
##  Min.   : 1.00   Min.   :   5.0   Min.   :1.000   Min.
:39.00   1:103
##  1st Qu.: 3.00   1st Qu.: 174.5   1st Qu.:1.000   1st
Qu.:57.00   2: 64
##  Median :11.00   Median : 268.0   Median :2.000   Median
:64.00
##  Mean   :10.71   Mean   : 309.9   Mean   :1.719   Mean
:62.57
##  3rd Qu.:15.00   3rd Qu.: 419.5   3rd Qu.:2.000   3rd
Qu.:70.00
##  Max.   :32.00   Max.   :1022.0   Max.   :2.000   Max.
:82.00
##     ph.ecog          ph.karno        pat.karno         meal.
cal
##  Min.   :0.0000   Min.   : 50.00   Min.   : 30.00   Min.   :
96.0
##  1st Qu.:0.0000   1st Qu.: 70.00   1st Qu.: 70.00   1st Qu.:
619.0
##  Median :1.0000   Median : 80.00   Median : 80.00   Median :
975.0
##  Mean   :0.9581   Mean   : 82.04   Mean   : 79.58   Mean   :
929.1
##  3rd Qu.:1.0000   3rd Qu.: 90.00   3rd Qu.: 90.00   3rd
Qu.:1162.5
##  Max.   :3.0000   Max.   :100.00   Max.   :100.00   Max.
:2600.0
##     wt.loss
##  Min.   :-24.000
##  1st Qu.:  0.000
```

```
##  Median :  7.000
##  Mean   :  9.719
##  3rd Qu.: 15.000
##  Max.   : 68.000
```

Análisis de correlación

```
correlation <- lung %>% select("inst", "time", "status", "age",
"ph.ecog",
"ph.karno", "pat.karno",
"meal.cal", "wt.loss")

correlations <- cor(correlation)

corrplot(correlations, type="upper", method="color")
```



A partir del gráfico anterior podemos decir que hay una fuerte correlación entre la variable "ph.karno" y "ph.ecog", y entre las variables "meal.cal" y "pat.karno".

Análisis de supervivencia

Se utiliza para la predicción, puntuación y análisis en los que se hace seguimiento de un sujeto durante un periodo de tiempo determinado desde el inicio hasta el término del evento. En este análisis, la variable de interés es conocida como "tiempo de vida", "tiempo de supervivencia" o "tiempo de falla". Las observaciones censuradas, que son aquellas que desaparecen del estudio o en las que no se produce el evento de interés durante el tiempo de observación, nos dan información parcial sobre la variable tiempo.

Preparamos la data

Es necesario especificar que *Y* es = variable de tiempo + variable de evento (muerte vs. censura), es por ello que creamos una nueva variable llamada "survival". La función `Surv` nos permite construir un objeto de supervivencia (variable de respuesta).

```
lung$survival <- with(lung, Surv( time, event = status))

lung$survival

##   [1]  455   210  1022+  310   361   218   166   170   567
613   707    61
##  [13]  301    81   371   520   574   118   390    12   473
26   107    53
##  [25]  814   965+   93   731   460   153   433   583    95
303   519   643
##  [37]  765    53   246   689     5   687   345   444   223
60   163    65
##  [49]  821+  428   230   840+  305    11   226   426   705
363   176   791
##  [61]   95   196+  167   806+  284   641   147   740+  163
```

```
655    88   245
##  [73]   30   477   559+  450   156   529+  429   351    15
181   283    13
##  [85]  212   524   288   363   199   550    54   558   207
92    60   551+
##  [97]  293   353   267   511+  457   337   201   404+  222
62   458+  353
## [109]  163    31   229   156   291   179   376+  384+  268
292+  142   413+
## [121]  266+  320   181   285   301+  348   197   382+  303+
296+  180   145
## [133]  269+  300+  284+  292+  332+  285   259+  110   286
270   225+  269
## [145]  225+  243+  276+  135    79    59   240+  202+  235+
239   252+  221+
## [157]  185+  222+  183   211+  175+  197+  203+  191+  105+
174+  177+
```

Curva de Kaplan - Meier (KM)

Grafica la probabilidad de supervivencia acumulada a lo largo del tiempo. Este procedimiento se realiza para observar la dinámica de supervivencia. Cuando no ocurre ningún evento, la curva es horizontal y cae verticalmente ante la ocurrencia de un evento. Utilizamos la función `survfit` del paquete `survival` para estimar la curva de KM; además, optamos por el intervalo de confianza "log-log", que es el más utilizado por producir intervalos que se comportan bien, es decir, se ajusta adecuadamente a los datos y brinda una estimación precisa de la probabilidad de supervivencia. No se considera ninguna agrupación, por lo que especificamos una intersección (~ 1).

```
KM.general <- survfit(survival ~ 1, data = lung, conf.type = "log-
log")
KM.general
```

```
## Call: survfit(formula = survival ~ 1, data = lung, conf.type =
“log-log”)
##
##        n events median 0.95LCL 0.95UCL
## [1,] 167    120    310     284     363
```

Vemos que la función `survfit` nos brinda un resumen de las curvas de supervivencia. Nos muestra el número de observaciones, el número de eventos, la mediana de supervivencia y los intervalos de confianza para la mediana.

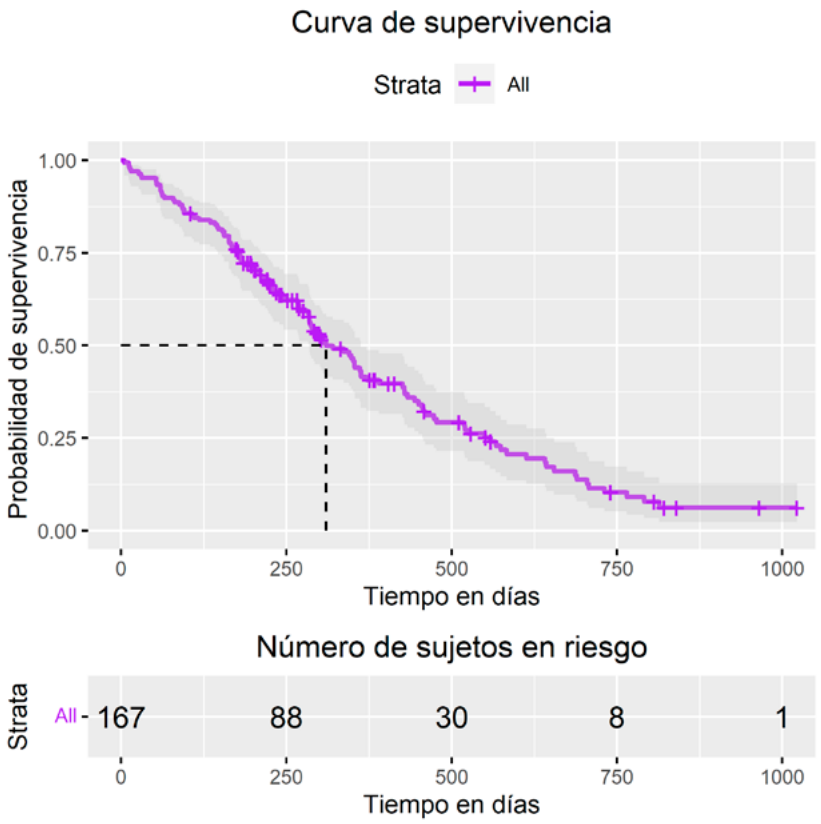
Utilizamos la función `ggsurvplot` del paquete `survminer` para graficar las curvas de supervivencia.

```
# creamos un objeto para centrar el título
centr <- theme(plot.title = element_text(hjust = 0.5))

#graficamos
ggsurvplot(KM.general,
            xlab=”Tiempo en días”,
            ylab=”Probabilidad de supervivencia “,
          title= “Curva de supervivencia”,
            risk.table=TRUE,
            conf.int=TRUE, pval = TRUE,
            surv.median.line=”hv”, ggtheme = centr,  risk.table.
title=”Número de sujetos en riesgo”, palette=”purple”)

## Warning in .pvalue(fit, data = data, method = method, pval =
pval, pval.coord = pval.coord, : There are no survival curves to
be compared.
##  This is a null model.
```

Utilizamos la función `ggsurvplot()` para crear un gráfico de la curva de supervivencia. Seguidamente, se especifican los títulos de los ejes x e y del gráfico, el título del gráfico y con `TRUE` se decide incluir una tabla de riesgo, intervalos de confianza y valores *p*. Además, en `surv.median.line` se define el tipo de línea que se utilizará

para trazar la línea mediana de la curva de supervivencia, en `ggtheme` el tema del gráfico y con `palette` la paleta de colores. De este modo, visualizamos la curva de supervivencia con información detallada sobre la probabilidad de supervivencia y el riesgo de eventos en diferentes puntos en el tiempo.

La advertencia indica que la función utilizada para calcular los valores *p*, no pudo encontrar múltiples grupos o categorías en los datos de supervivencia, lo cual no es un problema porque no se está comparando la supervivencia entre grupos debido a que solo hay un grupo conocido como "modelo nulo" o "modelo de referencia".

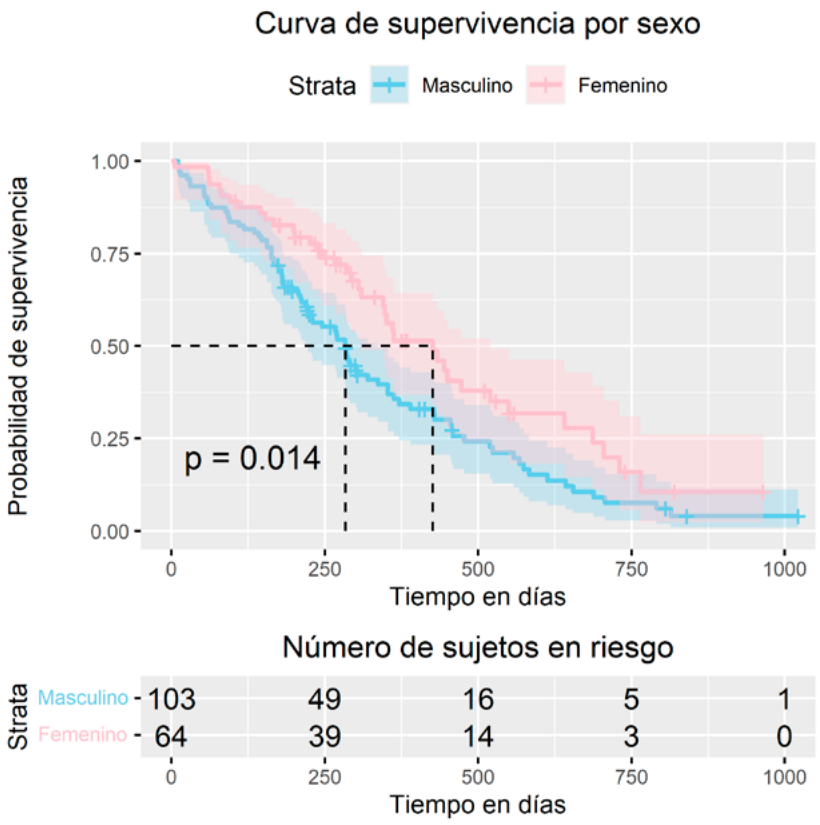


El gráfico anterior nos muestra el comportamiento de la población de estudio y además dibuja la línea horizontal y vertical de la mediana. Comparemos ahora los datos por género.

```
KM.SEXO <- survfit(survival ~ sex, data = lung, conf.type = "log-log")
KM.SEXO

## Call: survfit(formula = survival ~ sex, data = lung, conf.type = "log-log")
##
##         n events median 0.95LCL 0.95UCL
## sex=1 103     82    284     222     337
## sex=2  64     38    426     310     520
```

Graficamos:

```
ggsurvplot ((KM.SEXO),
            xlab="Tiempo en días",
            ylab="Probabilidad de supervivencia",
            title= "Curva de supervivencia por sexo",
            pval = TRUE,
            risk.table=TRUE,
            conf.int=TRUE,
            surv.median.line="hv",
            risk.table.title="Número de sujetos en riesgo",
legend.labs=c("Masculino", "Femenino"), palette=c("skyblue",
"pink"), ggtheme=centr)
```



El gráfico muestra la curva de supervivencia por sexo en un conjunto de datos de pacientes con cáncer de pulmón. En el eje *x* se muestra el tiempo en días y en el eje *y* se encuentra la probabilidad de supervivencia. Asimismo, hay dos líneas en el gráfico, una para el sexo masculino (línea azul) y otra para el sexo femenino (línea rosa). Observando ambas líneas, constatamos que la probabilidad de supervivencia para los hombres disminuye más rápidamente en el tiempo que para las mujeres, lo que sugiere que las mujeres tienen una mejor tasa de supervivencia ante el cáncer.

La tabla de riesgo que se encuentra debajo del gráfico muestra el número de sujetos en riesgo en diferentes puntos en el tiempo para cada sexo. Por otro lado, el valor *p* es menor que 0.05, lo que sugiere que sí hay una diferencia significativa en la supervivencia entre hombres y mujeres con cáncer de pulmón.

Regresión de Cox

En el análisis de supervivencia, el modelo de regresión más utilizado es la regresión de Cox, dada su flexibilidad y dado que, a la hora de interpretar los coeficientes, es algo más simple que el resto de los modelos propuestos, como la regresión logística binomial o de Poisson. Debido a que la interpretación de los coeficientes en la regresión de Cox suele ser más simple y directa que en los otros modelos.

Este modelo, denominado también modelo de *hazards* o de riesgos proporcionales, se utiliza para detectar qué relaciones existen entre el riesgo que se produce en un determinado individuo en el estudio y algunas variables independientes y/o explicativas; de esta forma, nos permite evaluar, entre un conjunto de variables, cuáles tienen relación o influencia sobre la función de riesgo y la función de supervivencia, ya que ambas funciones están conectadas.

Regresión simple

```
reg_simple <-coxph(survival~sex, data=lung )
reg_simple

## Call:
## coxph(formula = survival ~ sex, data = lung)
##
##         coef exp(coef) se(coef)      z      p
## sex2 -0.4792    0.6193   0.1966 -2.437 0.0148
##
## Likelihood ratio test=6.25  on 1 df, p=0.01244
## n= 167, number of events= 120
```

La variable "sexo" tiene un valor *p* de 0.01, lo cual indica una fuerte relación entre el sexo de los pacientes y la disminución del riesgo de muerte. Generalmente se considera que una relación es fuerte cuando el valor p es menor a 0.05 y el coeficiente de regresión es mayor a 0.3 o menor a -0.3. En este caso, el coeficiente de regresión para la variable "sexo" es -0.4792, lo cual indica que el riesgo de muerte para los pacientes de sexo masculino es 0.6193 veces menor que para los pacientes de sexo femenino.

Regresión múltiple

◊ *Método forward*

Modelos de primer orden

```
modelo1.1 <- coxph(survival ~ sex, data =  lung)
modelo1.1

## Call:
## coxph(formula = survival ~ sex, data = lung)
##
##         coef exp(coef) se(coef)      z      p
## sex2 -0.4792    0.6193   0.1966 -2.437 0.0148
##
## Likelihood ratio test=6.25  on 1 df, p=0.01244
## n= 167, number of events= 120

modelo1.2 <- coxph(survival ~ age, data =  lung)
modelo1.2

## Call:
## coxph(formula = survival ~ age, data = lung)
##
##        coef exp(coef) se(coef)     z      p
## age 0.01989   1.02009  0.01075 1.851 0.0642
##
## Likelihood ratio test=3.52  on 1 df, p=0.0605
## n= 167, number of events= 120

modelo1.3 <- coxph(survival ~ ph.karno, data =  lung)
modelo1.3

## Call:
## coxph(formula = survival ~ ph.karno, data = lung)
##
```

```
##                coef exp(coef)  se(coef)      z      p
## ph.karno -0.012201  0.987873  0.006701 -1.821 0.0686
##
## Likelihood ratio test=3.21  on 1 df, p=0.07318
## n= 167, number of events= 120

modelo1.4 <- coxph(survival ~ wt.loss, data =  lung)
modelo1.4

## Call:
## coxph(formula = survival ~ wt.loss, data = lung)
##
##               coef exp(coef)  se(coef)     z     p
## wt.loss 0.0001532 1.0001532 0.0066939 0.023 0.982
##
## Likelihood ratio test=0  on 1 df, p=0.9817
## n= 167, number of events= 120
```

Mientras mayor sea el *likelihood ratio test* o la prueba de razón de verosimilitud, mayor será el ajuste del modelo. Se observa, en este caso, que "sex" es la variable que entra primero en el modelo.

Modelos de segundo orden

```
modelo2.1 <- coxph(survival ~ sex + age, data =  lung)
modelo2.1

## Call:
## coxph(formula = survival ~ sex + age, data = lung)
##
##          coef exp(coef) se(coef)      z      p
## sex2 -0.44669   0.63974  0.19754 -2.261 0.0237
## age   0.01737   1.01752  0.01084  1.603 0.1089
##
```

```
## Likelihood ratio test=8.88  on 2 df, p=0.01179
## n= 167, number of events= 120

modelo2.2 <- coxph(survival ~ sex + ph.karno, data =  lung)
modelo2.2

## Call:
## coxph(formula = survival ~ sex + ph.karno, data = lung)
##
##               coef exp(coef)  se(coef)      z      p
## sex2     -0.472888  0.623200  0.196696 -2.404 0.0162
## ph.karno -0.011607  0.988460  0.006559 -1.770 0.0768
##
## Likelihood ratio test=9.28  on 2 df, p=0.009642
## n= 167, number of events= 120

modelo2.3 <- coxph(survival ~ sex + wt.loss, data =  lung)
modelo2.3

## Call:
## coxph(formula = survival ~ sex + wt.loss, data = lung)
##
##              coef exp(coef)  se(coef)      z      p
## sex2    -0.481179  0.618054  0.197013 -2.442 0.0146
## wt.loss -0.001010  0.998991  0.006682 -0.151 0.8799
##
## Likelihood ratio test=6.27  on 2 df, p=0.04351
## n= 167, number of events= 120
```

La prueba de razón de verosimilitud de “sex” y “ph.karno” es 9.28, es la mayor entre todos los modelos. En consecuencia, “pk.karno” es la segunda variable que ingresa al modelo.

Modelos de tercer orden

```
modelo3.1 <- coxph(survival ~ sex + ph.karno + age , data =
lung)
modelo3.1

## Call:
## coxph(formula = survival ~ sex + ph.karno + age, data = lung)
##
##                coef exp(coef)  se(coef)      z      p
## sex2     -0.453953  0.635113  0.197440 -2.299 0.0215
## ph.karno -0.008842  0.991197  0.007039 -1.256 0.2091
## age       0.012260  1.012336  0.011534  1.063 0.2878
##
## Likelihood ratio test=10.43  on 3 df, p=0.01523
## n= 167, number of events= 120

modelo3.2 <- coxph(survival ~ sex + ph.karno + wt.loss , data =
lung)
modelo3.2

## Call:
## coxph(formula = survival ~ sex + ph.karno + wt.loss, data =
lung)
##
##                coef exp(coef)  se(coef)      z      p
## sex2     -0.482265  0.617384  0.197292 -2.444 0.0145
## ph.karno -0.012483  0.987594  0.006685 -1.867 0.0619
## wt.loss  -0.004010  0.995998  0.006980 -0.574 0.5656
##
## Likelihood ratio test=9.62  on 3 df, p=0.02208
## n= 167, number of events= 120
```

Las variables “sex”, “ph.karno” y “age” son las que entran en el modelo.

Modelos de cuarto orden

```
modelo4.1 <- coxph(survival ~ sex + ph.karno + age + wt.loss,
data =  lung)
modelo4.1

## Call:
## coxph(formula = survival ~ sex + ph.karno + age + wt.loss,
data = lung)
##
##               coef exp(coef)  se(coef)      z      p
## sex2     -0.462931  0.629436  0.198194 -2.336 0.0195
## ph.karno -0.009711  0.990336  0.007206 -1.348 0.1778
## age       0.011816  1.011886  0.011549  1.023 0.3063
## wt.loss  -0.003525  0.996481  0.007065 -0.499 0.6178
##
## Likelihood ratio test=10.68  on 4 df, p=0.03034
## n= 167, number of events= 120
```

Las variables “sex”, “ph.karno” y “age” son las que entran en el modelo.

Cálculo del AIC del mejor modelo de cada orden

```
CalculoAIC <- list(modelo1.1 , modelo2.2, modelo3.1, modelo4.1)

#Especificar los nombres del modelo
NamesAIC <- c('Modelo1', 'Modelo2', 'Modelo3', 'Modelo4')

#Calcular el AIC de cada modelo
aictab(cand.set = CalculoAIC, modnames = NamesAIC)

##
## Model selection based on AICc:
##
```

```
##         K    AICc Delta_AICc AICcWt Cum.Wt      LL
## Modelo2 2 1011.02       0.00   0.40   0.40 -503.48
## Modelo3 3 1011.95       0.93   0.25   0.65 -502.90
## Modelo1 1 1012.01       0.99   0.25   0.90 -504.99
## Modelo4 4 1013.80       2.77   0.10   1.00 -502.77
```

Vemos que el Modelo 2 es el que tiene el AIC menor, en consecuencia, es el mejor modelo. Es decir, de acuerdo al método forward, las variables "sex" y "ph.karno" son las variables que mejor se relacionan con la variable "survival".

◊ *Método stepwise*

Como este método se inicia con una regresión con todas las variables, utilizaremos el modelo 4.1.

```
modelo4.1 <- coxph(survival ~ sex + ph.karno + age + wt.loss,
data =  lung)
stats::step(modelo4.1)

## Start:  AIC=1013.55
## survival ~ sex + ph.karno + age + wt.loss
##
##            Df    AIC
## - wt.loss   1 1011.8
## - age       1 1012.6
## - ph.karno  1 1013.3
## <none>        1013.5
## - sex       1 1017.3
##
## Step:  AIC=1011.8
## survival ~ sex + ph.karno + age
##
##            Df    AIC
## - age       1 1011.0
```

```
## - ph.karno  1 1011.4
## <none>        1011.8
## - sex       1 1015.3
##
## Step:  AIC=1010.95
## survival ~ sex + ph.karno
##
##            Df  AIC
## <none>        1011
## - ph.karno  1 1012
## - sex       1 1015

## Call:
## coxph(formula = survival ~ sex + ph.karno, data = lung)
##
##                coef exp(coef)  se(coef)      z      p
## sex2     -0.472888  0.623200  0.196696 -2.404 0.0162
## ph.karno -0.011607  0.988460  0.006559 -1.770 0.0768
##
## Likelihood ratio test=9.28  on 2 df, p=0.009642
## n= 167, number of events= 120
```

Este método confirma que las variables "sex" y "ph.karno" conforman el mejor modelo. Es decir, el modelo2.2 es el mejor.

Análisis post regresión

Prueba de la suposición de riesgos proporcionales

Esta suposición se comprueba a través de las pruebas estadísticas y gráficas de los residuos de Schoenfeld escalados. Los residuos de Schoenfeld son independientes del tiempo, por lo tanto, si un gráfico muestra un patrón no aleatorio contra el tiempo es evidencia de violación de esta suposición. Se rechaza la prueba si hay una relación significativa entre los residuos y el tiempo.

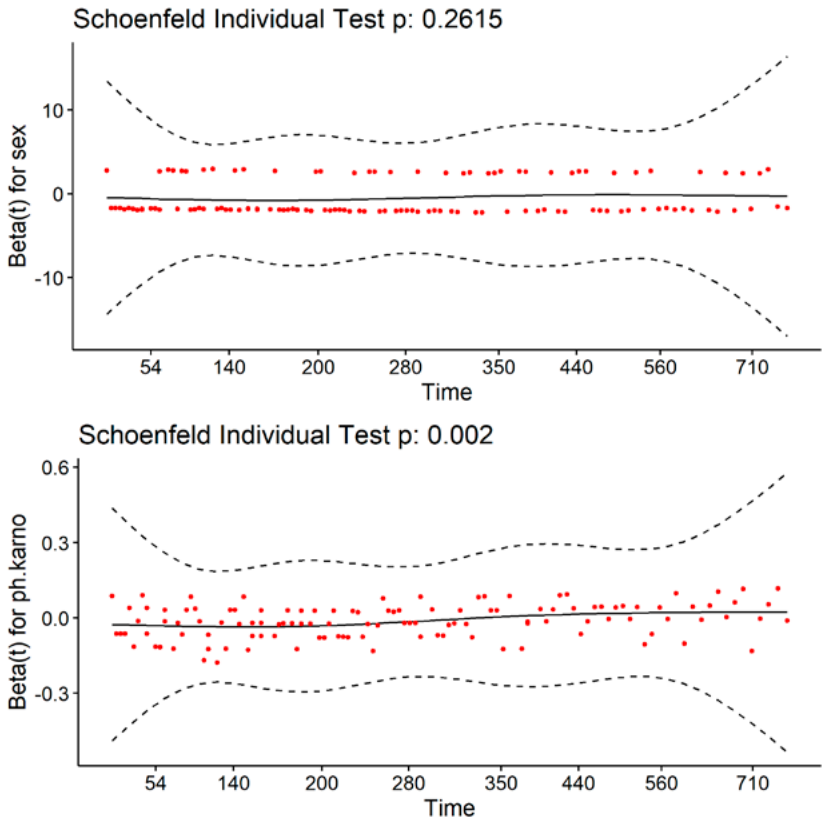
```
cox.zph(modelo2.2, transform="km", global=TRUE)

##           chisq df     p
## sex        1.26  1 0.262
## ph.karno   9.56  1 0.002
## GLOBAL    10.22  2 0.006
```

La función `cox.zph` se emplea en análisis de supervivencia para probar la proporcionalidad de riesgos en el modelo. El argumento "transform" indica la transformación que se aplicará a la variable de tiempo para obtener el estimador de riesgo proporcional de Cox, en este caso se utiliza la transformación de Kaplan-Meier (KM). El argumento `global` indica si se realizará una prueba global de proporcionalidad para todas las variables incluidas en el modelo o una prueba para cada variable individualmente.

Ten en cuenta que obtenemos una prueba global y una prueba separada para cada predictor. En general, si los valores *p* son menores que 0.05, se puede concluir que hay una violación significativa de la proporcionalidad de riesgos para el predictor correspondiente. Si hay una violación significativa, esto puede afectar la interpretación y las conclusiones del modelo.

Podemos trazar los residuos de Schoenfeld escalados directamente con `ggcoxzph` del paquete `survminer`.

```
ggcoxzph(cox.zph(modelo2.2))
```

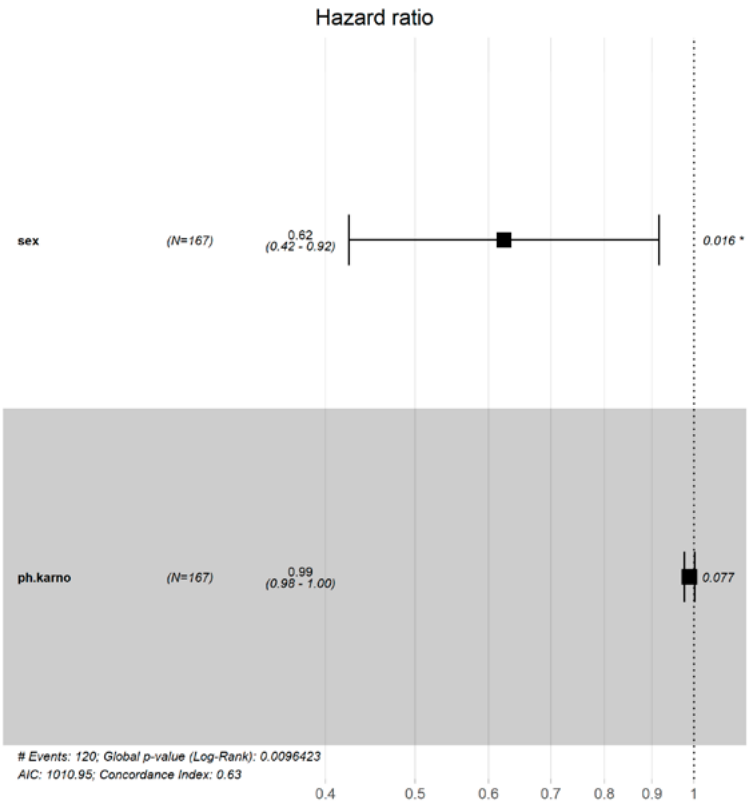


El eje horizontal representa el tiempo o los tiempos de seguimiento, mientras que el eje vertical muestra la función de riesgo proporcional para cada variable independiente en el modelo. La línea sólida representa la función de riesgo proporcional esperada bajo la hipótesis nula de que la proporcionalidad de riesgos se mantiene para todas las variables independientes en el modelo. La línea punteada en el gráfico representa el intervalo de confianza del 95% para el coeficiente de la variable.

Si la línea sólida no cruza la línea punteada en ningún momento, podemos asumir que no hay una violación significativa de la proporcionalidad de riesgos. Si los puntos en el gráfico están cerca de la línea sólida, esto indica que la proporcionalidad de los riesgos se mantiene y no hay violación importante de la hipótesis nula.

Se puede realizar otro gráfico de riesgos proporcionales con la función `ggforest`. El modelo es presentado en dos estructuras.

```
ggforest(modelo2.2, data = lung)
```



El comando anterior se utiliza para crear un gráfico forestal (*forest plot*) de los coeficientes estimados y sus intervalos de confianza para una regresión de Cox. Este tipo de gráfico es una forma común de visualizar los resultados de la regresión de Cox y es útil para comparar la magnitud y dirección de los efectos de diferentes variables predictoras en el tiempo de supervivencia.

Cada línea representa una variable del modelo de regresión de Cox y su impacto en el tiempo de supervivencia. La línea vertical en el centro del gráfico representa el valor nulo, lo que significa que si el intervalo de confianza de la línea de la variable cruza esta línea, entonces el efecto de la variable no es estadísticamente significativo.

El punto negro en cada línea representa el coeficiente estimado para la variable y la barra negra que se extiende desde el punto representa el intervalo de confianza del coeficiente.

El valor *p* asociado con cada variable se muestra al lado de la línea correspondiente. Si el valor *p* es menor que el nivel de significación predeterminado (generalmente 0.05), se considera que el efecto de la variable en el tiempo de supervivencia es estadísticamente significativo.

Prueba de hipótesis para evaluar la proporcionalidad de los *hazards*

El objetivo de esta prueba es verificar si la relación entre cada variable predictora y el tiempo de supervivencia es constante en el tiempo, es decir, si los riesgos proporcionales se mantienen constantes a lo largo del tiempo. La prueba de hipótesis se realiza para cada variable predictora individualmente y también para el modelo en su conjunto. Los resultados se presentan en forma de estadístico de prueba (Chi-cuadrado), grados de libertad (df) y valor *p*.

Veamos estadísticamente:

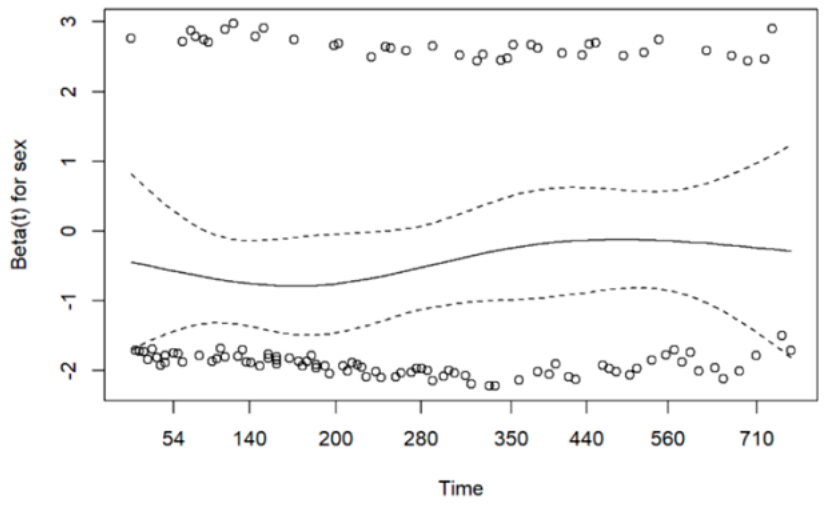
```
(rev.modelo2.2 <- cox.zph(modelo2.2))

##           chisq df     p
## sex        1.26  1 0.262
## ph.karno   9.56  1 0.002
## GLOBAL    10.22  2 0.006
```

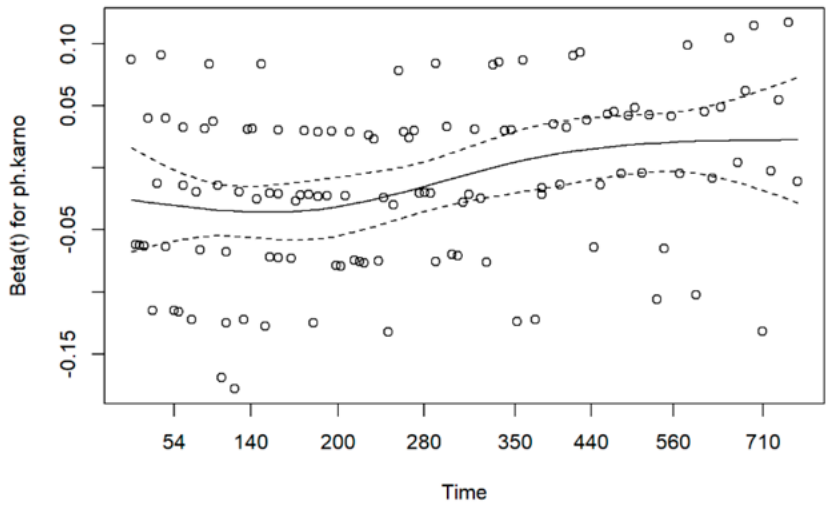
La salida muestra que para la variable "sex", el estadístico de prueba es 1.26, los grados de libertad son 1 y el valor *p* es 0.262. Esto indica que no hay suficiente evidencia para rechazar la hipótesis nula de que los riesgos proporcionales se mantienen constantes para esta variable en particular. Para la variable "ph.karno", el estadístico de prueba es 9.56, los grados de libertad son 1 y el valor *p* es 0.002. Esto indica que hay suficiente evidencia para rechazar la hipótesis nula de que los riesgos proporcionales se mantienen constantes para esta variable.

La última fila de la salida muestra la prueba de hipótesis para el modelo en su conjunto. El estadístico de prueba es 10.22, los grados de libertad son 2 y el valor *p* es 0.006. Por lo tanto, los resultados son estadísticamente significativos a un nivel de 0.05

Veamos gráficamente:

```
plot(rev.modelo2.2)
```





Los dos anteriores son gráficos de residuos de Schoenfeld para cada variable. En general, un gráfico de residuos de Schoenfeld bien ajustado debe mostrar un patrón aleatorio en el tiempo, lo que indica que la variable cumple con el supuesto de proporcionalidad de riesgos proporcionales. Si se observa un patrón no aleatorio en el gráfico, esto puede indicar que la variable no cumple con el supuesto de proporcionalidad de riesgos proporcionales y que se requiere una evaluación más detallada.

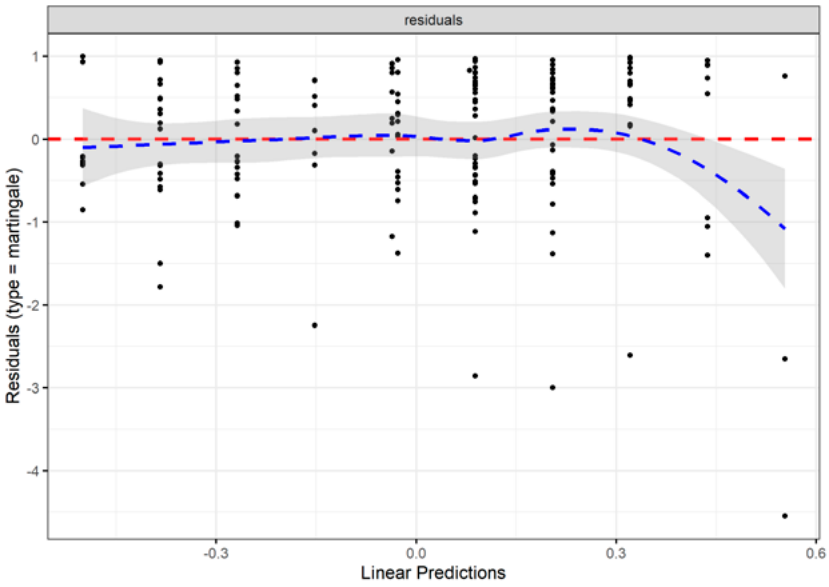
Prueba de observaciones influyentes

La observación o caso influyente es aquella que altera los coeficientes de regresión en una cantidad significativa cuando se incluye en el conjunto de datos para ajustar el modelo de regresión.

Si bien las observaciones influyentes no violan necesariamente ningún supuesto de regresión, pueden generar dudas sobre las conclusiones extraídas de su muestra.

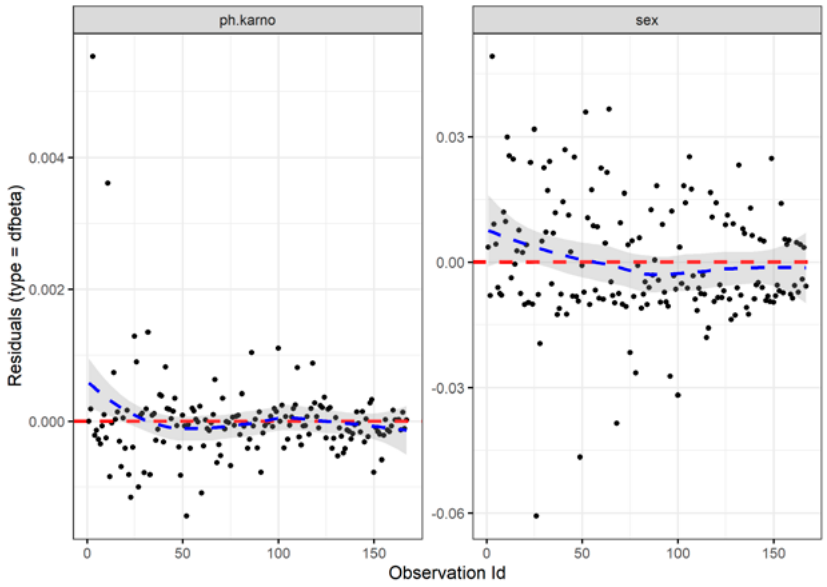
```
ggcoxdiagnostics(modelo2.2, type = , linear.predictions = TRUE)
```

El argumento `type` se utiliza para especificar el tipo de gráfico que se desea generar. El argumento `linear.predictions` se utiliza para especificar si se deben graficar las predicciones lineales del modelo en los gráficos. Si se establece en `TRUE`, se graficarán las predicciones lineales junto con los datos. Si se establece en `FALSE`, solo se graficarán los datos.



Si especificamos el argumento `type = "dfbeta"`, R traza los cambios estimados en los coeficientes de regresión al eliminar cada observación; asimismo, produce los cambios estimados en los coeficientes divididos por sus errores estándar. La opción `ggtheme = theme_bw()` se utiliza para establecer el tema del gráfico generado como blanco y negro, lo que hace que el gráfico sea más fácil de leer y de interpretar.

```
ggcoxdiagnostics(modelo2.2, type = "dfbeta",
                 linear.predictions = FALSE, ggtheme = theme_
bw())
```



Los puntos en el gráfico representan cada observación individual y la línea horizontal en cero indica el valor del coeficiente del modelo si se elimina esa observación. Los puntos que están alejados de cero indican que esa observación tiene una influencia importante en el coeficiente del modelo.

Prueba de colinealidad

Si el RR (riesgo relativo) es menor que 1 implica que la variable tiene un efecto protector y si es mayor que 1, que es factor de riesgo. Utilizamos el paquete `rms`.

```
vif(modelo2.2)

##      sex2 ph.karno
## 1.001124 1.001124
```

Se tiene una ligera colinealidad, es decir, las variables incluidas en el modelo presentan una correlación lineal fuerte entre ellas. En general, esto puede dificultar la interpretación precisa de los efectos de las variables individuales en el modelo y aumentar la incertidumbre en las estimaciones de los coeficientes de regresión. Sin

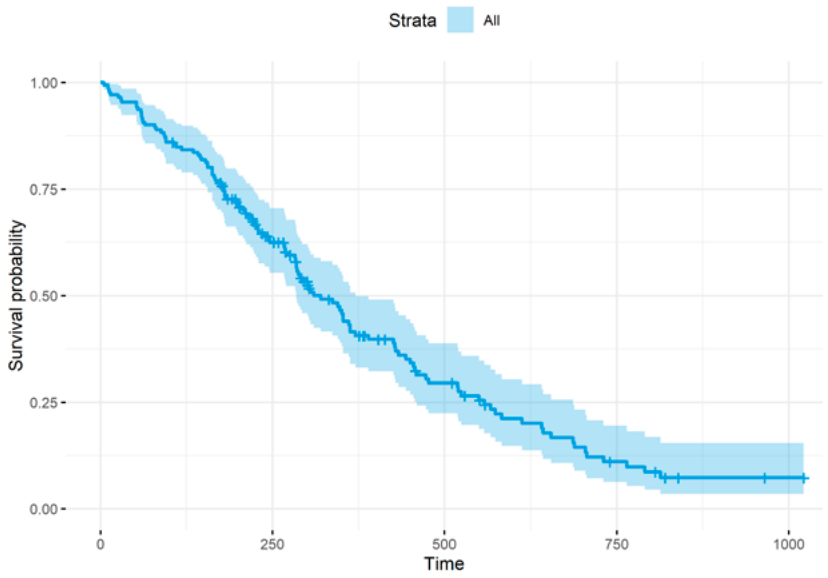
embargo, dado que los valores de VIF (factor de inflación de la varianza) son cercanos a 1, se considera que la colinealidad es leve y no afecta significativamente los resultados del modelo.

Distribución estimada de los tiempos de supervivencia.

Habiendo ajustado un modelo de Cox a los datos, es posible visualizar la proporción de supervivencia prevista en cualquier momento dado para un grupo de riesgo en particular. La función `survfit` estima la proporción de supervivencia, por defecto en los valores medios de las covariables.

```
ggsurvplot(survfit(modelo2.2, data = lung), color = "#2E9FDF",
           ggtheme = theme_minimal())
```

El comando crea un objeto survfit que contiene la información de la supervivencia. El primer argumento `modelo2.2` es el modelo de regresión de Cox que se ha ajustado previamente. Y el segundo argumento, `data = lung` indica la base de datos donde se encuentran los datos de la supervivencia. `Color = "#2E9FDF"` indica el color que se desea utilizar para el gráfico. En este caso, el color es un azul claro y `ggtheme = theme_minimal()` define el tema que se desea utilizar para el gráfico. En este caso, se utiliza el tema minimal.



El gráfico muestra la probabilidad de supervivencia en el eje vertical y el tiempo en el eje horizontal. La curva de supervivencia se representa mediante una línea que se desplaza hacia abajo a medida que el tiempo avanza.

Resumen del capítulo

En este capítulo resaltamos la importancia del análisis de supervivencia para la predicción de los eventos. Asimismo, realizamos un regresión simple con la variable sexo para conocer el grado de relación con la supervivencia de cáncer al pulmón, además de métodos múltiples que hemos presentado en capítulos anteriores. Para un análisis post regresión, las pruebas de *hazards* y de colinealidad, así como de residuos, son importantes.

En este libro, Antonio M. Quispe, médico epidemiólogo, docente, estadístico, investigador, consultor y comunicador científico vuelca toda su experiencia y habilidades cuantitativas para ofrecer una guía clara, precisa y práctica para realizar análisis estadísticos con R. El volumen constituye la segunda entrega de su trilogía de libros de texto del Club de Redacción de Artículos Científicos, intervención estructural diseñada para que todo estudiante e investigador joven pueda superar la inequidad de acceso a los recursos esenciales de la investigación. En estas páginas, los lectores podrán aprender a programar en R para manejar, visualizar y realizar un análisis exploratorio de datos, así como análisis bivariados y análisis de regresión multivariables, incluyendo los modelos generales de regresión lineal, logística, de Poisson, binomial y de Cox. De ahí que esta guía es altamente recomendable sobre todo para aquellos que deseen aventurarse en el fascinante mundo del análisis de datos y la investigación científica.

UTEC PRESS

9 786124 851544