INFERENCIA Y MODELOS ESTADÍSTICOS

Jacqueline Köhler C. y José Luis Jara V.

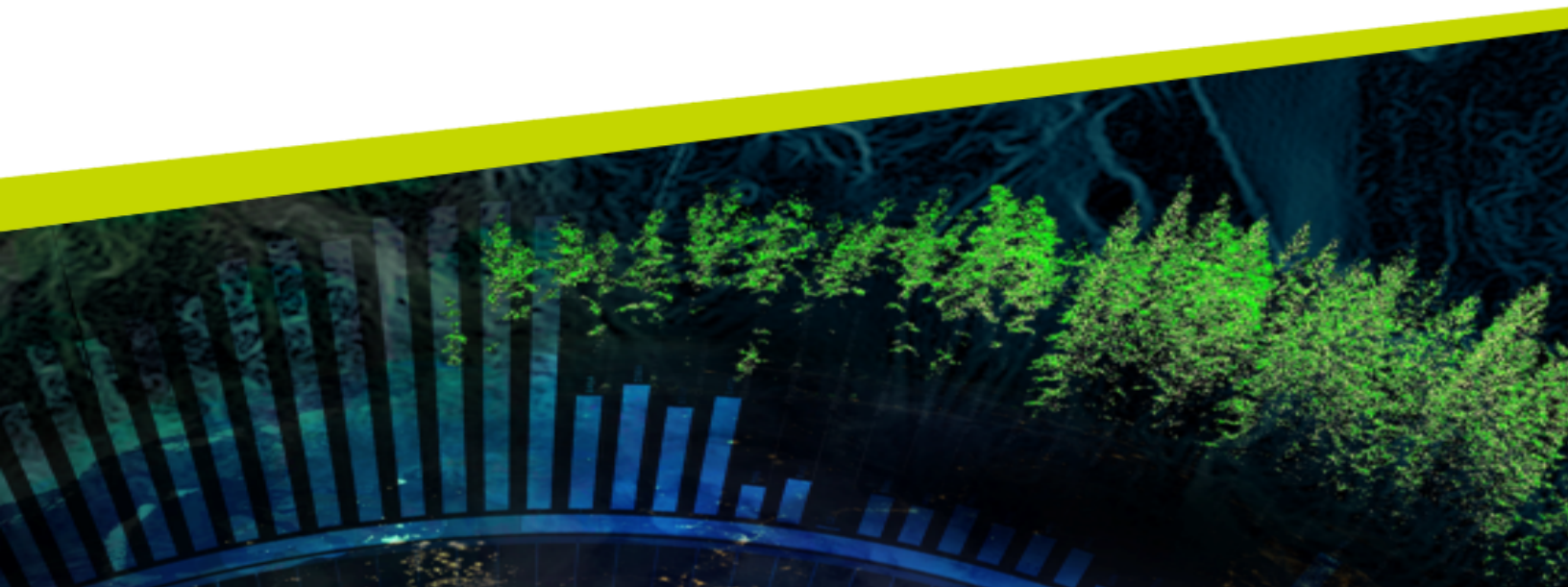


Tabla de contenido

Índice de tablas

Índice de figuras

Índice de scripts

CAPÍTULO 9. ANOVA DE UNA VÍA PARA MUESTRAS INDEPENDIENTES

En el capítulo 5 conocimos la prueba t de Student que permite, entre otras funciones, inferir acerca de la diferencia entre las medias de dos poblaciones a partir de dos muestras. Sin embargo, muchas veces necesitaremos realizar un procedimiento similar para $k \geq 3$ grupos. En el capítulo 8 nos enfrentamos a un escenario similar para cuando trabajamos con proporciones, para lo cual conocimos la prueba Q de Cochran. Aprendimos que esta última prueba es de tipo ómnibus: es decir, comprueba la igualdad de todos los grupos, pero que si encuentra diferencias no nos indica dónde están. En consecuencia, conocimos los procedimientos post-hoc para identificar entre qué grupos existen estas diferencias.

En el caso de las medias, cuando tenemos más de dos grupos también podemos usar una prueba ómnibus y algunos procedimientos post-hoc, para lo cual nos basaremos en las ideas presentadas por Lowry (1999, caps. 13-14); Glen (2021c); IBM (1989); Meier (2021, p. 4) y Berman (2000).

Intuitivamente, podríamos abordar este problema efectuando pruebas t independientes para cada pareja de grupos con un nivel de significación α. Por ejemplo, para tres muestras A, B y C con medias $\overline{x}_A$, $\overline{x}_B$ y $\overline{x}_C$, respectivamente, se tendrían tres pruebas t de Student para diferencia de medias:

1. $\overline{x}_A - \overline{x}_B$
2. $\overline{x}_A - \overline{x}_C$
3. $\overline{x}_B - \overline{x}_C$

Sin embargo, este enfoque presenta un grave inconveniente: por cada una de las pruebas t anteriores, se tiene una probabilidad α de cometer un error tipo I. Al efectuar cada una de las pruebas anteriores, la probabilidad total de que en alguna de ellas se cometa un error tipo I se acerca a $3 \cdot \alpha$, bastante superior al nivel de significación nominal establecido para la prueba[1]. El método de **análisis de varianza**, comúnmente conocido como **ANOVA** o AoV (del inglés Analysis of Variance), surge, en esencia, como un método para combatir este problema al comparar simultáneamente tres o más medias muestrales.

De manera similar, también existe el procedimiento ANOVA para muestras correlacionadas (que se aborda en el capítulo siguiente), semejante a la prueba t de Student con muestras pareadas. Los procedimientos ANOVA para muestras independientes y muestras correlacionadas corresponden al **análisis de varianza de una vía**, pues solo consideran una única variable independiente (de tipo categórica, un **factor**) cuyos niveles definen los grupos que se están comparando.

Existe además el **análisis de varianza de dos vías**, no abordado en el presente texto, el cual permite examinar simultáneamente los efectos de dos variables independientes e, incluso, determinar si ambas interactúan. De hecho, existen métodos para el análisis con más factores, que también están fuera del alcance de este curso.

Para explicar en detalle el procedimiento ANOVA de una vía para muestras independientes, consideremos el siguiente ejemplo: un ingeniero cuenta con tres algoritmos (A, B y C) para resolver un determinado problema (en iguales condiciones y para instancias de tamaño fijo, digamos con E elementos) y desea comparar su eficiencia. Para cada algoritmo, selecciona una muestra aleatoria independiente de instancias y registra el tiempo de ejecución (en milisegundos) para cada una de las instancias de la muestra correspondiente, obteniendo las siguientes observaciones:

- Algoritmo A: 23, 19, 25, 23, 20

[1] Aunque el cálculo exacto de esta probabilidad disjunta escapa a los alcances de este curso, es intuitivo ver que la probabilidad de no cometer un error de tipo I en cada prueba es $(1 - alpha)$. Así, la probabilidad de no cometer un error de tipo I en las tres comparaciones es $(1 - alpha)^3$. Luego, la probabilidad de cometer un error de tipo I para la hipótesis global (no hay diferencias entre los grupos) es $(1 - (1 - alpha)^3)$. Si, por ejemplo, nominalmente $\alpha = 0,05$, el nivel de significación para la hipótesis global sería aproximadamente 0.143

- Algoritmo B: 26, 24, 28, 23, 29
- Algoritmo C: 19, 24, 20, 21, 17

La pregunta detrás de ANOVA para este ejemplo es: ¿se diferencian los tiempos medios que requieren los algoritmos para resolver todas las posibles instancias del problema de tamaño E? De donde se desprende que:

H_0 : el tiempo de ejecución promedio para instancias de tamaño E es igual para los tres algoritmos.
H_A : el tiempo de ejecución promedio para instancias de tamaño E es diferente para al menos un algoritmo.

Notemos que, como en el caso de la prueba Q de Cochran, la hipótesis nula no es específica, sino que comprueba la igualdad de todas las medias, por lo que ANOVA es una prueba ómnibus.

9.1 CONDICIONES PARA USAR ANOVA DE UNA VÍA PARA MUESTRAS INDEPENDIENTES

Al igual que otras pruebas estudiadas en capítulos anteriores, el procedimiento ANOVA requiere que se cumplan algunas condiciones:

1. La escala con que se mide la variable dependiente tiene las propiedades de una escala de intervalos iguales.
2. Las k muestras son obtenidas de manera aleatoria e independiente desde la(s) población(es) de origen.
3. Se puede suponer razonablemente que la(s) población(es) de origen sigue(n) una distribución normal.
4. Las k muestras tienen varianzas aproximadamente iguales.

En nuestro ejemplo con los algoritmos, la primera condición se verifica, puesto que si para una instancia i un algoritmo tarda 20 [ms] mientras que otro algoritmo tarda 30 [ms], esa es la misma diferencia (10 milisegundos) que se presenta para una instancia j en que uno tarda 35 [ms] y el otro 45 [ms]. A su vez, el enunciado señala que el proceso seguido por el ingeniero garantiza el cumplimiento de la segunda condición.

La figura 9.1 (creada mediante el script 9.1, líneas 20–29) muestra gráficos Q-Q para cada muestra. Como se observan algunos valores que podrían ser atípicos y las muestras son pequeñas, es mejor que procedamos con cautela y usemos un nivel de significación $\alpha = 0,025$.

Una regla sencilla para comprobar la cuarta condición, llamada también **homogeneidad de las varianzas** u **homocedasticidad**, es comprobar que la razón entre la máxima y la mínima varianza muestral de los grupos no sea superior a 1,5.

$$\overline{x}_A = \frac{23+19+25+23+20}{5} = 22,0$$
$$s_A^2 = \frac{(23-22)^2+(19-22)^2+(25-22)^2+(23-22)^2+(20-22)^2}{4} = 6,0$$
$$\overline{x}_B = \frac{26+24+28+23+29}{5} = 26,0$$
$$s_B^2 = \frac{(26-26)^2+(24-26)^2+(28-26)^2+(23-26)^2+(29-26)^2}{4} = 6,5$$
$$\overline{x}_C = \frac{19+24+20+21+17}{5} = 20,2$$
$$s_C^2 = \frac{(19-20,2)^2+(24-20,2)^2+(20-20,2)^2+(21-20,2)^2+(17-20,2)^2}{4} = 6,7$$

En el caso del ejemplo, la muestra obtenida para el algoritmo A tiene la menor varianza, mientras que la muestra del algoritmo C tiene la mayor:

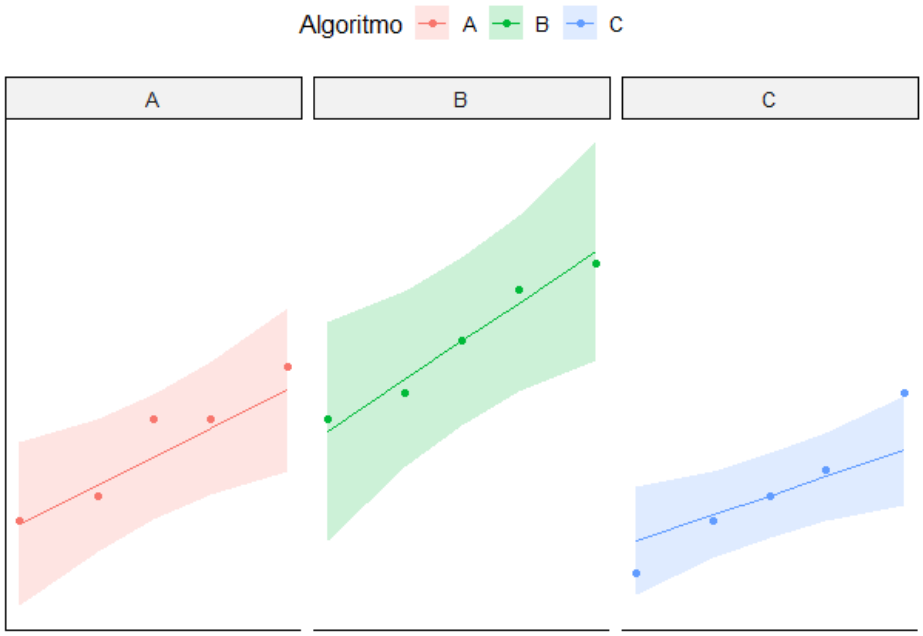


Figura 9.1: gráfico para comprobar el supuesto de normalidad en las tres muestras del ejemplo.

$$\frac{s_C^2}{s_A^2} = \frac{6,7}{6,0} = 1,117 \tag{9.1}$$

En consecuencia, la condición de homocedasticidad se verifica para el ejemplo.

Se ha encontrado que ANOVA es una prueba **robusta**, que resiste razonablemente bien a desviaciones en las condiciones de normalidad o de homocedasticidad, especialmente cuando las muestras tienen el mismo tamaño. Pero estas suposiciones sí están detrás de la lógica y matemática de la prueba, por lo que **no debemos ignorar** violaciones importantes a estas condiciones.

9.2 PROCEDIMIENTO ANOVA DE UNA VÍA PARA MUESTRAS INDEPENDIENTES

Como su nombre indica, ANOVA se centra en la **variabilidad** de las muestras, una generalización de la varianza, que se calcula en base a la suma de los cuadrados de las desviaciones, como muestra la ecuación 9.2, donde n corresponde al tamaño de la muestra, y $\overline{x}$ a la media muestral.

$$SS = \sum_{i=1}^{n} (x_i - \overline{x})^2 \tag{9.2}$$

9.2.1 Variabilidad total

La variabilidad total, SS_T, se calcula mediante la ecuación 9.2 considerando la totalidad de observaciones, vale decir, combinando las muestras correspondientes a los diferentes grupos.

$$\begin{aligned}\overline{x}_T &= \frac{23+19+25+23+20+26+24+28+23+29+19+24+20+21+17}{15} \\ &= 22,733\end{aligned}$$

$$\begin{aligned}SS_T = &(23-22,733)^2+(19-22,733)^2+(25-22,733)^2+(23-22,733)^2+(20-22,733)^2+ \\ &(26-22,733)^2+(24-22,733)^2+(28-22,733)^2+(23-22,733)^2+(29-22,733)^2+ \\ &(19-22,733)^2+(24-22,733)^2+(20-22,733)^2+(21-22,733)^2+(12-22,733)^2 \\ = &\ 164,933\end{aligned}$$

La variabilidad total puede descomponerse en dos partes: una de ellas corresponde a la variabilidad existente al interior de cada uno de los grupos (o variabilidad intra-grupos), *within groups* en inglés, denotada por SS_{wg}; la otra corresponde a la variabilidad entre los diferentes grupos, *between groups* en inglés, denotada como SS_{bg}. La ecuación 9.3 muestra una **identidad importante** que relaciona ambas componentes.

$$SS_T = SS_{bg} + SS_{wg} \tag{9.3}$$

9.2.2 Variabilidad entre grupos

La **variabilidad entre grupos** nos permite medir de manera agregada la magnitud de las diferencias entre las distintas medias muestrales. Se calcula como muestra la ecuación 9.4, donde:

- k: cantidad de grupos.
- n_i: cantidad de observaciones en el i-ésimo grupo.
- $\overline{x}_i$: media del i-ésimo grupo.
- $\overline{x}_T$: media de la muestra combinada.

$$SS_{bg} = \sum_{i=1}^{k} n_i(\overline{x}_i - \overline{x}_T)^2 \tag{9.4}$$

Esta medida corresponde a la suma de las desviaciones cuadradas de la media de cada grupo con respecto a la media combinada, donde la diferencia de cada grupo se pondera por la cantidad de observaciones que este contiene, a fin de mantener la representatividad de cada grupo. Mide el grado en que los grupos difieren unos de otros. Para el ejemplo tenemos:

$$SS_{bg} = 5(22,0-22,733)^2+5(26,0-22,733)^2+5(20,2-22,733)^2 = 88,133$$

9.2.3 Variabilidad al interior de cada grupo

La **variabilidad intra-grupos**, a su vez, corresponde a la suma total de las desviaciones cuadradas al interior de cada grupo, por lo que representa la variabilidad aleatoria de cada uno de los diferentes grupos. Esta medida se calcula de acuerdo a la ecuación 9.5, donde SS_i corresponde a la variabilidad del i-ésimo grupo, calculada mediante la ecuación 9.2.

$$SS_{wg} = \sum_{i=1}^{k} SS_i \tag{9.5}$$

Para el ejemplo:

$$\begin{aligned}
SS_A &= (23-22)^2 + (19-22)^2 + (25-22)^2 + (23-22)^2 + (20-22)^2 = 24,0 \\
SS_B &= (26-26)^2 + (24-26)^2 + (28-26)^2 + (23-26)^2 + (29-26)^2 = 26,0 \\
SS_C &= (19-20,2)^2 + (24-20,2)^2 + (20-20,2)^2 + (21-20,2)^2 + (17-20,2)^2 = 26,8 \\
SS_{wg} &= SS_A + SS_B + SS_C = 24,0 + 26,0 + 26,8 = 76,8
\end{aligned}$$

9.2.4 El estadístico de prueba F

En el capítulo 2 vimos que la varianza se calcula como:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

Podemos generalizar esta ecuación como muestra la ecuación 9.6, donde ν corresponde a los grados de libertad.

$$s^2 = \frac{1}{\nu}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{9.6}$$

En el contexto de análisis de varianza, llameremos MS, del inglés *mean square*, a la media de las desviaciones cuadradas. Para el caso de la variabilidad entre grupos, se tienen $\nu_{bg} = k - 1$ grados de libertad, donde k corresponde a la cantidad de grupos (para el ejemplo con los tres algortimos, $\nu_{bg} = 3 - 1 = 2$). Con ello, la media cuadrada entre grupos queda dada por la ecuación 9.7.

$$MS_{bg} = \frac{SS_{bg}}{\nu_{bg}} \tag{9.7}$$

Entonces, en nuestro ejemplo:

$$MS_{bg} = \frac{88,133}{2} = 44,067$$

De manera similar, los grados de libertad para la componente de la variabilidad al interior de los grupos está dada por la suma de los grados de libertad en cada grupo, como se ve en la ecuación 9.8 (siendo k la cantidad

de grupos), y su media de las desviaciones cuadradas se normaliza con estos grados de libertad (ecuación 9.9).

$$\nu_{wg} = \sum_{i=1}^{k}(n_k - 1) \tag{9.8}$$

$$MS_{wg} = \frac{SS_{wg}}{\nu_{wg}} \tag{9.9}$$

Así, para el ejemplo tenemos:

$$\begin{aligned} \nu_{wg} &= (5-1) + (5-1) + (5-1) = 12 \\ MS_{wg} &= \frac{76,8}{12} = 6,4 \end{aligned}$$

En ocasiones resulta útil conocer también la cantidad total de grados de libertad, que podemos obtener mediante la ecuación 9.10, donde n_T es el tamaño de la muestra combinada.

$$\nu_T = n_T - 1 = \nu_{bg} + \nu_{wg} \tag{9.10}$$

Si bien la relación entre MS_{bg} y MS_{wg} es compleja, en general se cumple que:

- Si la hipótesis nula es verdadera, MS_{bg} tiende a ser menor o igual que MS_{wg}.
- Si la hipótesis nula es falsa, MS_{bg} tiende a ser mayor que MS_{wg}.

Representamos esta relación mediante la razón F (el estadístico de prueba para ANOVA), que se calcula como muestra la ecuación 9.11, donde MS_{efecto} corresponde a una estimación de la varianza del efecto que se desea medir y MS_{error}, a la variabilidad aleatoria pura asociada a la situación. En este punto puede ser útil revisar nuevamente lo que ya hemos aprendido de la distribución F (capítulo 3).

$$F = \frac{MS_{\text{efecto}}}{MS_{\text{error}}} \tag{9.11}$$

En el ejemplo queremos estudiar si existe diferencia entre las medias de los grupos, por lo que $MS_{\text{efecto}} = MS_{bg}$. Asimismo, la variabilidad aleatoria está dada por la variabilidad al interior de los grupos, por lo que $MS_{\text{error}} = MS_{wg}$. Así:

$$F = \frac{44,067}{6,4} = 6,885$$

De manera similar a lo que hemos visto en otras pruebas, el p-valor corresponde al área bajo la cola superior de la distribución F (en este caso con 2 y 12 grados de libertad) mayor o igual al estadístico obtenido, que en R puede calcularse mediante la llamada `pf(6,885, 2, 12, lower.tail = FALSE)`, obteniéndose $p = 0,010$.

9.2.5 Resultado del procedimiento ANOVA

El resultado del procedimiento ANOVA suele representarse en forma tabular, como muestra la tabla 9.1.

Fuente	ν	SS	MS	F	p
Entre grupos (efecto)	2	88,133	44,067	6,885	0,010
Intra-grupos (error)	12	76,800	6,400		
TOTAL	14	164,933			

Tabla 9.1: resultado del procedimiento ANOVA.

Como es usual, la conclusión de esta prueba se efectúa comparando el valor p con el nivel de significación. En el ejemplo, $\alpha = 0,025$ y $p < \alpha$, por lo que rechazamos la hipótesis nula en favor de la hipótesis alternativa. En consecuencia, podemos concluir con 97,5 % de confianza que el tiempo de ejecución promedio es diferente para al menos uno de los algoritmos comparados.

Una observación importante que debemos tener en cuenta es que, si usamos ANOVA para casos con **solo dos grupos** (en su correspondientes versiones pareada o independiente), los resultados son equivalentes a los que obtendríamos con una **prueba t de Student**, y el estadístico F sería igual al cuadrado del estadístico t. No obstante, la prueba t puede ser unilateral o bilateral, mientras que el análisis de varianza es intrínsecamente unidireccional, pues la distribución F solo está definida para valores no negativos.

9.2.6 Resumen del procedimiento ANOVA de una vía para muestras independientes

El procedimiento ANOVA de una vía para variables independientes puede resumirse en los siguientes pasos:

1. Calcular la suma de los cuadrados de las desviaciones para la muestra combinada (SS_T).
2. Para cada grupo g, calcular la suma de los cuadrados de las desviaciones dentro de dicho grupo (SS_g).
3. Calcular la variabilidad entre grupos (SS_{bg}).
4. Calcular la variabilidad al interior de los grupos (SS_{wg}).
5. Calcular los grados de libertad (ν_T, ν_{bg} y ν_{wg}).
6. Calcular las medias de las desviaciones cuadradas (MS_{bg} y MS_{wg}).
7. Calcular el estadístico de prueba (F).
8. Obtener el valor p.

9.3 ANOVA DE UNA VÍA PARA MUESTRAS INDEPENDIENTES EN R

Desde luego, R nos ofrece funciones para realizar diferentes pruebas ANOVA, incluyendo la de una vía para muestras independientes.

La primera alternativa que conoceremos es la función `aov(formula, data)`, donde:

- `formula`: se escribe de la forma `variable_dependiente ~ variable_independiente`.
- `data`: data frame que contiene las variables especificadas en la fórmula.

Otra opción es usar la función `ezANOVA(data, dv, wid, between, return_aov)` del paquete `ez`, donde:

- `data`: data frame con los datos.
- `dv`: variable dependiente (numérica con escala de igual intervalo).
- `wid`: variable (factor) con el identificador de cada instancia.
- `between`: variable independiente (factor).

- `return_aov`: si es verdadero, devuelve un objeto de tipo aov para uso posterior.

Un parámetro adicional que no hemos mencionado es `type`, el cual no estudiaremos en detalle porque escapa a los alcances de este libro. Sin embargo, se debe tener en cuenta que este argumento permite incorporar algunas modificaciones y resguardos en caso que las muestras tengan diferentes tamaños o se tengan datos incompletos. Al trabajar con R, para la mayoría de los casos se recomienda mantener el valor por defecto (`type = 2`).

Una ventaja de `ezANOVA()` por sobre `aov()` es que, además de ejecutar la prueba ANOVA, realiza también la **prueba de homocedasticidad de Levene** (NIST/SEMATECH, 2013). Si bien no estudiaremos esta prueba en detalle, es pertinente mencionar que sirve para comprobar si k muestras tienen igual varianza, por lo que su resultado nos ayuda a verificar las condiciones requeridas para poder aplicar el procedimiento ANOVA de una vía para muestras independientes. Las hipótesis detrás de esta prueba son:

H_0: las varianzas de las k muestras son iguales.
H_A: al menos una de las muestras tiene varianza diferente a alguna de las demás.

El paquete `ez` contiene también la función `ezPlot(data, dv, wid, between, x)`, la cual nos permite ver gráficamente el tamaño del efecto medido. En general, los argumentos son los mismos que para `ezANOVA()`, con la salvedad del nuevo argumento `x`, que señala la variable que va en el eje horizontal del gráfico.

El script 9.1 muestra el procedimiento ANOVA de una vía para muestras independientes usando ambas funciones, con igual resultado al obtenido de manera manual. Además, genera el gráfico del tamaño del efecto medido, presentado en la figura 9.2.

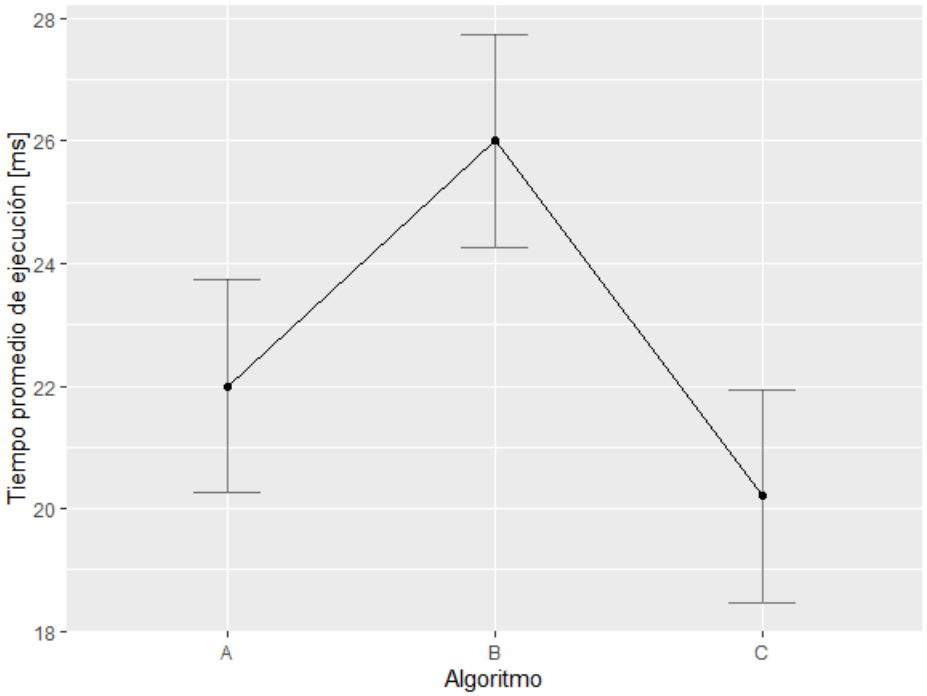


Figura 9.2: tamaño del efecto medido.

Script 9.1: procedimiento ANOVA de una vía para muestras independientes.

```
1 library(tidyverse)
2 library(ggpubr)
3 library(ez)
4
5 # Crear el data frame en formato ancho.
6 A <- c(23, 19, 25, 23, 20)
```

```r
B <- c(26, 24, 28, 23, 29)
C <- c(19, 24, 20, 21, 17)
datos <- data.frame(A, B, C)

# Llevar data frame a formato largo.
datos <- datos %>% pivot_longer(c("A", "B", "C"),
                                names_to = "algoritmo",
                                values_to = "tiempo")

datos[["algoritmo"]] <- factor(datos[["algoritmo"]])
datos[["instancia"]] <- factor(1:nrow(datos))

# Comprobción de normalidad.
g <- ggqqplot(datos,
              x = "tiempo",
              y = "algoritmo",
              color = "algoritmo")

g <- g + facet_wrap(~ algoritmo)
g <- g + rremove("x.ticks") + rremove("x.text")
g <- g + rremove("y.ticks") + rremove("y.text")
g <- g + rremove("axis.title")
print(g)

# Procedimiento ANOVA con aov().
cat("Procedimiento ANOVA usando aov\n\n")
prueba <- aov(tiempo ~ algoritmo, data = datos)
print(summary(prueba))

# Procedimiento ANOVA con ezANOVA().
cat("\n\nProcedimiento ANOVA usando ezANOVA\n\n")
prueba2 <- ezANOVA(
  data = datos,
  dv = tiempo,
  between = algoritmo,
  wid = instancia,
  return_aov = TRUE)

print(prueba2)

# Gráfico del tamaño del efecto.
g2 <- ezPlot(
  data = datos,
  dv = tiempo,
  wid = instancia,
  between = algoritmo,
  y_lab = "Tiempo promedio de ejecución [ms]",
  x = algoritmo
)

print(g2)
```

9.4 ANÁLISIS POST-HOC

Al aplicar el procedimiento ANOVA de una vía para muestras independientes a nuestro ejemplo pudimos concluir que **existe al menos un algoritmo cuyo tiempo promedio de ejecución es diferente al de los demás**. Ahora bien, si los algoritmos del ejemplo tienen por objeto resolver un problema crítico, de cuya rápida solución depende aumentar la productividad de una empresa o prevenir una situación de mucho riesgo, desde luego nos interesaría conocer cuál es el mejor (o el peor) de los algoritmos comparados a fin de poder garantizar un menor tiempo de respuesta. En consecuencia, necesitamos contar con algún método que permita determinar si los tiempos de ejecución de los algoritmos A y B difieren significativamente, o los de B y C, o bien los de A y C. En el contexto general, si tenemos k grupos, la cantidad de comparaciones (N) que deberíamos efectuar está dada por la ecuación 9.12.

$$N = \binom{k}{2} = \frac{k(k-1)}{2} \tag{9.12}$$

Al igual que aprendimos en el capítulo anterior para la prueba Q de Cochran, existen diversas pruebas *post-hoc* que podemos usar para este fin, algunas de las cuales exploraremos a continuación.

9.4.1 Correcciones de Bonferroni y Holm

Como ya estudiamos en el capítulo anterior, los factores de corrección de Bonferroni y Holm distribuyen el nivel de significación cuando se realizan múltiples comparaciones entre pares de grupos. Las fórmulas para calcularlos son las mismas que ya conocimos, pero ahora se realizan pruebas t para muestras independientes para cada par. R dispone de la función `pairwise.t.test(x, g, p.adjust.method, pool.sd, paired, alternative, ...)`, donde:

- `x`: vector con la variable dependiente.
- `g`: factor o vector de agrupamiento.
- `p.adjust.method`: señala qué método emplear para ajustar los valores p resultantes.
- `pool.sd`: valor booleano que indica si se usa o no varianza combinada.
- `paired`: valor booleano que indica si las pruebas t son pareadas (verdadero) o no.
- `alternative`: indica si la prueba es bilateral ("two.sided") o unilateral ("greater" o "less").
- `...`: argumentos adicionales que se pasan a la función `t.test()` que es llamada internamente

El script 9.2 muestra la realización de pruebas t para cada par de grupos usando tanto la corrección de Bonferroni como la de Holm, obteniéndose los resultados que se muestran en la figura 9.3. Debemos recordar que el nivel de significación que se entrega como argumento es el mismo que usamos en el procedimiento ANOVA.

Script 9.2: procedimientos *post-hoc* de Bonferroni y Holm en R.

```
library(tidyverse)

# Crear el data frame en formato ancho.
A <- c(23, 19, 25, 23, 20)
B <- c(26, 24, 28, 23, 29)
C <- c(19, 24, 20, 21, 17)
datos <- data.frame(A, B, C)

# Llevar data frame a formato largo.
datos <- datos %>% pivot_longer(c("A", "B", "C"),
```

```
                                names_to = "algoritmo",
                                values_to = "tiempo")

datos[["algoritmo"]] <- factor(datos[["algoritmo"]])
datos[["instancia"]] <- factor(1:nrow(datos))

# Establecer nivel de significación (el mismo usado en ANOVA).
alfa <- 0.025

# Procedimiento post-hoc de Bonferroni.
cat("Procedimiento post-hoc de Bonferroni\n\n")

bonferroni <- pairwise.t.test(datos[["tiempo"]],
                              datos[["algoritmo"]],
                              p.adj = "bonferroni",
                              pool.sd = TRUE,
                              paired = FALSE,
                              conf.level = 1 - alfa)

print(bonferroni)

# Procedimiento post-hoc de Holm.
cat("\n\nProcedimiento post-hoc de Holm\n\n")

holm <- pairwise.t.test(datos[["tiempo"]],
                        datos[["algoritmo"]],
                        p.adj = "holm",
                        pool.sd = TRUE,
                        paired = FALSE,
                        conf.level = 1 - alfa)

print(holm)
```

Los valores p obtenidos con ambos métodos son diferentes (un lector atento recordará que la corrección de Bonferroni es considerada muy conservadora). Sin embargo, en ambos casos podemos ver que únicamente los algoritmos B y C presentan una diferencia significativa al comparar el valor p ajustado que entrega R con el nivel de significación ($\alpha = 0,025$). Si miramos el gráfico del tamaño del efecto obtenido para el procedimiento ANOVA (figura 9.2), podemos concluir entonces con 97,5 % de confianza que el algoritmo C es más rápido que el algoritmo B.

9.4.2 Prueba HSD de Tukey

La prueba **HSD de Tukey** es más poderosa que los factores de corrección de Bonferroni y Holm. Se asemeja a estas últimas en que también busca diferencias significativas (de hecho, el nombre HSD se debe a las siglas inglesas para "diferencia honestamente significativa") entre los diferentes pares de medias, aunque usa un enfoque muy diferente: para ello emplea el estadístico Q, el cual sigue una distribución de rango estudiantizado[2], que para cualquier par de medias en los k grupos se calcula según la ecuación 9.13, donde:

- $\overline{x}_g$ es la mayor de las dos medias comparadas.
- $\overline{x}_p$ es la menor de las dos medias comparadas.
- MS_{wg} corresponde la media cuadrada intra-grupos (entregada por el procedimiento ANOVA).

[2]El detalle de la distribución de rango estudiantizado escapa a los alcances de este texto.

```
Procedimiento post-hoc de Bonferroni


Pairwise comparisons using t tests with pooled SD

data:  datos[["tiempo"]] and datos[["algoritmo"]]

  A     B
B 0.084 -
C 0.848 0.010

P value adjustment method: bonferroni


Procedimiento post-hoc de Holm


Pairwise comparisons using t tests with pooled SD

data:  datos[["tiempo"]] and datos[["algoritmo"]]

  A     B
B 0.056 -
C 0.283 0.010

P value adjustment method: holm
```

Figura 9.3: valores p obtenidos en las pruebas t para cada par de grupos mediante los métodos de Bonferroni y Holm.

- n_m es la cantidad de observaciones por cada muestra. Si las k muestras tienen tamaños diferentes, se calcula mediante la fórmula presentada en la ecuación 9.14.

$$Q = \frac{\overline{x}_g - \overline{x}_p}{\sqrt{\frac{MS_{wg}}{n_m}}} \tag{9.13}$$

$$n_m = \frac{k}{\sum_{i=1}^{k} \frac{1}{n_i}} \tag{9.14}$$

En la práctica, sin embargo, no necesitamos calcular el estadístico Q para cada par de medias, sino que basta con conocer el valor crítico de este estadístico para el nivel de significación α establecido (denotado por Q_α), el cual depende de la cantidad de grupos (k) y de los grados de libertad del error, ν_{wg} en el caso de ANOVA de una vía para muestras independientes. La llamada `qtukey(`α`,` n_m`,` ν_{wg}`, lower.tail = FALSE)` en R entrega el valor de Q_α.

El valor crítico Q_α nos permite determinar cuán grande debe ser la diferencia entre las medias de dos grupos para ser considerada significativa, lo cual se logra mediante la ecuación 9.15.

$$HSD_\alpha = Q_\alpha \cdot \sqrt{\frac{MS_{wg}}{n_m}} \tag{9.15}$$

Así, una diferencia entre las medias de dos grupos únicamente es significativa si es mayor o igual que HSD_α.

Para el ejemplo, se tenemos que $Q_\alpha = 4,324$, de donde:

$$HSD_{0,025} = 4,324 \cdot \sqrt{\frac{6,4}{5}} = 4,892$$

Recordando del procedimiento ANOVA que $\overline{x}_A = 22$, $\overline{x}_B = 26$ y $\overline{x}_C = 20,2$; tenemos:

$$\overline{x}_B - \overline{x}_A = 26 - 22 = 4$$

$$\overline{x}_A - \overline{x}_C = 22 - 20,2 = 1,8$$

$$\overline{x}_B - \overline{x}_C = 26 - 20,2 = 5,8$$

Así, la tercera diferencia es la única que supera $HSD_{0,025}$, con lo que solo existe diferencia significativa entre los tiempos promedio de ejecución de los algoritmos B y C, y se puede concluir que el algoritmo C es más rápido que el algoritmo B, lo que se condice con los resultados presentados en la figura 9.2.

R también permite realizar la prueba HSD de Tukey, como muestra el script 9.3. La función para ello es `TukeyHSD(x, which, ordered, conf.level)`, donde:

- `x`: un modelo ANOVA (objeto de tipo `aov`).
- `which`: string con el nombre de la variable para la que se calculan las diferencias.
- `ordered`: valor lógico que, cuando es verdadero, hace que los grupos se ordenen de acuerdo a sus medias a fin de obtener diferencias positivas.
- `conf.level`: nivel de confianza.

La figura 9.4 muestra el resultado obtenido para la prueba HSD de Tukey mediante el script 9.3.

Script 9.3: procedimiento *post-hoc* de Tukey.

```
library(tidyverse)

# Crear el data frame en formato ancho.
A <- c(23, 19, 25, 23, 20)
B <- c(26, 24, 28, 23, 29)
C <- c(19, 24, 20, 21, 17)
datos <- data.frame(A, B, C)

# Llevar data frame a formato largo.
datos <- datos %>% pivot_longer(c("A", "B", "C"),
                                names_to = "algoritmo",
                                values_to = "tiempo")

datos[["algoritmo"]] <- factor(datos[["algoritmo"]])
datos[["instancia"]] <- factor(1:nrow(datos))

# Establecer nivel de significación (el mismo usado en ANOVA).
alfa <- 0.025

# Procedimiento ANOVA.
anova <- aov(tiempo ~ algoritmo, data = datos)

# Prueba HSD de Tukey.
post_hoc <- TukeyHSD(anova,
                     "algoritmo",
                     ordered = TRUE,
```

```
27                       conf.level = 1 - alfa)
28
29 print(post_hoc)
```

```
  Tukey multiple comparisons of means
    97.5% family-wise confidence level
    factor levels have been ordered

Fit: aov(formula = tiempo ~ algoritmo, data = datos)

$algoritmo
    diff        lwr       upr     p adj
A-C  1.8 -3.0923417  6.692342 0.5176889
B-C  5.8  0.9076583 10.692342 0.0090297
B-A  4.0 -0.8923417  8.892342 0.0670199
```

Figura 9.4: resultado del procedimiento *post-hoc* HSD de Tukey.

En la figura 9.4 podemos apreciar que la columna `diff` muestra las diferencias de las medias entre grupos, obteniéndose resultados idénticos a los teóricos, y la columna `p.adj` entrega valores p asociados a cada diferencia, **ajustados** para compararlos con el nivel de significación original. Cabe destacar que el único valor p menor a este nivel ($\alpha = 0,025$) corresponde a la diferencia B-C, siendo esta última la única significativa, lo cual una vez más coincide con el resultado del procedimiento manual. También debemos notar que las columnas `lwr` y `upr` muestran el límite inferior y superior, respectivamente, del intervalo de $(1 - \alpha) \cdot 100\,\%$ confianza para la verdadera diferencia entre las medias de los grupos.

9.4.3 Prueba de comparación de Scheffé

Otra alternativa para hacer un análisis *post-hoc* es la **prueba de Scheffé**. Al igual que la corrección de Bonferroni, este método también es muy conservador al momento de efectuar comparaciones entre pares. No obstante, tiene la ventaja de que permite hacer comparaciones adicionales, además de todos los pares de grupos: por ejemplo, podríamos preguntar si un grupo es mejor que todos los demás. El ingeniero del ejemplo podría, tras encontrar mediante el procedimiento ANOVA que existen diferencias significativas, plantearse preguntas del siguiente tipo:

1. ¿Existe diferencia entre los tiempos de ejecución de los algoritmos A y B?
2. ¿Es el tiempo promedio de ejecución del algoritmo A distinto al tiempo de ejecución promedio de los algoritmos B y C?

La primera pregunta corresponde a una comparación entre pares, pero la segunda resulta más compleja. En realidad, podemos modelar escenarios para múltiples preguntas, usando para ello **contrastes**, que son **combinaciones lineales** de las medias de cada grupo. Para entender mejor esta idea, veamos la primera pregunta. Matemáticamente, puede formularse como las siguientes hipótesis:

H_0: $\mu_A - \mu_B = 0$
H_A: $\mu_A - \mu_B \neq 0$

La hipótesis nula puede expresarse, entonces, como una combinación lineal de la forma:

$$c_A \cdot \mu_A + c_B \cdot \mu_B + c_C \cdot \mu_C = 0$$

Que puede, a su vez, representarse vectorialmente como:

$$[c_A, c_B, c_C]$$

En este caso, resulta evidente que la combinación lineal es:

$$1 \cdot \mu_A - 1 \cdot \mu_B + 0 \cdot \mu_C = 0$$

Que corresponde al vector:

$$[1, -1, 0]$$

La segunda pregunta es algo más compleja, pero las hipótesis asociadas son:

H_0: $\mu_A - \frac{\mu_B + \mu_C}{2} = 0$
H_A: $\mu_A - \frac{\mu_B + \mu_C}{2} \neq 0$

Vectorialmente dada por:

$$\left[1, -\frac{1}{2}, -\frac{1}{2}\right]$$

Ahora que hemos establecido qué es un contraste, podemos comenzar a explicar el el procedimiento *post-hoc* de Scheffé, el cual ocupa el mismo nivel de significación empleado para el procedimiento ANOVA. Recordemos que, para el ejemplo, $\alpha = 0,025$, $\overline{x}_A = 22$, $\overline{x}_B = 26$ y $\overline{x}_C = 20,2$.

El primer paso consiste en determinar los contrastes a realizar. Supongamos que el ingeniero desea hacer todas las comparaciones entre pares y, además, comparar cada algoritmo contra los dos restantes. Podemos representar esto en forma matricial, donde cada fila de la matriz corresponde a un contraste:

$$\begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \\ 1 & -0,5 & -0,5 \\ -0,5 & 1 & -0,5 \\ -0,5 & -0,5 & 1 \end{bmatrix}$$

Luego calculamos los estimadores para cada contraste C_i:

$$\begin{aligned} C_1 &= |\overline{x}_A - \overline{x}_B| = 4,0 \\ C_2 &= |\overline{x}_A - \overline{x}_C| = 1,8 \\ C_3 &= |\overline{x}_B - \overline{x}_C| = 5,8 \\ C_4 &= |\overline{x}_A - \frac{\overline{x}_B}{2} - \frac{\overline{x}_C}{2}| = 1,1 \\ C_5 &= |-\frac{\overline{x}_A}{2} + \overline{x}_B - \frac{\overline{x}_C}{2}| = 4,9 \\ C_6 &= |-\frac{\overline{x}_A}{2} - \frac{\overline{x}_B}{2} + \overline{x}_C| = 3,8 \end{aligned}$$

El tercer paso consiste en calcular los valores críticos para la prueba de comparación de Scheffé, dados por la ecuación 9.16, donde:

- i es el número de fila del contraste.
- ν_{efecto}, ν_{error} y MS_{error} se obtienen desde la tabla ANOVA (tabla 9.1).
- F^* corresponde al percentil $1-\alpha$ de la distribución $F(\nu_{\text{efecto}}, \nu_{\text{error}})$.
- c_j es el peso del grupo j en la comparación i.
- n_j es el tamaño de la muestra para el grupo j.

$$VC_i = \sqrt{\nu_{\text{efecto}} \cdot F^* \cdot MS_{\text{error}} \cdot \sum_{j=1}^{k} \frac{c_j^2}{n_j}} \tag{9.16}$$

Así, para el ejemplo tenemos que $\nu_{\text{efecto}} = 2$, $\nu_{\text{error}} = 12$ y $MS_{\text{error}} = 6,4$. Podemos calcular F^* en R con la llamada `qf(1 - 0.025, 2, 12, lower.tail = TRUE)`, obteniéndose $F^* = 5,0959$. Debemos notar que en la ecuación 9.16, $\nu_{\text{efecto}} \cdot F^* \cdot MS_{\text{error}} = 2 \cdot 5,0959 \cdot 6,4 = 65,2275$ es constante para todos los contrastes. Así:

$$VC_1 = \sqrt{65,2275 \cdot \left(\frac{1^2}{5} + \frac{(-1)^2}{5} + \frac{0^2}{5}\right)} = 4,9891$$

$$VC_2 = \sqrt{65,2275 \cdot \left(\frac{1^2}{5} + \frac{0^2}{5} + \frac{(-1)^2}{5}\right)} = 4,9891$$

$$VC_3 = \sqrt{65,2275 \cdot \left(\frac{0^2}{5} + \frac{1^2}{5} + \frac{(-1)^2}{5}\right)} = 4,9891$$

$$VC_4 = \sqrt{65,2275 \cdot \left(\frac{1^2}{5} + \frac{(-0,5)^2}{5} + \frac{(-0,5)^2}{5}\right)} = 2,4945$$

$$VC_5 = \sqrt{65,2275 \cdot \left(\frac{(-0,5)^2}{5} + \frac{1^2}{5} + \frac{(-0,5)^2}{5}\right)} = 2,4945$$

$$VC_6 = \sqrt{65,2275 \cdot \left(\frac{(-0,5)^2}{5} + \frac{(-0,5)^2}{5} + \frac{1^2}{5}\right)} = 2,4945$$

Tabulemos los resultados obtenidos hasta ahora, como muestra la tabla 9.2.

i	C_i	VC_i
1	4,0	4,9891
2	1,8	4,9891
3	5,8	4,9891
4	1,1	2,4945
5	4,9	2,4945
6	3,8	2,4945

Tabla 9.2: estimadores y valores críticos para los contrastes de la prueba de comparación de Scheffé.

Finalmente evaluamos cada contraste, comparando el estimador C_i con el valor crítico correspondiente, VC_i. Si $C_i > VC_i$, la comparación es estadísticamente significativa. Podemos ver, entonces, que las comparaciones 3, 5 y 6 son significativas. Esto quiere decir que:

- Existe una diferencia significativa entre las eficiencias de los algoritmos B y C.
- El tiempo promedio de ejecución del algoritmo B es distinto del tiempo promedio de ejecución (combinado) de los algoritmos A y C.
- El tiempo promedio de ejecución del algoritmo C es distinto del tiempo promedio de ejecución (combinado) de los algoritmos A y B.

En R, este procedimiento puede hacerse mediante la función `ScheffeTest(x, which, contrasts, conf.level)` del paquete `DescTools`, donde:

- `x`: objeto `aov` con el resultado de ANOVA.
- `which`: variable independiente en la prueba.
- `contrasts`: matriz con los contrastes (cada contraste es una columna).
- `conf.level`: nivel de confianza.

El script 9.4 muestra el ejemplo en R, cuyo resultado se presenta en la figura 9.5. A diferencia del proceso manual, la función `ScheffeTest()` nos entrega un valor p ajustado para cada contraste e identifica aquellos que son relevantes para diferentes niveles de significación. Debemos tener en cuenta que el resultado es ligeramente diferente debido a errores de redondeo. Aquí, al igual que en el caso de la prueba HSD de Tukey, las columnas `lwr` y `upr` señalan los límites del intervalo de confianza para la verdadera diferencia entre las medias de los grupos.

```
  Posthoc multiple comparisons of means: Scheffe Test
    97.5% family-wise confidence level

$algoritmo
      diff     lwr.ci     upr.ci   pval
A-B   -4.0 -9.1079193  1.1079193 0.0808 .
A-C    1.8 -3.3079193  6.9079193 0.5479
B-C    5.8  0.6920807 10.9079193 0.0118 *
A-B,C -1.1 -5.5235879  3.3235879 0.7356
B-A,C  4.9  0.4764121  9.3235879 0.0138 *
C-A,B -3.8 -8.2235879  0.6235879 0.0540 .

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figura 9.5: valores p e intervalos de confianza para las diferencias de las medias obtenidos mediante la prueba de comparación de Scheffé.

Script 9.4: prueba de comparación de Scheffé.

```
library(tidyverse)
library(DescTools)

# Crear el data frame en formato ancho.
A <- c(23, 19, 25, 23, 20)
B <- c(26, 24, 28, 23, 29)
C <- c(19, 24, 20, 21, 17)
datos <- data.frame(A, B, C)

# Llevar data frame a formato largo.
datos <- datos %>% pivot_longer(c("A", "B", "C"),
                                names_to = "algoritmo",
                                values_to = "tiempo")

datos[["algoritmo"]] <- factor(datos[["algoritmo"]])
datos[["instancia"]] <- factor(1:nrow(datos))

# Establecer nivel de significación (el mismo usado en ANOVA).
alfa <- 0.025

# Procedimiento ANOVA.
anova <- aov(tiempo ~ algoritmo, data = datos)
```

```r

# Crear matriz de contrastes.
contrastes <- matrix(c(1, -1, 0,
                       1, 0, -1,
                       0, 1, -1,
                       1, -0.5, -0.5,
                       -0.5, 1, -0.5,
                       -0.5, -0.5, 1),
                     nrow=6,
                     byrow = TRUE
)

# Trasponer matriz de contrastes (para que cada contraste sea una columna).
contrastes <- t(contrastes)

# Hacer prueba de Scheffé.
scheffe <- ScheffeTest(x = anova,
                       which = "algoritmo",
                       contrasts = contrastes,
                       conf.level = 1 - alfa
)

print(scheffe)
```

Un detalle importante a tener en cuenta es que podemos hacer la llamada a `ScheffeTest()` sin entregar los argumentos `which` y `contrasts`, en cuyo caso únicamente se contrastan todos los pares, como en las pruebas *post-hoc* precedentes.

9.5 EJERCICIOS PROPUESTOS

1. Si se tienen datos de tres grupos (A, B y C), ¿por qué no se pueden hacer tres comparaciones con pares de grupos con la prueba t de Student (A-B, A-C, B-C) vista en el capítulo 5?
2. Si SS es la suma de desviaciones cuadradas, ¿qué son SS entre grupos, SS al interior de los grupos y SS total?
3. Define la razón F. ¿Por qué se espera que sea cercana a 1 si es que las poblaciones tienen medias similares?
4. ¿Qué significa que un procedimiento ANOVA sea de una vía?
5. ¿Cuándo un procedimiento ANOVA de una vía es equivalente a una prueba T de Student?
6. ¿Qué sería ANOVA de dos vías?
7. ¿Cuándo aplica el procedimiento ANOVA de una vía para muestras independientes?
8. ¿Cuáles son las hipótesis contrastadas en el procedimiento ANOVA de una vía para muestras independientes?
9. Enumera las condiciones (o supuestos) del procedimiento ANOVA de una vía para muestras independientes para tener confiabilidad.
10. Investiga con más detalle por qué se dice que un procedimiento ANOVA es una prueba omnibus.
11. Investiga en qué consiste, para qué sirve y cómo se aplica en R la prueba de Levene.
12. Investiga algún procedimiento *post-hoc* no abordado en este capítulo, junto con la forma de aplicarlo en R.
13. El conjunto de datos `chickwts`, disponible en R, registra el peso de 71 pollitos a las seis semanas de nacidos y el tipo de alimento que cada pollito recibió. Para este conjunto de datos:
    a) Verifica si se cumplen las condiciones para efectuar un procedimiento ANOVA de una vía para

muestras independientes.

b) Independientemente del resultado anterior, efectúa el procedmiento ANOVA de una vía para muestras independientes a fin de determinar si existen diferencias en el peso de los pollitos de acuerdo al tipo de alimento recibido.

c) En caso de identificar que existen diferencias significativas, lleva a cabo los análisis post-hoc y determina qué tipos de alimento presentan dichas diferencias. Compara los resultados obtenidos con los diferentes métodos.

CAPÍTULO 10. ANOVA DE UNA VÍA PARA MUESTRAS CORRELACIONADAS

En el capítulo 9 conocimos el procedimiento ANOVA de una vía para muestras independientes, que podemos entender como una extensión de la prueba t de Student para muestras independientes. De manera similar, ahora abordaremos el **procedimiento ANOVA de una vía para muestras correlacionadas** (también llamado **ANOVA para medidas repetidas** o **ANOVA intra-sujetos**) que puede asociarse a la prueba t con muestras pareadas, pero ahora con tres o más mediciones (o condiciones) en lugar de dos. Para ello tomaremos como base la explicación que ofrece Lowry (1999, cap. 15).

En este caso podemos distinguir entre dos escenarios:

- **Diseño con medidas repetidas:** a cada sujeto se le toman medidas en las diferentes condiciones, por ejemplo, registrar los tiempos de ejecución para una misma instancia de un problema con k algoritmos diferentes.
- **Diseño con bloques aleatorios:** cada bloque contiene diferentes sujetos agrupados según una determinada característica, por ejemplo, registrar tiempos de ejecución usando instancias de grafos diferentes, pero similares (como que tengan el mismo número de vértices y aristas), para los k algoritmos.

El método es el mismo en ambos casos e intenta controlar estadísticamente la variación introducida por factores distintos al que se desea estudiar, usando para ello varias mediciones de un sujeto (o grupos de sujetos parecidos). Si bien el diseño con bloques aleatorios es común, especialmente en medicina, este apunte usa las medidas repetidas en su discusión, ya que son más comunes en el área de la informática.

Como es habitual, usemos un ejemplo para ver cómo se lleva a cabo el procedimiento ANOVA de una vía para muestras correlacionadas. Supongamos que un estudiante de un curso de programación debe comparar la eficiencia de cuatro algoritmos de ordenamiento: *quicksort*, *bubblesort*, *radixsort* y *mergesort*. Para ello, ha seleccionado aleatoriamente 6 arreglos de igual tamaño y registrado, para cada uno de ellos, el tiempo de ejecución utilizado por cada algoritmo (en milisegundos), como muestra la tabla 10.1[1].

Instancia	Quicksort	Bubblesort	Radixsort	Mergesort
1	23,2	31,6	30,1	25,0
2	22,6	29,3	28,4	25,7
3	23,4	30,7	28,7	25,7
4	23,3	30,8	28,3	23,7
5	21,8	29,8	29,9	25,5
6	23,9	30,3	29,1	24,7

Tabla 10.1: tiempos de ejecución para las diferentes instancias con cada algoritmo del ejemplo.

En este caso, la lógica es muy similar a la que ya conocimos para ANOVA con muestras independientes. Sin embargo, existe una diferencia importante al trabajar con muestras correlacionadas: no toda la variabilidad es pura e inevitable, sino que una parte de ella se debe a diferencias individuales preexistentes entre los sujetos (por ejemplo, un arreglo puede estar ordenado desde el inicio, mientras otro podría estar en orden inverso).

Recordemos que la pregunta detrás de ANOVA es: ¿se diferencian las medias poblacionales?, por lo que nuestras hipótesis son:

H_0: El tiempo de ejecución promedio es igual para los cuatro algoritmos.
H_A: El tiempo de ejecución promedio es diferente para al menos un algoritmo.

[1] Los valores aquí expuestos son ficticios.

10.1 CONDICIONES PARA USAR ANOVA DE UNA VÍA PARA MUESTRAS CORRELACIONADAS

Al igual que otras pruebas que hemos conocido en capítulos anteriores, este procedimiento requiere que se cumplan algunas condiciones:

1. La escala con que se mide la variable dependiente tiene las propiedades de una escala de intervalos iguales.
2. Las mediciones son independientes al interior de cada grupo.
3. Se puede suponer razonablemente que la(s) población(es) de origen sigue(n) una distribución normal.
4. La matriz de varianzas-covarianzas es esférica. Como explica Horn (2008, p. 1), esta condición establece que las varianzas entre los diferentes niveles de las medidas repetidas deben ser iguales.

Veamos si nuestro ejemplo cumple con las condiciones. La primera se verifica, puesto que el tiempo, como toda magnitud física, tiene una escala de intervalos iguales (de hecho tiene escala de razón). A su vez, el enunciado señala que el proceso seguido por el ingeniero garantiza el cumplimiento de la segunda condición.

La figura 10.1 (creada mediante el script 10.1, líneas 20–26) muestra gráficos Q-Q para cada grupo, donde se puede apreciar que no se observan valores que pudieran ser considerados atípicos y se puede suponer razonablemente que las distribuciones se asemejan a la normal.

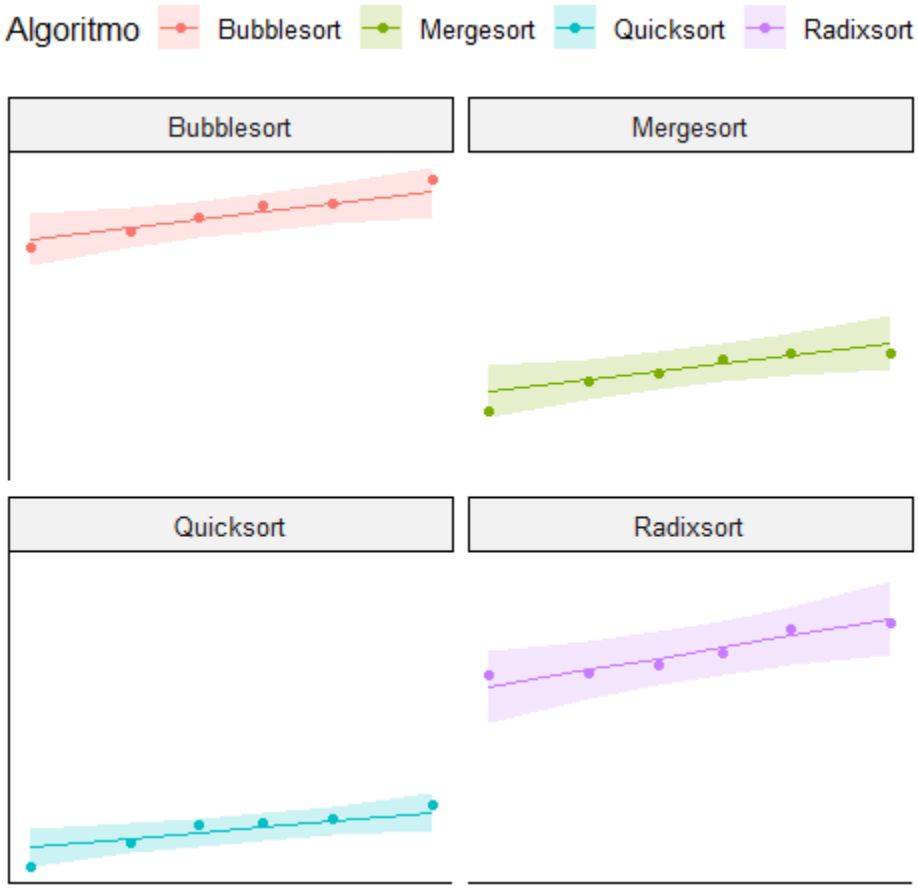


Figura 10.1: gráfico para comprobar el supuesto de normalidad en las muestras del ejemplo.

La prueba de esfericidad es más compleja, por lo que no se aborda en este texto. En principio, podemos examinar las diferencias entre las varianzas registradas para cada algoritmo (que se muestran en la tabla

10.2). Podemos ver que las diferencias parecen más bien "pequeñas" si se considera que los tiempos promedio están el rango de 23 a 30 [ms], por lo que podríamos asumir que son iguales. No obstante, la función de R `ezANOVA()` incluye una prueba para verificar esta condición: la **prueba de esfericidad de Mauchly**, e incluso proporciona un método para controlar posibles violaciones, como veremos más adelante en este capítulo.

	Mergesort	Quicksort	Radixsort
Bubblesort	0.05	0.12	0.07
Mergesort		0.06	0.01
Quicksort			-0.05

Tabla 10.2: diferencias de las varianza del tiempo de ejecución entre cada par de algoritmos.

10.2 PROCEDIMIENTO ANOVA DE UNA VÍA PARA MUESTRAS CORRELACIONADAS

Al igual que para el caso de muestras independientes, el procedimiento ANOVA para muestras correlacionadas opera en base a la variabilidad, calculada en base a la suma de los cuadrados de las desviaciones. Recordemos la forma general de este cálculo:

$$SS = \sum_{i=1}^{n} (x_i - \overline{x})^2$$

10.2.1 Variabilidad total, entre grupos e intra-grupos

Los primeros pasos son idénticos a los que ya estudiamos para ANOVA de una vía con muestras independientes, y consisten en calcular la variabilidad total (es decir, para las muestras combinadas), la variabilidad entre grupos y la variabilidad intra-grupos, denotadas por SS_T, SS_{bg} y SS_{wg}, respectivamente.

$$\begin{aligned} SS_T &= 224,930 \\ SS_{bg} &= 213,045 \\ SS_{wg} &= 11,885 \end{aligned}$$

10.2.2 Variabilidad entre sujetos

Como mencionamos en la introducción de este capítulo, al trabajar con muestras correlacionadas es necesario descartar la variabilidad debida a las diferencias preexistentes entre los diferentes sujetos ($SS_{sujetos}$), pues

estas son ajenas al factor en estudio, y solo nos interesa conservar la variabilidad pura (SS_{error}). Así, en ANOVA para muestras correlacionadas aparece una nueva identidad, dada por la ecuación 10.1.

$$SS_{wg} = SS_{sujetos} + SS_{error} \tag{10.1}$$

De manera similar a la que ya empleamos en cálculos previos, la variabilidad entre sujetos está dada por la ecuación 10.2, donde:

- k corresponde a la cantidad de observaciones (medidas) por cada sujeto.
- $\overline{x}_i$ es la media de las observaciones del i-ésimo sujeto.
- $\overline{x}_T$ es la media combinada de las mediciones.

$$SS_{sujetos} = k \cdot \sum_{i=1}^{n} (\overline{x}_i - \overline{x}_T)^2 \tag{10.2}$$

La tabla 10.3 muestra una vez más las observaciones del ejemplo, incluyendo el tiempo promedio de ejecución para cada instancia.

Instancia	**Quicksort**	**Bubblesort**	**Radixsort**	**Mergesort**	$\overline{x}$
1	23,2	31,6	30,1	25,0	27,475
2	22,6	29,3	28,4	25,7	26,500
3	23,4	30,7	28,7	25,7	27,125
4	23,3	30,8	28,3	23,7	26,525
5	21,8	29,8	29,9	25,5	26,750
6	23,9	30,3	29,1	24,7	27,000

Tabla 10.3: tiempos de ejecución y tiempo medio de ejecución para las diferentes instancias del ejemplo.

$$\begin{aligned} SS_{sujetos} = 4 \cdot [&(27,475 - 26,896)^2 + (26,500 - 26,896)^2 + (27,125 - 26,896)^2 + \\ &+ (26,525 - 26,8967)^2 + (26,750 - 26,896)^2 + (27,000 - 26,896)^2] = 2,857 \end{aligned}$$

$$SS_{error} = SS_{wg} - SS_{sujetos} = 11,885 - 2,857 = 9,028$$

10.2.3 El estadístico de prueba F

Al igual que en el capítulo 9, calculamos ahora los grados de libertad ya conocidos (recordemos que ahora k corresponde a la cantidad de mediciones por sujeto):

$$\begin{aligned} \nu_T = &\quad n_T - 1 = 24 - 1 &= 23 \\ \nu_{bg} = &\quad k - 1 = 4 - 1 &= 3 \\ \nu_{wg} = &\quad n_T - k = 24 - 4 &= 20 \end{aligned}$$

Puesto que anteriormente descompusimos la variabilidad intra-grupos en variabilidad intra-sujetos y variabilidad del error, necesitamos también identificar los grados de libertad correspondientes a cada componente, dados por las ecuaciones 10.3 y 10.4, respectivamente.

$$\nu_{sujetos} = \quad n_{sujetos} - 1 \tag{10.3}$$
$$\nu_{error} = \quad \nu_{wg} - \nu_{sujetos} \tag{10.4}$$

Para el ejemplo, tenemos:

$$\begin{aligned} \nu_{sujetos} = &\quad 6 - 1 \quad &= 5 \\ \nu_{error} = &\quad 20 - 5 \quad &= 15 \end{aligned}$$

Las medias cuadradas del procedimiento ANOVA de una vía para muestras correlacionadas son, respectivamente, la media cuadrada entre grupos para el efecto (igual que para muestras independientes) y la media cuadrada del error, dada por la ecuación 10.5.

$$MS_{error} = \frac{SS_{error}}{\nu_{error}} \tag{10.5}$$

Así, tenemos que:

$$\begin{aligned} MS_{efecto} = MS_{bg} &\quad = \tfrac{213{,}045}{3} &= 71,015 \\ MS_{error} &\quad = \tfrac{9{,}028}{15} &= 0,602 \end{aligned}$$

En consecuencia:

$$F = \frac{MS_{\text{efecto}}}{MS_{\text{error}}} = \frac{71,015}{0,602} = 117,992$$

Al hacer la llamada `pf(117.992, 3, 15, lower.tail = FALSE)` para calcular el valor p, obtenemos $p = 1,177 \cdot 10^{-10}$.

10.2.4 Resultado del procedimiento ANOVA

Una vez más, el resultado del procedimiento se representa en la forma tabular, como muestra la tabla 10.4.

Fuente	ν	**SS**	**MS**	**F**	**p**
Efecto	3	213,045	71,015	117,991	$1,177 \cdot 10^{-10}$
Error	15	9,028	0,602		
TOTAL	23	224,930			

Tabla 10.4: resultado del procedimiento ANOVA.

El valor p obtenido es muy menor a cualquier nivel de significación típico que podamos considerar, por lo que rechazamos la hipótesis nula en favor de la hipótesis alternativa. Así, el estudiante del ejemplo concluye con más de 99 % de confianza que existen diferencias significativas entre al menos dos de los algoritmos de ordenamiento comparados.

10.2.5 Resumen del procedimiento ANOVA de una vía para muestras correlacionadas

El procedimiento ANOVA de una vía para variables independientes puede resumirse en los siguientes pasos:

1. Calcular la suma de los cuadrados de las desviaciones para la muestra combinada (SS_T).
2. Para cada grupo g, calcular la suma de los cuadrados de las desviaciones dentro de dicho grupo (SS_g).
3. Calcular la variabilidad entre grupos (SS_{bg}).
4. Calcular la variabilidad al interior de los grupos (SS_{wg}).
5. Calcular la variabilidad intra-sujetos y la variabilidad del error ($SS_{sujetos}$ y SS_{error}).
6. Calcular los grados de libertad relevantes (ν_T, $\nu_{efecto} = \nu_{bg}$ y ν_{error}).
7. Calcular las medias cuadradas ($MS_{efecto} = MS_{bg}$ y MS_{error}).
8. Calcular el estadístico de prueba (F).
9. Obtener el valor p.

10.3 ANOVA DE UNA VÍA PARA MUESTRAS CORRELACIONADAS EN R

Para efectuar el procedimiento ANOVA de una vía para muestras correlacionadas en R, podemos usar las mismas funciones ya estudiadas en el capítulo 9: `aov()` y `ezANOVA()`, como ilustra el script 10.1.

En el caso de `aov()`, podemos apreciar que la fórmula entregada en la llamada (líneas 31–32 del script 10.1) es bastante diferente a la del capítulo 9. Esto se debe a que esta función realiza por defecto un procedimiento ANOVA para muestras independientes, por lo que se debe explicitar en la fórmula que se requiere descartar la variabilidad entre sujetos. La figura 10.2 muestra el resultado obtenido, idéntico al presentado en la tabla 10.4 salvo por ligeras diferencias de redondeo.

```
Resultado de la prueba ANOVA para muestras correlacionadas con aov

Error: Instancia
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  5  2.857  0.5714

Error: Instancia:Algoritmo
          Df Sum Sq Mean Sq F value   Pr(>F)
Algoritmo  3 213.04   71.01     118 1.18e-10 ***
Residuals 15   9.03    0.60
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 10.2: resultado del procedimiento ANOVA usando la función `aov()`.

La llamada a `ezANOVA()`, en cambio, es muy similar a la ya conocida, como se puede apreciar en las líneas 39–40 del script 10.1. Es importante destacar que la única diferencia con respecto a la llamada realizada en el capítulo 9 es que ahora **el argumento** `between` empleado en el capítulo 9 ha sido **reemplazado por** `within` para la variable independiente. Esta diferencia indica a `ezANOVA()` que se trata de un procedimiento ANOVA con muestras correlacionadas. La figura 10.3 muestra, una vez más, que se obtiene el mismo resultado.

Script 10.1: procedimiento ANOVA de una vía para muestras correlacionadas.

```r
library(tidyverse)
library(ggpubr)
library(ez)

# Crear el data frame.
instancia <- factor(1:6)
Quicksort <- c(23.2, 22.6, 23.4, 23.3, 21.8, 23.9)
Bubblesort <- c(31.6, 29.3, 30.7, 30.8, 29.8, 30.3)
Radixsort <- c(30.1, 28.4, 28.7, 28.3, 29.9, 29.1)
Mergesort <- c(25.0, 25.7, 25.7, 23.7, 25.5, 24.7)
datos <- data.frame(instancia, Quicksort, Bubblesort, Radixsort, Mergesort)

# Llevar data frame a formato largo.
datos <- datos %>% pivot_longer(c("Quicksort", "Bubblesort", "Radixsort",
                                  "Mergesort"),
                                names_to = "algoritmo", values_to = "tiempo")

datos[["algoritmo"]] <- factor(datos[["algoritmo"]])

# Comprobción de normalidad.
g <- ggqqplot(datos, x = "tiempo", y = "algoritmo", color = "algoritmo")
g <- g + facet_wrap(~ algoritmo)
g <- g + rremove("x.ticks") + rremove("x.text")
g <- g + rremove("y.ticks") + rremove("y.text")
g <- g + rremove("axis.title")
print(g)

# Procedimiento ANOVA con aov.
cat("Procedimiento ANOVA usando aov\n\n")

prueba <- aov(tiempo ~ algoritmo + Error(instancia/(algoritmo)),
              data = datos)

print(summary(prueba))

# Procedimiento ANOVA con ezANOVA().
cat("\n\nProcedimiento ANOVA usando ezANOVA\n\n")

prueba2 <- ezANOVA(data = datos, dv = tiempo, within = algoritmo,
                   wid = instancia, return_aov = TRUE)

print(summary(prueba2$aov))
cat("\n\nPero ezANOVA entrega más información.\n")
cat("El resultado de la prueba de esfericidad de Mauchly:\n\n")
print(prueba2[["Mauchly's Test for Sphericity"]])

cat("\n\nY factores de corrección para cuando no se cumple la\n")
cat("condición de esfericidad:\n\n")
print(prueba2$`Sphericity Corrections`)

# Gráfico del tamaño del efecto.
g2 <- ezPlot(data = datos, dv = tiempo, wid = instancia, within = algoritmo,
             y_lab = "Tiempo promedio de ejecución [ms]", x = algoritmo)

print(g2)
```

```
Resultado de la prueba ANOVA para muestras correlacionadas con ezANOVA

Error: Instancia
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  5  2.857  0.5714

Error: Instancia:Algoritmo
          Df Sum Sq Mean Sq F value   Pr(>F)
Algoritmo  3 213.04   71.01     118 1.18e-10 ***
Residuals 15   9.03    0.60
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 10.3: resultado del procedimiento ANOVA usando la función `ezANOVA()`.

Habíamos mencionado que otra ventaja de `ezANOVA()` es que verifica la condición de esfericidad mediante la prueba de esfericidad de Mauchly, cuyo resultado se muestra en la figura 10.4. Podemos apreciar que el valor p obtenido en esta prueba es muy alto ($p = 0,555$), de lo que se desprende que los datos del ejemplo sí cumplen con la condición de esfericidad (hipótesis nula de la prueba de Mauchly).

```
Resultado de la prueba de esfericidad de Mauchly

     Effect         W         p p<.05
2 Algoritmo 0.3367911 0.5545469
```

Figura 10.4: resultado de la prueba de esfericidad de Mauchly realizada por `ezANOVA()`.

Ahora bien, existen dos correcciones que suelen emplearse cuando se producen violaciones a la condición de esfericidad: la de **Greenhouse-Geisser** y la de **Huynd-Feldt**. Ambas estiman la esfericidad, denotada por ϵ, y corrigen el valor p de ANOVA en base a dicha estimación (ajustando los grados de libertad de la distribución F usada en el cálculo). La corrección de Greenhouse-Geisser es más conservadora y tiende a subestimar ϵ cuando esta es cercana a 1, por lo que se recomienda su uso para $\epsilon < 0,75$. Para $\epsilon \geq 0,75$ (estimada con el método Greenhouse-Geisser, de acuerdo a Karadimitriou y Marshall (2016)) suele emplearse la estimación de Huynd-Feldt, algo más liberal (Lærd Statistics, 2020b). `ezANOVA()` lleva a cabo ambas correcciones y reporta para cada una de ellas tanto la estimación de la esfericidad como el valor p corregido, como se aprecia en la figura 10.5:

- GGe: estimación de ϵ con el método de Greenhouse-Geisser.
- p[gg]: valor p tras la corrección de Greenhouse-Geisser.
- HFe: estimación de ϵ con el método de Huynd-Feldt.
- p[HF]: valor p tras la corrección de Huynd-Feldt.

```
Factores de corrección para cuando no se cumple la condición de esfericidad

     Effect       GGe        p[GG] p[GG]<.05      HFe        p[HF] p[HF]<.05
2 Algoritmo 0.6803135 8.377723e-08         * 1.154155 1.177725e-10         *
```

Figura 10.5: correcciones de esfericidad realizadas por `ezANOVA()`.

Si los datos del ejemplo no cumplieran con la esfericidad, deberíamos considerar `p[GG]` como p valor de la prueba, y no el valor (sin corregir) de la tabla entregada por `ezANOVA()` de la figura 10.3. Una vez más, podemos graficar el tamaño del efecto medido (script 10.1, líneas 52–55), obteniéndose como resultado el gráfico de la figura 10.6.

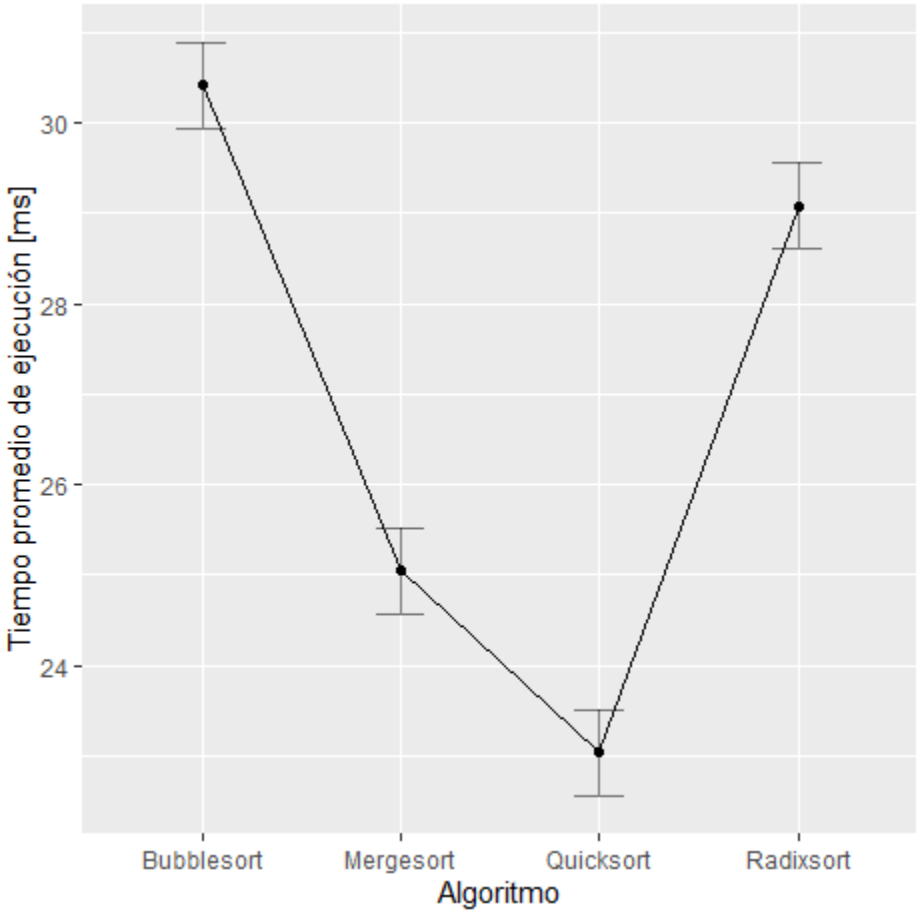


Figura 10.6: Tamaño del efecto medido.

10.4 PROCEDIMIENTOS POST-HOC

Podemos ocupar los mismos procedimientos post-hoc estudiados en el capítulo 9 tras realizar un procedimiento ANOVA de una vía con muestras correlacionadas.

En el caso de las correcciones de Bonferroni y Holm, lo único que cambia es que ahora debemos asignar el valor `TRUE` al argumento `paired` de la función `pairwise.t.test()`.

En cuanto a las pruebas HSD de Tukey y de Scheffé, su realización se dificulta al usar la tabla ANOVA resultante de un proceso de una vía para muestras correlacionadas. Esto se debe a que el formato del objeto `aov` resultante difiere al que se obtiene al realizar un procedimiento ANOVA para muestras independientes, por lo que las funciones del paquete `DescTools` arrojan un error. No obstante, existe una alternativa. El primer paso consiste en crear un modelo mixto (concepto que va más allá de los alcances de este documento) mediante la función `lme(formula, data, random)` del paquete `nlme`, donde:

- `formula`: tiene la forma <variable dependiente>∼<variable categórica>.
- `data`: matriz de datos en formato largo.
- `random`: fórmula de la forma ∼1|<identificador del sujeto>.

Como segundo paso, estimamos la media de la variable dependiente, con su respectivo intervalo de confianza, para cada nivel de la variable categórica. Para esto usamos la función `emmeans(object, specs)` del paquete homónimo, donde:

- `object`: modelo mixto construido en el paso previo.
- `specs`: nombre del factor en estudio, delimitado por comillas.

Por último, estimamos las medias de las diferencias para los contrastes entre pares, con su error estándar y los valores p correspondientes, mediante la función `pairs(x, adjust)`, donde:

- `x`: diferencias obtenidas en el párrafo precedente.
- `adjust`: método para ajustar los valores p. "tukey" para el método HSD de Tukey, "scheffe" para el método de Scheffé.

Los mecanismos para estimar las medias marginales (emmeans) y para construir otros contrastes con el método de Scheffé van más allá de los alcances de este libro, pero pueden encontrarse en la documentación de los paquetes R involucrados.

El script 10.2 efectúa las pruebas post-hoc para el ejemplo, obteniéndose los resultados de la figura 10.7. Considerando el ajuste para múltiples pruebas de Tukey, podemos concluir con 99 % de confianza que todos los algoritmos tienen tiempos de ejecución distintos, con la excepción del par *bubblesort/radixsort*, que presentan el mismo tiempo de ejecución medio.

Script 10.2: pruebas post-hoc para el ejemplo.

```
library(tidyverse)
library(nlme)
library(emmeans)
library(ez)

# Crear el data frame.
instancia <- factor(1:6)
Quicksort <- c(23.2, 22.6, 23.4, 23.3, 21.8, 23.9)
Bubblesort <- c(31.6, 29.3, 30.7, 30.8, 29.8, 30.3)
Radixsort <- c(30.1, 28.4, 28.7, 28.3, 29.9, 29.1)
Mergesort <- c(25.0, 25.7, 25.7, 23.7, 25.5, 24.7)
datos <- data.frame(instancia, Quicksort, Bubblesort, Radixsort, Mergesort)

# Llevar data frame a formato largo.
datos <- datos %>% pivot_longer(c("Quicksort", "Bubblesort", "Radixsort",
                                  "Mergesort"),
                                names_to = "algoritmo", values_to = "tiempo")

datos[["algoritmo"]] <- factor(datos[["algoritmo"]])

# Nivel de significación.
alfa <- 0.01

# Procedimiento ANOVA.
anova <- ezANOVA(data = datos, dv = tiempo, within = algoritmo,
                 wid = instancia, return_aov = TRUE)

# Procedimiento post-hoc de Bonferroni.
bonferroni <- pairwise.t.test(datos[["tiempo"]], datos[["algoritmo"]],
                              p.adj = "bonferroni", paired = TRUE)

cat("Corrección de Bonferroni\n")
print(bonferroni)

# Procedimiento post-hoc de Holm.
holm <- pairwise.t.test(datos[["tiempo"]], datos[["algoritmo"]],
                        p.adj = "holm", paired = TRUE)

cat("\n\nCorrección de Holm\n")
```

```r
print(holm)

# Procedimiento post-hoc HSD de Tukey.
mixto <- lme(tiempo ~ algoritmo, data = datos, random = ~1|instancia)
medias <- emmeans(mixto, "algoritmo")
tukey <- pairs(medias, adjust = "tukey")

cat("\n\nPrueba HSD de Tukey\n\n")
print(tukey)

# Procedimiento post-hoc de Scheffé
cat("\n\nComparación de Scheffé\n")
scheffe <- pairs(medias, adjust = "scheffe")
print(scheffe)
```

10.5 EJERCICIOS PROPUESTOS

1. ¿Cuándo aplica el procedimiento ANOVA de una vía para muestras correlacionadas? ¿Con qué otros nombres se encuentra?
2. Explica a qué se refiere "eliminar la variabilidad de los sujetos".
3. ¿Cuáles son las hipótesis contrastadas en el procedimiento ANOVA de una vía para muestras correlacionadas?
4. Enumera las condiciones (o supuestos) del procedimiento ANOVA de una vía para muestras correlacionadas para tener confiabilidad.
5. El conjunto de datos `ChickWeight` registra el peso (en gramos) de 50 pollitos al momento de nacer y al cabo de varios días después de nacidos. Identifica si existen diferencias significativas en el peso de los pollitos al momento de nacer, al cabo de 4 días y luego de 8 días. En caso de detectarse tales diferencias, indica cuáles son.

```
            Corrección de Bonferroni

Pairwise comparisons using paired t tests

data:  Tiempo and Algoritmo

          Bubblesort Mergesort Quicksort
Mergesort 0.00112    -         -
Quicksort 1.4e-05    0.07196   -
Radixsort 0.09232    0.00088   0.00039

P value adjustment method: bonferroni

Corrección de Holm

Pairwise comparisons using paired t tests

data:  Tiempo and Algoritmo

          Bubblesort Mergesort Quicksort
Mergesort 0.00059    -         -
Quicksort 1.4e-05    0.02399   -
Radixsort 0.02399    0.00059   0.00033

P value adjustment method: holm

Prueba HSD de Tukey

 contrast               estimate    SE df t.ratio p.value
 Bubblesort - Mergesort     5.37 0.445 15  12.058 <.0001
 Bubblesort - Quicksort     7.38 0.445 15  16.589 <.0001
 Bubblesort - Radixsort     1.33 0.445 15   2.996 0.0403
 Mergesort - Quicksort      2.02 0.445 15   4.531 0.0020
 Mergesort - Radixsort     -4.03 0.445 15  -9.062 <.0001
 Quicksort - Radixsort     -6.05 0.445 15 -13.594 <.0001

Degrees-of-freedom method: containment
P value adjustment: tukey method for comparing a family of 4 estimates

Comparación de Scheffé
 contrast               estimate    SE df t.ratio p.value
 Bubblesort - Mergesort     5.37 0.445 15  12.058 <.0001
 Bubblesort - Quicksort     7.38 0.445 15  16.589 <.0001
 Bubblesort - Radixsort     1.33 0.445 15   2.996 0.0642
 Mergesort - Quicksort      2.02 0.445 15   4.531 0.0040
 Mergesort - Radixsort     -4.03 0.445 15  -9.062 <.0001
 Quicksort - Radixsort     -6.05 0.445 15 -13.594 <.0001

Degrees-of-freedom method: containment
P value adjustment: scheffe method with rank 3
```

Figura 10.7: resultados de las pruebas post-hoc para el ejemplo.

CAPÍTULO 11. INFERENCIA NO PARAMÉTRICA CON MEDIANAS

En el capítulo 8 conocimos algunos métodos no paramétricos que podemos usar para inferir sobre frecuencias cuando nuestro conjunto de datos no cumple con las condiciones para poder usar, por ejemplo, las pruebas de Wald o de Wilson. Mencionamos también que este problema también puede ocurrir para el caso de inferir con medias, por lo que en este capítulo conoceremos alternativas no paramétricas para las pruebas t de Student (para una y dos medias) y ANOVA (para más de dos medias). Para ello nos basaremos principalmente en Lowry (1999, caps. 11a, 12a, 14a, 15a), Glen (2021b) y Lærd Statistics (2020a).

11.1 PRUEBAS PARA UNA O DOS MUESTRAS

En el capítulo 5 aprendimos que la prueba t de Student es adecuada para inferir acerca de una o dos medias muestrales, siempre y cuando se verifiquen algunas condiciones. En el caso de la prueba t de una muestra (o de la diferencia de dos muestras pareadas):

1. Las observaciones son independientes entre sí.
2. Las observaciones provienen de una distribución cercana a la normal.

En el caso de dos muestras independientes:

1. Cada muestra cumple las condiciones para usar la distribución t.
2. Las muestras son independientes entre sí.

Es importante mencionar también que la distribución normal es continua, de donde se desprende que la escala de medición empleada para la medición de las muestras debe ser de intervalos iguales.

Como ya vimos en el capítulo 8, si usamos la prueba t en un escenario en que no se cumple alguna de estas condiciones, el resultado no sería válido pues carecería de sentido y, en consecuencia, también lo harían las conclusiones que se obtengan a partir de él.

11.1.1 Prueba de suma de rangos de Wilcoxon

La **prueba de suma de rangos de Wilcoxon**, también llamada **prueba U de Mann-Whitney** o **prueba de Wilcoxon-Mann-Whitney**, es una alternativa no paramétrica a la prueba t de Student con muestras independientes. Pese a ser no paramétrica, requiere verificar el cumplimiento de las siguientes condiciones:

1. Las observaciones de ambas muestras son independientes.
2. La escala de medición empleada debe ser a lo menos ordinal, de modo que tenga sentido hablar de relaciones de orden (“igual que”, “menor que”, “mayor o igual que”).

Consideremos el siguiente contexto para estudiar la aplicación de esta prueba: una empresa de desarrollo de software desea evaluar la usabilidad de dos interfaces alternativas, A y B, para un nuevo producto de software. Con este fin, la empresa ha seleccionado al azar a 23 voluntarias y voluntarios, quienes son asignados de manera

aleatoria a dos grupos, cada uno de los cuales debe probar una de las interfaces ($n_A = 12$, $n_B = 11$). Cada participante debe evaluar 6 aspectos de usabilidad de la interfaz, cada uno de los cuales se mide con una escala Likert de 7 puntos, donde 1 significa "muy malo" y 7, "muy bueno". La valoración que cada participante da a la interfaz evaluada corresponde al promedio simple de las puntuaciones de los 6 aspectos evaluados. La tabla 11.1 muestra las evaluaciones realizadas por cada participante.

	Interfaz A	**Interfaz B**
	2,7	5,0
	6,6	1,4
	1,6	5,6
	5,1	4,6
	3,7	6,7
	6,1	2,7
	5,0	1,3
	1,4	6,3
	1,8	3,7
	1,5	1,3
	3,0	6,8
	5,3	
Media	3,65	4,13

Tabla 11.1: evaluación de las interfaces de usuario A y B.

En este caso, si bien se cumple la condición de independencia de la prueba t de Student, no podemos usar esta prueba por dos razones: primero, no todas las escalas Likert pueden asegurar que son de igual intervalo. En el ejemplo, si dos participantes califican un aspecto de la interfaz A con notas 3 y 5, mientras que dos participantes califican esos aspectos con notas 4 y 6 para la interfaz B, ¿se podría asegurar que en ambos casos existe la misma diferencia de usabilidad (2 puntos)? Pocas escalas Likert tienen estudios de reproducibilidad que aseguren esta consistencia, por lo que no podríamos asumir que la escala es de intervalos iguales en este ejemplo. En segundo lugar, al revisar los histogramas para las muestras (figura 11.1) podemos observar que las distribuciones no se asemejan a una normal.

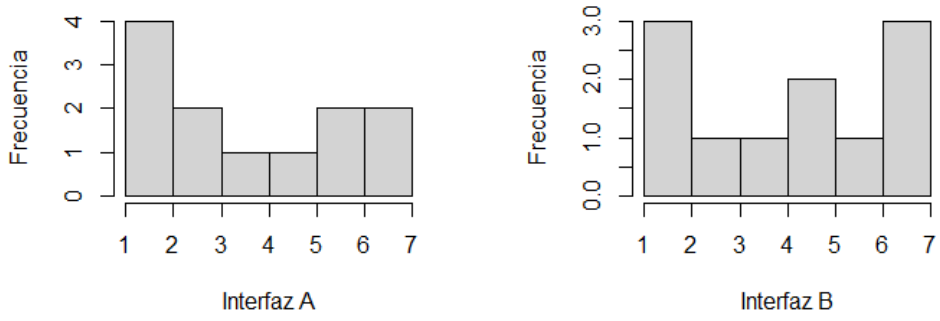


Figura 11.1: histogramas de las muestras.

Como alternativa, podemos usar la prueba no paramétrica de Wilcoxon-Mann-Whitney, cuyas hipótesis para el ejemplo son:

H_0: no hay diferencia en la usabilidad de ambas interfaces (se distribuyen de igual forma).
H_A: sí hay diferencia en la usabilidad de ambas interfaces (distribuciones distintas).

Al igual que en el caso de la prueba χ^2 de Pearson, estas hipótesis no hacen referencia a algún parámetro de

una supuesta distribución para las poblaciones de puntuaciones de usabilidad, es decir, nos entregan menos información que la prueba paramétrica equivalente.

El primer paso de la prueba consiste en combinar todas las observaciones en un único conjunto de tamaño $n_T = n_A + n_B$ y ordenarlo de menor a mayor. A cada elemento se le asigna un **valor de rango** (*rank* en inglés) de 1 a n_T, de acuerdo a la posición que ocupa en el conjunto ordenado. En caso de que un valor aparezca más de una vez, cada repetición toma como valor el rango promedio de todas las ocurrencias del valor. La tabla 11.2 muestra el resultado de este proceso. Podemos notar que hay dos observaciones con valor $1,3$ a las que le corresponderían los rangos 1 y 2, por lo que, en consecuencia, ambas reciben el mismo valor de rango, igual al promedio $1,5$. Esto también ocurre para las puntuaciones $1,4$; $2,7$; $3,7$ y $5,0$.

Observación	**Muestra**	**Rango**
1,3	B	1,5
1,3	B	1,5
1,4	A	3,5
1,4	B	3,5
1,5	A	5,0
1,6	A	6,0
1,8	A	7,0
2,7	A	8,5
2,7	B	8,5
3,0	A	10,0
3,7	A	11,5
3,7	B	11,5
4,6	B	13,0
5,0	A	14,5
5,0	B	14,5
5,1	A	16,0
5,3	A	17,0
5,6	B	18,0
6,1	A	19,0
6,3	B	20,0
6,6	A	21,0
6,7	B	22,0
6,8	B	23,0

Tabla 11.2: muestras combinadas con rango.

A continuación, se suman los rangos asociados a las observaciones de cada muestra, y para la muestra combinada. Así, para la muestra A obtenemos:

$$T_A = 3,5 + 5,0 + 6,0 + 7,0 + 8,5 + 10,0 + 11,5 + 14,5 + 16,0 + 17,0 + 19,0 + 21,0 = 139$$

De manera análoga, para la muestra B se tiene:

$$T_B = 1,5 + 1,5 + 3,5 + 8,5 + 11,5 + 13,0 + 14,5 + 18,0 + 20,0 + 22,0 + 23,0 = 137$$

La suma de rangos para la muestra combinada siempre está dada por la ecuación 11.1.

$$T_T = \frac{n_T \cdot (n_T + 1)}{2} \tag{11.1}$$

Para el ejemplo:

$$T_T = \frac{23 \cdot (23+1)}{2} = 276$$

Trabajar con los rangos en lugar de las observaciones nos ofrece dos ventajas: la primera es que el foco solo está en las relaciones de orden entre las observaciones, sin necesidad de que estas provengan de una escala de intervalos iguales. La segunda es que esta transformación facilita conocer de manera sencilla algunas propiedades del conjunto de datos. Por ejemplo, la suma de rangos de la muestra se determina siempre mediante la ecuación 11.1 y la media de rangos de la muestra combinada es siempre como muestra la ecuación 11.2.

$$\mu = \frac{n_T \cdot (n_T+1)}{2} \cdot \frac{1}{n_T} = \frac{n_T+1}{2} \tag{11.2}$$

Para el ejemplo:

$$\mu = \frac{23+1}{2} = 12$$

En consecuencia, la hipótesis nula en el dominio de los rangos es que las medias de los rangos de las dos muestras son iguales. Si la hipótesis nula fuera cierta, las observaciones en ambas muestras serían similares, por lo que, al ordenar la muestra combinada, ambas muestras se mezclarían de manera homogénea. En consecuencia, deberíamos esperar que los promedios de rangos para cada muestra se aproximen al rango promedio de la muestra combinada, es decir, que T_A y T_B se aproximen a los siguientes valores:

$$\mu_A = n_A \cdot \frac{(n_T)+1}{2} = 12 \cdot \frac{(23+1)}{2} = 144$$
$$\mu_B = n_B \cdot \frac{(n_T)+1}{2} = 11 \cdot \frac{(23+1)}{2} = 132$$

La prueba de Wilcoxon-Mann-Whitney tiene dos variantes, una para muestras grandes y otra para muestras pequeñas, que se diferencian a partir de este punto.

11.1.1.1 Prueba de suma de rangos de Wilcoxon para muestras grandes

Hasta ahora, hemos determinado que:

- El valor observado $T_A = 139$ pertenece a una distribución muestral con media $\mu_A = 144$.
- El valor observado $T_B = 137$ pertenece a una distribución muestral con media $\mu_B = 132$.

Bajo el supuesto de que la hipótesis nula sea verdadera, podríamos demostrar que las distribuciones muestrales de T_A y T_B tienen la misma desviación estándar, dada por la ecuación 11.3.

$$\sigma_T = \sqrt{\frac{n_A \cdot n_B \cdot (n_T+1)}{12}} \tag{11.3}$$

Con lo que:

$$\sigma_T = \sqrt{\frac{12 \cdot 11 \cdot (23+1)}{12}} = 16,248$$

Cuando **ambas muestras tienen tamaño mayor o igual a 5**, siguiendo un procedimiento similar al descrito en la primera sección del capítulo 4, podemos demostrar que las distribuciones muestrales de T_A y T_B tienden a aproximarse a la distribución normal. En consecuencia, una vez conocidas la media y la desviación estándar de una distribución normal para la muestra, podemos calcular el estadístico z para T_A o T_B, dado por la ecuación 11.4, donde:

- T_{obs} es cualquiera de los valores observados, T_A o T_B.
- μ_{obs} es la media de la distribución muestral de T_{obs}.
- σ_T es la desviación estándar de la distribución muestral de T_{obs} (es decir, el error estándar).

$$z = \frac{(T_{obs} - \mu_{obs}) \pm 0,5}{\sigma_T} \tag{11.4}$$

Puesto que las distribuciones muestrales de T son intrínsecamente discretas (solo pueden asumir valores con decimales cuando existen rangos empatados), debemos emplear un factor de corrección de continuidad:

- $-0,5$ si $T_{obs} > \mu_{obs}$.
- $0,5$ si $T_{obs} < \mu_{obs}$.

Volviendo al ejemplo, tenemos:

$$z_A = \frac{(139 - 144) + 0,5}{16,248} = -0,277$$
$$z_B = \frac{(137 - 132) - 0,5}{\sigma_T} = 0,277$$

Los valores z obtenidos a partir de T_A y T_B siempre tienen igual valor absoluto y signos opuestos, por lo que no importa cuál de ellos usemos para la prueba de significación estadística. No obstante, debemos tener muy claro el significado del signo de z: si para el ejemplo tuviésemos como hipótesis alternativa que la interfaz A es mejor que la interfaz B, entonces esperaríamos que las observaciones de mayor rango estuvieran en el grupo A, por lo que z_A tendría que ser positivo.

El valor z obtenido permite calcular el valor p para una hipótesis alternativa unilateral (pues solo delimita la región de rechazo en una de las colas de la distribución normal estándar subyacente). Así, para el ejemplo, cuya hipótesis alternativa es bilateral, podemos calcular el valor p en R mediante la llamada `2 * pnorm(-0.277, mean = 0, sd = 1, lower.tail = TRUE)`, obteniéndose como resultado $p = 0,782$.

Evidentemente, el valor p obtenido es muy alto, por lo que fallamos al rechazar la hipótesis nula. En consecuencia, podemos concluir que no hay diferencia significativa en la usabilidad de las dos interfaces.

11.1.1.2 Prueba de suma de rangos de Wilcoxon para muestras pequeñas

Cuando las muestras son pequeñas (menos de 5 observaciones), no podemos usar el supuesto de normalidad del apartado anterior, por lo que necesitamos una vía alternativa. Este método sirve también para muestras más grandes, con resultados equivalentes a los ya obtenidos.

Aprovechando una vez más las ventajas de considerar los rangos en lugar de las observaciones originales, podemos calcular el máximo valor posible para la suma de rangos de cada muestra como indica la ecuación 11.5. Fijémonos en que el valor máximo para la suma de rangos de una muestra se produce cuando esta contiene los n_x rangos mayores de la muestra combinada.

$$T_{x[max]} = n_x \cdot n_y + \frac{n_x \cdot (n_x + 1)}{2} \tag{11.5}$$

Así, para el ejemplo:

$$T_{A[max]} = 12 \cdot 11 + \frac{12 \cdot (12 + 1)}{2} = 210$$
$$T_{B[max]} = 11 \cdot 12 + \frac{11 \cdot (11 + 1)}{2} = 198$$

Con esto podemos definir un nuevo estadístico de prueba U, como muestra la ecuación 11.6.

$$U_x = T_{x[max]} - T_x \tag{11.6}$$

Por lo que:

$$U_A = 210 - 139 = 71$$
$$U_B = 198 - 137 = 61$$

El valor del estadístico de prueba es el mínimo entre U_A y U_B, por lo que $U = 61$.

Debemos notar que siempre se cumple la identidad presentada en la ecuación 11.7, por lo que podemos escoger cualquiera de los valores U obtenidos para realizar el resto del procedimiento.

$$U_A + U_B = n_A \cdot n_B \tag{11.7}$$

Si la hipótesis nula fuese cierta, esperaríamos que:

$$U_A = T_{A[max]} - \mu_A = 210 - 144 = 66$$
$$U_B = T_{B[max]} - \mu_B = 198 - 132 = 66$$

Formalmente, entonces, si la hipótesis nula fuera verdadera, esperaríamos que:

$$U_A = U_B = \frac{n_A \cdot n_B}{2}$$

En consecuencia, la pregunta asociada a la prueba de hipótesis es: si la hipótesis nula es verdadera (no hay diferencias significativas en la usabilidad de ambas interfaces), ¿qué tan probable es obtener un valor de U al menos tan pequeño como el observado ($U = 61$)? Para responder a esta pregunta, seguimos un procedimiento similar al que ya conocimos para la prueba exacta de Fisher (capítulo 8): se calculan todas las formas en que n_T rangos podrían combinarse en dos grupos de tamaños n_A y n_B, y luego se determina la proporción de las combinaciones que produzcan un valor de U al menos tan pequeño como el encontrado. Pero ¡existen 676.039 combinaciones posibles!

Aunque R no ofrece herramientas para calcular el valor p a partir del estadístico U (pues utiliza el estadístico W, propuesto por Frank Wilcoxon en 1945, que lleva a los mismos resultados), afortunadamente existen tablas que permiten conocer el máximo valor de U para el cual se rechaza la hipótesis nula para un nivel de significación dado sin tener que revisar todas las combinaciones. Considerando $\alpha = 0,05$ para una prueba bilateral, el valor crítico es $U = 33$ (Real Statistics Using Excel, s.f.). Puesto que $61 > 33$, fallamos al rechazar la hipótesis nula, por lo que concluimos con 95 % de confianza que no existe una diferencia estadísticamente significativa en la usabilidad de ambas interfaces.

11.1.1.3 Prueba de suma de rangos de Wilcoxon en R

Como ya dijimos, la implementación de esta prueba en R usa el estadístico W (introducido por Wilcoxon) en lugar del estadístico U empleado por Mann y Whitney. Es por ello que que esta prueba se realiza mediante la función `wilcox.test(x, y, paired = FALSE, alternative, mu, conf.level)`, donde:

- `x, y`: vectores numéricos con las observaciones. Para aplicar la prueba con una única muestra, `y` debe ser nulo (por defecto, lo es).
- `paired`: booleano con valor falso para indicar que las muestras son independientes (se asume por defecto).
- `alternative`: señala el tipo de hipótesis alternativa: bilateral ("two.sided") o unilateral ("less" o "greater").
- `mu`: valor nulo para la media (solo para la prueba con una muestra).
- `conf.level`: nivel de confianza.

El script 11.1 muestra la aplicación de esta prueba para el ejemplo, obteniéndose los resultados que se presentan en la figura 11.2.

```
Wilcoxon rank sum test with continuity correction

data:  a and b
W = 61, p-value = 0.7816
alternative hypothesis: true location shift is not equal to 0
```

Figura 11.2: resultado de la prueba de Mann-Whitney (en rigor, de la prueba para el ejemplo.

Script 11.1: prueba de Mann-Whitney para el ejemplo.

```
# Ingresar los datos.
a <- c(2.7, 6.6, 1.6, 5.1, 3.7, 6.1, 5.0, 1.4, 1.8, 1.5, 3.0, 5.3)
b <- c(5.0, 1.4, 5.6, 4.6, 6.7, 2.7, 1.3, 6.3, 3.7, 1.3, 6.8)

# Establecer nivel de significación.
alfa <- 0.05

# Hacer la prueba de Mann-Whitney.
prueba <- wilcox.test(a, b, alternative = "two.sided", conf.level = 1 - alfa)
print(prueba)
```

11.1.2 Prueba de rangos con signo de Wilcoxon

La **prueba de rangos con signo de Wilcoxon** es, conceptualmente, parecida a la prueba de suma de rangos de Wilcoxon presentada en la sección anterior. Sin embargo, en este caso es la alternativa no paramétrica a la prueba t de Student con muestras pareadas. Las condiciones que se deben cumplir para usar esta prueba son:

1. Los pares de observaciones son independientes.
2. La escala de medición empleada para las observaciones es intrínsecamente continua.
3. La escala de medición empleada para ambas muestras debe ser a lo menos ordinal.

Consideremos ahora un nuevo contexto para la aplicación de esta prueba. Una empresa de desarrollo desea evaluar la usabilidad de dos interfaces alternativas, A y B, para un nuevo producto de software, a fin de

determinar si, como asegura el departamento de diseño, es mejor la interfaz A. Para ello, la empresa ha seleccionado a 10 participantes al azar, quienes deben evaluar 6 aspectos de usabilidad de cada interfaz, cada uno de los cuales se mide con una escala Likert de 7 puntos, donde 1 significa "muy malo" y 7, "muy bueno". La valoración que un participante da a la interfaz evaluada corresponde al promedio simple de las puntuaciones de los 6 aspectos evaluados. La tabla 11.3 muestra las evaluaciones realizadas por cada participante a cada una de las interfaces.

Participante	**Interfaz A**	**Interfaz B**
1	2,9	6,0
2	6,1	2,8
3	6,7	1,3
4	4,7	4,7
5	6,4	3,1
6	5,7	1,8
7	2,7	2,9
8	6,9	4,0
9	1,7	2,3
10	6,4	1,6

Tabla 11.3: evaluación de las interfaces de usuario A y B.

Formalmente, las hipótesis son:

H_0: las mismas personas no perciben diferencia en la usabilidad de ambas interfaces.
H_A: las mismas personas consideran que la interfaz A tiene mejor usabilidad que la interfaz B.

La mecánica inicial para esta prueba consiste en calcular las diferencias entre cada par de observaciones y obtener luego su valor absoluto. Generalmente se descartan aquellas instancias con diferencia igual a cero, pues no aportan información relevante al procedimiento. A continuación se ordenan las diferencias absolutas en orden creciente y se les asignan rangos de manera correlativa del mismo modo que en la prueba de Wilcoxon-Mann-Whitney. Una vez asignados los rangos, se les incorpora el signo asociado a la diferencia. La tabla 11.4 ilustra el proceso descrito.

Participante	**Interfaz A**	**Interfaz B**	**A−B**	**|A−B|**	**Rango absoluto**	**Rango con signo**
4	4,7	4,7	0,0	0	-	-
7	2,7	2,9	-0,2	0,2	1	-1
9	1,7	2,3	-0,6	0,6	2	-2
8	6,9	4,0	2,9	2,9	3	+3
1	2,9	6,0	-3,1	3,1	4	-4
2	6,1	2,8	3,3	3,3	5,5	+5,5
5	6,4	3,1	3,3	3,3	5,5	+5,5
6	5,7	1,8	3,9	3,9	7	+7
10	6,4	1,6	4,8	4,8	8	+8
3	6,7	1,3	5,4	5,4	9	+9

Tabla 11.4: asignación de rangos con signo.

Una vez realizado este proceso, se calcula el estadístico de prueba W, correspondiente a la suma de los rangos con signo. Debemos notar que, tras eliminar aquellas observaciones con diferencia igual a 0, el tamaño de las muestras para el ejemplo es $n = 9$. Así:

$$W = -1 + -2 + 3 + -4 + 5,5 + 5,5 + 7 + 8 + 9 = 31$$

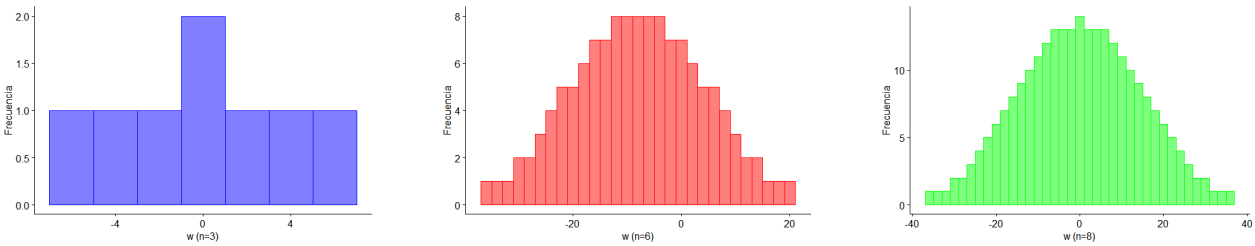
Desde luego, el máximo valor posible para W, W_{max} corresponde a la suma de los n rangos sin signo (todos positivos) (ecuación 11.2) y, además, $W_{min} = -|W_{max}|$.

Para entender mejor la distribución de W, una muestra de tamaño n genera n rangos no empatados sin signo (columna "Rango absoluto" de la tabla 11.4). A su vez, cada uno de dichos rangos podría tomar valores positivos o negativos, por lo que para W se tienen 2^n combinaciones para los signos de los rangos. La tabla 11.5 muestra todas las posibles combinaciones para $n = 3$. Si la hipótesis nula fuese cierta, los rangos positivos y negativos se distribuirían de manera homogénea, por lo que esperaríamos que el valor de W fuese cercano a 0 (hipótesis nula en el dominio de los rangos).

La figura 11.3 muestra la distribución de W para distintos valores de n. En ella podemos apreciar que, a medida que n aumenta, la distribución de W se aproxima cada vez más a una distribución normal centrada en $\mu_W = 0$.

Rango			
1	**2**	**3**	W
+	+	+	6
+	+	-	0
+	-	+	2
+	-	-	-4
-	+	+	4
-	+	-	-2
-	-	+	0
-	-	-	-6

Tabla 11.5: valores que puede adoptar el estadístico W para $n = 3$.



(a) $n = 3$ (b) $n = 6$ (c) $n = 8$

Figura 11.3: distribución de W.

La desviación estándar de la distribución muestral de W está dada por la ecuación 11.8.

$$\sigma_W = \sqrt{\frac{n \cdot (n+1) \cdot (2n+1)}{6}} \tag{11.8}$$

Para el ejemplo:

$$\sigma_W = \sqrt{\frac{9 \cdot (9+1) \cdot (2 \cdot 9+1)}{6}} = 16,882$$

Puesto que estamos trabajando bajo el supuesto de normalidad, calculamos el estadístico de prueba z, dado por la ecuación 11.9.

$$z = \frac{W - 0,5}{\sigma_W} \tag{11.9}$$

Así, para el ejemplo tenemos que:

$$z = \frac{31 - 0,5}{16,882} = 1,807$$

Una vez conocido el estadístico de prueba, podemos obtener el valor p mediante la llamada `pnorm(1.807, mean = 0, sd = 1, lower.tail = FALSE)` (no multiplicamos por 2, pues es una prueba unilateral), obteniendo como resultado $p = 0,035$. Considerando un nivel de significación $\alpha = 0,05$, rechazamos la hipótesis nula en favor de la hipótesis alternativa. En consecuencia, concluimos con 95 % de confianza que la usabilidad de la interfaz A es mejor que la de la interfaz B.

Siempre debemos tener en cuenta que el supuesto de normalidad es válido únicamente para $n > 10$, por lo que en caso de que las muestras sean más pequeñas, tenemos que consultar la tabla de valores críticos para la distribución W.

En R, la prueba de rangos con signo de Wilcoxon está implementada en la misma función que en el caso de muestras independientes, pero ahora la llamada es `wilcox.test(x, y, paired = TRUE, alternative, conf.level)`, donde:

- `x, y`: vectores numéricos con las observaciones.
- `paired`: booleano con valor verdadero para indicar que las muestras son pareadas.
- `alternative`: señala el tipo de hipótesis alternativa: bilateral ("two.sided") o unilateral ("less" o "greater").
- `paired`: indica si las muestras están o no pareadas.
- `conf.level`: nivel de confianza.

Así, el valor por defecto para el parámetro `paired` es `FALSE`, en cuyo caso se efectúa la prueba de suma de rangos de Wilcoxon; mientras que si explícitamente indicamos `paired = TRUE`, se aplica la prueba de rangos con signo de Wilcoxon.

El script 11.2 muestra la aplicación de la prueba de rangos con signo de Wilcoxon para el ejemplo, obteniéndose los resultados que se presentan en la figura 11.4. Es importante tener en cuenta en cuenta que R usa una variante ligeramente diferente. En lugar del estadístico de prueba W, calcula el estadístico V, correspondiente a la suma de los rangos con signo positivo.

```
Wilcoxon signed rank test with continuity correction

data:  a and b
V = 38, p-value = 0.03778
alternative hypothesis: true location shift is greater than 0
```

Figura 11.4: resultado de la prueba de rangos con signo de Wilcoxon para el ejemplo.

Script 11.2: prueba de rangos con signo de Wilcoxon para el ejemplo.

```
# Ingresar los datos.
a <- c(2.9, 6.1, 6.7, 4.7, 6.4, 5.7, 2.7, 6.9, 1.7, 6.4)
b <- c(6.0, 2.8, 1.3, 4.7, 3.1, 1.8, 2.9, 4.0, 2.3, 1.6)

# Establecer nivel de significación.
alfa <- 0.05

# Hacer la prueba de rangos con signo de Wilcoxon.
prueba <- wilcox.test(a, b, alternative = "greater", paired = TRUE,
                      conf.level = 1 - alfa)

print(prueba)
```

11.2 PRUEBAS PARA MÁS DE DOS MUESTRAS

Al igual que existen alternativas no paramétricas para inferir con una o dos medias muestrales, también las hay para cuando se tienen más de dos muestras. Conoceremos ahora alternativas no paramétricas para el procedimiento ANOVA de una vía, tanto para muestras independientes como para muestras correlacionadas.

11.2.1 Prueba de Kruskal-Wallis

En el capítulo 9 estudiamos el procedimiento ANOVA de una vía para $k > 2$ muestras independientes, el cual requiere el cumplimiento de los siguientes supuestos:

1. La escala con que se mide la variable dependiente tiene las propiedades de una escala de intervalos iguales.
2. Las k muestras son obtenidas de manera aleatoria e independiente desde la(s) población(es) de origen.
3. Se puede suponer razonablemente que la(s) población(es) de origen sigue(n) una distribución normal.
4. Las k muestras tienen varianzas aproximadamente iguales.

Si bien ANOVA es usualmente robusto ante desviaciones leves de las condiciones (excepto la segunda) cuando las muestras son de igual tamaño, no ocurre lo mismo cuando los tamaños de las muestras difieren. En este caso, una alternativa es emplear la **prueba de Kruskal-Wallis**, cuyas condiciones son:

1. La variable independiente debe tener a lo menos dos niveles (aunque, para dos niveles, se suele usar la prueba de Wilcoxon-Mann-Whitney).
2. La escala de la variable dependiente debe ser, a lo menos, ordinal.
3. Las observaciones son independientes entre sí.

Para ilustrar esta prueba, tomemos el ejemplo de un ingeniero que cuenta con cuatro algoritmos (A, B, C y D) para resolver un determinado problema (en iguales condiciones y para instancias de tamaño fijo) y desea comparar su eficiencia. Para cada algoritmo, selecciona una muestra aleatoria independiente de instancias y registra el tiempo de ejecución (en milisegundos) del algoritmo en cuestión para cada una de las instancias de la muestra correspondiente, obteniendo las siguientes mediciones:

- Algoritmo A: 21, 22, 22, 23, 23, 23, 23, 24, 24, 24, 25, 26
- Algoritmo B: 15, 17, 18, 18, 19, 19, 20, 20, 21
- Algoritmo C: 9, 10, 10, 10, 10, 11, 11, 12, 12, 12, 12, 13, 14, 15
- Algoritmo D: 15, 15, 16, 16, 16, 18, 18, 18

Las hipótesis a contrastar son, entonces:

H_0: todos los algoritmos son igual de eficientes (o, de manera similar, ningún algoritmo es menos ni más eficiente que los demás).
H_A: al menos uno de los algoritmos presenta una eficiencia diferente a al menos algún otro algoritmo.

El procedimiento de la prueba de Kruskal-Wallis tiene elementos similares a los descritos en las pruebas no paramétricas para una y dos medias. El primer paso consiste en combinar las muestras y luego asignar el rango a cada elemento, obteniéndose para el ejemplo el resultado de la tabla 11.6.

A continuación se calcula la suma (T_x) y la media (M_x) de los rangos en cada grupo y en la muestra combinada. La tabla 11.7 presenta los valores obtenidos para el ejemplo, incluyendo además el tamaño muestral (n_x).

De manera similar a ANOVA, se requiere determinar la diferencia entre las medias grupales. Para ello se calculan las desviaciones cuadradas de las medias grupales de los rangos con respecto a la media total de los rangos. Así, la variabilidad entre grupos está dada por la ecuación 11.10.

Observaciones				Ranking de obs.			
A	**B**	**C**	**D**	**A**	**B**	**C**	**D**
21	15	9	15	31,5	15,5	1,0	15,5
22	17	10	15	33,5	21,0	3,5	15,5
22	18	10	16	33,5	24,0	3,5	19,0
23	18	10	16	36,5	24,0	3,5	19,0
23	19	10	16	36,5	27,5	3,5	19,0
23	19	11	18	36,5	27,5	8,5	24,0
23	20	11	18	36,5	29,5	8,5	24,0
24	20	12	18	40,0	29,5	8,5	24,0
24	21	12		40,0	31,5	8,5	
24		12		40,0		8,5	
25		12		42,0		8,5	
26		13		43,0		12,0	
		14				13,0	
		15				15,5	

Tabla 11.6: asignación de rangos a la muestra combinada.

	A	**B**	**C**	**D**	**Combinada**
n	12	9	14	8	43
T	449,50	230,00	106,50	160,00	946,00
M	37,46	25,56	7,61	20,00	22,00

Tabla 11.7: resumen de los rangos.

$$SS_{bg(R)} = \sum_{i=1}^{k} n_i \cdot (M_i - M_T)^2 \tag{11.10}$$

Para el ejemplo, entonces:

$$SS_{bg(R)} = n_A \cdot (M_A - M_T)^2 + n_B \cdot (M_B - M_T)^2 + n_C \cdot (M_C - M_T)^2 + n_D \cdot (M_D - M_T)^2 = \\ 12 \cdot (37,46 - 22)^2 + 9 \cdot (25,56 - 22)^2 + 14 \cdot (7,61 - 22)^2 + 8 \cdot (20 - 22)^2 = 5.913,21$$

La hipótesis nula, llevada al dominio de los rangos, es que los rangos medios de los distintos grupos no serán muy diferentes entre sí. Podría esperarse que el valor nulo para $SS_{bg(R)}$ fuera 0, no obstante, no es así. Supongamos por un momento que tenemos 3 muestras con dos observaciones cada una, con lo que tendríamos un total de 6 rangos. Dichos rangos pueden combinarse de 90 maneras distintas para formar tres grupos con dos elementos. La distribución muestral de $SS_{bg(R)}$ estaría dada, entonces, por los valores de $SS_{bg(R)}$ obtenidos para cada una de las 90 combinaciones, de los cuales únicamente 6 son iguales a 0 y todos los restantes, mayores que 0 (recuerde que es matemáticamente imposible obtener desviaciones cuadradas con valor negativo). La media de la distribución muestral para $SS_{bg(R)}$ está dada por la ecuación 11.11.

$$\mu_{SS} = (k-1)\frac{n_T \cdot (n_T + 1)}{12} \tag{11.11}$$

Para el ejemplo, entonces, tenemos que el valor nulo es:

$$\mu_{SS} = (4-1)\frac{43 \cdot (43+1)}{12} = 473$$

Llegado este punto, se define el estadístico de prueba H, el cual se construye en torno al valor obtenido para $SS_{bg(R)}$ y parte de la fórmula empleada para calcular el valor nulo, como muestra la ecuación 11.12.

$$H = \frac{SS_{bg(R)}}{\frac{n_T \cdot (n_T+1)}{12}} = \frac{12 \cdot SS_{bg(R)}}{n_T \cdot (n_T+1)} \tag{11.12}$$

En consecuencia, el valor del estadístico de prueba para el ejemplo es:

$$H = \frac{12 \cdot 5.913,21}{43 \cdot (43+1)} = 37.5$$

Cuando cada uno de los k grupos tiene a lo menos 5 observaciones, el estadístico de prueba H sigue una distribución χ^2 con $\nu = k - 1$ grados de libertad. Así, podemos calcular el valor p para el ejemplo (en R) mediante la llamada `pchisq(37.5, 3, lower.tail = FALSE)`, obteniéndose como resultado $p = 3.606 \cdot 10^{-8}$. Este valor indica que la evidencia es suficientemente fuerte como para rechazar la hipótesis nula en favor de la hipótesis alternativa, incluso para un nivel de significación $\alpha = 0,01$. En consecuencia, podemos concluir con 99 % de confianza que existen diferencias significativas entre los tiempos promedio de ejecución de los algoritmos A, B, C y D.

Fijémonos en que, al igual que ANOVA, la prueba de Kruskal-Wallis es de tipo ómnibus, por lo que no entrega información en relación a cuáles grupos presentan diferencias. En consecuencia, una vez más es necesario efectuar un análisis post-hoc cuando se detectan diferencias significativas. De manera similar a la estudiada en el capítulo 9, podemos hacer comparaciones entre pares de grupos con la prueba de Wilcoxon-Mann-Whitney (equivalentes a las realizadas con la prueba t de Student para ANOVA de una vía para muestras independientes), usando alguno de los factores de corrección que ya conocimos en el capítulo 8 (por ejemplo, Holm y Bonferroni) (Amat Rodrigo, 2016c).

En R, podemos ejecutar la prueba de Kruskal-Wallis mediante la función `kruskal.test(formula, data)`, donde:

- `formula`: tiene la forma <variable dependiente>∼ <variable independiente (factor)>.
- `data`: matriz de datos en formato largo.

Para los procedimientos post-hoc, las pruebas de Bonferroni y Holm pueden realizarse mediante la función `pairwise.wilcox.test(x, g, p.adjust.method, paired = FALSE)`, donde:

- `x`: vector con la variable dependiente.
- `g`: factor o agrupamiento.
- `p.adjust.method`: puede ser "holm" o "bonferroni", entre otras alternativas.
- `paired`: valor booleano que indica si la prueba es pareada (verdadero) o no. Para la prueba de Kruskal-Wallis debe ser `FALSE`.

El script 11.3 muestra la realización de la prueba de Kruskal-Wallis para el ejemplo e incorpora el procedimiento post-hoc de Holm. Los resultados se presentan en la figura 11.5. Podemos ver que el valor p difiere ligeramente al obtenido anteriormente, debido a errores de redondeo. A partir de los resultados del procedimiento post-hoc, considerando un nivel de significación $\alpha = 0,01$, podemos concluir con 99 % de confianza que existen diferencias significativas entre los tiempos promedio de ejecución de todos los pares de algoritmos con excepción de los algoritmos B y D.

Script 11.3: prueba de Kruskal-Wallis y el procedimiento post-hoc de Holm para el ejemplo.

```
# Construir la mariz de datos.
A <- c(24, 23, 26, 21, 24, 24, 25, 22, 23, 22, 23, 23)
B <- c(17, 15, 18, 20, 19, 21, 20, 18, 19)
C <- c(10, 11, 14, 11, 15, 12, 12, 10, 9, 13, 12, 12, 10, 10)
D <- c(18, 16, 18, 15, 16, 15, 18, 16)
Tiempo <- c(A, B, C, D)

```

```
        Kruskal-Wallis rank sum test

        data:  Tiempo by Algoritmo
        Kruskal-Wallis chi-squared = 37.714, df = 3,
        p-value = 3.249e-08


        Pairwise comparisons using Wilcoxon rank sum test with continuity correction

        data:  datos$Tiempo and datos$Algoritmo

          A       B       C
        B 0.00060 -       -
        C 9.3e-05 0.00042 -
        D 0.00060 0.02738 0.00060

        P value adjustment method: holm
```

Figura 11.5: resultado de la prueba de Kruskal-Wallis y el procedimiento post-hoc de Holm para el ejemplo.

```r
Algoritmo <- c(rep("A", length(A)),
               rep("B", length(B)),
               rep("C", length(C)),
               rep("D", length(D)))

Algoritmo <- factor(Algoritmo)

datos <- data.frame(Tiempo, Algoritmo)

# Establecer nivel de significación
alfa <- 0.01

# Hacer la prueba de Kruskal-Wallis.
prueba <- kruskal.test(Tiempo ~ Algoritmo, data = datos)
print(prueba)

# Efectuar procedimiento post-hoc de Holm si se encuentran diferencias
# significativas.
if(prueba$p.value < alfa) {
  post_hoc <- pairwise.wilcox.test(datos$Tiempo,
                                   datos$Algoritmo,
                                   p.adjust.method = "holm",
                                   paired = FALSE)

  print(post_hoc)
}
```

Notemos que `pairwise.wilcox.test()` solo reporta los p valores ajustados. Si queremos conocer el tamaño del efecto de las diferencias detectadas, debemos realizar las correspondientes pruebas de Wilcoxon-Mann-Whitney para todos los pares de grupos que presenten diferencias significativas.

11.2.2 Prueba de Friedman

Como es natural suponer, podemos considerar la **prueba de Friedman** como una alternativa no paramétrica al procedimiento ANOVA de una vía con muestras correlacionadas descrito en el capítulo 10. Sin embargo, debemos saber que no es exactamente una extensión de esta prueba, puesto que no considera las diferencias relativas entre sujetos (como lo hace ANOVA y la prueba de rangos con signo de Wilcoxon), y en consecuencia, como señala Baguley (2012), el poder estadístico es bastante menor.

Recordemos las condiciones que se deben verificar para poder aplicar la prueba ANOVA para muestras correlacionadas:

1. La escala con que se mide la variable dependiente tiene las propiedades de una escala de intervalos iguales.
2. Las mediciones son independientes al interior de cada grupo.
3. Se puede suponer razonablemente que la(s) población(es) de origen sigue(n) una distribución normal.
4. La matriz de varianzas-covarianzas es esférica. Como explica Horn (2008, p. 1), esta condición establece que las varianzas entre los diferentes niveles de las medidas repetidas deben ser iguales.

Existen situaciones en las que no podemos comprobar que la escala de medición de la variable dependiente sea de intervalos iguales:

- Cuando las observaciones se miden en una escala logarítmica (por ejemplo, la escala de pH para medir la acidez o la escala de Richter para medir la intensidad de los sismos).
- Cuando las mediciones provienen de una escala ordinal, por ejemplo, un orden de preferencia.
- Cuando las mediciones de base provienen de una escala ordinal. Por ejemplo, cuando se suman o promedian puntajes de diversos elementos evaluados con una escala Likert.

Las condiciones requeridas por la prueba de Friedman son las siguientes:

1. La variable independiente debe ser categórica y tener a lo menos tres niveles.
2. La escala de la variable dependiente debe ser, a lo menos, ordinal.
3. Los sujetos son una muestra aleatoria e independiente de la población.

Como ejemplo para esta prueba, supongamos ahora que un equipo de desarrolladores desea establecer qué interfaz gráfica (A, B o C) resulta más atractiva para los usuarios de un nuevo sistema, por lo que han seleccionado una muestra aleatoria representativa de los distintos tipos de usuarios y les han solicitado evaluar 6 aspectos de cada interfaz con una escala Likert de 5 puntos, donde el valor 1 corresponde a una valoración muy negativa y 5, a una muy positiva. La tabla 11.8 muestra las puntuaciones totales asignadas por cada participante a las diferentes interfaces. En consecuencia, las hipótesis a contrastar son:

H_0: las interfaces tienen preferencias similares.
H_A: al menos una interfaz obtiene una preferencia distinta a las demás.

Usuario	A	B	C
1	21	6	13
2	10	21	25
3	7	18	18
4	21	7	20
5	24	24	24
6	27	13	8
7	17	13	29

Tabla 11.8: evaluación realizada por los usuarios a cada una de las distintas interfaces.

El primer paso del proceso consiste en asignar rangos a las observaciones de cada sujeto. La interfaz con puntuación más baja recibe un rango de 1 y la más alta, un rango de 3 (generalizando, si se tienen k

observaciones pareadas, se asignan rangos con valores 1 a k). En caso de empate, se asigna el promedio de los rangos correspondientes. La tabla 11.9 muestra el resultado de este proceso.

	Originales			**Rangos**		
Usuario	**A**	**B**	**C**	**A**	**B**	**C**
1	21	6	13	3	1	2
2	10	21	25	1	2	3
3	7	18	18	1	2,5	2,5
4	21	7	20	3	1	2
5	24	24	24	2	2	2
6	27	13	8	3	2	1
7	17	13	29	2	1	3

Tabla 11.9: ranking de las interfaces por usuario.

La hipótesis nula para la prueba de Friedman es que, los rangos promedio de cada interfaz serán muy similares. Si denotamos el rango promedio de un grupo (interfaz) por M_x, para cada grupo esperamos, entonces, que se cumpla la igualdad de la ecuación 11.13, donde k es la cantidad de grupos.

$$M_x = \frac{k+1}{2} \tag{11.13}$$

A continuación se calculan algunas estadísticas de resumen, donde n corresponde al tamaño de cada muestra y M, a la media de los rangos (tabla 11.10).

	A	**B**	**C**	**Combinada**
n	7	7	7	21
M	2,14	1,64	2,21	2

Tabla 11.10: resumen de los rangos.

Con estos valores, podemos definir una medida para la variabilidad de los grupos agregados, dada por la ecuación 11.14.

$$SS_{bg(R)} = \sum_{i=1}^{k} n_i \cdot (M_i - M_T)^2 \tag{11.14}$$

Haciendo el cálculo para el ejemplo, tenemos:

$$SS_{bg(R)} = 7 \cdot [(2,14-2)^2 + (1,64-2)^2 + (2,21-2)^2] = 1,357$$

Con el resultado anterior, podemos ahora calcular el estadístico de prueba (11.15), que sigue una distribución χ^2 con $k-1$ grados de libertad.

$$\chi^2 = \frac{SS_{bg(R)}}{\frac{k\cdot(k+1)}{12}} = \frac{12 \cdot SS_{bg(R)}}{k \cdot (k+1)} \tag{11.15}$$

Para el ejemplo:

$$\chi^2 = \frac{12 \cdot 1,357}{3 \cdot (3+1)} = 1,357$$

Una vez más, calculamos el valor p mediante la llamada `pchisq(1.357, 2, lower.tail = FALSE)`, obteniéndose $p = 0,507$. Considerando un nivel de significación $\alpha = 0,05$, se falla al rechazar la hipótesis nula. En consecuencia, concluimos con 95 % de confianza que no hay diferencias significativas de preferencia entre las distintas interfaces.

En este caso no es necesario realizar un procedimiento post-hoc, pues la prueba ómnibus no encontró diferencias estadísticamente significativas. No obstante, si fuese necesario, podemos efectuar una prueba de rangos con signo de Wilcoxon por cada par de grupos y aplicar algún factor de corrección.

Para hacer la prueba de Friedman en R, podemos usar la función `friedman.test(formula, data)`, donde:

- `formula`: tiene la forma <variable dependiente>∼ <variable independiente>| <identificador de sujeto o bloque>.
- `data`: matriz de datos en formato largo.

Para los procedimientos post-hoc, podemos aplicar los ajustes de Bonferroni y Holm mediante la función `pairwise.wilcox.test()`, del mismo modo descrito para la prueba de Kruskal-Wallis, cuidando en este caso que el argumento `paired` debe tomar forzosamente el valor `TRUE`. Si además queremos conocer el tamaño del efecto detectado para aquellos pares identificados como relevantes, debemos realizar las correspondientes pruebas de rangos con signo de Wilcoxon para todos los pares de grupos que presenten diferencias significativas (Amat Rodrigo, 2016b).

El script 11.4 muestra la realización de la prueba de Friedman para el ejemplo, cuyo resultado se presenta en la figura 11.6, e incorpora el procedimiento post-hoc de Holm por fines académicos, ya que solo debería realizase si la prueba ómnibus encuentra diferencias significativas.

Script 11.4: prueba de Friedman y el procedimiento post-hoc de Holm para el ejemplo.

```
# Construir la mariz de datos.
A <- c(21, 10, 7, 21, 24, 27, 17)
B <- c(6, 21, 18, 7, 24, 13, 13)
C <- c(13, 25, 18, 20, 24, 8, 29)

Puntuacion <- c(A, B, C)

Interfaz <- c(rep("A", length(A)),
              rep("B", length(B)),
              rep("C", length(C)))

Sujeto <- rep(1:7, 3)

Interfaz <- factor(Interfaz)

datos <- data.frame(Sujeto, Puntuacion, Interfaz)

# Establecer nivel de significación
alfa <- 0.05

# Hacer la prueba de Friedman.
prueba <- friedman.test(Puntuacion ~ Interfaz | Sujeto, data = datos)
print(prueba)

# Efectuar procedimiento post-hoc de Holm si se encuentran diferencias
# significativas.
if(prueba$p.value < alfa) {
  post_hoc <- pairwise.wilcox.test(datos$Puntuacion,
                                   datos$Interfaz,
                                   p.adjust.method = "holm",
                                   paired = TRUE)

```

```
33   print(post_hoc)
34 }
```

```
                Friedman rank sum test

                data:  Puntuacion and Interfaz and Sujeto
                Friedman chi-squared = 1.6522, df = 2, p-value = 0.4378
```

Figura 11.6: valores p obtenidos en las pruebas t para cada par de grupos mediante los métodos de Bonferroni y Holm.

11.3 EJERCICIOS PROPUESTOS

1. ¿Qué riesgos se corren si se aplica la prueba t de Student con dos muestras que no cumplen con las suposiciones que hace esta prueba?
2. La prueba de Wilcoxon-Mann-Whitney es una alternativa no paramétrica ¿para qué versión de la prueba t de Student?
3. ¿Qué suposiciones hace la prueba de Wilcoxon-Mann-Whitney?
4. Explica cómo la prueba de Wilcoxon-Mann-Whitney construye el ranking de los datos.
5. ¿Qué estadístico usa la prueba de Wilcoxon-Mann-Whitney y cómo se calcula?
6. ¿Por qué a la prueba de Wilcoxon-Mann-Whitney también se le conoce como U-test?
7. La prueba de los rangos con signo de Wilcoxon es una alternativa no paramétrica ¿para qué versión de la prueba t de Student?
8. ¿Qué suposiciones hace la prueba de los rangos con signo de Wilcoxon?
9. Explica cómo la prueba de los rangos con signo de Wilcoxon construye el ranking de los datos.
10. ¿Qué estadístico usa la prueba de los rangos con signo de Wilcoxon y cómo se calcula?
11. ¿Cuándo es más relevante preocuparse de las violaciones de las condiciones del procedimiento ANOVA para muestras independientes?
12. Explica cómo la prueba de Kruskal-Wallis construye el ranking de los datos.
13. ¿Qué estadístico usa la prueba de Kruskal-Wallis y cómo se calcula? ¿Qué distribución sigue dicho estadístico?
14. ¿Cuál es la hipótesis nula de la prueba de Kruskal-Wallis?
15. ¿Qué suposiciones hace la prueba de Kruskal-Wallis?
16. Explique cómo la prueba de Friedman construye el ranking de los datos.
17. ¿Qué estadístico usa la prueba de Friedman y cómo se calcula? ¿Qué distribución sigue dicho estadístico?
18. ¿Cuál es la hipótesis nula de la prueba de Friedman?
19. ¿Qué suposiciones hace la prueba de Friedman?

CAPÍTULO 12. REMUESTREO

Los **métodos basados en remuestreo** son una buena alternativa a emplear cuando necesitamos inferir sobre parámetros distintos a la media o la proporción, o bien cuando no se cumplen las condiciones requeridas por las pruebas ya conocidas. Además, algunos de estos métodos son más precisos que los tradicionales. Pese a estas ventajas, los métodos basados en remuestreo realizan enormes cantidades de cómputos, por lo que en la práctica requieren de herramientas de software para su aplicación. Si bien existen métodos de remuestreo paramétricos y semiparamétricos, en este capítulo abordaremos las principales técnicas de remuestro no paramétricas, basándonos en las ideas descritas por Amat Rodrigo (2016a) y Hesterberg y col. (2003).

12.1 BOOTSTRAPPING

A partir de lo que hemos aprendido hasta ahora, ya tenemos bastante claro que, en estadística, el ideal es contar con varias muestras grandes. Pero muchas veces solo disponemos de una muestra bastante pequeña. Sin embargo, si esta muestra es representativa de la población, esperaríamos que las observaciones que ella contiene aparecieran con frecuencias similares a las de la población. El método de **bootstrapping** se construye en torno a esta idea y, en términos generales, sigue los siguientes pasos:

1. Crear una gran cantidad B de nuevas muestras (cientos o miles) a partir de la muestra original. Cada muestra debe tener el mismo tamaño que la original y se construye mediante **muestreo con reposición**. Esto quiere decir que, al seleccionar un elemento de la muestra original, se devuelve a ella antes de tomar el siguiente, por lo que podría ser reelegido.
2. Calcular la distribución bootstrap y obtener el estadístico de interés para cada una de las muestras.
3. Usar la distribución bootstrap, la cual entrega información acerca de la forma, el centro y la variabilidad de la distribución muestral del estadístico de interés.

A los lectores atentos les habrá llamado la atención que, a diferencia de las pruebas y procedimientos anteriores, el segundo paso del método de bootstrapping habla de un **estadístico de interés** en lugar de la media o la proporción (como las pruebas estudiadas hasta ahora). Esto se debe a que, en general, puede aplicarse para casi **cualquier estadístico**.

Esta técnica, además de contrastar hipótesis, permite construir intervalos de confianza para el parámetro estimado de la población.

12.1.1 Bootstrapping para una muestra

Supongamos que la investigadora Helen Chufe desea evaluar un nuevo algoritmo de clasificación y determinar el tiempo promedio de ejecución (en milisegundos) para instancias de tamaño fijo del problema. Para ello ha realizado pruebas con 10 instancias del problema y registrado los tiempos de ejecución, presentados en la tabla 12.1. La figura 12.1 muestra la distribución del tiempo de ejecución para la muestra.

Evidentemente, la muestra es pequeña ($n = 10$) y su distribución está fuertemente desviada hacia la izquierda, por lo que Chufe ha decidido emplear bootstrapping como alternativa para enfrentar estos datos problemáticos.

Instancia	1	2	3	4	5	6	7	8	9	10
Tiempo (ms)	79	75	84	75	94	82	76	90	79	88

Tabla 12.1: tiempo de ejecución para cada instancia de la muestra.

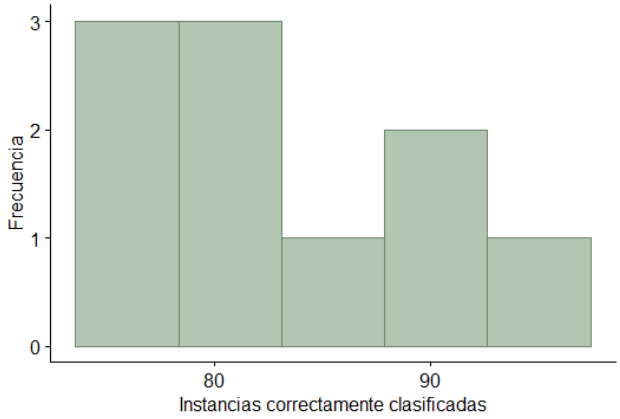


Figura 12.1: distribución del tiempo de ejecución para la muestra.

Para ilustrar el proceso paso a paso, consideremos inicialmente $B = 10$ remuestreos y calculemos la media para cada uno. La tabla 12.2 muestra en cada columna una de las muestras obtenidas, con sus respectivas medias en la última fila.

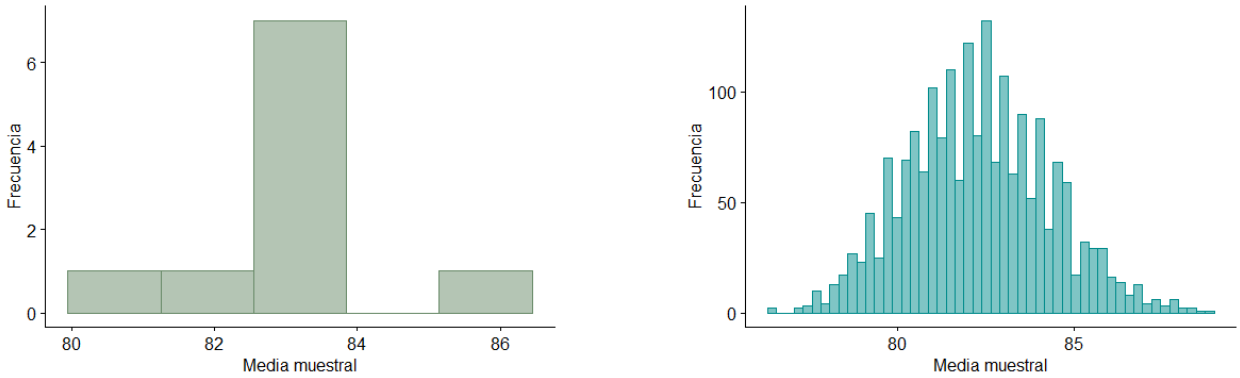
Original	**Bt 1**	**Bt 2**	**Bt 3**	**Bt 4**	**Bt 5**	**Bt 6**	**Bt 7**	**Bt 8**	**Bt 9**	**Bt 10**
79	84	84	84	94	94	94	76	82	75	79
75	94	75	82	88	75	75	88	88	82	79
84	79	82	84	90	90	94	79	79	94	94
75	79	88	79	90	82	94	76	75	94	82
94	88	79	79	90	82	76	84	75	79	84
82	75	84	79	75	76	75	75	79	76	82
76	88	82	84	75	76	79	75	90	88	94
90	88	79	79	75	82	75	79	88	88	79
79	84	79	90	94	90	88	75	75	79	75
88	75	94	88	82	88	76	94	90	82	82
82,2	**83,4**	**82,6**	**82,8**	**85,3**	**83,5**	**82,6**	**80,1**	**82,1**	**83,7**	**83,0**

Tabla 12.2: muestra original y remuestreos de bootstrap

La figura 12.2 muestra la distribución bootstrap de la media para los 10 remuestreos del ejemplo (figura 12.2a) y para 2.000 remuestreos (figura 12.2b). En ella podemos ver claramente que, a medida que la cantidad de muestras bootstrap crece, la distribución bootstrap de la media se asemeja cada vez más a la distribución normal, por lo que se acerca a la forma que esperaríamos para la distribución muestral. La figura 12.2b también nos da una idea acerca de la variabilidad de las medias de los diferentes remuestreos. Sin embargo, podemos conocer mejor esta variabilidad calculando el error estándar, dado por la ecuación 12.1, donde $\overline{x}_i^*$ es la media del i-ésimo remuestreo y B es la cantidad de remuestreos realizados.

$$SE_{b,\overline{x}} = \sqrt{\frac{1}{B-1} \cdot \sum_{i=1}^{B} \left(\overline{x}_i^* - \frac{1}{B} \cdot \sum_{i=1}^{B} \overline{x}_i^* \right)^2} \tag{12.1}$$

Fijémonos en que la subexpresión $\frac{1}{B} \cdot \sum_{i=1}^{B} \overline{x}_i^*$ de la ecuación 12.1 corresponde al promedio de las medias de



(a) 10 remuestreos.

(b) 2.000 remuestreos.

Figura 12.2: distribución bootstrap de la media

los remuestreos (media de la distribución bootstrap), por lo que la subexpresión $\sum_{i=1}^{B}\left(\overline{x}_i^* - \frac{1}{B}\cdot\sum_{i=1}^{B}\overline{x}_i^*\right)^2$ es, a su vez, la suma de las desviaciones cuadradas, con lo que resulta evidente la semejanza con el cálculo del error estándar para la media de una muestra presentado en el capítulo 4:

$$SE_{\overline{x}} = \frac{s}{\sqrt{n}}$$

Para el ejemplo con $B = 10$, entonces, tenemos que la media de la distribución bootstrap es:

$$\frac{1}{B}\cdot\sum_{i=1}^{B}\overline{x}_i^* = 83,4 + 82,6 + 82,8 + 85,3 + 83,5 + 82,6 + 80,1 + 82,1 + 83,7 + 83,0 = 82,91$$

Con lo que la suma de las desviaciones cuadradas es:

$$\begin{aligned}\sum_{i=1}^{B}(\overline{x}_i^* - 82,91)^2 &= (83,4-82,91)^2 + (82,6-82,91)^2 + (82,8-82,91)^2 + \\ &+(85,3-82,91)^2 + (83,5-82,91)^2 + (82,6-82,91)^2 + (80,1-82,91)^2 + \\ &+(82,1-82,91)^2 + (83,7-82,91)^2 + (83,0-82,91)^2 = 15,689\end{aligned}$$

En consecuencia, el error estándar de la distribución bootstrap es:

$$SE_{B,\overline{x}} = \sqrt{\frac{1}{B-1}\cdot 15,689} = 1,320$$

Otra medida que suele emplearse para la distribución bootstrap es el **sesgo**, que indica cuánto se aleja el estadístico de interés de la muestra original (θ) de la media de la distribución bootstrap, como muestra la ecuación 12.2.

$$sesgo = \theta - \frac{1}{B}\cdot\sum_{i=1}^{B}\overline{x}_i^* \tag{12.2}$$

Para el ejemplo:

$$sesgo = \overline{x} - \frac{1}{B} \cdot \sum_{i=1}^{B} \overline{x}_i^* = 82,2 - 82,91 = -0,71$$

Ahora que ya conocemos la distribución bootstrap para la media, podemos entonces **construir un intervalo de confianza para la media de la población**, para lo que abordaremos diferentes alternativas.

Cuando la distribución bootstrap se asemeja a la normal y el sesgo es pequeño en comparación con el estimador calculado (como en este caso), podemos construir el intervalo de confianza usando la distribución t, del mismo modo que conocimos en el capítulo 4, como muestra la ecuación 12.3, usando el error estándar de la distribución bootstrap. El valor crítico de t, t^*, se obtiene para el nivel de significación establecido para el estudio y $\nu = n - 1$ grados de libertad (no olvidemos que n es el tamaño de la muestra original).

$$\theta \pm t^* \cdot SE_{B,\overline{x}} \tag{12.3}$$

Así, si consideramos para este ejemplo un nivel de significación $\alpha = 0,01$, el valor crítico de t (para dos colas) con 9 grados de libertad es $t^* = 3,25$. En consecuencia, el intervalo de confianza resultante para la media de la población es:

$$82,2 \pm 3,25 \cdot 1,320 = (77,910; 86,490)$$

Otra alternativa cuando la distribución bootstrap se asemeja a la normal, que tiene en cuenta posibles asimetrías, es construir el intervalo de confianza en base a cuantiles. En este caso, para $\alpha = 0,01$, los límites del intervalo están dados por los percentiles 1 y 99 de la distribución bootstrap:

$$(80,280; 85,156)$$

Cuando los intervalos de confianza obtenidos por ambos métodos son muy diferentes, es clara señal de que no podemos asumir que la distribución bootstrap se asemeja a la normal. En general, lo más recomendable es usar otro esquema, llamado BCa (del inglés *bias-corrected accelerated*), es decir, con sesgo corregido y acelerado. No se detalla aquí el procedimiento, pues requiere el empleo de software.

Desde luego, es inviable usar bootstrapping sin software. R ofrece el paquete `boot`, con las funciones `boot(data, statistic, R)` para generar la distribución bootstrap y `boot.ci(boot.out, conf, type)` para calcular los intervalos de confianza, donde:

- `data`: el conjunto de datos. En caso de matrices y data frames, se considera cada fila como una observación con múltiples variables.
- `statistic`: función que se aplica a los datos y devuelve un vector con el (o los) estadístico(s) de interés.
- `R`: cantidad de remuestreos bootstrap (B).
- `boot.out`: objeto de la clase `boot`, generado por la función `boot()`.
- `conf`: nivel de confianza ($1 - \alpha$).
- `type`: string o vector que indica los tipos de intervalo de confianza a construir ("`norm`" para el basado en la distribución normal, "`perc`" para el basado en los percentiles y "`bca`" para el método recomendado).

Debemos mencionar que la función `boot()` puede recibir otros muchos argumentos, los cuales escapan al alcance de los contenidos aquí expuestos. El script 12.1 construye intervalos de confianza mediante bootstrapping para el ejemplo, con $B = 2000$ y manteniendo el nivel de significación $\alpha = 0,01$. En las líneas 15–17 se construye la función para el estadístico de interés (en este caso la media), que luego usa la función `boot()` para generar la distribución bootstrap (líneas 19–20), obteniéndose el resultado que se presenta en la figura 12.3.

Podemos ver gráficamente esta distribución mediante un histograma y un gráfico Q-Q (figura 12.4), gracias a la llamada a la función `plot()` con el resultado entregado por `boot()` como argumento (línea 23).

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = muestra, statistic = media, R = B)


Bootstrap Statistics :
    original  bias    std. error
t1*     82.2 0.06125     1.98329
```

Figura 12.3: distribución bootstrap generada mediante `boot()` para la media.

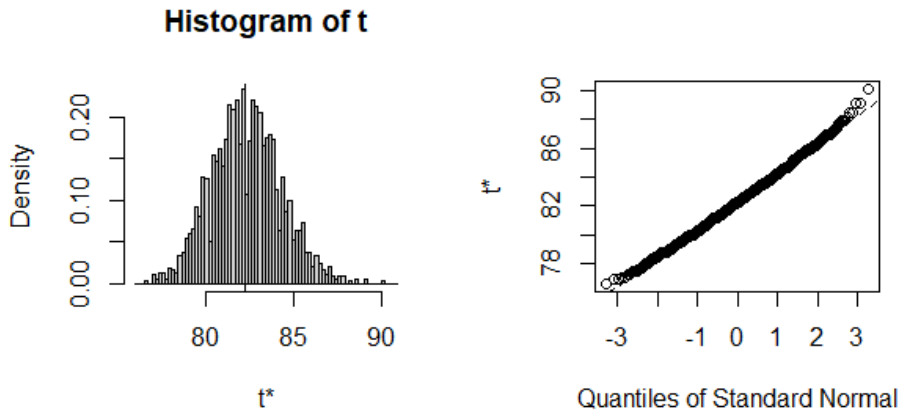


Figura 12.4: histograma y gráfico Q-Q de la distribución bootstrap generada mediante `boot()` para la media.

En las líneas 26–39 se muestra el uso de `boot.ci()` para construir los intervalos de confianza mediante diferentes métodos, obteniéndose los siguientes resultados:

- Intervalo de confianza usando aproximación normal: $(77,03; 87,25)$.
- Intervalo de confianza usando percentiles: $(77,4; 87,7)$.
- Intervalo de confianza BCa: $(77,48; 87,90)$.

Las líneas 42–45 muestran otra alternativa para construir la distribución bootstrap por medio del paquete `bootES`, que ofrece la función `bootES(data, R, ci.type, ci.conf, plot, ...)`. Esta función realiza internamente una llamada a la función `boot()` descrita en los párrafos precedentes, pero no requiere implementar previamente la función para el cálculo de la media. Debemos tener en cuenta que aquí solo se muestran algunos de los argumentos, a saber:

- `data`: conjunto de datos.
- `R`: cantidad de remuestreos bootstrap (B).
- `ci.type`: tipo de intervalo de confianza a construir (opcional), con las mismas opciones descritas para `boot.ci()`.
- `ci.conf`: nivel de significación para el intervalo de confianza (opcional, por defecto 0.95).
- `plot`: por defecto con valor `FALSE`, cuando es `TRUE` genera una figura con el histograma y el gráfico Q-Q de la ditribución bootstrap.
- `...`: permite pasar otros argumentos para la función `boot()` subyacente.

Las figuras 12.5 y 12.6 muestran los resultados obtenidos, ligeramente diferentes a los anteriores. A partir de

estos últimos podemos concluir que tenemos 95 % de confianza de que el algoritmo tarda entre 77,48 ms y 87,90 ms en ejecutar las instancias del tamaño seleccionado.

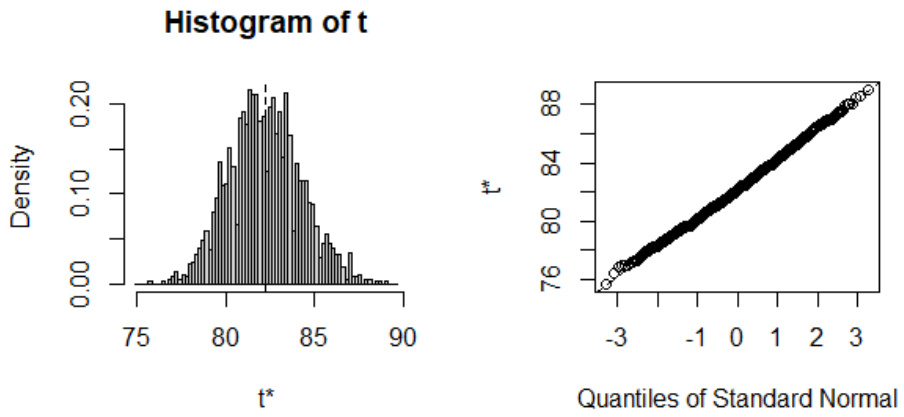


Figura 12.5: histograma y gráfico Q-Q de la distribución bootstrap generada mediante `bootES()` para la media.

```
99.00% bca Confidence Interval, 2000 replicates
Stat        CI (Low)    CI (High)   bias        SE
82.200      77.482      87.900      0.061       1.983
```

Figura 12.6: distribución bootstrap e intervalo de confianza para la media de la población generada mediante `bootES()`.

Script 12.1: construcción de un intervalo de confianza para la media poblacional mediante bootstrapping.

```r
library(boot)
library(bootES)

# Crear muestra inicial, mostrar su histograma y calcular la media.
muestra <- c(79, 75, 84, 75, 94, 82, 76, 90, 79, 88)
datos <- data.frame(muestra)

# Establecer cantidad de remuestreos y nivel de significación.
B = 2000
alfa <- 0.01

cat("Paquete boot\n")

# Construir distribución bootstrap usando el paquete boot.
media <- function(valores, i) {
  mean(valores[i])
}

set.seed(432)
distribucion_b <- boot(muestra, statistic = media, R = B)
print(distribucion_b)

# Graficar distribución bootstrap.
print(plot(distribucion_b))

```

```r
# Construir intervalos de confianza.
intervalo_t <- boot.ci(distribucion_b, conf = 1 - alfa, type = "norm")

cat("\n\nIntervalo de confianza usando distribución t:\n")
print(intervalo_t)

intervalo_per <- boot.ci(distribucion_b, conf = 1 - alfa, type = "perc")

cat("\n\nIntervalo de confianza usando percentiles:\n")
print(intervalo_per)

intervalo_bca <- boot.ci(distribucion_b, conf = 1 - alfa, type = "bca")

cat("\n\nIntervalo de confianza BCa:\n")
print(intervalo_bca)

# Construir distribución bootstrap usando el paquete bootES.
set.seed(432)

distribucion_bootstrapES <- bootES(muestra, R = B, ci.type = "bca",
                                   ci.conf = 1 - alfa, plot = TRUE)

print(distribucion_bootstrapES)
```

Supongamos ahora que Helen desea hacer una prueba de hipótesis para ver si el tiempo promedio de ejecución del algoritmo para instancias del tamaño seleccionado es mayor a 75 milisegundos. Así, tenemos que:

Denotando como μ al tiempo medio que tarda el algoritmo de Helen para resolver instancias de tamaño fijo del problema, entonces:
H_0: $\mu = 75$ [ms]
H_A: $\mu > 75$ [ms]

El contraste de hipótesis requiere siempre generar la distribución centrada en el valor nulo para, a partir de ella, obtener el valor p. Sabemos que la distribución bootstrap se centra *alrededor* del valor observado, por lo que debemos desplazarla para cumplir con esta condición. Para lograrlo, simplemente necesitamos restar a cada observación de la distribución bootstrap la diferencia entre su valor promedio y el valor nulo.

Para calcular el valor p, seguimos la fórmula señalada en la ecuación 12.4, donde:

- r: cantidad de observaciones en la distribución bootstrap (desplazada) a lo menos tan extremas como el estadístico observado.
- B: cantidad de repeticiones bootstrap consideradas en la simulación.

$$p = \frac{r+1}{B+1} \tag{12.4}$$

Tras hacer la prueba (script 12.2), obtenemos que $p = 0,001$, menor que el nivel de significación, por lo que la evidencia es suficientemente fuerte para rechazar la hipótesis nula en favor de la hipótesis alternativa. En consecuencia, concluimos con 99,5 % de confianza que el tiempo de ejecución promedio del algoritmo para instancias del tamaño seleccionado supera los 75 milisegundos.

Script 12.2: inferencia sobre la media de una muestra con bootstrapping.

```r
library(boot)

set.seed(432)

# Crear muestra inicial, mostrar su histograma y calcular la media.
muestra <- c(79, 75, 84, 75, 94, 82, 76, 90, 79, 88)
valor_observado <- mean(muestra)
datos <- data.frame(muestra)

# Construir distribución bootstrap.
B <- 2000

media <- function(valores, i) {
  mean(valores[i])
}

distribucion_b <- boot(muestra, statistic = media, R = B)

# Desplazar la distribución bootstrap para que se centre en
# el valor nulo.
valor_nulo <- 75
desplazamiento <- mean(distribucion_b[["t"]]) - valor_nulo
distribucion_nula <- distribucion_b[["t"]] - desplazamiento

# Determinar el valor p.
p <- (sum(distribucion_nula > valor_observado) + 1) / (B + 1)
cat("Valor p:", p)
```

12.1.2 Bootstrapping para dos muestras independientes

El proceso para comparar dos poblaciones mediante bootstraping es similar al que ya conocimos para una única población. Si tenemos dos muestras independientes A y B provenientes de dos poblaciones diferentes, de tamaños n_A y n_B respectivamente, los pasos a seguir son:

1. Fijar la cantidad B de repeticiones bootstrap.
2. En cada repetición, hacer un remuestreo con reposición de tamaño n_A a partir de la muestra A y otro de tamaño n_B a partir de la muestra B.
3. En cada repetición, calcular el estadístico de interés para generar la distribución bootstrap.
4. Construir el intervalo de confianza para el estadístico de interés.

Supongamos que una Universidad desea estudiar la diferencia entre las calificaciones finales de hombres y mujeres que rinden una asignatura inicial de programación por primera vez. Para ello, disponen de las notas (en escala de 1,0 a 7,0) de 27 hombres y 19 mujeres:

- Hombres: 1,3; 1,5; 1,6; 1,7; 1,7; 1,9; 2,3; 2,4; 2,6; 2,6; 2,7; 2,8; 3,2; 3,7; 4,1; 4,4; 4,5; 4,8; 5,2; 5,2; 5,3; 5,5; 5,5; 5,6; 5,6; 5,7; 5,7
- Mujeres: 3,5; 3,6; 3,8; 4,3; 4,5; 4,5; 4,9; 5,1; 5,3; 5,3; 5,5; 5,8; 6,0; 6,3; 6,3; 6,4; 6,4; 6,6; 6,7

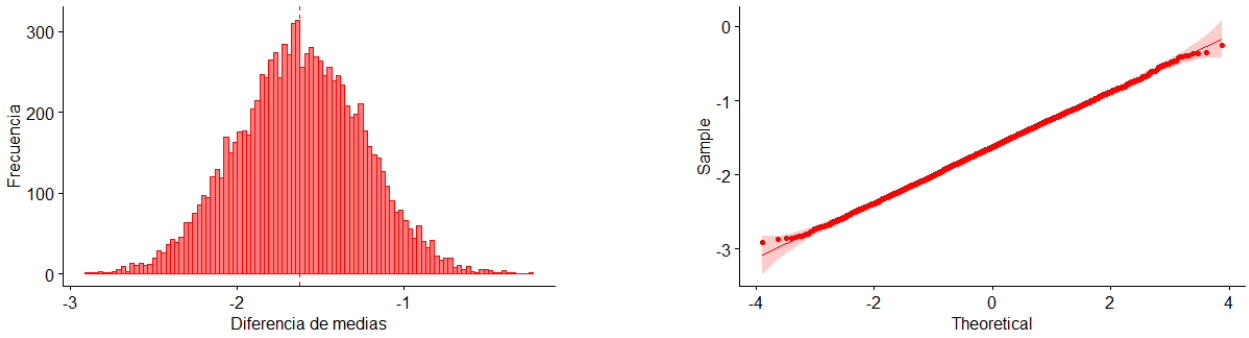
Tras aplicar pruebas de Shapiro-Wilk (figura 12.7), los investigadores han comprobado que las notas de los varones no siguen una distribución normal, por lo que han decidido usar bootstrapping para la prueba de hipótesis, con un nivel de significación $\alpha = 0,05$ y $B = 9999$ repeticiones.

```
Shapiro-Wilk normality test

data:  hombres
W = 0.88357, p-value = 0.005742


Shapiro-Wilk normality test

data:  mujeres
W = 0.93022, p-value = 0.1748
```

Figura 12.7: pruebas de normalidad de Shappiro-Wilk para ambas muestras.

La media observada (en la muestra original) para la calificación final de las mujeres es $\overline{x}_m = 5,305$, mientras que para los hombres es $\overline{x}_h = 3,670$. Así, la diferencia observada es $\overline{x}_h - \overline{x}_m = -1,635$.

La distribución bootstrap de la diferencia de medias se asemeja a la normal (figura 12.8), con media $\overline{x} = -1,628$ y desviación estándar $s = 0,377$.



(a) histograma con la media añadida.

(b) gráfico Q-Q.

Figura 12.8: distribución bootstrap de la diferencia de medias.

Al construir el intervalo de confianza mediante el método BCa para la distribución bootstrap, R nos entrega como resultado el intervalo $(-2,372; -0,894)$. En consecuencia, concluimos con 95 % de confianza que las mujeres tienen, en promedio, mejor calificación final que los hombres, con una diferencia de entre 0,894 y 2,372 puntos.

El script 12.3 muestra el desarrollo de este ejemplo en R, el cual usa la función `two.boot(sample1, sample2, FUN, R)` del paquete `simpleboot`, donde:

- `sample1, sample2`: muestras originales.
- `FUN`: función que, para cada muestra, calcula el estadístico de interés θ.
- `R`: cantidad de remuestreos con repetición.

Esta función opera generando remuestreos para cada una de las muestras originales, y calculando en cada iteración el estadístico $\theta_1 - \theta_2$, donde los subíndices señalan la muestra correspondiente.

Script 12.3: bootstraping para la diferencia de medias.

```r
library(simpleboot)
library(boot)
library(ggpubr)

set.seed(432)

```

```r
# Ingresar datos originales
hombres <- c(1.3, 1.5, 1.6, 1.7, 1.7, 1.9, 2.3, 2.4, 2.6, 2.6, 2.7,
             2.8, 3.2, 3.7, 4.1, 4.4, 4.5, 4.8, 5.2, 5.2, 5.3, 5.5,
             5.5, 5.6, 5.6, 5.7, 5.7)

mujeres <- c(3.5, 3.6, 3.8, 4.3, 4.5, 4.5, 4.9, 5.1, 5.3, 5.3, 5.5,
             5.8, 6.0, 6.3, 6.3, 6.4, 6.4, 6.6, 6.7)

n_hombres <- length(hombres)
n_mujeres <- length(mujeres)

sexo <- c(rep("Hombre", n_hombres), rep("Mujer", n_mujeres))
nota <- c(hombres, mujeres)
datos <- data.frame(nota, sexo)

# Comprobar normalidad de las muestras.
print(shapiro.test(hombres))
print(shapiro.test(mujeres))

# Calcular la diferencia observada entre las medias muestrales.
media_hombres <- mean(hombres)
media_mujeres <- mean(mujeres)
diferencia_observada <- media_hombres - media_mujeres

cat("diferencia observada:", media_hombres - media_mujeres, "\n\n")

# Establecer el nivel de significación.
alfa <- 0.05

# Crear la distribución bootstrap.
B <- 9999
distribucion_bootstrap <- two.boot(hombres, mujeres, FUN = mean, R = B)

# Examinar la distribución bootstrap.
valores <- data.frame(distribucion_bootstrap$t)
colnames(valores) <- "valores"

histograma <- gghistogram(valores, x = "valores", color = "red",
                          fill = "red", bins = 100,
                          xlab = "Diferencia de medias",
                          ylab = "Frecuencia", add = "mean")

print(histograma)

qq <- ggqqplot(valores, x = "valores", color = "red")
print(qq)

cat("Distribución bootstrap:\n")
cat("\tMedia:", mean(valores$valores), "\n")
cat("\tDesviación estándar:", sd(valores$valores), "\n\n")

# Construir el intervalo de confianza.
intervalo_bca <- boot.ci(distribucion_bootstrap, conf = 1 - alfa,
                         type = "bca")

print(intervalo_bca)
```

Supongamos ahora que el estudio del ejemplo desea determinar, con un nivel de significación $\alpha = 0,05$, si la diferencia entre las calificaciones finales de hombres y mujeres es igual a 1,5 puntos. Para ello, formulamos

las siguientes hipótesis:

Sean μ_h y μ_m las calificaciones finales de hombres y mujeres, respectivamente, que rinden una asignatura inicial de programación por primera vez en la Universidad en estudio, entonces:
H_0: $\mu_h - \mu_m = 1,5$
H_A: $\mu_h - \mu_m \neq 1,5$

Tras aplicar bootstrapping para la prueba de hipótesis (script 12.4), obtenemos un valor p de $p = 0,364$, superior al nivel de significación, por lo que fallamos al rechazar la hipótesis nula. En consecuencia, concluimos con 95 % de confianza que la diferencia en la calificación final entre hombres y mujeres es de 1,5 puntos.

Script 12.4: bootstraping para inferir acerca de la diferencia de medias.

```
library(simpleboot)
library(boot)
library(ggpubr)

set.seed(432)

# Ingresar datos originales
hombres <- c(1.3, 1.5, 1.6, 1.7, 1.7, 1.9, 2.3, 2.4, 2.6, 2.6, 2.7,
             2.8, 3.2, 3.7, 4.1, 4.4, 4.5, 4.8, 5.2, 5.2, 5.3, 5.5,
             5.5, 5.6, 5.6, 5.7, 5.7)

mujeres <- c(3.5, 3.6, 3.8, 4.3, 4.5, 4.5, 4.9, 5.1, 5.3, 5.3, 5.5,
             5.8, 6.0, 6.3, 6.3, 6.4, 6.4, 6.6, 6.7)

n_hombres <- length(hombres)
n_mujeres <- length(mujeres)

sexo <- c(rep("Hombre", n_hombres), rep("Mujer", n_mujeres))
nota <- c(hombres, mujeres)
datos <- data.frame(nota, sexo)

# Calcular la diferencia observada entre las medias muestrales.
media_hombres <- mean(hombres)
media_mujeres <- mean(mujeres)
valor_observado <- media_hombres - media_mujeres

# Crear la distribución bootstrap.
B <- 9999
valor_nulo <- 1.5
distribucion_bootstrap <- two.boot(hombres, mujeres, FUN = mean, R = B)
desplazamiento <- mean(distribucion_bootstrap[["t"]]) - valor_nulo
distribucion_nula <- distribucion_bootstrap[["t"]] - desplazamiento

# Determinar el valor p.
p <- (sum(abs(distribucion_nula) > abs(valor_observado)) + 1) / (B + 1)
cat("Valor p:", p)
```

12.1.3 Bootstrapping para dos muestras pareadas

En este caso, el procedimiento resulta muy sencillo. A partir de las dos muestras originales, se crea una nueva muestra con la diferencia entre ambas, y luego se realiza el proceso especificado para la construcción de un intervalo de confianza que ya conocimos para el caso de una única muestra.

Supongamos ahora, que la Universidad del ejemplo anterior desea saber si existe diferencia entre las calificaciones obtenidas en la primera y la segunda prueba de un curso inicial de programación. Para ello, dispone de las calificaciones (en escala de 1,0 a 7,0) obtenidas en ambas pruebas para una muestra de 20 estudiantes, como muestra la tabla 12.3. Han decidido llevar a cabo el estudio mediante bootstrapping con $B = 3999$ repeticiones y un nivel de significación $\alpha = 0,05$, para lo cual han creado en R el script 12.5, obteniendo los resultados que se presentan en la figura 12.9.

Alumno	**Prueba 1**	**Prueba 2**
1	3,5	5,2
2	2,7	5,1
3	1,0	5,9
4	1,8	4,8
5	1,6	1,4
6	4,3	2,3
7	5,8	6,8
8	6,4	5,3
9	3,9	3,1
10	4,3	3,8
11	3,4	4,6
12	5,3	1,2
13	5,8	3,9
14	5,3	2,0
15	2,0	1,7
16	1,3	3,3
17	4,0	6,0
18	5,3	4,8
19	1,6	6,9
20	3,6	1,3

Tabla 12.3: calificaciones de los estudiantes en la primera y la segunda prueba de un curso inicial de programación.

```
95.00% bca Confidence Interval, 3999 replicates
Stat        CI (Low)    CI (High)   bias        SE
0.325       -0.656      1.439       0.001       0.541
```

Figura 12.9: intervalo de confianza BCa para la media de las diferencias.

A partir del resultado anterior, concluimos con 95 % de confianza que la diferencia de las medias para las calificaciones de la primera y la segunda evaluación se encuentra en el intervalo $(-0,656; 1,439)$, por lo que no hay una diferencia estadísticamente significativa.

Script 12.5: bootstraping para la media de las diferencias.

```
library(bootES)

set.seed(432)

# Ingresar datos originales.
alumno <- 1:20

prueba_1 <- c(3.5, 2.7, 1.0, 1.8, 1.6, 4.3, 5.8, 6.4, 3.9, 4.3, 3.4,
              5.3, 5.8, 5.3, 2.0, 1.3, 4.0, 5.3, 1.6, 3.6)

prueba_2 <- c(5.2, 5.1, 5.9, 4.8, 1.4, 2.3, 6.8, 5.3, 3.1, 3.8, 4.6,
              1.2, 3.9, 2.0, 1.7, 3.3, 6.0, 4.8, 6.9, 1.3)
```

```r

# Establecer nivel de significación.
alfa <- 0.05

# Calcular la diferencia entre ambas observaciones.
diferencia <- prueba_2 - prueba_1

# Generar la distribución bootstrap y su intervalo de confianza.
B <- 3999

distribucion_bootstrapES <- bootES(diferencia, R = B, ci.type = "bca",
                                   ci.conf = 1 - alfa, plot = FALSE)

print(distribucion_bootstrapES)
```

Ahora la Universidad del ejemplo desea saber si la diferencia entre las calificaciones obtenidas en la primera y la segunda prueba de un curso inicial de programación es de 5 décimas. Así, considerando un nivel de significación $\alpha = 0,05$, los investigadores formulan las siguientes hipótesis:

H_0: $\mu_{dif} = 0,5$
H_0: $\mu_{dif} \neq 0,5$

Tras efectuar la prueba de hipótesis mediante bootstrapping (script 12.6) obtienen un valor p de $p = 0,573$, por lo que la evidencia no es suficientemente fuerte como para rechazar la hipótesis nula. En consecuencia, los investiadores concluyen con 95 % de confianza que la diferencia de las calificaciones obtenidas en ambas evaluaciones es de 5 décimas.

Script 12.6: bootstraping para inferir acerca de la media de las diferencias.

```r
library(bootES)

set.seed(432)

# Ingresar datos originales.
alumno <- 1:20

prueba_1 <- c(3.5, 2.7, 1.0, 1.8, 1.6, 4.3, 5.8, 6.4, 3.9, 4.3, 3.4,
              5.3, 5.8, 5.3, 2.0, 1.3, 4.0, 5.3, 1.6, 3.6)

prueba_2 <- c(5.2, 5.1, 5.9, 4.8, 1.4, 2.3, 6.8, 5.3, 3.1, 3.8, 4.6,
              1.2, 3.9, 2.0, 1.7, 3.3, 6.0, 4.8, 6.9, 1.3)

# Establecer nivel de significación.
alfa <- 0.05

# Calcular la diferencia entre ambas observaciones.
diferencia <- prueba_2 - prueba_1

# Calcular la media observada de las diferencias.
valor_observado <- mean(diferencia)

# Generar la distribución bootstrap y su intervalo de confianza.
B <- 3999
valor_nulo <- 0.5

distribucion_bootstrapES <- bootES(diferencia, R = B, ci.type = "bca",
                                   ci.conf = 1 - alfa, plot = FALSE)

distribucion_nula <- distribucion_bootstrapES[["t"]] - valor_nulo

```

```r

# Determinar el valor p.
p <- (sum(abs(distribucion_nula) > abs(valor_observado)) + 1) / (B + 1)
cat("Valor p:", p)
```

12.2 PRUEBAS DE PERMUTACIONES

En el capítulo 8 conocimos la prueba exacta de Fisher, la cual obtiene un valor p exacto tras calcular todas las permutaciones de los datos con iguales valores marginales en una tabla de contingencia como alternativa para muestras pequeñas de la prueba y considerar únicamente aquellas permutaciones que ocurren con igual o menor probabilidad que la obtenida para los datos del estudio.

La prueba exacta de Fisher es lo que se conoce como una **prueba exacta de permutaciones**, cuyo único requisito es la **intercambiabilidad**: si se cumple la hipótesis nula, todas las permutaciones pueden ocurrir con igual probabilidad. En la práctica, este tipo de métodos puede emplearse para diversos estadísticos, tales como la proporción, la media y la varianza. Puesto que el valor p entregado por las pruebas de permutaciones es exacto, no es posible obtener un intervalo de confianza.

En términos generales, las pruebas exactas de permutaciones para la diferencia entre dos grupos A y B (puede extenderse esta idea para más grupos) de tamaños n_A y n_B, respectivamente, sigue los siguientes pasos:

1. Calcular la diferencia entre el estadístico de interés observado para ambos grupos.
2. Juntar ambas muestras en una muestra combinada.
3. Obtener todas las permutaciones de la muestra combinada en que se pueden distribuir las observaciones en dos grupos de tamaños n_A y n_B.
4. Construir la distribución de las posibles diferencias, calculando la diferencia entre el estadístico de interés obtenido para ambos grupos en cada una de las permutaciones.
5. Calcular el valor p exacto, dado por la proporción de permutaciones en que el valor (absoluto, si es bilateral) de la diferencia calculada es menor/mayor o igual al valor (absoluto si es bilateral) de la diferencia observada.

Puesto que las pruebas exactas de permutaciones requieren calcular todas las permutaciones, solo resultan adecuadas para muestras pequeñas, pues requieren de una enorme cantidad de cómputos. En consecuencia, si la muestra es grande, suele tomarse una muestra aleatoria de las permutaciones posibles, procedimiento que suele denominarse **simulación de Monte Carlo**, y a partir de ella calcular un valor p aproximado dado por la ecuación 12.4.

Podemos ver que en la ecuación 12.4 se suma 1 tanto al numerador como al denominador. Esto corresponde a una corrección que debemos aplicar puesto que el método de Monte Carlo no es insesgado.

De los párrafos anteriores se desprende que las pruebas de permutaciones (exactas o no) son adecuadas para el contraste de hipótesis con dos o más muestras, pues determinan una significación estadística (valor p).

En términos generales, el procedimiento para efectuar una prueba de permutaciones usando simulaciones de Monte Carlo no es muy distinto al de bootstrapping, aunque hay algunas diferencias fundamentales en el trasfondo:

1. Formular las hipótesis a contrastar (e identificar el estadístico de interés θ).
2. Crear una gran cantidad P de permutaciones (generalmente terminada en 9 para simplificar los cómputos) a partir de las muestras originales, usando **muestreo sin reposición sobre la muestra combinada**, y obtener el estadístico θ para cada una de las muestras.
3. Generar la distribución que el estadístico θ tendría si la hipótesis nula fuese cierta.

4. Determinar la probabilidad de encontrar un valor de θ al menos tan extremo como el observado en la distribución generada.

Debemos fijarnos en que, a diferencia de bootstrapping, las pruebas de permutaciones usan muestreo sin reposición puesto que, si la hipótesis nula fuera cierta, cada permutación de los valores obtenidos en la muestra combinada sería igualmente probable. Así, lo que se hace en cada repetición es tomar una muestra sin repetición de la muestra original (es decir, "reordenar" las observaciones) y asignar aleatoriamente cada observación a uno de los grupos, respetando los tamaños n_A y n_B de las muestras originales.

12.2.1 Prueba de permutaciones para comparar una variable continua en dos muestras independientes

El profesor de una asignatura inicial de programación, que se imparte para estudiantes de primer año de Ingeniería y estudiantes de último año de otras carreras que pueden cursar dicha asignatura como electivo, desea estudiar si existen diferencias en el rendimiento académico de ambos grupos. Para ello, considera una muestra de $n_A = 20$ estudiantes de primer año de Ingeniería y $n_B = 12$ estudiantes de último año de otras carreras.

El profesor ha decidido comparar el promedio de calificaciones finales de ambos grupos, usando para ello una prueba de permutaciones con $P = 5999$ repeticiones y un nivel de significación $\alpha = 0,05$. La diferencia observada para las muestras originales es $\overline{x}_A - \overline{x}_B = -0,017$, sugiriendo que los estudiantes de Ingeniería tienen peores calificaciones. Así, las hipótesis a contrastar son:

Denotando como μ_A al promedio de calificaciones finales de estudiantes de primer año de Ingeniería en el curso inicial de programación bajo estudio, y como μ_B al promedio de calificaciones finales de estudiantes de último año de otras carreras en el mismo curso, entonces:
H_0: $\mu_A - \mu_B = 0$
H_A: $\mu_A - \mu_B \neq 0$

Tras hacer la prueba, la distribución generada se asemeja bastante a la normal, aunque con una ligera asimetría hacia la derecha (figura 12.10), y el valor p obtenido para el contraste de hipótesis es $p = 0,969$, por lo que concluye con 95 % de confianza que no existe diferencia entre las calificaciones finales de ambos grupos de estudiantes.

Intrigado por este resultado, pues el profesor tiene la fuerte sensación de que, en general, los estudiantes de Ingeniería tienen más calificaciones deficientes que los estudiantes de otras carreras, ha decidido hacer un nuevo estudio con las mismas muestras, comparando ahora la diferencia en la variabilidad (manteniendo la misma cantidad de repeticiones e igual nivel de significación). Así:

Denotando como σ_A a la varianza de las calificaciones finales de estudiantes de primer año de Ingeniería en el curso inicial de programación bajo estudio, y como σ_B a la varianza de las calificaciones finales de estudiantes de último año de otras carreras en el mismo curso, entonces:
H_0: $\sigma_A - \sigma_B = 0$
H_A: $\sigma_A - \sigma_B \neq 0$

La diferencia observada entre las varianzas de la muestra original es $\sigma x_A - \sigma x_B = 2,560$, sugiriendo que la variabilidad de las calificaciones obtenidas por los estudiantes de ingeniería es mayor. Tras efectuar el contraste de hipótesis, obtiene como resultado $p = 0,003$, evidencia suficiente para rechazar la hipótesis nula en favor de la hipótesis alternativa. Así, el profesor concluye que su percepción no es del todo errada, puesto que la variabilidad de las calificaciones es significativamente mayor para los estudiantes de Ingeniería.

Para hacer estos estudios, el profesor desarrolló en R el script 12.7. A pesar de que existen algunos paquetes de R para realizar pruebas de permutaciones, hemos decidido en esta ocasión implementar el procedimiento creando la función `contrastar_hipotesis_permutaciones()`, cuya especificación puede leerse en el script

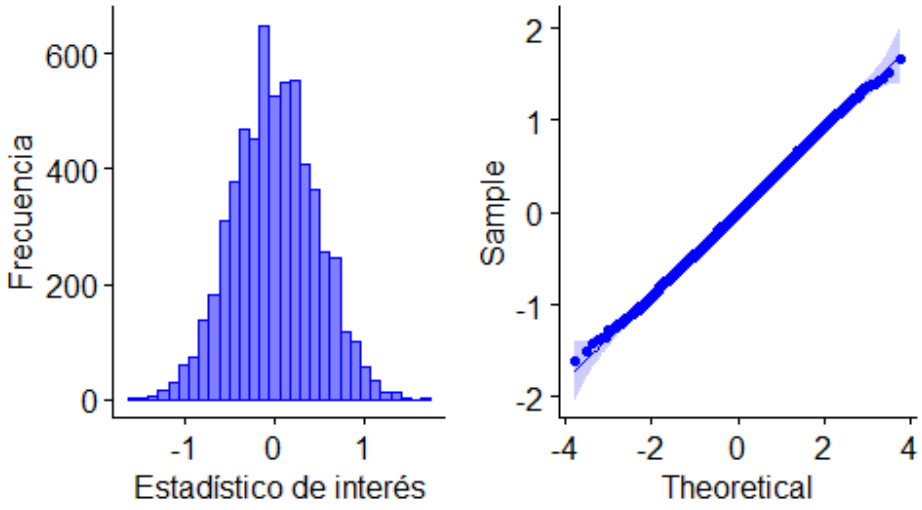


Figura 12.10: histograma y gráfico Q-Q de la distribución para la diferencia de medias generada mediante permutaciones.

12.7, la cual realiza el proceso y arroja como resultado el valor p resultante. Podemos ver que esta función opera usando como estadístico de interés la diferencia de un estadístico θ entre dos muestras y que la función que calcula dicho estadístico θ para una muestra se entrega como argumento.

Script 12.7: pruebas de permutaciones para variables numéricas.

```
library(ggpubr)

set.seed(432)

# Función para calcular la diferencia de medias.
# Argumentos:
# - muestra_1, muestra_2: vectores numéricos con las muestras a comparar.
# - FUN: función del estadístico E para el que se calcula la diferencia.
# Valor:
# - diferencia E_1 - E_2.
calcular_diferencia <- function(muestra_1, muestra_2, FUN){
  diferencia <- FUN(muestra_1) - FUN(muestra_2)
  return(diferencia)
}

# Función para hacer una permutación y calcular el estadístico
# de interés.
# Argumentos:
# - muestra_1, muestra_2: vectores numéricos con las muestras a comparar.
# - FUN: función del estadístico E para el que se calcula la diferencia.
# Valor:
# - diferencia E_1 - E_2.
permutar <- function(muestra_1, muestra_2, FUN) {
  n_1 <- length(muestra_1)
  n_2 <- length(muestra_2)

  # Hacer la permutación.
  permutacion <- sample(c(muestra_1, muestra_2), size = n_1 + n_2,
                        replace = FALSE)

  # Asignar elementos a los dos grupos.
```

```r
  permutacion_1 <- permutacion[1 : n_1]
  permutacion_2 <- permutacion[n_1 + 1 : n_2]

  # Calcular y devolver la diferencia de medias.
  return(calcular_diferencia(permutacion_1, permutacion_2, FUN))
}

# Función para calcular el valor p.
# Argumentos:
# - distribucion: distribución nula del estadístico de interés.
# - valor_observado: valor del estadístico de interés para las muestras
#   originales.
# - repeticiones: cantidad de permutaciones a realizar.
# - alternative: tipo de hipótesis alternativa. "two.sided" para
#   hipótesis bilateral, "greater" o "less" para hipótesis unilaterales.
# Valor:
# - el valorp calculado.
calcular_valor_p <- function(distribucion, valor_observado,
                             repeticiones, alternative) {
  if(alternative == "two.sided") {
    numerador <- sum(abs(distribucion) > abs(valor_observado)) + 1
    denominador <- repeticiones + 1
    valor_p <- numerador / denominador
  }
  else if(alternative == "greater") {
    numerador <- sum(distribucion > valor_observado) + 1
    denominador <- repeticiones + 1
    valor_p <- numerador / denominador
  }
  else {
    numerador <- sum(distribucion < valor_observado) + 1
    denominador <- repeticiones + 1
    valor_p <- numerador / denominador
  }

  return(valor_p)
}

# Función para graficar una distribución.
# Argumentos:
# - distribucion: distribución nula del estadístico de interés.
# - ...: otros argumentos a ser entregados a gghistogram y ggqqplot.
graficar_distribucion <- function(distribucion, ...) {
  observaciones <- data.frame(distribucion)

  histograma <- gghistogram(observaciones, x = "distribucion",
                            xlab = "Estadístico de interés",
                            ylab = "Frecuencia", ...)

  qq <- ggqqplot(observaciones, x = "distribucion", ...)

  # Crear una única figura con todos los gráficos de dispersión.
  figura  <- ggarrange(histograma, qq, ncol = 2, nrow = 1)
  print(figura)
}

# Función para hacer la prueba de permutaciones.
# Argumentos:
# - muestra_1, muestra_2: vectores numéricos con las muestras a comparar.
```

```r
# - repeticiones: cantidad de permutaciones a realizar.
# - FUN: función del estadístico E para el que se calcula la diferencia.
# - alternative: tipo de hipótesis alternativa. "two.sided" para
#   hipótesis bilateral, "greater" o "less" para hipótesis unilaterales.
# - plot: si es TRUE, construye el gráfico de la distribución generada.
# - ...: otros argumentos a ser entregados a graficar_distribucion.
contrastar_hipotesis_permutaciones <- function(muestra_1, muestra_2,
                                               repeticiones, FUN,
                                               alternative, plot, ...) {
  cat("Prueba de permutaciones\n\n")
  cat("Hipótesis alternativa:", alternative, "\n")
  observado <- calcular_diferencia(muestra_1, muestra_2, FUN)
  cat("Valor observado:", observado, "\n")

  distribucion <- rep(NA, repeticiones)

  for(i in 1:repeticiones) {
    distribucion[i] <- permutar(muestra_1, muestra_2, FUN)
  }

  if(plot) {
    graficar_distribucion(distribucion, ...)
  }

  valor_p <- calcular_valor_p(distribucion, observado, repeticiones,
                              "two.sided")

  cat("Valor p:", valor_p, "\n\n")
}

# Crear muestras iniciales.
a <- c(5.4, 4.7, 6.3, 2.9, 5.9, 5.1, 2.1, 6.2, 1.6, 6.7, 3.0, 3.3,
       5.0, 4.1, 3.3, 3.4, 1.2, 3.8, 5.8, 4.2)

b <- c(4.0, 4.1, 4.3, 4.3, 4.3, 4.2, 4.3, 4.3, 4.4, 4.1, 4.3, 4.0)

# Hacer pruebas de permutaciones para la media y la varianza.
R = 5999

contrastar_hipotesis_permutaciones(a, b, repeticiones = R, FUN = mean,
                                   alternative = "two.sided", plot = TRUE,
                                   color = "blue", fill = "blue")

contrastar_hipotesis_permutaciones(a, b, repeticiones = R, FUN = var,
                                   alternative = "two.sided", plot = FALSE)
```

12.2.2 Prueba de permutaciones para comparar medias de más de dos muestras correlacionadas

Supongamos ahora que un estudiante de un curso de programación necesita comparar la eficiencia de tres algoritmos de ordenamiento: *quicksort*, *bubblesort* y *mergesort*. Para ello, ha seleccionado aleatoriamente 6 arreglos de igual tamaño y registrado para cada uno de ellos el tiempo de ejecución utilizado por cada algoritmo (en milisegundos) bajo iguales condiciones, como muestra la tabla 12.4.

Instancia	Quicksort	Bubblesort	Mergesort
1	11,2	15,7	12,0
2	22,6	29,3	25,7
3	23,4	30,7	25,7
4	23,3	30,8	23,7
5	21,8	29,8	25,5
6	40,1	50,3	44,7

Tabla 12.4: tiempos de ejecución para las diferentes instancias con cada algoritmo del ejemplo.

Tras comprobar mediante la figura 12.11 que no se cumple la condición de normalidad, el estudiante ha decidido usar permutaciones para resolver su problema. Para ello, ha considerado un nivel de significación $\alpha = 0,01$ y un total de 2999 repeticiones, obteniendo como resultado un valor p $p = 0,0003$, mucho menor que el nivel de significación. En consecuencia, concluye con 99 % de confianza que el tiempo de ejecución promedio es significativamente diferente para al menos uno de los algoritmos.

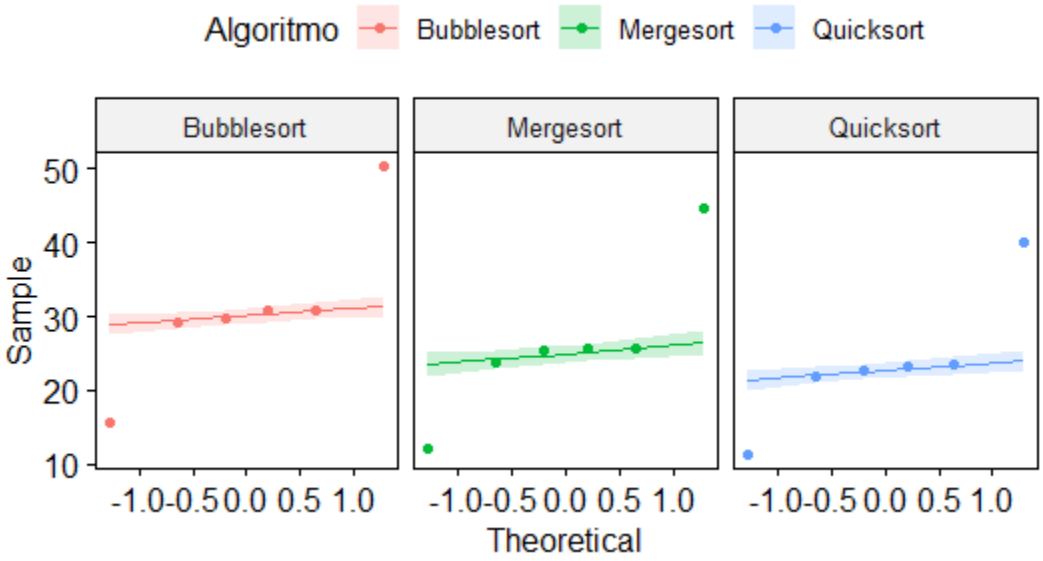


Figura 12.11: gráfico Q-Q para comprobar el supuesto de normalidad para el ejemplo.

A fin de determinar qué algoritmos difieren en su tiempo promedio de ejecución, ha decidido llevar a cabo un procedimiento post-hoc, calculando los valores p para las medias de las diferencias entre cada par de grupos para las diferentes permutaciones, obteniendo los resultados que se presentan en la figura 12.12. En consecuencia, el estudiante concluye con 99 % de confianza, que existen diferencias significativas en el tiempo promedio de ejecución entre los algoritmos Quicksort y Bubblesort y los algoritmos Bubblesort y Mergesort. Al estudiar las diferencias observadas, puede ver que Bubblesort es menos eficiente que los dos algoritmos restantes.

El script 12.8 corresponde a la solución desarrollada por el estudiante.

Script 12.8: prueba de permutaciones para muestras correlacionadas.

```
library(boot)
library(ggpubr)
library(ez)
library(tidyverse)

# Crear el data frame.
```

```
Análisis post-hoc (permutaciones) para la diferencia de las medias
--------------------------------------------------------
Valores p:
Quicksort - Bubblesort: 0.000
Quicksort - Mergesort: 0.116
Bubblesort - Mergesort: 0.003

Diferencias observadas:
Quicksort - Bubblesort: -7.367
Quicksort - Mergesort: -2.483
Bubblesort - Mergesort: 4.883
```

Figura 12.12: resultado del procedimiento post-hoc.

```r
Quicksort <- c(11.2, 22.6, 23.4, 23.3, 21.8, 40.1)
Bubblesort <- c(15.7, 29.3, 30.7, 30.8, 29.8, 50.3)
Mergesort <- c(12.0, 25.7, 25.7, 23.7, 25.5, 44.7)
Instancia <- factor(1:6)
datos_anchos <- data.frame(Instancia, Quicksort, Bubblesort, Mergesort)

datos_largos <- datos_anchos %>% pivot_longer(c("Quicksort", "Bubblesort",
                                                "Mergesort"),
                                              names_to = "Algoritmo",
                                              values_to = "Tiempo")

datos_largos[["Algoritmo"]] <- factor(datos_largos[["Algoritmo"]])

# Verificar condición de normalidad.
g <- ggqqplot(datos_largos, "Tiempo", facet.by = "Algoritmo",
              color = "Algoritmo")

print(g)

# Establecer nivel de significación.
alfa <- 0.01

# Obtener el valor observado, correspondiente al estadístico F entregado
# por ANOVA para la muestra original.
anova <- ezANOVA(datos_largos, dv = Tiempo, within = Algoritmo,
                 wid = Instancia, return_aov = TRUE)

valor_observado <- anova[["ANOVA"]][["F"]]

# Generar permutaciones.
R = 2999
permutaciones <- list()
copia_ancha <- data.frame(datos_anchos)

set.seed(432)

for(i in 1:R) {
  copia_ancha[, 2:4] <- t(apply(copia_ancha[, 2:4], 1, sample))

  copia_larga <- copia_ancha %>% pivot_longer(c("Quicksort", "Bubblesort",
                                                "Mergesort"),
                                              names_to = "Algoritmo",
```

```
                                                values_to = "Tiempo")

  copia_larga[["Algoritmo"]] <- factor(copia_larga[["Algoritmo"]])
  permutaciones <- append(permutaciones, list(copia_larga))
}

# Generar distribución de estadísticos F con las permutaciones.
distribucion <- c()

for(i in 1:R) {
  datos <- as.data.frame(permutaciones[i])

  anova <- ezANOVA(datos, dv = Tiempo, within = Algoritmo, wid = Instancia,
                   return_aov = TRUE)

  distribucion <- c(distribucion, anova[["ANOVA"]][["F"]])
}

# Obtener valor p.
p <- (sum(distribucion > valor_observado) + 1) / (R + 1)
cat("ANOVA de una vía para muestras pareadas con permutaciones\n")
cat("p =", p, "\n\n")

# Análisis post-hoc.
# Función para calcular la media de las diferencias para dos columnas de una
# matriz de datos en formato ancho.
media_diferencias <- function(datos, columna_1, columna_2) {
  media <- mean(datos[[columna_1]] - datos[[columna_2]])
  return(media)
}

# Función para generar la distribuciones de la diferencia de medias a
# partir de las permutaciones.
distribucion_diferencias <- function(permutaciones, columna_1, columna_2) {
  R <- length(permutaciones)
  distribucion <- c()

  for(i in 1:R) {
    datos <- as.data.frame(permutaciones[i])

    datos <- datos %>% pivot_wider(names_from = "Algoritmo",
                                   values_from = "Tiempo")

    diferencia <- media_diferencias(datos, columna_1, columna_2)
    distribucion <- c(distribucion, diferencia)
  }

  return(distribucion)
}

if (p < alfa) {
  quick <- 2
  bubble <- 3
  merge <- 4

  # Calcular diferencias observadas en la muestra original.
  dif_obs_quick_bubble <- media_diferencias(datos_anchos, quick, bubble)
  dif_obs_quick_merge <- media_diferencias(datos_anchos, quick, merge)
  dif_obs_bubble_merge <- media_diferencias(datos_anchos, bubble, merge)
```

```
108
109   # Generar distribuciones para diferencias entre pares a partir de las
110   # permutaciones.
111   dif_quick_bubble <- distribucion_diferencias(permutaciones, quick, bubble)
112   dif_quick_merge <- distribucion_diferencias(permutaciones, quick, merge)
113   dif_bubble_merge <- distribucion_diferencias(permutaciones, bubble, merge)
114
115   # Obtener valores p.
116   num <- sum(abs(dif_quick_bubble) > (abs(dif_obs_quick_bubble) + 1))
117   den <- R + 1
118   p_quick_bubble <- num / den
119
120   num <- sum(abs(dif_quick_merge) > abs(dif_obs_quick_merge) + 1)
121   den <- R + 1
122   p_quick_merge <- num / den
123
124   num <- sum(abs(dif_bubble_merge) > abs(dif_obs_bubble_merge) + 1)
125   den <- R + 1
126   p_bubble_merge <- num / den
127
128   cat("\n\n")
129   cat("Análisis post-hoc (permutaciones) para la diferencia de las medias\n")
130   cat("-----------------------------------------------------------\n")
131   cat("Valores p:\n")
132
133   cat(sprintf("Quicksort - Bubblesort: %.3f\n", p_quick_bubble))
134   cat(sprintf("Quicksort - Mergesort: %.3f\n", p_quick_merge))
135   cat(sprintf("Bubblesort - Mergesort: %.3f\n", p_bubble_merge))
136
137   cat("\nDiferencias observadas:\n")
138   cat(sprintf("Quicksort - Bubblesort: %.3f\n", dif_obs_quick_bubble))
139   cat(sprintf("Quicksort - Mergesort: %.3f\n", dif_obs_quick_merge))
140   cat(sprintf("Bubblesort - Mergesort: %.3f\n", dif_obs_bubble_merge))
141 }
```

12.3 EJERCICIOS PROPUESTOS

1. En tus palabras, ¿qué son las técnicas de remuestreo?
2. Explica si ¿podría considerarse que la prueba exacta de Fisher usa técnicas de remuestreo?
3. ¿En qué se parecen y en qué se diferencias las técnicas de bootstrapping y permutación?
4. En tus palabras, ¿qué es una simulación Monte Carlo?
5. ¿Cómo realizarías bootstrapping para determinar si la estatura media de estudiantes de Las Condes es igual a la estatura media de estudiantes de La Pintana?
6. ¿Cómo usarías Monte Carlo para verificar que un medicamento para bajar el colesterol funciona en un grupo de 30 personas escogidas al azar?
7. ¿Cómo usarías bootstrapping para analizar si tres algoritmos necesitan tiempos similares en procesar 25 instancias de prueba del problema de la mochila?
8. ¿Cómo usarías Monte Carlo para conocer si un medicamento para bajar la presión funciona al administrarlo a un grupo de personas hipertensas, comparando con un grupo de personas hipertensas recibiendo placebo y un grupo de control de personas no hipertensas?

CAPÍTULO 13. REGRESIÓN LINEAL

En el capítulo 2 introdujimos los gráficos de dispersión como una herramienta que nos permite identificar posibles relaciones entre dos variables cuantitativas. En este capítulo estudiaremos la **regresión lineal simple** (RLS), herramienta que sistematiza esta idea, basándonos en los textos de Diez y col. (2017, pp. 331-355), Field y col. (2012, pp. 245-311) e Irizarry (2019, pp. 535-545)

La RLS asume que la relación entre dos variables, x e y, puede ser modelada mediante **una recta** de la forma que se presenta en la ecuación 13.1, donde:

- β_0 y β_1 son los parámetros del modelo lineal.
- x es la variable explicativa o **predictor** (variable independiente).
- y es la variable **de respuesta** o **de salida** (variable dependiente).

$$\hat{y} = \beta_0 + \beta_1 x \tag{13.1}$$

Llamamos **intercepción** (*intercept*, en inglés) al parámetro β_0, que corresponde al punto en que la recta corta el eje y. A su vez, denominamos **pendiente** al parámetro β_1, el cual determina la inclinación de la recta del modelo.

Si tuviéramos una relación lineal perfecta entre ambas variables, significaría que se podríamos conocer el valor exacto de y con solo conocer el valor de x. Sin embargo, como podemos apreciar en la figura 13.1, rara vez los datos se ajustan al modelo con exactitud.

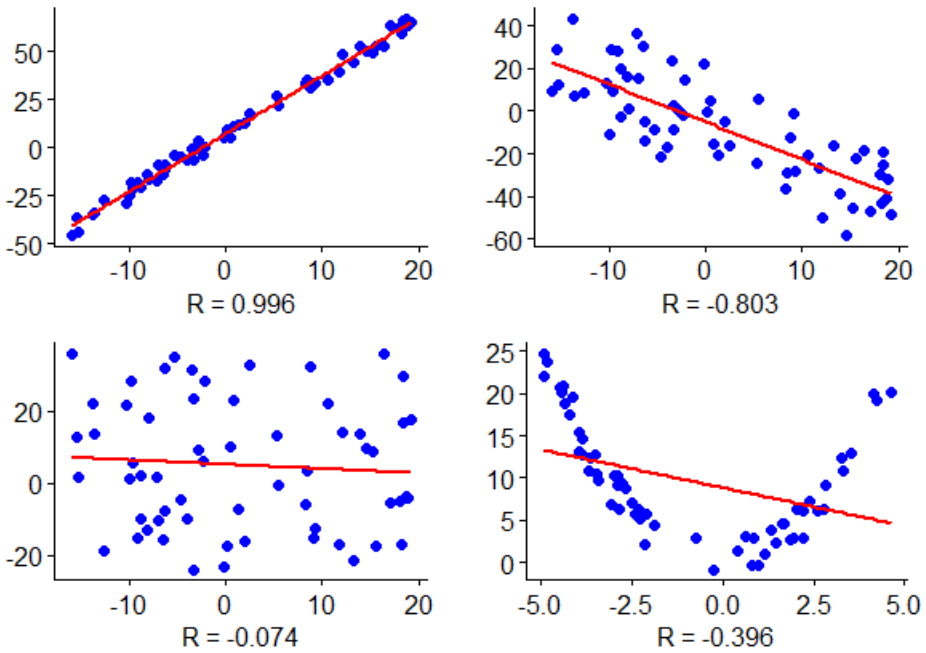


Figura 13.1: modelos lineales para cuatro conjuntos de datos.

Los gráficos de la fila superior de la figura 13.1 muestran dos tendencias lineales, siendo la izquierda una relación directa y muy fuerte, y la de la derecha, inversa y algo más débil. En el caso de los gráficos de la fila inferior, los datos en el de la izquierda no se aglutinan en torno a la recta marcada, y los de la derecha, presentan un escenario donde ambas variables se relacionan clara y fuertemente, pero de manera no lineal.

Siempre tenemos que tener en cuenta que, si los datos presentan una tendencia no lineal, debemos usar herramientas más avanzadas que la regresión lineal simple.

Fijémonos en el siguiente modelo lineal, que corresponde a la línea roja en el gráfico de arriba a la izquierda de la figura 13.1:

$$\hat{y} = 7 + 3x$$

En él, si $x = 5$, entonces $\hat{y} = 22$. $\hat{y}$ es un estimador que podemos entender de la siguiente manera: dado un valor de x, el valor de y es, en promedio, $\hat{y}$. En otras palabras, $\hat{y}$ corresponde al valor esperado de y para un determinado valor de x. En la práctica, existe una diferencia entre el valor esperado $\hat{y}$ y el valor observado de y. Esta diferencia se denomina **residuo** y se denota e. Así, tenemos que el valor observado de y está dado por la ecuación 13.2.

$$y = \hat{y} + e \tag{13.2}$$

Otra forma de entender el residuo es como la distancia que separa a la observación de la recta. Si la observación se encuentra por sobre esta última, entonces $e > 0$. En caso contrario, $e < 0$. Puesto que los residuos sirven para evaluar qué tan bien se ajusta un modelo lineal al conjunto de datos, suelen mostrarse en un **gráfico de residuos**, el cual es sencillamente un gráfico de dispersión donde la variable predictora se representa en su escala original y el eje y muestra el residuo para cada observación. La figura 13.2 muestra los residuos para los modelos lineales de la figura 13.1.

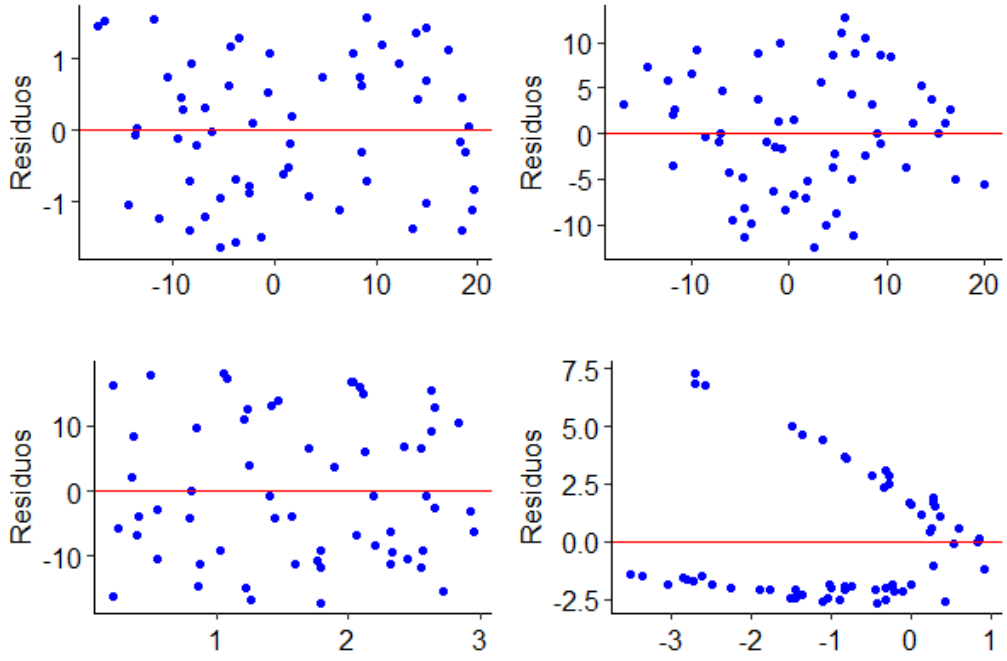


Figura 13.2: residuos para los modelos lineales de la figura 13.1.

13.1 CORRELACIÓN

Hasta ahora hemos hablado de la fuerza de una relación lineal entre dos variables, concepto que hemos asociado implícitamente a la magnitud de los residuos. Formalmente, podemos medir la fuerza de una relación lineal mediante la **correlación**. Una de las formas más sencillas para calcularla es el coeficiente de correlación de Pearson, dado por la ecuación 13.3, donde:

- $\overline{x}$, $\overline{y}$ son las medias de las variables x e y en la muestra.
- s_x, s_y corresponden a las desviaciones estándar de las de las variables x e y en la muestra.
- n es el tamaño de la muestra.

$$R = \frac{1}{n-1} \cdot \sum_{i=1}^{n} \frac{x_i - \overline{x}}{s_x} \cdot \frac{y_i - \overline{y}}{s_y} \tag{13.3}$$

La correlación siempre toma un valor entre -1 y 1. Mientras más débil sea la relación entre dos variables, su valor será más cercano a 0. El signo de la correlación indica si la relación es directa ($R > 0$) o inversa ($R < 0$). Para comprender mejor esta idea, fijémonos los coeficientes de correlación obtenidos para cada modelo lineal de la figura 13.1, indicados en las etiquetas del eje x. Podemos ver que, en el caso de abajo a la derecha, la relación es muy fuerte, pero no lineal, por lo que R, al considerar solo una recta, toma un valor relativamente bajo.

Para los ejemplos de este capítulo usaremos el conjunto de datos `mtcars`, disponible en R, que contiene diversas características para $n = 32$ modelos de automóviles de los años 1973 y 1974. La tabla 13.1 describe brevemente cada una de las variables de dicho conjunto.

Columna	Descripción
mpg	Rendimiento, en millas (EEUU) por galón [millas/galón].
cyl	Cantidad de cilindros del motor.
disp	Volumen útil de los cilindros de un motor, en centímetros cúbicos [cc].
hp	Potencia del motor, en caballos de fuerza [hp].
drat	Relación del eje trasero (proporción).
wt	Peso total, en miles de libras.
qsec	Tiempo mínimo para recorrer un cuarto de milla (desde el reposo), en segundos [s].
vs	Tipo de motor (0 = en forma de V, 1 = recto).
am	Transmisión (0 = automática, 1 = manual).
gear	Número de marchas hacia adelante.
carb	Número de carburadores.

Tabla 13.1: descripción de las variables para el conjunto de datos `mtcars` usados en este capítulo.

Si consideramos a x como el rendimiento del vehículo y a y como la potencia del motor, cuya relación se muestra gráficamente en la figura 13.3, tenemos que: $\overline{x} = 20,091$, $s_x = 6,027$, $\overline{y} = 146,688$ y $s_y = 68,563$. En consecuencia, la correlación es:

$$R = \frac{1}{32-1} \cdot \sum_{i=1}^{n} \frac{x_i - 20,091}{6,027} \cdot \frac{y_i - 146,688}{68,563} = -0,776$$

En R, podemos calcular la correlación entre dos variables usando la función `cor(x, y)`, donde `x` es el predictor e `y` la respuesta. Para el ejemplo obtenemos que $R = -0,7761684$, lo que coincide con el resultado teórico teniendo en cuenta que la diferencia se debe únicamente al redondeo.

Adicionalmente, cuando `x` es una matriz de datos, la función `cor(x)` nos entrega una **matriz de correlación**, que contiene las correlaciones entre todos los pares de variables. La figura 13.4 muestra la matriz de correlación

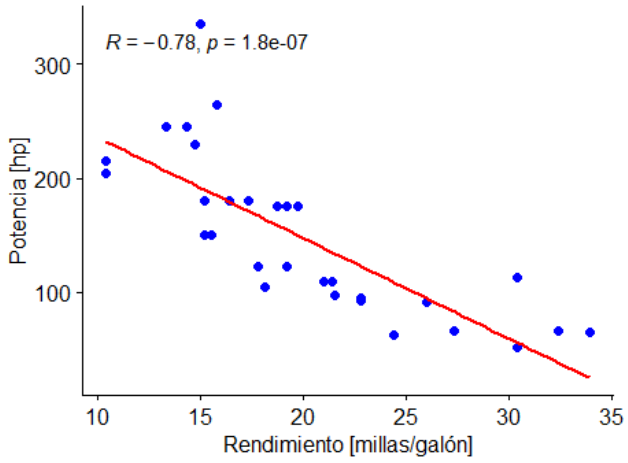


Figura 13.3: Relación entre el rendimiento y la potencia.

(redondeada al segundo decimal) para el conjunto de datos `mtcars`. Podemos ver que, naturalmente, la matriz de correlación es simétrica y que su diagonal solo contiene unos.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
mpg	1.00	-0.85	-0.85	-0.78	0.68	-0.87	0.42	0.66	0.60	0.48	-0.55
cyl	-0.85	1.00	0.90	0.83	-0.70	0.78	-0.59	-0.81	-0.52	-0.49	0.53
disp	-0.85	0.90	1.00	0.79	-0.71	0.89	-0.43	-0.71	-0.59	-0.56	0.39
hp	-0.78	0.83	0.79	1.00	-0.45	0.66	-0.71	-0.72	-0.24	-0.13	0.75
drat	0.68	-0.70	-0.71	-0.45	1.00	-0.71	0.09	0.44	0.71	0.70	-0.09
wt	-0.87	0.78	0.89	0.66	-0.71	1.00	-0.17	-0.55	-0.69	-0.58	0.43
qsec	0.42	-0.59	-0.43	-0.71	0.09	-0.17	1.00	0.74	-0.23	-0.21	-0.66
vs	0.66	-0.81	-0.71	-0.72	0.44	-0.55	0.74	1.00	0.17	0.21	-0.57
am	0.60	-0.52	-0.59	-0.24	0.71	-0.69	-0.23	0.17	1.00	0.79	0.06
gear	0.48	-0.49	-0.56	-0.13	0.70	-0.58	-0.21	0.21	0.79	1.00	0.27
carb	-0.55	0.53	0.39	0.75	-0.09	0.43	-0.66	-0.57	0.06	0.27	1.00

Figura 13.4: matriz de correlación para el conjunto de datos `mtcars`

13.2 REGRESIÓN LINEAL MEDIANTE MÍNIMOS CUADRADOS

Si bien existen diversos métodos para ajustar un modelo lineal, el más empleado es el de la **línea de mínimos cuadrados**, que minimiza la suma de los cuadrados de los residuos (ecuación 13.4).

$$\text{mín} \sum_{i=1}^{n} e_i^2 \tag{13.4}$$

El método de mínimos cuadrados tiene las ventajas de ser fácil de calcular y de tomar en cuenta la discrepancia entre la magnitud del residuo y su efecto. Como señalan Diez y col. (2017, p. 341), "por ejemplo, desviarse por 4 suele ser más de dos veces peor que desviarse por 2". No obstante, para aplicar este método debemos verificar que se cumplan algunas condiciones:

1. Los datos deben presentar una relación lineal.
2. La distribución de los residuos debe ser cercana a la normal.
3. La variabilidad de los puntos en torno a la línea de mínimos cuadrados debe ser aproximadamente constante.
4. Las observaciones deben ser independientes entre sí. Esto significa que no se puede usar regresión lineal con series de tiempo (tema que va más allá de los alcances de este texto).

Los gráficos de residuos reflejan cuando no se cumplen las condiciones anteriores. Por ejemplo, el gráfico inferior derecho de la figura 13.1 aplica regresión lineal entre un par de variables cuya relación es, en realidad, cuadrática. Esta relación se puede ver en la forma en que se distribuyen los puntos en el gráfico de residuos correspondiente de la figura 13.2.

La figura 13.5 muestra, en la fila superior, dos modelos lineales en los que los datos no cumplen las condiciones, con sus respectivos gráficos de residuos en la fila inferior. A la izquierda, se viola la condición de normalidad de los residuos, que no se distribuyen aleatoriamente en torno a la línea 0. A la derecha, no se respeta la condición de homocedasticidad, y la variabilidad de los puntos en torno a la línea de mínimos cuadrados no es aproximadamente constante, lo que genera una característica forma de embudo en el gráfico de residuos.

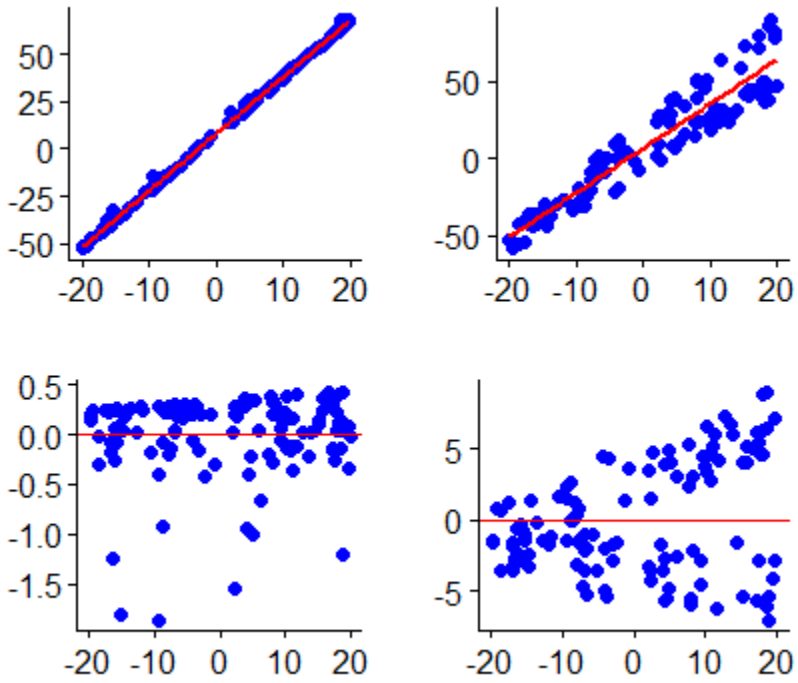


Figura 13.5: modelos lineales (fila superior) que violan alguna condición y sus residuos (fila inferior).

El primer paso que debemos seguir cuando queremos determinar la recta de mínimos cuadrados para un conjunto de datos consiste en estimar la pendiente (β_1) mediante la ecuación 13.5, donde:

- s_x y s_y son las desviaciones estándares muestrales de las variables x e y, respectivamente.
- R corresponde a la correlación entre ambas variables.

$$b_1 = \frac{s_y}{s_x} \cdot R \tag{13.5}$$

El punto $(\overline{x}, \overline{y})$, donde $\overline{x}$ e $\overline{y}$ son las medias muestrales para las variables representadas en los ejes x e y respectivamente, siempre pertenece a la recta de mínimos cuadrados, por lo que podemos calcular la intercepción mediante la ecuación 13.6 (Winner, 2021, p. 4).

$$b_0 = \overline{y} - b_1 \cdot \overline{x} \tag{13.6}$$

Cuando contamos con más de una variable para construir una regresión lineal simple (RLS), lo más adecuado es que escojamos como predictor aquella variable que tenga la correlación más fuerte con la variable de respuesta. Para el caso del conjunto de datos `mtcars`, la variable que presenta la correlación más fuerte con el rendimiento (`mpg`) es el peso del automóvil (`wt`), con $R = -0,87$, como podemos ver en la figura 13.4. Así, si usamos como predictor la variable peso, tenemos que la pendiente de la recta es:

$$b_1 = \frac{6,027}{0,978} \cdot -0,868 = -5,344$$

A su vez, la intercepción está dada por:

$$b_0 = 20,091 + 5,344 \cdot 3,217 = 37,285$$

Por lo que la recta ajustada mediante mínimos cuadrados es:

$$\widehat{\texttt{mpg}} = 37,285 - 5,344 \cdot \texttt{wt}$$

Desde luego, R ofrece una función que permite ajustar la recta de mínimos cuadrados para un par de variables: `lm(formula, data)`, donde:

- `formula`: tiene la forma <variable de respueta>∼<variable predictora>.
- `data`: matriz de datos.

El script 13.1 ajusta la línea de mínimos cuadrados para la variable de respuesta rendimiento (`mpg`), con la variable peso (`wt`) como predictor, mediante el uso de `lm()`. En la figura 13.6, bajo el encabezado `Coefficients`, podemos ver que los valores estimados para los parámetros de la RLS coinciden con los obtenidos previamente.

Un detalle interesante es que, en la línea 8 del script 13.1, usamos la llamada `print(summary(modelo))` en lugar de `print(modelo)`, lo que nos entrega información más detallada del modelo ajustado (figura 13.6). La segunda solo muestra los coeficientes obtenidos.

```
Call:
lm(formula = mpg ~ wt, data = datos)

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
```

Figura 13.6: regresión lineal simple para predecir el rendimiento de un automóvil a partir de su peso.

La figura 13.7 muestra la recta ajustada para el rendimiento de un automóvil de acuerdo a su peso (script 13.1, líneas 11–15), donde podemos observar que los datos presentan una relación lineal, aunque algunos puntos parecen estar algo alejados de la recta ajustada. A su vez, la figura 13.8 muestra diversos gráficos obtenidos mediante la línea 18 del script 13.1, donde vemos que la variabilidad de los residuos no es muy grande (figura 13.8a) y, en general, sigue una distribución razonablemente cercana a la normal (figura 13.8b), aunque en ambas figuras se aprecian unos pocos modelos que se comportan como valores atípicos. Además, podemos suponer que las observaciones son independientes entre sí y, evidentemente, no corresponden a una serie de tiempo. Con este análisis verificamos las condiciones 1 y 4 para emplear la regresión lineal de mínimos cuadrados, aunque las dos condiciones restantes parecen no cumplirse.

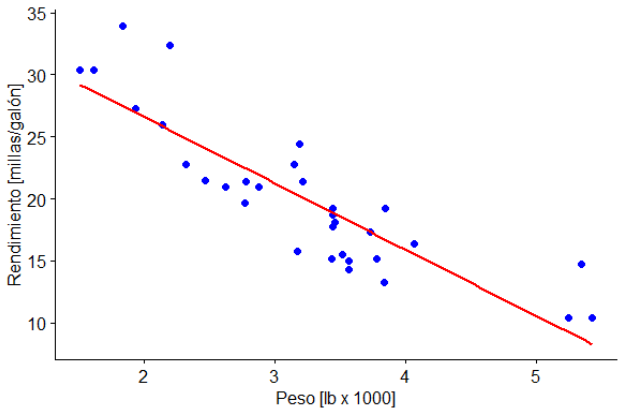


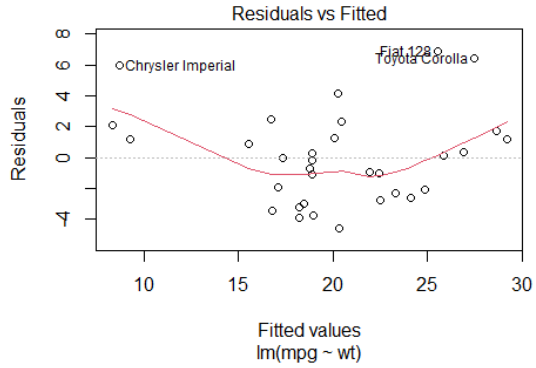
Figura 13.7: recta ajustada para el rendimiento de un automóvil de acuerdo a su peso.

Si bien para fines del ejercicio supondremos que las condiciones se cumplen, vale la pena que revisemos algunas características que, según Pardoe y col. (2018), se observan en el gráfico de los residuos cuando sí se verifican todas las condiciones o, en otras palabras, cuando el modelo de RLS es apropiado:

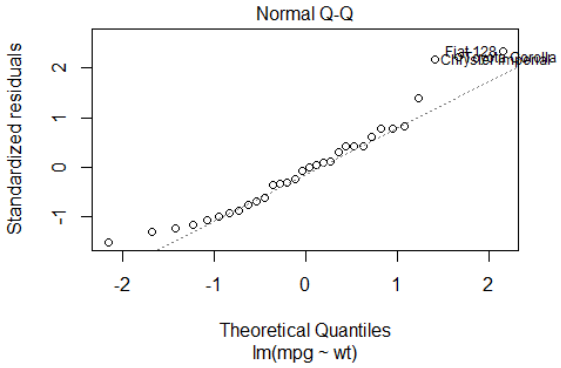
1. Un gráfico en que los residuos se distribuyen aleatoriamente en torno a la línea de valor 0, sugiere que es razonable suponer que las variables presentan una relación lineal.
2. Cuando los residuos forman una "banda horizontal´´ en torno a la línea de valor 0, sugiere una variabilidad aproximadamente constante de los residuos.
3. La ausencia de residuos que se alejen del patrón que forman los demás sugiere la ausencia de valores atípicos.

Script 13.1: ajuste de una regresión lineal simple.

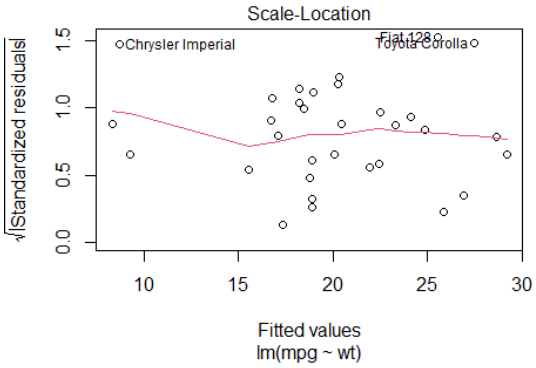
```
1  library(ggpubr)
2
3  # Cargar los datos.
4  datos <- mtcars
5
6  # Ajustar modelo con R.
7  modelo <- lm(mpg ~ wt, data = datos)
8  print(summary(modelo))
9
10 # Graficar el modelo.
11 p <- ggscatter(datos, x = "wt", y = "mpg", color = "blue", fill = "blue",
12                 xlab = "Peso [lb x 1000]", ylab = "Rendimiento [millas/galón]")
13
14 p <- p + geom_smooth(method = lm, se = FALSE, colour = "red")
15 print(p)
16
17 # Crear gráficos para evaluar el modelo.
18 plot(modelo)
19
```



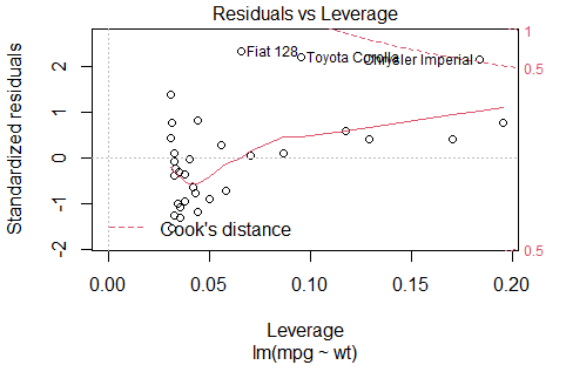
(a) residuos.



(b) distribución de los residuos.



(c) residuos estandarizados.



(d) apalancamiento.

Figura 13.8: gráficos para evaluar el modelo lineal.

```
# Ingresar algunas instancias artificiales.
mpg <- c(23.714, 19.691, 19.242, 12.430, 10.090, 9.565, 18.171, 26.492, 7.054,
         24.447, 15.683, 17.403, 13.465, 18.850, 29.493)

wt <- c(2.973, 4.532, 2.332, 3.016, 4.220, 4.286, 2.580, 3.084, 3.816, 2.775,
        3.251, 3.013, 4.951, 2.644, 2.218)

nuevos <- data.frame(mpg, wt)

# Usar el modelo para predecir el rendimiento de los nuevos  y ver los
# residuos resultantes.
predicciones <- predict(modelo, nuevos)
residuos <- nuevos$mpg - predicciones
nuevos <- data.frame(nuevos, residuos)

r <- ggscatter( nuevos, x = "wt", y = "residuos", color = "blue",
                fill = "blue", xlab = "Peso [lb * 1000]", ylab = "Residuo")

r <- r + geom_hline(yintercept = 0, colour = "red")
print(r)
```

Una de las etapas más importantes en un proceso de análisis es la **interpretación de los parámetros** del modelo. La pendiente explica la diferencia esperada en el valor de la respuesta y si el predictor x se incrementa

en una unidad. Así, para el ejemplo se espera que, al incrementar en el peso del automóvil en 1.000 libras, el rendimiento se reduzca en 5,344 millas por galón de combustible. A su vez, la intercepción corresponde a la respuesta que se obtendría en promedio si x fuese igual a 0, suponiendo que el modelo fuese válido para $x = 0$, lo que no siempre ocurre. De hecho, para el ejemplo, es imposible que un automóvil carezca de masa.

El párrafo anterior ilustra una limitación propia de cualquier modelo: este, al ser una simplificación de la realidad, tiene validez únicamente en el rango de valores de los datos originales, por lo que la **extrapolación** (es decir, estimar valores fuera del rango de los datos originales) puede conllevar a errores al asumir que el modelo es válido donde aún no ha sido analizado. El peso de los automóviles del ejemplo varía entre 1.500 y 5.500 libras aproximadamente, por lo que si lo usáramos para predecir el rendimiento de un vehículo de 7.000 libras o de solo 980, el resultado podría carecer de validez.

Desde luego, debemos tener en cuenta también las condiciones de diseño del modelo. El resultado podría ser equivocado si intentáramos predecir, por ejemplo, el rendimiento de un automóvil moderno (recordemos que el conjunto de datos solo contiene vehículos de los años 1973 y 1974). Más aún, la variable de respuesta carece absolutamente de sentido si pensamos, por ejemplo, en un automóvil eléctrico.

13.3 USO DEL MODELO

Supongamos que queremos predecir el rendimiento de un auto norteamericano (modelo 1974) cuyo peso es de 4.260 libras (es decir, $\texttt{wt} = 4,260$). Para ello, basta con reemplazar el valor del predictor en el modelo:

$$\widehat{\texttt{mpg}} = 37,285 - 5,344 \cdot 4,260 = 14,520$$

En R, la función `predict(object, newdata)` nos permite usar un modelo (en este caso, una RLS) para predecir una respuesta. Los argumentos de esta función son:

- `object`: el modelo a emplear.
- `newdata`: matriz de datos con las nuevas instancias para las que se desea efectuar la predicción, la cual debe tener todas las columnas presentes en la fórmula del modelo (para el ejemplo, mpg y wt).

La línea 31 del script 13.1 ilustra el uso de esta función para un conjunto de 15 instancias generadas artificialmente.

En la línea 32 se calculan los residuos para posteriormente graficarlos (líneas 35–39) a fin de tener una idea preliminar acerca de la calidad de las predicciones. Como resultado obtenemos la figura 13.9, donde podemos observar que los residuos varían en un rango bastante más amplio que para el conjunto de datos original (figura 13.8a).

13.4 REGRESIÓN LINEAL CON UN PREDICTOR CATEGÓRICO

Las variables categóricas también nos pueden servir para predecir una respuesta. En este capítulo solo estudiaremos el caso de una variable dicotómica (es decir, con solo dos niveles), idea que profundizaremos en el siguiente capítulo.

Para usar una variable categórica con dos niveles, tenemos que convertirla a formato numérico, para lo cual creamos una nueva **variable indicadora** que toma los valores 0 y 1. Hacer este proceso en R es bastante

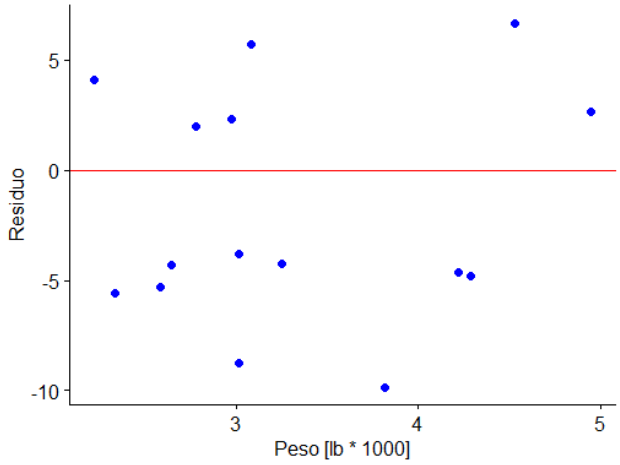


Figura 13.9: residuos obtenidos tras usar el modelo para predecir el rendimiento de nuevos automóviles.

sencillo, como muestra el script 13.2. En la práctica, rara vez tendremos que realizar este paso, pues las funciones de R que ajustan modelos lo hacen automáticamente cuando encuentran predictores categóricos.

Script 13.2: reemplazar una variable dicotómica por una variable indicadora.

```
# Crear un data frame con una variable dicotómica.
alumno <- 1:5
sexo <- factor(c("F", "M", "F","F", "M"))
datos <- data.frame(alumno, sexo)

# Crear una variable indicadora para sexo, con valor 0
# para hombres y 1, para mujeres.
es_mujer <- rep(1, length(sexo))
es_mujer[sexo == "M"] <- 0

# Reemplazar la variable sexo por lavariable indicadora.
datos <- cbind(datos, es_mujer)
datos[["sexo"]] <- NULL
```

El conjunto de datos **mtcars** ya cuenta con un par de variables que cumplen con esta característica: la transmisión (`am`) y la forma del motor (`vs`). De estas dos variables, la forma del motor tiene una correlación más fuerte con el rendimiento, por lo que la usaremos como ejemplo para crear un modelo RLS. Al crear el modelo (script 13.3) obtenemos como resultado la recta representada en el gráfico superior de la figura 13.10, con los residuos del gráfico inferior en la misma figura. A su vez, la figura 13.11 muestra los valores obtenidos para los parámetros del modelo.

Cuando usamos un predictor dicotómico siempre se cumple la condición de que los datos presentan una relación lineal. Sin embargo, debemos verificar que la distribución de los residuos de ambos grupos se asemeje a la normal y que tengan varianzas similares. El panel superior de la figura 13.10 muestra que, en efecto, las variabilidades de los residuos de ambos grupos son independientes y la figura 13.12 muestra que la distribución de los residuos se acerca a la normal para ambos tipos de transmisión, por lo que se verifican las condiciones.

Script 13.3: alternativa robusta para comparar entre múltiples grupos correlacionados.

```
library(ggpubr)

# Cargar los datos.
datos <- mtcars

# Ajustar modelo con R.
```

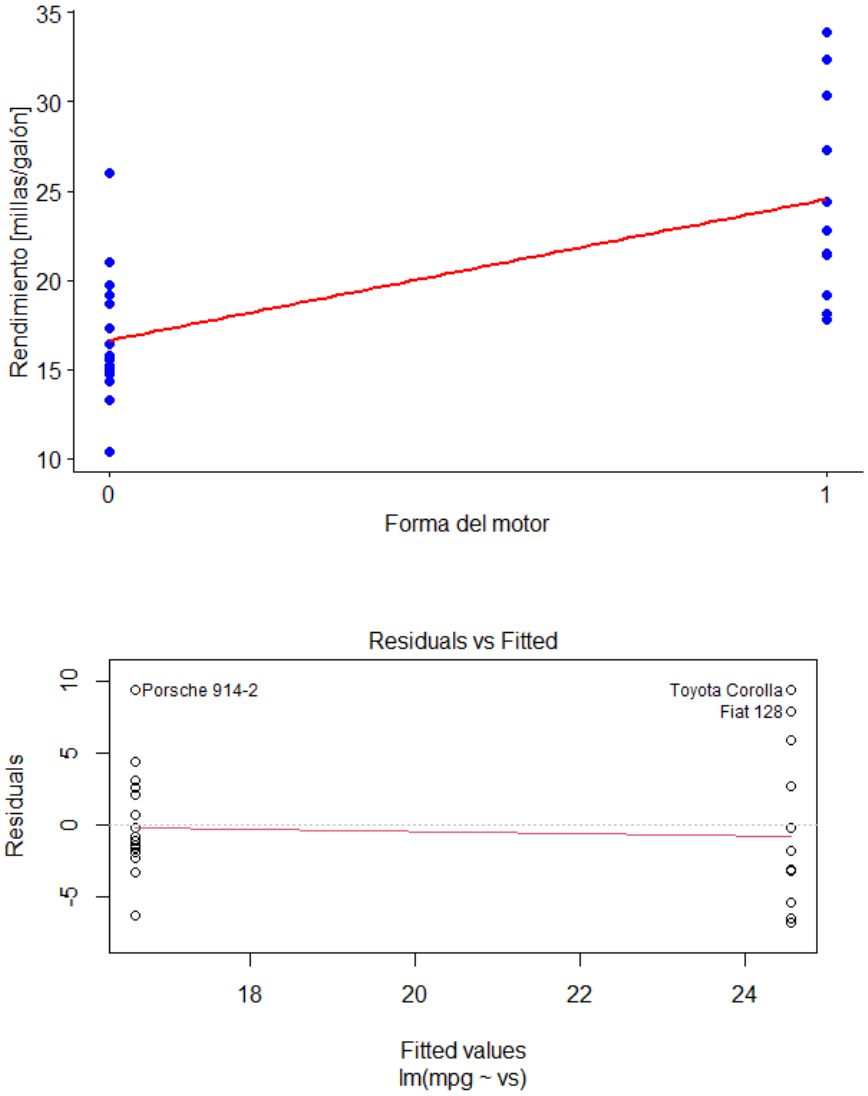


Figura 13.10: modelo de regresión lineal y gráfico de residuos para el ejemplo con un predictor dicotómico.

```
modelo <- lm(mpg ~ vs, data = datos)
print(summary(modelo))

# Graficar el modelo.
p <- ggscatter(datos, x = "vs", y = "mpg", color = "blue", fill = "blue",
               xlab = "Forma del motor", ylab = "Rendimiento [millas/galón]",
               xticks.by = 1)

p <- p + geom_smooth(method = lm, se = FALSE, colour = "red")
print(p)

# Crear gráficos para evaluar el modelo.
plot(modelo)

#Graficar residuos.
residuos <- modelo$residuals
datos <- cbind(datos, residuos)
datos[["vs"]] <- factor(datos[["vs"]])
```

```
Call:
lm(formula = mpg ~ vs, data = datos)

Residuals:
   Min     1Q Median     3Q    Max
-6.757 -3.082 -1.267  2.828  9.383

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   16.617      1.080  15.390 8.85e-16 ***
vs             7.940      1.632   4.864 3.42e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.581 on 30 degrees of freedom
Multiple R-squared:  0.4409,Adjusted R-squared:  0.4223
F-statistic: 23.66 on 1 and 30 DF,  p-value: 3.416e-05
```

Figura 13.11: recta de mínimos cuadrados para el ejemplo con un predictor dicotómico.

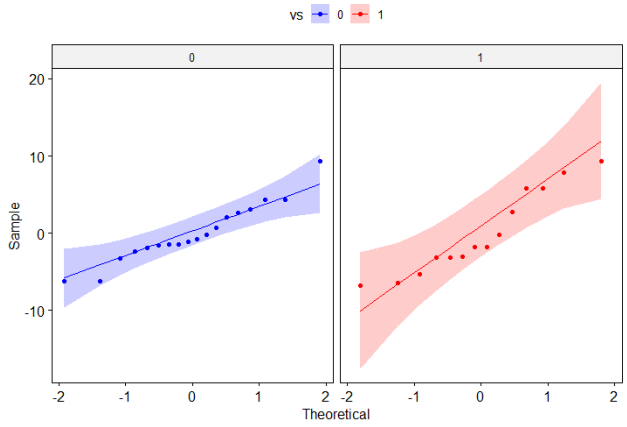


Figura 13.12: distribución de los residuos.

```
25
26 r <- ggqqplot(datos, x = "residuos", facet.by = "vs", color = "vs",
27                palette = c("blue", "red"))
28
29 print(r)
```

13.5 EVALUACIÓN DE UN MODELO DE RLS

Hasta ahora hemos visto cómo ajustar y usar un modelo de RLS. Sin embargo, no hemos realizado algunas verificaciones importantes antes de hacer predicciones, pues puede ocurrir que la recta ajustada esté fuertemente influenciada por un pequeño grupo de valores atípicos o que no pueda generalizarse para otras muestras.

13.5.1 Influencia de los valores atípicos

Hemos dicho que, en muchas ocasiones, los valores atípicos influyen significativamente en el incumplimiento de las condiciones que debemos verificar para poder usar una regresión lineal. Sin embargo, no todos los valores atípicos son perjudiciales. La figura 13.13 muestra, para seis conjuntos de datos, los gráficos de dispersión (incluyendo la línea de regresión) y sus respectivos gráficos de los residuos. En cada uno de ellos se evidencia la presencia de al menos un valor atípico. Como señalan Diez y col. (2017, p. 349):

- En (1) hay un valor atípico que se aleja mucho de la nube de puntos, pero que no parece tener mucha influencia en la línea de regresión.
- En (2), se observa un valor atípico, a la derecha y bastante cercano a la línea de regresión, que no parece tener gran influencia.
- Nuevamente aparece un valor atípico a la derecha en (3), el cual parece ser el causante de que la línea de regresión no se ajuste muy bien a la nube principal de puntos.
- En (4), los datos se agrupan en dos nubes, una principal y la otra (secundaria) con cuatro valores atípicos. La nube secundaria parece influenciar fuertemente la línea de regresión, haciendo que se ajuste pobremente a los datos de la nube principal.
- La nube principal no evidencia tendencia alguna (pendiente cercana a cero) en (5), y el valor atípico a la derecha parece ejercer una gran influencia en la línea de regresión.
- En (6) se observa un valor atípico a la izquierda que se aleja bastante de la nube principal. Sin embargo, no parece ejercer mucha influencia en la línea de regresión y se sitúa cerca de ella.

Los valores atípicos que se alejan horizontalmente del centro de la nube principal de puntos pueden, potencialmente, tener una gran influencia en el ajuste de la línea de regresión. Este fenómeno se conoce como **apalancamiento** (*leverage* en inglés), pues dichos puntos parecen tirar de la línea hacia ellos. Cuando un valor atípico ejerce efectivamente esta influencia, decimos que es un **punto influyente**. Una forma de saber si un punto es o no influyente es determinar la línea de regresión sin considerar dicho punto y ver cuánto se aleja este último de la nueva línea. Si miramos la figura 13.8d, podemos detectar dos observaciones atípicas que influyen en el ajuste del modelo.

Si bien puede resultar tentador descartar los valores atípicos antes de ajustar un modelo, no es pertinente llevar a cabo esta acción sin hacer un riguroso análisis previo. En muchos casos los valores atípicos resultan ser las observaciones más interesantes. Diez y col. (2017, p. 349) ilustran esta idea con el ejemplo de acciones de la bolsa con valores excepcionalmente altos. Si omitieran estos valores atípicos, los agentes de bolsa perderían los mejores negocios.

Un buen método para identificar valores atípicos es usar los residuos estandarizados (figura 13.8c) (es decir, divididos por la estimación de su desviación estándar), pues esto nos permite establecer un rango fijo de valores aceptables y, en consecuencia, fijar un criterio para comparar residuos de distintos modelos.

Debemos ser cuidadosos cuando usemos como predictores variables categóricas que tengan pocas observaciones en alguno de sus niveles, pues cuando esto ocurre, dichas observaciones se convierten en puntos influyentes.

13.5.2 Bondad de ajuste

Una medida muy útil que podemos usar para evaluar la **bondad de ajuste** de un modelo de regresión lineal con respecto a las observaciones es el **coeficiente de determinación**, que corresponde al cuadrado de la correlación, por lo que suele también denominarse R-cuadrado (R^2) (Glen, 2021a). Esta medida, cuyo valor varía entre 0 y 1, corresponde al porcentaje de la variabilidad de la respuesta que es explicado por el predictor, dado por la ecuación 13.7, donde s_e^2 es la varianza de los residuos.

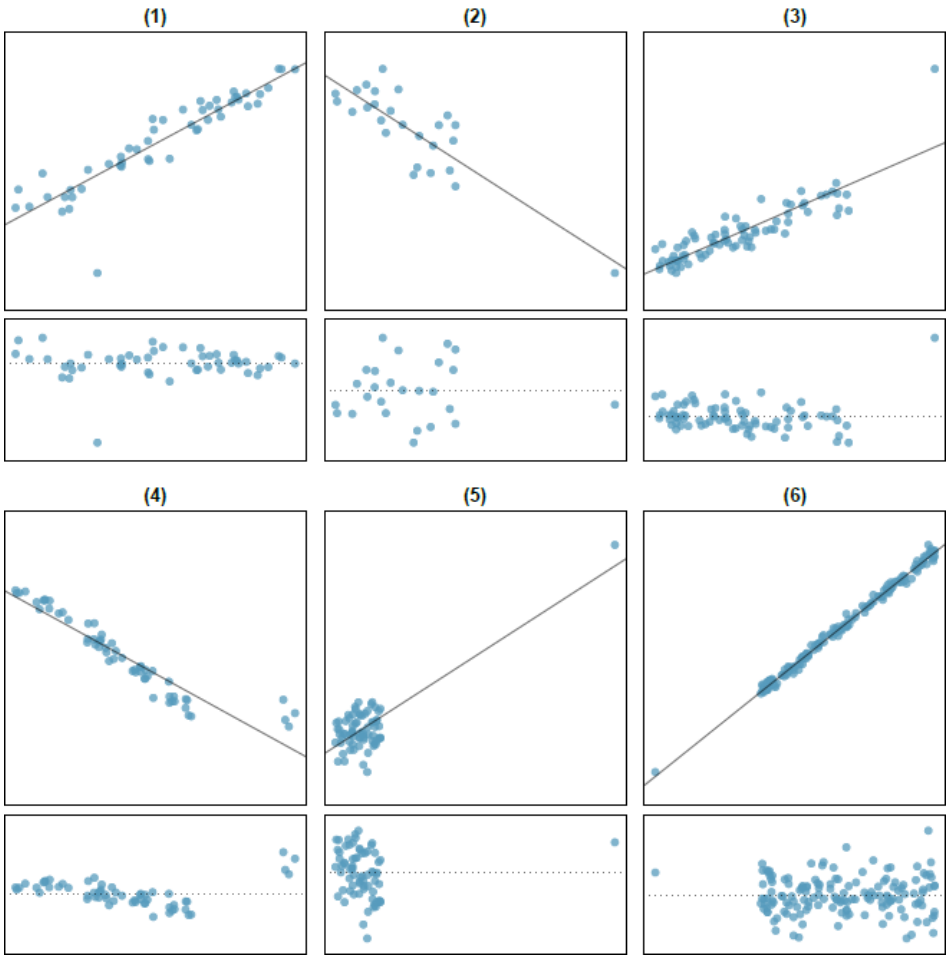


Figura 13.13: seis modelos de regresión lineal con sus respectivos gráficos de residuos. Fuente: Diez y col. (2017, p. 350).

$$R^2 = \frac{s_y^2 - s_e^2}{s_y^2} \tag{13.7}$$

Para el ejemplo, como podemos verificar en la penúltima línea de la descripción del modelo obtenido (figura 13.6) bajo el nombre de `Multiple R-squared`, tenemos:

$$R^2 = -0,868^2 = 0,753$$

En consecuencia, la recta de regresión lineal, construida con el peso del vehículo como predictor, explica 75,3 % de la variabilidad en el rendimiento.

13.5.3 Validación cruzada

Hasta ahora hemos visto cómo detectar valores atípicos que influyan en la recta y cómo determinar si la recta se ajusta bien a la muestra. Sin embargo, nos falta verificar si el modelo puede generalizarse. Una estrategia frecuente para esto es la **validación cruzada**, en la que el conjunto de datos se separa en dos fragmentos:

- **Conjunto de entrenamiento:** suele contener entre el 80 % y el 90 % de las observaciones (aunque es frecuente encontrar que solo contenga el 70 % de ellas), escogidas de manera aleatoria, y se emplea para ajustar la recta con el método de mínimos cuadrados.
- **Conjunto de prueba:** contiene el 10 % a 30 % restante de las instancias, y se usa para evaluar el modelo con datos nuevos.

Estos porcentajes se definen con el propósito de contar con la mayor cantidad de datos posible para ajustar el modelo, resguardando que el conjunto de prueba sea lo suficientemente grande como para obtener una buena estimación de la calidad del modelo.

La idea detrás de este método es evaluar cómo se comporta el modelo con datos que no ha visto previamente, en comparación al comportamiento con el conjunto de entrenamiento. Una buena métrica que podemos usar para esta tarea es el **error cuadrático medio**, o MSE por sus siglas en inglés, pues es lo que el método de mínimos cuadrados busca minimizar.

El script 13.4 aborda, una vez más, el ajuste de una RLS para predecir el rendimiento de un automóvil a partir de su peso, pero esta vez usando validación cruzada. Como resultado, obtenemos el modelo de la figura 13.14. Fijémonos en que, para el conjunto de entrenamiento, el error cuadrático medio es $MSE_e = 5,652$, mientras que para el conjunto de prueba obtenemos $MSE_p = 17,516$, bastante más elevado (¡más del triple!). Esto sugiere que el modelo puede estar sobreajustado, es decir, que se adapta bien a los datos del conjunto de entrenamiento pero no tanto al conjunto de prueba, por lo que podría ser imprudente suponer que puede ser generalizado. Sin embargo, esto puede deberse a la separación aleatoria de los datos. Al ejecutar el script 13.4 reemplazando la semilla aleatoria por 125, obtenemos el resultado de la figura 13.15. Podemos notar que los parámetros del modelo son algo diferentes a los obtenidos con la semilla 101. Además, ahora el error cuadrático medio para el conjunto de entrenamiento es $MSE_e = 8,596$ y para el conjunto de prueba, $MSE_p = 9,122$. Estos últimos valores son muy parecidos, por lo que este segundo modelo sí podría ser generalizable.

```
Call:
lm(formula = mpg ~ wt, data = entrenamiento)

Residuals:
  Min     1Q Median     3Q    Max
-3.602 -1.854 -0.212  1.590  4.684

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  34.4745     2.1780  15.829 8.89e-13 ***
wt           -4.5761     0.6566  -6.969 9.16e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.494 on 20 degrees of freedom
Multiple R-squared:  0.7083,Adjusted R-squared:  0.6938
F-statistic: 48.57 on 1 and 20 DF,  p-value: 9.159e-07
```

Figura 13.14: recta de mínimos cuadrados usando validación cruzada.

Script 13.4: ajuste de una regresión lineal simple usando validación cruzada.

```
1 # Cargar los datos.
```

```r
datos <- mtcars

# Crear conjuntos de entrenamiento y prueba.
set.seed(101)
n <- nrow(datos)
n_entrenamiento <- floor(0.7 * n)
muestra <- sample.int(n = n, size = n_entrenamiento, replace = FALSE)
entrenamiento <- datos[muestra, ]
prueba  <- datos[-muestra, ]

# Ajustar modelo con el conjunto de entrenamiento.
modelo <- lm(mpg ~ wt, data = entrenamiento)
print(summary(modelo))

# Calcular error cuadrado promedio para el conjunto de entrenamiento.
mse_entrenamiento <- mean(modelo$residuals ** 2)
cat("MSE para el conjunto de entrenamiento:", mse_entrenamiento, "\n")

# Hacer predicciones para el conjunto de prueba.
predicciones <- predict(modelo, prueba)

# Calcular error cuadrado promedio para el conjunto de prueba.
error <- prueba[["mpg"]] - predicciones
mse_prueba <- mean(error ** 2)
cat("MSE para el conjunto de prueba:", mse_prueba)
```

```
Call:
lm(formula = mpg ~ wt, data = entrenamiento)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7972 -2.7338 -0.0359  1.3380  6.5505

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  38.1048     2.3760  16.037 6.97e-13 ***
wt           -5.6044     0.7381  -7.593 2.59e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.075 on 20 degrees of freedom
Multiple R-squared:  0.7425,Adjusted R-squared:  0.7296
F-statistic: 57.66 on 1 and 20 DF,  p-value: 2.586e-07
```

Figura 13.15: otra recta de mínimos cuadrados usando validación cruzada.

13.5.4 Validación cruzada de k pliegues

Una buena manera de mejorar la estimación del error cuadrático medio es obtener más observaciones, de acuerdo al ya conocido teorema del límite central. Para esto, se puede usar una nueva manera de remuestreo: la **validación cruzada de k pliegues** (en inglés *k-fold cross validation*). La idea de fondo es la misma de

la validación cruzada expuesta en el apartado anterior: usar un conjunto de entrenamiento para ajustar el modelo y otro de prueba para evaluarlo. Sin embargo, esta variante modifica este proceso a fin de obtener k estimaciones del error. Para ello se separa el conjunto de datos en k subconjuntos de igual tamaño y, como explica Amat Rodrigo (2016d), realizamos k estimaciones del error cuadrático medio de la siguiente manera:

1. Para cada uno de los k subconjuntos:
   a) Tomar uno de los k subconjuntos del conjunto de entrenamiento y reservarlo como conjunto de prueba.
   b) Ajustar la recta de mínimos cuadrados usando para ello los $k-1$ subconjuntos restantes.
   c) Estimar el error cuadrático medio usando para ello el conjunto de prueba.
2. Estimar el error cuadrático medio del modelo, correspondiente a la media de los k errores cuadrados medios obtenidos en el paso 1.

En R, podemos realizar este proceso de forma bastante sencilla gracias a la función `train(formula, method = ‘‘lm’’, trControl = trainControl(method = ‘‘cv’’, number)` del paquete `caret`, donde:

- `formula`: fórmula que se emplea en las llamadas internas a `lm()`.
- `number`: cantidad de pliegues (k).

El lector atento habrá notado que hemos asignado valores fijos a algunos de los argumentos de la función `train()`. Esto se debe a que este método sirve para ajustar muchos otros modelos además de la RLS. El script 13.5 crea, una vez más, una RLS para predecir el rendimiento de un automóvil a partir de su peso, usando ahora validación cruzada de 5 pliegues.

Script 13.5: ajuste de una regresión lineal simple usando validación cruzada de 5 pliegues.

```
library(caret)

# Cargar los datos.
datos <- mtcars

# Crear conjuntos de entrenamiento y prueba.
set.seed(101)
n <- nrow(datos)
n_entrenamiento <- floor(0.7 * n)
muestra <- sample.int(n = n, size = n_entrenamiento, replace = FALSE)
entrenamiento <- datos[muestra, ]
prueba  <- datos[-muestra, ]

# Ajustar modelo usando validación cruzada de 5 pliegues.
modelo <- train(mpg ~ wt, data = entrenamiento, method = "lm",
                trControl = trainControl(method = "cv", number = 5))

print(summary(modelo))

# Hacer predicciones para el conjunto de entrenamiento.
predicciones_entrenamiento <- predict(modelo, entrenamiento)

# Calcular error cuadrado promedio para el conjunto de prueba.
error_entrenamiento <- entrenamiento[["mpg"]] - predicciones_entrenamiento
mse_entrenamiento <- mean(error_entrenamiento ** 2)
cat("MSE para el conjunto de entrenamiento:", mse_entrenamiento, "\n")

# Hacer predicciones para el conjunto de prueba.
predicciones_prueba <- predict(modelo, prueba)

# Calcular error cuadrado promedio para el conjunto de prueba.
error_prueba <- prueba[["mpg"]] - predicciones_prueba
mse_prueba <- mean(error_prueba ** 2)
cat("MSE para el conjunto de prueba:", mse_prueba)
```

Un aspecto importante a tener en cuenta es que la función `train()` ajusta el modelo final con la totalidad del conjunto de entrenamiento, por lo que el error cuadrático medio para el conjunto de prueba y los parámetros del modelo son los mismos que ya habíamos obtenido. Sin embargo, la estimación del error cuadrático medio para el conjunto de entrenamiento es diferente: al usar la semilla 101 se obtiene $MSE_e = 2,785$ y para la semilla 125, $MSE_e = 3,482$, valores bastante más parecidos entre sí.

13.5.5 Validación cruzada dejando uno fuera

Cuando la muestra disponible es pequeña, tema que reforzaremos en el capítulo siguiente, una buena alternativa es usar **validación cruzada dejando uno fuera** (*leave-one-out cross validation* en inglés). El esquema es el mismo que para validación cruzada con k pliegues, pero ahora usaremos tantos pliegues como observaciones tenga el conjunto de entrenamiento. En otras palabras, hacemos una iteración por cada elemento del conjunto de entrenamiento, reservando una única observación para validación. En R, la llamada a `train()` es muy similar a la que hicimos para validación cruzada con k pliegues: solo cambia el argumento `trControl`, cuyo valor ahora debe ser `trainControl(method = "LOOCV")`.

13.6 INFERENCIA PARA REGRESIÓN LINEAL

También podemos usar los modelos de RLS para hacer inferencia, procedimiento que ilustraremos mediante el siguiente ejemplo: el gerente de una empresa de desarrollo de software cree que, mientras más *stakeholders* tiene un proyecto, menos requisitos funcionales tiene el software a desarrollar. Para llevar a cabo el estudio pertinente, seleccionó aleatoriamente los datos de 15 proyectos de entre los 200 que ha desarrollado la empresa hasta la fecha, los cuales se muestran en la tabla 13.2.

requisitos	**stakeholders**
11	8
10	8
12	6
14	6
8	8
13	7
18	3
15	1
20	3
16	4
21	5
13	4
10	4
9	9
21	2

Tabla 13.2: requisitos funcionales y cantidad de *stakeholders* para diferentes proyectos desarrollados por la empresa.

Para llevar a cabo su estudio, el gerente ha ajustado un modelo de regresión lineal usando para ello el script

13.6. La recta ajustada y el gráfico de residuos se muestran en la figura 13.16.

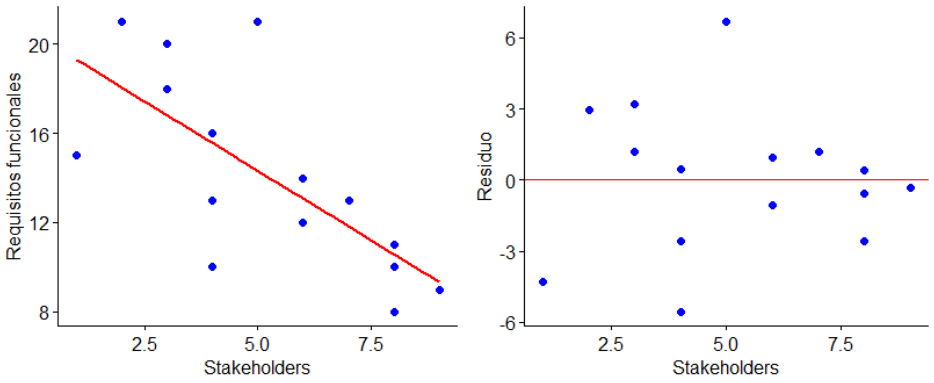


Figura 13.16: regresión lineal para la cantidad de requisitos funcionales de acuerdo a la cantidad de *stakeholders*.

Script 13.6: regresión lineal para la cantidad de requisitos funcionales de acuerdo a la cantidad de *stakeholders*.

```
library(ggpubr)

# Crear los datos originales.
requisitos <- c(11, 10, 12, 14, 8, 13, 18, 15, 20, 16, 21, 13, 10, 9, 21)
stakeholders <- c(8, 8, 6, 6, 8, 7, 3, 1, 3, 4, 5, 4, 4, 9, 2)
datos <- data.frame(requisitos, stakeholders)

# Ajustar modelo.
modelo <- lm(requisitos ~ stakeholders, data = datos)
print(summary(modelo))

# Graficar el modelo.
p <- ggscatter(
  datos, x = "stakeholders", y = "requisitos", color = "blue", fill = "blue",
  xlab = "Stakeholders", ylab = "Requisitos funcionales")

p <- p + geom_smooth(method = lm, se = FALSE, colour = "red")

# Graficar los residuos.
b_1 <- modelo$coefficients[2]
b_0 <- modelo$coefficients[1]
residuos <- datos[["requisitos"]] - (b_1 * datos[["stakeholders"]] + b_0)
datos <- data.frame(datos, residuos)

r <- ggscatter(datos, x = "stakeholders", y = "residuos", color = "blue",
               fill = "blue", xlab = "Stakeholders", ylab = "Residuo")

r <- r + geom_hline(yintercept = 0, colour = "red")

g <- ggarrange(p, r, ncol = 2, nrow = 1)
print(g)

# Verificar normalidad de los residuos.
cat("Prueba de normalidad para los residuos\n")
print(shapiro.test(datos$residuos))
```

Puesto que la correlación entre ambas variables es relativamente fuerte ($R = -0,706$), podemos comprobar

que los datos siguen una tendencia lineal. Al aplicar la prueba de normalidad de Shapiro-Wilk a los residuos, concluimos que estos siguen una distribución cercana a la normal ($p = 0,924$). Podemos apreciar en la figura 13.16 que la variabilidad de los residuos es relativamente constante. Por otra parte, las observaciones son independientes entre sí, pues han sido seleccionadas de manera aleatoria y corresponden a menos del 10 % de la población. En consecuencia, se verifica el cumplimiento de todas las condiciones necesarias para emplear un modelo de RLS ajustado mediante mínimos cuadrados.

```
Call:
lm(formula = requisitos ~ stakeholders, data = datos)

Residuals:
    Min      1Q  Median      3Q     Max
-5.5624 -1.8160  0.4234  1.1840  6.6840

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   20.5482     1.9810  10.373 1.17e-07 ***
stakeholders  -1.2464     0.3466  -3.596  0.00326 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.184 on 13 degrees of freedom
Multiple R-squared:  0.4987,Adjusted R-squared:  0.4601
F-statistic: 12.93 on 1 and 13 DF,  p-value: 0.003255
```

Figura 13.17: descripción detallada del modelo obtenido por el gerente para el ejemplo.

En la descripción del modelo (figura 13.17) podemos notar, bajo el encabezado `Coefficients`, una tabla con dos filas: una por cada parámetro del modelo, donde la primera corresponde a la intercepción y la segunda, a la pendiente. A su vez, la primera columna identifica los parámetros del modelo y la segunda presenta sus valores estimados. Como toda estimación tiene asociado un margen de error, la tercera columna muestra el error estándar para cada parámetro. Las dos columnas restantes requieren de una explicación algo más detallada, por lo que no las describiremos aquí.

El gerente, con la intención de evaluar a una abatida estudiante en práctica que aún no ha cursado su asignatura de estadística, le ha entregado los resultados obtenidos y le ha preguntado si los datos sustentan su teoría de que la cantidad de requisitos funcionales disminuye a medida que la cantidad de *stakeholders* aumenta. Tras muchas horas buscando información, la estudiante ha formulado las siguientes hipótesis:

H_0: $\beta_1 = 0$. La pendiente del modelo es igual a 0 o, lo que es lo mismo, la cantidad de *stakeholders* no explica en absoluto la cantidad de requisitos funcionales.
H_A: $\beta_1 < 0$.

Puesto que el valor p entregado por R corresponde a una prueba bilateral (fijarse en el valor absoluto que incluye el título de la columna: `Pr(>|t|)`), en el caso unilateral se debe considerar la mitad de este valor. En consecuencia, el gerente concluye, con 99 % de confianza ($p < 0.002$), que en efecto la cantidad de requisitos funcionales disminuye a medida que la cantidad de *stakeholders* aumenta.

13.7 EJERCICIOS PROPUESTOS

1. ¿Qué asume la regresión lineal, qué variables involucra y con qué parámetros trabaja?

2. ¿Cómo lucen los gráficos de dispersión de una relación lineal fuerte, de una débil y de una nula?
3. ¿Por qué hay que mirar un gráfico de dispersión de los datos al pensar en regresión lineal?
4. Describe cómo se usa la línea de regresión para predecir.
5. ¿Qué son los residuos? ¿Qué valores pueden tomar? ¿Qué utilidad tienen?
6. Explica qué mide la correlación, cómo se calcula y qué valores puede tomar.
7. Explica cómo funciona el método de los mínimos cuadrados.
8. ¿Qué condiciones necesita el método de los mínimos cuadrados para ser confiable?
9. ¿Cómo lucen los gráficos de residuos que no cumplen con alguno de los requisitos enunciados en el ejercicio anterior?
10. Explica cómo se interpretan los parámetros estimados con regresión por mínimos cuadrados.
11. Explica qué mide el coeficiente de determinación R^2, cómo se calcula y qué valores puede tomar.
12. Explica cómo se interpretan los parámetros de la regresión lineal cuando la variable predictora es categórica.
13. Explica qué es apalancamiento y por qué es importante detectarlo.
14. ¿Cómo lucen los gráficos de datos (y residuos) que pueden tener problemas de apalancamiento?
15. ¿Cuáles son las hipótesis que se contrastan al hacer inferencia con la regresión lineal?
16. Investiga cómo se usa la función `lm()` de R y qué información entrega.

CAPÍTULO 14. REGRESIÓN LINEAL MÚLTIPLE

En el capítulo anterior conocimos los principios detrás de la regresión lineal, considerando para ello una única variable predictora y una variable de respuesta. Sin embargo, en la vida real es más frecuente que un fenómeno sea explicado por muchas variables. En consecuencia, en este capítulo presentaremos un nuevo modelo lineal más complejo: la regresión lineal múltiple (RLM), correspondiente al caso de una única respuesta con múltiples predictores. Para ello tomaremos como base los textos de Field y col. (2012, pp. 245-311) y Diez y col. (2017, pp. 372-385).

Una regresión lineal con múltiples variables tiene la forma que se presenta en la ecuación 14.1, donde:

- Cada x_i es un predictor.
- Cada β_i corresponde a un parámetro del modelo.
- k es la cantidad de predictores.
- $\hat{y}$ es una estimación de la respuesta.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + +\beta_k x_k \tag{14.1}$$

Una vez más, al ajustar el modelo mediante el método de mínimos cuadrados, buscamos minimizar la suma de los cuadrados de los residuos (ecuación 14.2), proceso que se vuelve más complejo a medida que aumenta la cantidad de variables por lo que suele hacerse mediante el uso de software.

$$\text{mín} \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{14.2}$$

Al igual que en el caso de la regresión lineal simple (RLS), la RLM requiere verificar algunas condiciones:

1. La distribución de los residuos debe ser cercana a la normal.
2. La variabilidad de los residuos debe ser aproximadamente constante.
3. Los residuos deben ser independientes entre sí.
4. Cada variable se relaciona linealmente con la respuesta.

Para los ejemplos de este capítulo usaremos una vez más el conjunto de datos `mtcars`, cuyas variables describimos en la tabla 13.1. Como punto de partida, recordemos la RLS para predecir el rendimiento de un automóvil (usando el mismo conjunto de datos) a partir de su peso y ajustada con todo el conjunto de entrenamiento (figura 13.7).

Ahora consideremos una RLM con dos predictores para el rendimiento: el peso (columna `wt`) y el tiempo mínimo requerido para recorrer un cuarto de milla (columna `qsec`), modelo que podemos obtener mediante el script 14.1 y que se muestra en la figura 14.1. El procedimiento para ajustar la RLM en R es el mismo que usamos en el capítulo anterior, pero ahora en el lado derecho de la fórmula para ajustar el modelo tenemos que combinar ambos predictores.

Script 14.1: regresión lineal para predecir el rendimiento de un automóvil a partir de dos variables.

```
library(scatterplot3d)

# Cargar los datos.
datos <- mtcars

# Ajustar modelo usando validación cruzada de 5 pliegues.
modelo <- lm(mpg ~ wt + qsec, data = datos)
print(summary(modelo))

```

```
Call:
lm(formula = mpg ~ wt + qsec, data = datos)

Residuals:
    Min      1Q  Median      3Q     Max
-4.3962 -2.1431 -0.2129  1.4915  5.7486

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,Adjusted R-squared:  0.8144
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

Figura 14.1: descripción del modelo lineal para predecir el rendimiento de un automóvil a partir de dos variables.

```
# Graficar modelo ajustado.
g <- scatterplot3d(datos$wt, datos$qsec, datos$mpg, type = "p",
                   highlight.3d = TRUE, pch = 20, xlab = "Peso [lb x 1000]",
                   ylab = "Rendimiento [millas/galón]",
                   zlab = "1/4 de milla [s]")

g$plane3d(modelo ,draw_polygon = TRUE, draw_lines = TRUE)
print(g)
```

Para usar este modelo a fin de predecir valores para la respuesta a partir de un nuevo conjunto de datos, usamos una vez más la función `predict()`, del mismo modo que vimos en el capítulo 13 para la RLS.

Como en este caso tenemos dos predictores, lo que se ajusta ya no es una recta, sino un plano, como muestra la figura 14.2. Así, ya no tiene sentido hablar de la pendiente de la recta al momento de interpretar los parámetros del modelo. Un análisis de regresión lineal con múltiples variables busca aislar la relación entre cada predictor y la respuesta, por lo que el coeficiente β_i, asociado al i-ésimo predictor, representa el cambio esperado que se produce en la respuesta al incrementar dicho predictor en una unidad, **manteniendo constantes todos los demás predictores**. Si b_1 es el parámetro ajustado para el peso y b_2 es el parámetro ajustado para el cuarto de milla, b_1 puede entenderse como la pendiente del plano de la figura 14.1 con respecto al eje y, mientras que b_2 puede entenderse como la pendiente del mismo plano con respecto al eje z. A su vez, la intercepción fija la posición del plano con respecto al origen.

Desde luego, podemos extender esta idea para más de dos predictores. En tal caso ajustamos un hiperplano cuya forma y ubicación están dadas por los parámetros del modelo.

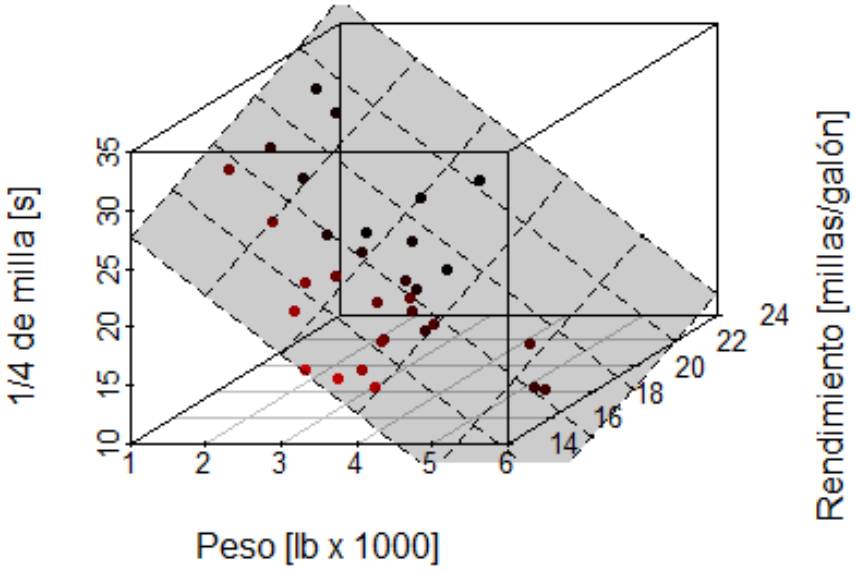


Figura 14.2: plano ajustado para la RLM con dos predictores.

14.1 RLM CON PREDICTORES CATEGÓRICOS

En el capítulo 13 habíamos señalado que para usar una variable categórica con dos niveles como predictor, esta debe ser transformada en una variable indicadora. Desde luego, podemos extender la misma idea para el caso de variables categóricas con más niveles.

Imaginemos que tenemos una variable categórica con k niveles. El primer paso consiste en crear $k-1$ nuevas variables artificiales. A continuación, para cada una de estas nuevas variables, escogemos un nivel diferente de la variable original y asignamos un 1 a todas las observaciones que tengan ese nivel y un 0 a las restantes. Para entender mejor esta idea, supongamos que un conjunto de datos tiene la variable categórica `tipo`, con cuatro niveles: `A`, `B`, `C` y `D`. Creamos tres nuevas variables, por ejemplo, `tipo_A`, `tipo_B` y `tipo_C`. La variable `tipo_A` contiene tantas observaciones como la variable original, con un 1 para aquellas observaciones en que `tipo` toma el valor `A` y un 0 para las restantes. Para las variables `tipo_B` y `tipo_C` se procede de manera análoga. Puesto que solo se crean $k-1$ variables artificiales, se descarta aquella correspondiente al nivel `D` de la variable `tipo`.

En R, podemos hacer esta tarea de manera sencilla mediante la función `dummy.data.frame(data, names, drop)` del paquete `dummies`, donde:

- `data`: matriz de datos.
- `names`: nombres de las columnas para las que se desea crear variables artificiales. Si se omite este argumento, se crean variables artificiales para todas las variables categóricas y de tipo string.
- `drop`: indicador booleano que, cuando es verdadero, descarta la variable original del resultado.

Es importante señalar que esta función crea una variable artificial por cada nivel de la variable categórica,

por lo que debemos descartar una de ellas.

El script 14.2 muestra el funcionamiento de `dummy.data.frame()` para una matriz de datos con dos variables categóricas. La tabla 14.1 muestra, en las columnas de la izquierda, el conjunto de datos original y en las de la izquierda, el conjunto de datos con las nuevas variables artificiales.

sujeto	sexo	tipo	valor
1	F	B	1.68
2	F	D	2.79
3	M	A	1.92
4	M	B	2.26
5	M	A	2.10
6	M	C	2.63
7	F	D	2.19
8	M	D	3.62
9	F	D	2.76
10	F	A	1.26

sujeto	sexoM	tipoB	tipoC	tipoD	valor
1	0	1	0	0	1.68
2	0	0	0	1	2.79
3	1	0	0	0	1.92
4	1	1	0	0	2.26
5	1	0	0	0	2.10
6	1	0	1	0	2.63
7	0	0	0	1	2.19
8	1	0	0	1	3.62
9	0	0	0	1	2.76
10	0	0	0	0	1.26

Tabla 14.1: creación de variables artificiales para una matriz de datos con variables categóricas.

Script 14.2: creación de variables artificiales para variables categóricas.

```
library(dummies)

# Crear una matrz de datos.
sujeto <- 1:10
sexo <- c("F", "F", "M", "M", "M", "M", "F", "M", "F", "F")
tipo <- c("B", "D", "A", "B", "A", "C", "D", "D", "D", "A")
valor <- c(1.68, 2.79, 1.92, 2.26, 2.1, 2.63, 2.19, 3.62, 2.76, 1.26)
datos <- data.frame(sujeto , sexo , tipo, valor)

# Crear variables artificiales.
datos.dummy <- dummy.data.frame(datos , drop = TRUE)
datos.dummy[["sexoF"]] <- NULL
datos.dummy[["tipoA"]] <- NULL

# Crear modelos lineales.
m1 <- lm(valor ~ sexo + tipo, datos)
print(m1)

m2 <- lm(valor ~ sexoM + tipoB + tipoC + tipoD, datos.dummy)
print(m2)
```

Finalmente, para usar la variable categórica como predictor, agregamos al modelo todas las variables artificiales creadas a partir de ella. Cabe señalar que la función `lm()` realiza internamente este proceso cuando recibe una variable categórica entre los predictores (las variables indicadoras descartadas en el script 14.2 replican el resultado que entrega `lm()`). No obstante, al usar el modelo, debemos fijarnos en que la cantidad de predictores categóricos sea la misma, como también en que la cantidad y el orden de sus niveles coincida con los del conjunto de entrenamiento.

El script 14.2 ajusta además dos modelos, el primero usando el conjunto de datos original y el segundo, el conjunto de datos transformado para que tenga variables indicadoras en reemplazo de las variables categóricas. Si nos fijamos en la figura 14.3, podemos ver que en ambos casos el modelo tiene los mismos predictores.

```
Call:
lm(formula = valor ~ sexo + tipo, data = datos)

Coefficients:
(Intercept)        sexoM        tipoB        tipoC        tipoD
     1.2139       0.8191       0.3465       0.5970       1.4213


Call:
lm(formula = valor ~ sexoM + tipoB + tipoC + tipoD, data = datos.dummy)

Coefficients:
(Intercept)        sexoM        tipoB        tipoC        tipoD
     1.2139       0.8191       0.3465       0.5970       1.4213
```

Figura 14.3: resultado del script 14.2.

14.2 CONDICIONES PARA USAR RLM

Llegado este punto, necesitamos examinar con más detalle las condiciones que debemos cumplir para que un modelo de regresión lineal sea generalizable:

1. Las variables predictoras deben ser cuantitativas o dicotómicas (de ahí la necesidad de variables indicadoras para manejar más de dos niveles).
2. La variable de respuesta debe ser cuantitativa y continua, sin restricciones para su variabilidad.
3. Los predictores deben tener algún grado de variabilidad (su varianza no debe ser igual a cero). En otras palabras, no pueden ser constantes.
4. No debe existir **multicolinealidad**. Esto significa que no deben existir relaciones lineales *fuertes* entre dos o más predictores (coeficientes de correlación altos).
5. Los residuos deben ser homocedásticos (con varianzas similares) para cada nivel de los predictores.
6. Los residuos deben seguir una distribución cercana a la normal centrada en cero.
7. Los valores de la variable de respuesta son independientes entre sí.
8. Cada predictor se relaciona linealmente con la variable de respuesta.

14.3 EVALUACIÓN DEL AJUSTE DE UNA RLM

En el capítulo anterior introdujimos el coeficiente de determinación (R^2) como instrumento para evaluar la bondad de ajuste de una regresión lineal. Sin embargo, cuando el modelo es multivariado, la función ya conocida para estimar R^2 genera una mala estimación del porcentaje de la varianza explicada por el modelo, pues los grados de libertad asociados a la variabilidad de los residuos es ahora diferente, como muestra la ecuación 14.3, donde n es el tamaño de la muestra y k, la cantidad de variables predictoras.

$$\nu = n - k - 1 \tag{14.3}$$

Así, para evaluar una RLM tenemos que usar un coeficiente de determinación ajustado. Algunos autores han propuesto distintas maneras de efectuar este ajuste, una de las cuales presentamos en la ecuación 14.4. También podemos usar este ajuste cuando tenemos un único predictor, aunque la diferencia en este caso suele ser muy pequeña como para ser relevante.

$$R^2 = 1 - \frac{s_e^2}{s_y^2} \cdot \frac{n-1}{n-k-1} \tag{14.4}$$

Existen otras alternativas para evaluar la bondad de ajuste de un modelo que se basan en el **principio de parsimonia**, también llamado navaja de Occam, el cual indica que un modelo debe mantenerse tan simple como sea posible. Dos de ellas son el **criterio de información de Akaike**, abreviado AIC, y el **criterio bayesiano de Schwarz** (BIC o SBC), que penalizan el modelo por contener variables adicionales, por lo que mientras menor sea su valor, mejor será el modelo. Si bien el cálculo de estas medidas no se detalla aquí por ser un tópico más avanzado, podemos obtenerlas en R mediante las funciones `AIC(object)` y `BIC(object)`, donde `object` corresponde a un modelo lineal ajustado.

Para el modelo que habíamos ajustado usando únicamente el peso como predictor, obtenemos $AIC = 166,03$ y $BIC = 170,43$. Del mismo modo, para el modelo que usa como predictores el peso y el cuarto de milla, en cambio, tenemos que que $AIC = 156,72$ y $BIC = 162,58$. En consecuencia, el segundo modelo parece ser "mejor" bajo estos criterios.

Otra opción, adecuada cuando necesitamos saber cuáles predictores son estadísticamente significativos, es observar los valores p asociados a cada predictor. Habitualmente consideraremos significativos aquellos predictores para los cuales $p < 0,05$.

14.4 COMPARACIÓN DE MODELOS

En la sección anterior vimos que métricas como el AIC o el BIC nos pueden resultar útiles para comparar dos modelos de regresión lineal, considerando la noción general que un modelo es mejor mientras menor sea su valor de AIC (o BIC). Si calculamos el AIC para cada uno de los modelos ajustados hasta ahora (script 14.3), veremos que el AIC del modelo con dos predictores es menor. Sin embargo, al ser una medida relativa, hasta ahora no contamos con una prueba estadística que nos permita determinar si la diferencia es significativa.

Cuando los modelos son jerárquicos, es decir, el segundo incorpora nuevos predictores además de mantener los del primer modelo, podemos hacer una prueba de hipótesis usando los coeficientes de determinación para ver si la diferencia es significativa. El estadístico de prueba se calcula mediante la ecuación 14.5, donde:

- n: cantidad de observaciones.
- k_2: cantidad de predictores en el modelo con más variables (segundo modelo).
- R^2_{cambio}: diferencia entre los valores de R^2 del segundo y el primer modelo.
- k_{cambio}: diferencia entre la cantidad de predictores del segundo y el primer modelo.
- R_2^2: coeficiente de determinación del segundo modelo.

$$F_{cambio} = \frac{(n-k_2-1)R^2_{cambio}}{k_{cambio}(1-R_2^2)} \tag{14.5}$$

Así, para los dos modelos ajustados hasta ahora tenemos:

$$F_{cambio} = \frac{(32-2-1)(0,8264-0,7528)}{1(1-0,8264)} = 12,295$$

Notemos que el estadístico de prueba sigue una distribución F con k_{cambio} y $n-k_2-1$ grados de libertad, a partir de lo cual podemos determinar el valor p correspondiente mediante la llamada `pf(12.295, 1, 29, lower.tail = FALSE)`, obteniendo como resultado $p = 0,0015$. En consecuencia, podemos concluir con 95 % de confianza que la diferencia es significativa, por lo que el segundo modelo es mejor.

Como ya es habitual, en R podemos hacer esta tarea de forma simple gracias a la función `anova(object, ...)`, que recibe como argumentos los diferentes modelos a comparar. La interpretación del resultado de esta prueba es sencilla: si el valor p obtenido es significativo, entonces el modelo más complejo (con más predictores) es mejor. Al ejecutar el script 14.3, podemos ver que obtenemos el mismo resultado que con la prueba F.

Script 14.3: comparación de dos modelos lineales.

```
# Cargar datos.
datos <- mtcars

# Ajustar modelo con el peso como predictor.
modelo_1 <- lm(mpg ~ wt, data = datos)
print(summary(modelo_1))
aic_1 <- AIC(modelo_1)
cat("Modelo 1: AIC =", AIC(modelo_1), "\n")

# Ajustar modelo con el peso y el cuarto de milla como predictores.
modelo_2 <- lm(mpg ~ wt + qsec, data = datos)
print(summary(modelo_2))
aic_2 <- AIC(modelo_2)
cat("Modelo 2: AIC =", AIC(modelo_2), "\n")

# Comparar ambos modelos.
comparacion <- anova(modelo_1, modelo_2)
print(comparacion)
```

14.5 SELECCIÓN DE PREDICTORES

La cuarta condición para emplear RLM indica que debemos evitar la multicolinealidad. Esto es importante porque el ajuste de un modelo RLM asume que podemos cambiar una variable predictora, *manteniendo las otras constantes*. Cuando las variables predictoras están correlacionadas, se hace imposible cambiar el valor de una sin alterar también a las demás, desestabilizando la estimación de los coeficientes del modelo que indican cómo influye cada variable predictora en la variable de salida de forma *independiente*.

Cuando existe colinealidad, los valores de los coeficientes varían enormemente si se agregan o quitan unos pocos datos de entrenamiento y se reduce el poder estadístico del modelo. Por esta razón, es importante que escojamos con cuidado las variables predictoras a considerar en un modelo RML. Es más, existe un riesgo adicional cuando los predictores están correlacionados: el orden en que se agreguen puede influenciar la calidad del modelo. En consecuencia, necesitamos contar con alguna estrategia que nos permita determinar cuáles predictores incluir. Existen diversas alternativas para esta tarea.

El método más adecuado, aunque también el más complejo, es la **regresión jerárquica**. Es el que debemos considerar al momento de intentar probar una teoría y consiste en comenzar por incorporar en primer lugar aquellos predictores ya conocidos, en orden de importancia, en base a investiagiones previas. Una vez incorporados todos los predictores ya conocidos, podemos incorporar otros nuevos si creemos que existen **buenas y justificadas razones** para ello. Antes de la masificación de los computadores y de entornos como R, ¡esta era la única alternativa viable!

En R, podemos realizar este método con ayuda de la función `update(object, formula)`, que nos permite incorporar o quitar variables del modelo, donde:

- `object`: modelo previamente ajustado, en este caso con `lm()`.
- `formula`: actualización de la fórmula para el nuevo modelo.

En este caso no contamos con investigaciones previas que nos permitan formular una teoría, por lo que nos limitaremos a proporcionar un ejemplo de cómo usar esta función (script 14.4), cuyos resultados presentamos en la figura 14.4.

Script 14.4: incorporación y eliminación de variables en un modelo de RLM.

```
# Cargar datos.
datos <- mtcars

# Ajustar modelo inicial con la variable wt como predictor.
modelo <- lm(mpg ~ wt, data = datos)
cat("=== Modelo inicial ===\n")
print(modelo)

# Incorporar el predictor cyl.
modelo <- update(modelo, . ~ . + cyl)
cat("=== Modelo con predictores wt y cyl ===\n")
print(modelo)

# Quitar el predictor wt.
modelo <- update(modelo, . ~ . - wt)
cat("=== Modelo con predictor cyl ===\n")
print(modelo)

# Agregar predictores wt y drat, y quitar predictor cyl.
modelo <- update(modelo, . ~ . + wt + drat - cyl)
cat("=== Modelo con predictores wt y drat ===\n")
print(modelo)
```

Si, en lugar de probar una teoría, lo que queremos es **explorar los datos**, podemos usar otras estrategias.

Selección hacia adelante Se crea un modelo inicial nulo, es decir, sin predictores, para el cual únicamente se estima la intercepción. A continuación, se escoge como primer predictor aquel que tenga la correlación más alta con la variable de respuesta. Si dicho predictor incrementa la capacidad predictiva del modelo, se retiene en el modelo y se procede a seleccionar un segundo predictor. El modelo con una única variable ajustado al inicio de este capítulo tiene como predictor el peso de los automóviles, variable que en efecto tiene la más alta correlación con el rendimiento (variable de respuesta). El coeficiente de determinación para este modelo es $R^2 = 0,7528$, lo cual significa que explica aproximadamente el 75 % de la variabilidad de la respuesta. En consecuencia, existe aún un 25 % de variabilidad de la respuesta que aún no ha sido explicado.
Para la selección de predictores adicionales, en cada repetición se escoge aquel predictor (que no haya sido previamente agregado al modelo) que tenga la **máxima correlación semi-parcial con la respuesta**, es decir, que explique la máxima porción de la varianza no cubierta por el modelo ya existente. Si la inclusión de este nuevo predictor mejora el poder predictivo del modelo, se incorpora de manera definitiva y se evalúa el siguiente predictor.
Adicionalmente, se evalúa si la inclusión de cada nuevo predictor mejora (es decir, reduce) el AIC. Si ninguno de los posibles predictores restantes logra reducir este indicador, se detiene la inclusión de nuevos predictores.

Eliminación hacia atrás Es el proceso inverso a la eliminación hacia atrás, puesto que se comienza desde un modelo con todas las variables para luego eliminar predictores uno a uno y evaluar el AIC. Si este último se reduce, se elimina dicho predictor y se reevalúa la contribución de los predictores que aún se encuentran en el modelo. Una vez más, el proceso se repite hasta que no es posible reducir el AIC.

Regresión escalonada En términos generales, opera de manera análoga a la selección hacia adelante. Sin embargo, para reducir el potencial efecto debido al orden de incorporación de los predictores, cada vez que se incorpora uno nuevo se evalúa qué ocurre al descartar el menos importante. En consecuencia, el modelo es permanentemente reevaluado para descartar posibles redundancias entre predictores.

```
=== Modelo inicial ===

Call:
lm(formula = mpg ~ wt, data = datos)

Coefficients:
(Intercept)           wt
     37.285       -5.344

=== Modelo con predictores wt y cyl ===

Call:
lm(formula = mpg ~ wt + cyl, data = datos)

Coefficients:
(Intercept)           wt          cyl
     39.686       -3.191       -1.508

=== Modelo con predictor cyl ===

Call:
lm(formula = mpg ~ cyl, data = datos)

Coefficients:
(Intercept)          cyl
     37.885       -2.876

=== Modelo con predictores wt y drat ===

Call:
lm(formula = mpg ~ wt + drat, data = datos)

Coefficients:
(Intercept)           wt         drat
     30.290       -4.783        1.442
```

Figura 14.4: resultado del script 14.4.

Todos los subconjuntos Una alternativa más exhaustiva, altamente costosa en tiempo de ejecución, es usar un algoritmo de fuerza bruta en el que se exploran todos los subconjuntos de predictores.

Es importante reiterar que solo debemos usar estos métodos si estamos **explorando datos**, pues de lo contrario incurriríamos en faltas a la ética. Veamos por ejemplo el trabajo de Montero Muñoz y col. (2012), donde estudian algunas implicaciones clínicas en el uso de antibióticos en ancianos mayores de 80 años. Podemos ver que, en la recolección de datos, recopilaron variables que la medicina ha probado a lo largo de su historia que son importantes al evaluar la salud de un paciente y que pueden ser afectadas por el uso de medicamentos. Un equipo poco ético podría haber empleado otras variables registradas en las bases de datos de origen, por ejemplo, el monto de la pensión del paciente. ¿Qué consecuencias podrían existir si tuviéramos una correlación espuria de esta variable con la variable de respuesta? ¡Podríamos afectar las vidas de millones de personas si tal estudio concluyera que los antibióticos son más nocivos para ancianos con bajas pensiones!

Todas las estrategias mencionadas están disponibles en R con ayuda de la ya conocida función `update()` y las funciones `add1(object, scope)` y `drop1(object, scope)`, donde:

- `object`: un modelo ajustado.
- `scope`: fórmula que proporciona los términos a agregar o quitar.

La función `add1()` evalúa la incorporación de cada nuevo predictor potencial (separadamente) a un modelo base y entrega algunas métricas para el efecto que tiene su incorporación, entre ellas el AIC. El mejor nuevo predictor corresponde, entonces, a aquella variable con el menor AIC. Las líneas 15 y 22 del script 14.5 ilustran su uso, obteniéndose los resultados que se muestran en la figura 14.5.

```
Single term additions

Model:
mpg ~ 1
       Df Sum of Sq     RSS     AIC
<none>              1126.05 115.943
cyl     1    817.71  308.33  76.494
disp    1    808.89  317.16  77.397
hp      1    678.37  447.67  88.427
drat    1    522.48  603.57  97.988
wt      1    847.73  278.32  73.217
qsec    1    197.39  928.66 111.776
vs      1    496.53  629.52  99.335
am      1    405.15  720.90 103.672
gear    1    259.75  866.30 109.552
carb    1    341.78  784.27 106.369


Single term additions

Model:
mpg ~ wt
       Df Sum of Sq    RSS    AIC
<none>              278.32 73.217
cyl     1    87.150 191.17 63.198
disp    1    31.639 246.68 71.356
hp      1    83.274 195.05 63.840
drat    1     9.081 269.24 74.156
qsec    1    82.858 195.46 63.908
vs      1    54.228 224.09 68.283
am      1     0.002 278.32 75.217
gear    1     1.137 277.19 75.086
carb    1    44.602 233.72 69.628
```

Figura 14.5: resultado de las llamadas a `add1()` en el script 14.5

De manera similar, la función `drop1()` evalúa (separadamente) la eliminación potencial de cada predictor presente en un modelo base y entrega las mismas métricas que `add1()` para el efecto que tiene su eliminación. El mejor predictor a descartar es, una vez más, aquel que lleva a la mayor reducción en AIC. La línea 29 del script 14.5 ilustra su uso, obteniéndose los resultados que se muestran en la figura 14.6.

Script 14.5: Evaluación de variables a incorporar y eliminar en un modelo de RLM.

```
1 # Cargar datos.
2 datos <- mtcars
3
4 # Ajustar modelo nulo.
5 nulo <- lm(mpg ~ 1, data = datos)
```

```
Single term deletions

Model:
mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
       Df Sum of Sq    RSS    AIC
<none>              147.49 70.898
cyl     1    0.0799 147.57 68.915
disp    1    3.9167 151.41 69.736
hp      1    6.8399 154.33 70.348
drat    1    1.6270 149.12 69.249
wt      1   27.0144 174.51 74.280
qsec    1    8.8641 156.36 70.765
vs      1    0.1601 147.66 68.932
am      1   10.5467 158.04 71.108
gear    1    1.3531 148.85 69.190
carb    1    0.4067 147.90 68.986
```

Figura 14.6: resultado de la llamada a `drop1()` en el script 14.5

```r
# cat("=== Modelo nulo ===\n")
# print(summary(nulo))

# Ajustar modelo completo.
completo <- lm(mpg ~ ., data = datos)
# cat("=== Modelo completo ===\n")
# print(summary(completo))

# Evaluar variables para incorporar.
print(add1(nulo, scope = completo))
cat("\n\n")

# Agregar la variable con menor AIC.
modelo <- update(nulo, . ~ . + wt)

# Evaluar variables para incorporar.
print(add1(modelo, scope = completo))
cat("\n\n")

# Agregar la variable con menor AIC.
modelo <- update(modelo, . ~ . + cyl)

# Evaluar variables para eliminar.
print(drop1(completo, scope = completo))
cat("\n\n")

# Eliminar la variable con menor AIC.
modelo <- update(modelo, . ~ . - cyl)
```

Por supuesto, R en la práctica ya cuenta con funciones que implementan los métodos para seleccionar predictores antes descritos (excepto la regresión jerárquica, por supuesto). Los tres primeros pueden efectuarse mediante la función `step(object, scope, direction, trace)`, que usa `add1()` y `drop1()` de manera iterativa, donde:

- `object`: es un modelo ya ajustado que es usado como punto de partida.
- `scope`: es una lista de fórmulas que define el rango de modelos a explorar.
- `direction`: indica el tipo de selección a realizar, donde "`forward`" corresponde a selección hacia ade-

lante; "`backward`", a eliminación hacia atrás, y "`both`", a regresión escalonada.
- `trace`: argumento opcional que indica si se quiere ver por consola el proceso realizado.

El script 14.6 muestra el funcionamiento de la función `step()` para seleccionar los predictores a incorporar en un modelo donde la respuesta es, una vez más, el rendimiento de un automóvil.

Script 14.6: selección de predictores a incluir en una RLM.

```
1  library(leaps)
2
3  # Cargar datos.
4  datos <- mtcars
5
6  # Ajustar modelo nulo.
7  nulo <- lm(mpg ~ 1, data = datos)
8  cat("=== Modelo nulo ===\n")
9  print(summary(nulo))
10
11 # Ajustar modelo completo.
12 completo <- lm(mpg ~ ., data = datos)
13 cat("=== Modelo completo ===\n")
14 print(summary(completo))
15
16 # Ajustar modelo con selección hacia adelante.
17 adelante <- step(nulo, scope = list(upper = completo), direction = "forward",
18                  trace = 0)
19
20 cat("=== Modelo con selección hacia adelante ===\n")
21 print(summary(adelante))
22 cat("AIC =", AIC(adelante), "\n\n")
23
24 # Ajustar modelo con eliminación hacia atrás.
25 atras <- step(completo, scope = list(lower = nulo), direction = "backward",
26               trace = 0)
27
28 cat("=== Modelo con eliminación hacia atrás ===\n")
29 print(summary(atras))
30 cat("AIC =", AIC(atras), "\n\n")
31
32 # Ajustar modelo con regresión escalonada.
33 escalonado <- step(nulo, scope = list(lower = nulo, upper = completo),
34                    direction = "both", trace = 0)
35
36 cat("=== Modelo con regresión escalonada ===\n")
37 print(summary(escalonado))
38 cat("AIC =", AIC(escalonado), "\n\n")
39
40 # Ajustar modelo con todos los subconjuntos.
41 modelos <- regsubsets(mpg ~ ., data = datos, method = "exhaustive",
42                       nbest = 1, nvmax = 10)
43
44 print(plot(modelos))
```

La línea 7 del script 14.6 ajusta el modelo nulo, obteniéndose el resultado que se muestra en la figura 14.7. De manera similar, la línea 12 ajusta el modelo completo (figura 14.8).

Las líneas 17–18 usan el método de selección hacia adelante, comenzando desde el modelo nulo. Notemos que en el argumento `scope` entregamos, en este caso, el modelo completo con todos los posibles predictores. No obstante, no es necesario siempre comenzar desde el modelo nulo o evaluar hasta el modelo completo.

```
Call:
lm(formula = mpg ~ 1, data = datos)

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
```

Figura 14.7: modelo nulo.

En la llamada a `step()` de las líneas 17–18, el argumento `trace` tiene por defecto el valor 1, por lo que la función muestra información de los diferentes modelos evaluados (las figuras 14.9 y 14.10 muestran esta información para todas las iteraciones). Notemos que se comienza por calcular el AIC para el modelo nulo, y luego se determina el AIC (y otras métricas) que se obtendría para diferentes modelos, cada uno de los cuales contendría solo una de las variables restantes. El predictor que se escoge es aquel que genera el modelo con menor AIC (el peso del automóvil en la iteración 1).

En la siguiente iteración se conserva el peso como predictor y se evalúa el AIC de los modelos que incorporan un predictor adicional. Una vez más, se incorpora aquel con menor AIC, correspondiente en esta ocasión a la cantidad de cilindros.

El proceso se detiene una vez que ninguno de los predictores logra un AIC menor que el del modelo ya obtenido en la iteración previa. El modelo final obtenido mediante selección hacia adelante se presenta en la figura 14.11, que se consigue en tres iteraciones y considera, además del peso del automóvil y el número de cilindros, los caballos de fuerza bruta (`hp`).

De manera similar, las figuras 14.12 y 14.13 muestran los modelos resultantes al usar eliminación hacia atrás y regresión escalonada, respectivamente (líneas 25–34). Notemos que, en estos casos, la llamada a `step()` se realiza con el argumento `trace = 0`, por lo que no nos muestra información acerca de los diferentes modelos evaluados.

Las líneas 41–42 del script 14.6, por otra parte, muestran la aplicación del método de todos los subconjuntos para determinar el mejor modelo, proceso que hacemos mediante la función `regsubsets(formula, data, nbest, nvmax, force.in, force.out, method = ‘‘exhaustive’’)` del paquete `leaps`, donde:

- `formula`: indica la variable de respuesta y los posibles predictores.
- `data`: matriz de datos.
- `nbest`: cantidad de modelos a reportar por cada tamaño de subconjunto. Si, por ejemplo, `nbest = 3`, entonces se reportan los tres mejores modelos con un único predictor, los tres mejores modelos con dos predictores, etc.
- `nvmax`: fija una cantidad máxima de predictores a considerar.
- `force.in`: argumento opcional que toma la forma de un vector con los índices de las columnas que deben ser forzosamente consideradas en los modelos evaluados.
- `force.out`: argumento opcional que toma la forma de un vector con los índices de las columnas que deben ser forzosamente excluidas en los modelos evaluados.

La figura 14.14 representa gráficamente el resultado de la aplicación del método de todos los subconjuntos. En ella, el eje x lista todas las variables predictoras y el eje y corresponde al BIC de cada modelo, que es el criterio utilizado por defecto por la función `regsubsets()`. Fijémonos en que el eje y no es una escala graduada, sino

```
Call:
lm(formula = mpg ~ ., data = datos)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4506 -1.6044 -0.1196  1.2193  4.6271

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.30337   18.71788   0.657   0.5181
cyl         -0.11144    1.04502  -0.107   0.9161
disp         0.01334    0.01786   0.747   0.4635
hp          -0.02148    0.02177  -0.987   0.3350
drat         0.78711    1.63537   0.481   0.6353
wt          -3.71530    1.89441  -1.961   0.0633 .
qsec         0.82104    0.73084   1.123   0.2739
vs           0.31776    2.10451   0.151   0.8814
am           2.52023    2.05665   1.225   0.2340
gear         0.65541    1.49326   0.439   0.6652
carb        -0.19942    0.82875  -0.241   0.8122
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.65 on 21 degrees of freedom
Multiple R-squared:  0.869,Adjusted R-squared:  0.8066
F-statistic: 13.93 on 10 and 21 DF,  p-value: 3.793e-07
```

Figura 14.8: modelo completo.

una lista ordenada del BIC para cada uno de los modelos evaluados. Así, la figura corresponde a una matriz en que cada fila está asignada a un modelo, donde se colorea el recuadro para las variables presentes en dicho modelo. Así, el mejor modelo (es decir, el que tiene el menor BIC) es aquel que contiene como predictores el peso (`wt`), el cuarto de milla (`qsec`) y el tipo de transmisión (`am`), además de la intercepción.

También es importante fijarnos en que, en este caso particular, obtenemos el mismo modelo al usar selección hacia adelante y regresión escalonada. Del mismo modo, la eliminación hacia atrás y el método de todos los subconjuntos generan un mismo modelo. Si usamos el AIC como indicador, el primero de estos dos modelos es el mejor. No obstante, el segundo es mejor si consideramos el BIC.

14.6 EVALUACIÓN DE UN MODELO DE RLM

Para ilustrar el proceso de evaluación de un modelo de RLM, consideremos el modelo obtenido mediante eliminación hacia atrás en la sección precedente, es decir, el modelo que tiene como predictores el peso, el cuarto de milla y el tipo de marcha. Al graficar este modelo (script 14.7, línea 6) obtenemos los gráficos de la figura 14.15, donde podemos ver que la distribución de los residuos se desvía un poco de la normal y que parece haber algunas observaciones atípicas influenciando el ajuste del modelo.

```
Start:  AIC=115.94
mpg ~ 1

       Df Sum of Sq     RSS     AIC
+ wt    1    847.73  278.32  73.217
+ cyl   1    817.71  308.33  76.494
+ disp  1    808.89  317.16  77.397
+ hp    1    678.37  447.67  88.427
+ drat  1    522.48  603.57  97.988
+ vs    1    496.53  629.52  99.335
+ am    1    405.15  720.90 103.672
+ carb  1    341.78  784.27 106.369
+ gear  1    259.75  866.30 109.552
+ qsec  1    197.39  928.66 111.776
<none>               1126.05 115.943

Step:  AIC=73.22
mpg ~ wt

       Df Sum of Sq    RSS    AIC
+ cyl   1    87.150 191.17 63.198
+ hp    1    83.274 195.05 63.840
+ qsec  1    82.858 195.46 63.908
+ vs    1    54.228 224.09 68.283
+ carb  1    44.602 233.72 69.628
+ disp  1    31.639 246.68 71.356
<none>               278.32 73.217
+ drat  1     9.081 269.24 74.156
+ gear  1     1.137 277.19 75.086
+ am    1     0.002 278.32 75.217
```

Figura 14.9: modelos evaluados por la función `step()` durante el proceso de selección hacia adelante (parte 1).

14.6.1 Identificación de valores con sobreinfluencia

Existen diversos estadísticos que nos permiten evaluar la influencia de una observación en el ajuste de un modelo de regresión lineal:

1. **Residuo estandarizado**: los residuos deben seguir una distribución normal estándar, por lo que se esperaría que el 95 % de ellos se encuentre entre -1,96 y 1,96, y el 99 % entre -2,58 y 2,58.
2. **Valor predicho ajustado:** corresponde al valor predicho si se excluyera dicho punto en el ajuste del modelo. Si el punto no ejerce gran influencia en el ajuste del modelo, se esperaría que este valor fuera muy cercano al predicho cuando dicho punto sí es considerado para el ajuste.
3. **Residuo estudiantizado:** está dado por el valor predicho ajustado dividido por el error estándar. Una característica importante de esta medida es que es estandarizada y sigue una distribución t, por lo que puede emplearse para hacer comparaciones entre distintos modelos de regresión. Sin embargo, esta medida solo indica cuánto influye la presencia de un punto en el conjunto de entrenamiento en su valor predicho, pero no proporciona información alguna en cuanto a la influencia de la observación en el modelo como un todo.
4. **Diferencia en ajuste:** más conocido como DFFit, es la diferencia entre el valor predicho para la observación evaluada cuando esta es considerada en el ajuste del modelo y cuando no lo es.
5. **Diferencia en betas:** más conocido como DFBeta, corresponde a la diferencia entre los valores de un

```
Step:  AIC=63.2
mpg ~ wt + cyl

       Df Sum of Sq    RSS    AIC
+ hp    1   14.5514 176.62 62.665
+ carb  1   13.7724 177.40 62.805
<none>              191.17 63.198
+ qsec  1   10.5674 180.60 63.378
+ gear  1    3.0281 188.14 64.687
+ disp  1    2.6796 188.49 64.746
+ vs    1    0.7059 190.47 65.080
+ am    1    0.1249 191.05 65.177
+ drat  1    0.0010 191.17 65.198

Step:  AIC=62.66
mpg ~ wt + cyl + hp

       Df Sum of Sq    RSS    AIC
<none>              176.62 62.665
+ am    1    6.6228 170.00 63.442
+ disp  1    6.1762 170.44 63.526
+ carb  1    2.5187 174.10 64.205
+ drat  1    2.2453 174.38 64.255
+ qsec  1    1.4010 175.22 64.410
+ gear  1    0.8558 175.76 64.509
+ vs    1    0.0599 176.56 64.654
```

Figura 14.10: modelos evaluados por la función `step()` durante el proceso de selección hacia adelante (parte 2).

parámetro cuando es estimado usando todas las observaciones y cuando es estimado sin considerar la observación evaluada. Se calcula para cada parámetro del modelo. Se consideran preocupantes aquellas observaciones en que este estimador es mayor a 1.

6. **Distancia de Cook:** es una medida del efecto que tiene una observación en particular combinadamente en todos los parámetros de un modelo. Aquellos valores para los cuales la distancia de Cook sea mayor a 1 pueden ser considerados como potencialmente problemáticos.
7. **Apalancamiento:** estima la influencia del valor observado en los valores predichos. Toma valores entre 0 y 1. Un apalancamiento igual a 0 señala que un punto no ejerce influencia alguna, mientras que un valor de 1 indica que la influencia ejercida por esa observación es total. Se consideran preocupantes aquellas observaciones para las cuales esta medida supere en dos o tres veces el apalancamiento promedio, dado por la ecuación 14.6, donde k es la cantidad de predictores en el modelo y n la cantidad de observaciones empleadas para el ajuste.

$$\overline{x}_{Apalancamiento} = \frac{k+1}{n} \tag{14.6}$$

8. **Razón de covarianza:** corresponde a la razón entre los determinantes de la matriz de covarianzas cuando se consideran todas las observaciones y cuando se omite la observación en estudio. Aquellas observaciones para las cuales el valor de esta medida estén fuera del intervalo definido por la ecuación 14.7 se consideran preocupantes.

$$|Covratio - 1| < \frac{3(k+1)}{n} \tag{14.7}$$

```
Call:
lm(formula = mpg ~ wt + cyl + hp, data = datos)

Residuals:
    Min      1Q  Median      3Q     Max
-3.9290 -1.5598 -0.5311  1.1850  5.8986

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 38.75179    1.78686  21.687  < 2e-16 ***
wt          -3.16697    0.74058  -4.276 0.000199 ***
cyl         -0.94162    0.55092  -1.709 0.098480 .
hp          -0.01804    0.01188  -1.519 0.140015
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.512 on 28 degrees of freedom
Multiple R-squared:  0.8431,Adjusted R-squared:  0.8263
F-statistic: 50.17 on 3 and 28 DF,  p-value: 2.184e-11

AIC = 155.4766
```

Figura 14.11: modelo obtenido mediante selección hacia adelante.

El script 14.7 muestra la detección de posibles valores atípicos que puedan estar influenciando el modelo de la figura 14.12, obteniéndose los resultados presentados en la figura 14.16. Al examinar estos resultados, en conjunto con los gráficos de la figura 14.15, las observaciones correspondientes al Chrysler Imperial, el Fiat 128 y el Toyota Corolla parecen ser candidatos a eliminación para la generación del modelo. Sin embargo, la distancia de Cook estimada para todas las observaciones potencialmente influyentes están lejos de sobrepasar el valor recomendado, por lo que en este caso no parece ser necesario quitarlos, aun cuando algunas muestren valores un tanto alto de apalancamiento y covarianza. Eliminar datos para construir un modelo debe tener una **sólida justificación**, y por ningún motivo debe hacerse por conveniencia, lo que sería **profundamente antiético**. Consideremos, por ejemplo, que la *pobreza extrema* o los *super-ricos* son casos extremos que, usualmente, no deben eliminarse de un modelo que intente estudiar la distribución de los ingresos de un país.

Script 14.7: identificación de valores atípicos.

```
# Cargar datos.
datos <- mtcars

# Ajustar modelo.
modelo <- lm(mpg ~ wt + qsec + am, data = datos)
plot(modelo)

# Reducir matriz de datos para que solo contenga los predictores
# empleados y la respuesta.
predictores <- names(coef(modelo))[-1]
datos <- datos[, c(predictores, "mpg")]

# Construir una matriz de datos con la respuesta predicha, los
# residuos y algunas estadísticas para evaluar la influencia de
# cada observación.
resultados <- data.frame(respuesta_predicha = fitted(modelo))
```

```
Call:
lm(formula = mpg ~ wt + qsec + am, data = datos)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4811 -1.5555 -0.7257  1.4110  4.6610

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.6178     6.9596   1.382 0.177915
wt           -3.9165     0.7112  -5.507 6.95e-06 ***
qsec          1.2259     0.2887   4.247 0.000216 ***
am            2.9358     1.4109   2.081 0.046716 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.459 on 28 degrees of freedom
Multiple R-squared:  0.8497,Adjusted R-squared:  0.8336
F-statistic: 52.75 on 3 and 28 DF,  p-value: 1.21e-11

AIC = 154.1194
```

Figura 14.12: modelo obtenido mediante eliminación hacia atrás.

```
Call:
lm(formula = mpg ~ wt + cyl + hp, data = datos)

Residuals:
    Min      1Q  Median      3Q     Max
-3.9290 -1.5598 -0.5311  1.1850  5.8986

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 38.75179    1.78686  21.687  < 2e-16 ***
wt          -3.16697    0.74058  -4.276 0.000199 ***
cyl         -0.94162    0.55092  -1.709 0.098480 .
hp          -0.01804    0.01188  -1.519 0.140015
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.512 on 28 degrees of freedom
Multiple R-squared:  0.8431,Adjusted R-squared:  0.8263
F-statistic: 50.17 on 3 and 28 DF,  p-value: 2.184e-11

AIC = 155.4766
```

Figura 14.13: modelo obtenido mediante regresión escalonada.

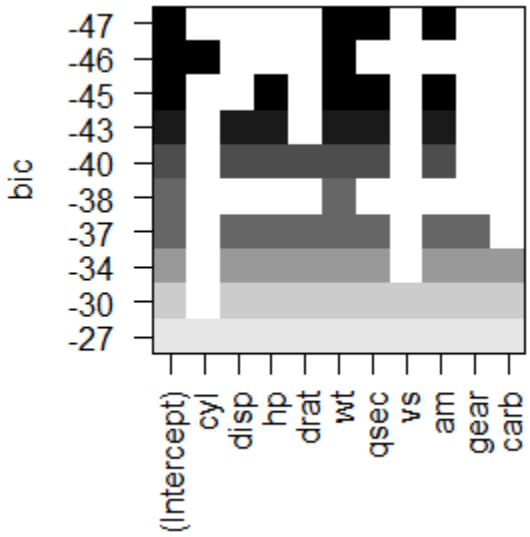
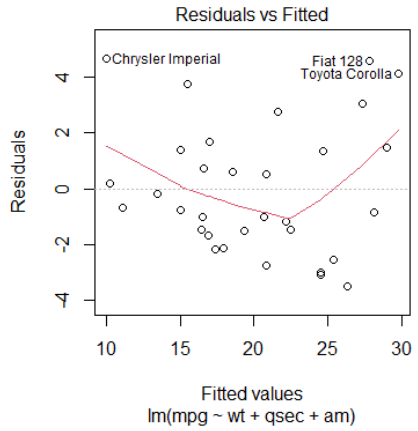
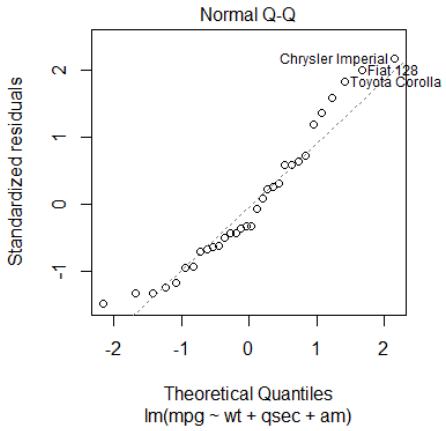


Figura 14.14: representación gráfica de los mejores modelos encontrados mediante el método de todos los subconjuntos.

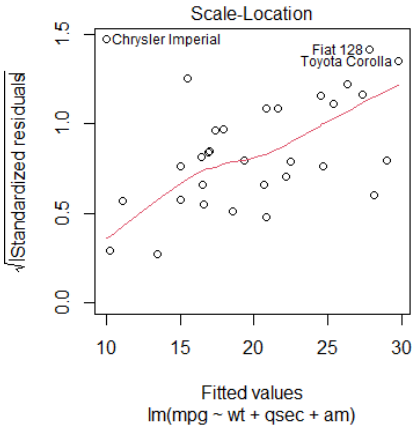
```
resultados[["residuos_estandarizados"]] <- rstandard(modelo)
resultados[["residuos_estudiantizados"]] <-rstudent(modelo)
resultados[["distancia_Cook"]] <- cooks.distance(modelo)
resultados[["dfbeta"]] <- dfbeta(modelo)
resultados[["dffit"]] <- dffits(modelo)
resultados[["apalancamiento"]] <- hatvalues(modelo)
resultados[["covratio"]] <- covratio(modelo)

cat("Identificación de valores atípicos:\n")
# Observaciones con residuos estandarizados fuera del 95% esperado.
sospechosos1 <- which(abs(
  resultados[["residuos_estandarizados"]]) > 1.96)

cat("- Residuos estandarizados fuera del 95% esperado:",
    sospechosos1, "\n")

# Observaciones con distancia de Cook mayor a uno.
sospechosos2 <- which(resultados[["cooks.distance"]] > 1)

cat("- Residuos con una distancia de Cook alta:",
    sospechosos2, "\n")

# Observaciones con apalancamiento mayor igual al doble del
# apalancamiento promedio.

apal_medio <- (ncol(datos) + 1) / nrow(datos)
sospechosos3 <- which(resultados[["apalancamiento"]] > 2 * apal_medio)

cat("- Residuos con apalancamiento fuera de rango:",
    sospechosos3, "\n")

# Observaciones con DFBeta mayor o igual a 1.
sospechosos4 <- which(apply(resultados[["dfbeta"]] >= 1, 1, any))
names(sospechosos4) <- NULL

cat("- Residuos con DFBeta >= 1:",
```



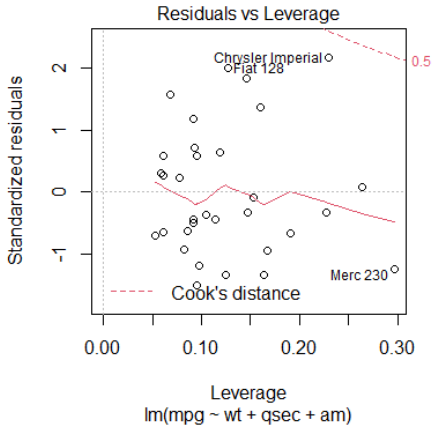
(a) residuos.



(b) distribución de los residuos.



(c) residuos estandarizados.



(d) apalancamiento.

Figura 14.15: gráficos disponibles en R (base) para evaluar un modelo lineal.

```
    sospechosos4, "\n")

# Observaciones con razón de covarianza fuera de rango.
inferior <- 1 - 3 * apal_medio
superior <- 1 + 3 * apal_medio
sospechosos5 <- which(resultados[["covratio"]] < inferior |
                      resultados[["covratio"]] > superior)

cat("- Residuos con razón de covarianza fuera de rango:",
    sospechosos5, "\n")

# Resumen de valores sospechosos.
sospechosos <- c(sospechosos1, sospechosos2, sospechosos3,
                 sospechosos4, sospechosos5)

sospechosos <- sort(unique(sospechosos))

cat("\nResumen de valores sospechosos:\n")
cat("Apalancamiento promedio:", apal_medio, "\n")
```

```r

cat("Intervalo razón de covarianza: [", inferior, "; ",
    superior, "]\n\n", sep = "")

print(round(resultados[sospechosos, c("distancia_Cook", "apalancamiento",
                                      "covratio")], 3))
```

```
Identificación de valores atípicos:
- Residuos estandarizados fuera del 95% esperado: 17 18
- Residuos con una distancia de Cook alta:
- Residuos con apalancamiento fuera de rango: 9 16
- Residuos con DFBeta >= 1: 3 5 6 9 25 28 32
- Residuos con razón de covarianza fuera de rango: 15 16

Resumen de valores sospechosos:
Apalancamiento promedio: 0.125
Intervalo razón de covarianza: [0.625; 1.375]

                    distancia_Cook apalancamiento covratio
Datsun 710                   0.058          0.095     0.919
Hornet Sportabout            0.013          0.093     1.182
Valiant                      0.038          0.097     1.045
Merc 230                     0.162          0.297     1.313
Cadillac Fleetwood           0.008          0.227     1.474
Lincoln Continental          0.001          0.264     1.570
Chrysler Imperial            0.348          0.230     0.724
Fiat 128                     0.146          0.128     0.715
Pontiac Firebird             0.045          0.068     0.856
Lotus Europa                 0.088          0.161     1.050
Volvo 142E                   0.063          0.124     1.016
```

Figura 14.16: identificación de valores atípicos.

14.6.2 Verificación de las condiciones

A fin de que el modelo sea generalizable, tenemos que verificar el cumplimiento de las condiciones descritas en las primeras páginas de este capítulo. Es sencillo comprobar que las variables predictoras son dicotómicas o numéricas a nivel de intervalo y que ninguna de ellas corresponde a una constante. Adicionalmente, las observaciones son independientes entre sí por tratarse de modelos diferentes de automóviles que no parecen seguir un criterio de selección (más que los años de fabricación). A su vez, podemos comprobar que la variable dependiente es numérica a nivel de intervalo sin restricciones.

Al examinar la matriz de correlación (figura 13.4), sin embargo, podemos apreciar una relación positiva moderada entre el tiempo mínimo para recorrer un cuarto de milla y el rendimiento ($R = 0,419$).

La verificación de otras condiciones resulta algo más compleja, por lo que requiere de herramientas matemáticas adicionales.

14.6.2.1 Independencia de los residuos

Esta condición significa que no debe existir autocorrelación en los residuos. Podemos probarlo con una prueba estadística específica conocida con el nombre de sus autores: la prueba de Durbin–Watson, que verifica si dos residuos adyacentes (un retardo), o incluso más alejados, están correlacionados (Durbin & Watson, 1950, 1951). A diferencia de la mayoría de las pruebas de hipótesis, esta prueba define tres regiones: rechazo de H_0, no rechazo de H_0 y una región no concluyente, por lo que existen dos valores críticos, los cuales se buscan en tablas publicadas.

La función `durbinWatsonTest(model)`, del paquete `car`, nos permite aplicar la prueba de Durbin-Watson a los residuos. Sin embargo, debemos tener en cuenta que los resultados de esta prueba dependen del orden de los datos, por lo que al reordenar los datos se podrían obtener resultados diferentes. Al aplicar esta prueba para el ejemplo (script 14.8, línea 12) obtenemos un valor $p = 0,236$, por lo que podemos concluir que los residuos son, en efecto, independientes.

14.6.2.2 Distribución normal de los residuos

Tal como mencionamos previamente, la figura 14.15b muestra que los residuos podrían alejase un poco de la distribución normal. Al aplicar la prueba de Shapiro-Wilk (script 14.8, línea 16), obtenemos como resultado $p = 0,080$, por lo que podemos asumir que el supuesto se cumple, aunque manteniendo cautela por la cercanía con el nivel de significación.

14.6.2.3 Homocedasticidad de los residuos

El gráfico de la figura 14.15a muestra algunos residuos que se alejan levemente del rango general, pero que no parecen ser muy problemáticos.

Una prueba adecuada para verificar esta condición es la de Breusch-Pagan-Godfrey (Glen, 2016), cuya hipótesis nula es que las varianzas de los residuos son iguales. En R, esta prueba está implementada en la función `ncvTest(model)` del paquete `car`. Al usarla para el ejemplo (script 14.8, línea 20), obtenemos como resultado $p = 0,212$, por lo que podemos concluir que el supuesto de homocedasticidad se cumple.

14.6.2.4 Multicolinealidad

Esta condición establece que no debe existir una relación lineal entre dos o más predictores. En otras palabras, la correlación entre variables no debe ser muy alta (o muy baja, si la relación es inversa).

Como hemos dejado entrever a lo largo de este capítulo, el incumplimiento de esta condición puede causar diversos problemas:

- Estimaciones poco confiables de los parámetros. Cuando dos (o más) predictores están perfectamente correlacionados, existen en realidad infinitos valores para sus parámetros que resuelven el problema

de optimización de minimizar la suma de los residuos cuadrados, por lo que la variabilidad de dichos parámetros puede ser muy elevada.
- Limita la magnitud de R, puesto que si los predictores están correlacionados en realidad explican la misma porción de la variabilidad de la respuesta.
- Dificulta evaluar la importancia de cada predictor, por el mismo motivo anterior.

Podemos revisar esta condición mediante el factor de inflación de varianza (VIF) y el estadístico tolerancia ($1/VIF$).

El VIF para cada predictor i se calcula mediante la ecuación 14.8, donde R_i^2 se obtiene usando todos los predictores restantes para ajustar una RLM con el predictor i como respuesta (Frost, 2021).

$$VIF_i = \frac{1}{1 - R_i^2} \tag{14.8}$$

Aunque no hay un acuerdo general, muchos autores usan el valor $VIF \geq 10$ como umbral para preocuparse, aunque hay autores que consideran críticos valores más conservadores, de 5 o incluso de 2,5 (Frost, 2021). También se ha encontrado que si el VIF promedio es mayor a 1, podría haber sesgo en el modelo.

En el caso de la tolerancia, la literatura sugiere que valores bajo 0,2 podrían ser problemáticos, aunque algunos académicos creen que valores cercanos a 0,4 deberían ser revisados.

El paquete `car` de R incluye la función `vif(model)` para calcular el factor de inflación de la varianza. Al usarla para el ejemplo (script 14.8, líneas 23–29) obtenemos los resultados de la figura 14.17.

```
Verificar la multicolinealidad:
- VIFs:
      wt     qsec       am
2.482952 1.364339 2.541437
- Tolerancias:
      wt      qsec        am
0.4027465 0.7329556 0.3934781
- VIF medio: 2.129576
```

Figura 14.17: verificación de condición de multicolinealidad.

Si miramos los factores de inflación de la varianza, en general no parecen ser preocupantes. Sin embargo, los estadísticos de tolerancia son preocupantes. Adicionalmente, el VIF promedio también indica que el modelo podría estar sesgado. Es necesario, entonces, buscar un modelo más confiable. Podríamos, por ejemplo, eliminar la variable menos significativa (`am`, que tiene el menor valor p), obteniendo el modelo con dos predictores de la figura 14.1, el que tendríamos que evaluar y comparar con este modelo de tres predictores (ver los últimos ejercicios propuestos).

Script 14.8: iverificación de condiciones para el modelo.

```
1  library(car)
2
3  # Cargar datos.
4  datos <- mtcars
5
6  # Ajustar modelo.
7  modelo <- lm(mpg ~ wt + qsec + am, data = datos)
8
9  # Comprobar independencia de los residuos.
10 cat("Prueba de Durbin-Watson para autocorrelaciones ")
11 cat("entre errores:\n")
12 print(durbinWatsonTest(modelo))
13
```

```r
# Comprobar normalidad de los residuos.
cat("\nPrueba de normalidad para los residuos:\n")
print(shapiro.test(modelo$residuals))

# Comprobar homocedasticidad de los residuos.
cat("Prueba de homocedasticidad para los residuos:\n")
print(ncvTest(modelo))

# Comprobar la multicolinealidad.
vifs <- vif(modelo)
cat("\nVerificar la multicolinealidad:\n")
cat("- VIFs:\n")
print(vifs)
cat("- Tolerancias:\n")
print(1 / vifs)
cat("- VIF medio:", mean(vifs), "\n")
```

14.6.3 Validación cruzada

Al igual que en el caso de regresión lineal simple, también podemos usar validación cruzada como herramienta para mejorar la estimación del error cuadrado medio. No hay diferencia en la realización de este proceso con respecto a lo estudiado en el capítulo anterior, por lo que no se aborda aquí con mayor detalle.

14.6.4 Tamaño de la muestra

Otro aspecto importante a tener en cuenta al momento de determinar si un modelo de RLM es confiable es el tamaño de la muestra. Existen distintas reglas que simplifican la tarea de determinar el tamaño de la muestra, pero lo cierto es que mientras más observaciones tengamos, mejor. Una de las reglas más simplistas en verificar que se tengan al menos 10 o 15 observaciones por cada predictor. De acuerdo a este criterio, la muestra del ejemplo cuenta con 32 observaciones, con 3 variables predictoras en el modelo, por lo que estaríamos en el borde inferior recomendado.

Considerando, además, que ya hemos encontrado indicios de que el modelo de tres variables podría no ser confiable, parece ser más pertinente usar un modelo más sencillo, como el de la figura 14.1.

14.7 EJERCICIOS PROPUESTOS

1. ¿En qué se diferencia la regresión lineal simple (RLS) de la regresión lineal multivariada (RLM)?
2. ¿Cómo luce la ecuación de un modelo RLM?
3. Si un modelo RLS y un modelo RLM usan una misma variable, ¿tendrán los mismos coeficientes?
4. ¿Qué son los predictores colineales y por qué hay que identificarlos?
5. Explica qué es el coeficiente de determinación R^2 ajustado y para qué se usa.

6. Explica qué son los modelos nulo y completo en el contexto RML.
7. Explica en qué consiste la estrategia de selección de variables hacia adelante.
8. Explica en qué consiste la estrategia de eliminación de variables hacia atrás.
9. Enumera las condiciones necesarias para aplicar RLM.
10. Explica qué se puede ver en los gráficos que entrega la función `plot(modelo)` cuando `modelo` es una RLM.
11. Verifica si el modelo presentado en la figura 14.1 cumple las condiciones para poder usar RLM.
12. ¿Son significativamente distintos los modelos de las figuras 14.1 y 14.12?

CAPÍTULO 15. REGRESIÓN LOGÍSTICA

En los capítulos 13 y 14 estudiamos la regresión lineal, útil para predecir una respuesta numérica a partir de una o más variables, aunque con una serie de condiciones. Como explican Field y col. (2012, pp. 312-345), la **regresión logística** es un **modelo lineal generalizado**, que admite una variable de respuesta cuyos residuos sigan una distribución diferente a la normal.

La regresión logística relaciona la distribución de la variable de respuesta con un modelo lineal usando como función de enlace la **función logística estándar**, también conocida como `logit()`, que presentamos en la ecuación 15.1 y mostramos gráficamente en la figura 15.1. Esta función describe una **transición de cero a uno**, por lo que resulta especialmente útil para representar la **probabilidad** de que ocurra algún evento: un valor cercano a cero indica que es muy poco probable, mientras un valor cercano a 1 corresponde a una alta probabilidad (lógicamente, un valor de $0,5$ indica que es igualmente probable que el evento ocurra o no). Así, la regresión logística resulta adecuada para predecir una respuesta dicotómica, pues puede ser asociada a una **distribución binomial**.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{15.1}$$

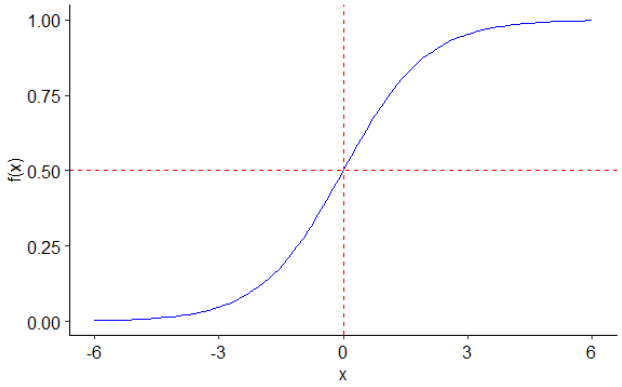


Figura 15.1: función logística.

Para entender mejor esta idea, necesitamos introducir el concepto de ***odds*** (Cerda y col., 2013), el cual no tiene una traducción directa al castellano, pero que puede entenderse como "oportunidad" o "chance", aunque a veces se traduce incorrectamente como "probabilidad". Matemáticamente, el *odds ratio* está dado por la ecuación 15.2, por lo que se define como la razón entre la probabilidad de que ocurra un evento y la probabilidad de que este no ocurra.

$$odds = \frac{p}{1 - p} \tag{15.2}$$

Tomemos el ejemplo que usan (Cerda y col., 2013): supongamos que los registros históricos dicen que en junio llueve 12 días. Así, la probabilidad de que un día de junio sea lluvioso es:

$$p = \frac{12}{30} = 0,4$$

Pero la oportunidad de que el día sea lluvioso es:

$$odds = \frac{12}{18} = 0,67$$

Ambas medidas presentan la misma información, pero de manera diferente. Cuando un evento e tiene las mismas posibilidades de ocurrir o no, $p(e) = 0,5$ y $odds(e) = 1$.

Suponiendo que el logaritmo de los *odds* sigue una distribución normal, podemos relacionar los *odds* y las probabilidades como muestran las ecuaciones 15.3 y 15.4.

$$z = \log(\frac{p}{1-p}) \tag{15.3}$$

$$p = \text{logit}(z) = \frac{1}{1+e^{-z}} \tag{15.4}$$

A su vez, también podemos asociar z a otras variables, como muestra la ecuación 15.5.

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \tag{15.5}$$

Así, la regresión logística nos permite **asociar** la probabilidad de ocurrencia de un evento e a una combinación lineal de variables predictoras $x_1, x_2, \ldots, x_n$, de acuerdo a la ecuación 15.6.

$$p(e) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n)}} \tag{15.6}$$

De esta forma podemos seleccionar una división o **umbral** que permita **predecir** la ocurrencia de un evento e o, de forma equivalente, **clasificar** un objeto en dos categorías posibles:

- Para valores menores que el umbral se predice que el evento e "no ocurre", otorgándose una clasificación cero ($\hat{y} = 0$) o negativa ($\hat{y} = -$).
- Mientras que para valores mayores o iguales que el umbral se predice que el evento "sí ocurre", a lo que corresponde una clasificación uno ($\hat{y} = 1$) o positiva ($\hat{y} = +$).

Si bien es usual utilizar el valor $p(e) = 0,5$ como umbral, esto no es obligatorio ni siempre conveniente, como veremos más adelante.

En capítulos precedentes explicamos que el ajuste de un modelo de regresión lineal se realiza mediante la resolución de un problema de optimización que busca minimizar la suma de las desviaciones cuadradas entre las respuestas predichas y las observadas. En el caso de la regresión logística, también se ajusta mediante la resolución de un problema de optimización, donde buscamos minimizar la diferencia entre las respuestas observadas y las respuestas predichas. Como las respuestas corresponden a una **variable dicotómica**, esta optimización se realiza usando la función de verosimilitud $\mathcal{L}(p)$, aunque, por conveniencia, se suele optimizar el logaritmo natural de la verosimilitud, tal y como se muestra en la ecuación 15.7.

$$\begin{aligned}
\mathcal{L}(p) &= P(y_1, y_2, \ldots, y_n|p) \text{ with } y_i \in \{0,1\} \\
&= \prod_{i=1}^{n} p^{y_i}(1-p)^{y_i} \\
&\text{Luego,} \\
\ln \mathcal{L}(p) &= \sum_{i=1}^{n} \left[y_i \ln p_{(y_i=1)} + (1-y_i)\ln(1-p_{(y_i=1)}) \right]
\end{aligned} \tag{15.7}$$

15.1 EVALUACIÓN DE UN CLASIFICADOR

Una forma de evaluar modelos de clasificación, entre ellos los de regresión logística, es de acuerdo a la cantidad de errores cometidos (Zelada, 2017). Para ello, el primer paso consiste en construir una tabla de contingencia (también llamada matriz de confusión) para las respuestas predichas y observadas, como muestra la tabla 15.1, bastante similar a la que ya conocimos para explicar los errores de decisión en la prueba de hipótesis (tabla 4.1). Las cuatro celdas de la matriz de confusión contienen:

- **Verdaderos positivos** (VP): cantidad de instancias correctamente clasificadas como pertenecientes a la clase positiva.
- **Falsos positivos** (FP): cantidad de instancias erróneamente clasificadas como pertenecientes a la clase positiva.
- **Falsos negativos** (FN): cantidad de instancias erróneamente clasificadas como pertenecientes a la clase negativa.
- **Verdaderos negativos** (VN): cantidad de instancias correctamente clasificadas como pertenecientes a la clase negativa.

		Real		
		1 (+)	**0 (−)**	**Total**
Clasificación	**1 (+)**	VP	FP	$VP + FP$
	0 (−)	FN	VN	$FN + VN$
	Total	$VP + FN$	$FP + VN$	n

Tabla 15.1: tabla de contingencia para evaluar un clasificador.

La **exactitud** (*accuracy*) del modelo corresponde a la proporción de observaciones correctamente clasificadas, dada por la ecuación 15.8.

$$\text{exactitud} = \frac{VP + VN}{n} \tag{15.8}$$

A su vez, el **error** del modelo corresponde a la proporción de observaciones clasificadas de manera equivocada (ecuación 15.9).

$$\text{error} = \frac{FP + FN}{n} = 1 - \text{exactitud} \tag{15.9}$$

La **sensibilidad** (*sensitivity* o *recall*, ecuación 15.10) indica cuán apto es el modelo para detectar aquellas observaciones pertenecientes a la clase positiva.

$$\text{sensibilidad} = \frac{VP}{VP + FN} \tag{15.10}$$

De manera análoga, la **especificidad** (*specificity*, ecuación 15.11) permite determinar cuán exacta es la asignación de elementos a la clase positiva. También puede entenderse como la aptitud del modelo para correctamente asignar observaciones a la clase negativa.

$$\text{especificidad} = \frac{VN}{FP + VN} \tag{15.11}$$

La **precisión** (*precision*) o valor predictivo positivo (VPP, ecuación 15.12) indica la proporción de instancias clasificadas como positivas que realmente lo son.

$$VPP = \frac{VP}{VP + FP} \tag{15.12}$$

Asimismo, el **valor predictivo negativo** (VPN, ecuación 15.13) señala la proporción de instancias correctamente clasificadas como pertenecientes a la clase negativa.

$$VPN = \frac{VN}{FN + VN} \tag{15.13}$$

Otra herramienta útil es la **curva de calibración**, también llamada curva ROC por las siglas inglesas para *receiver-operating characteristic*, que muestra la relación entre la sensibilidad y la especificidad del modelo (Glen, 2017). Este gráfico también permite evaluar la precisión del modelo, puesto que mientras más se aleje la curva de la diagonal, mayor es la precisión. Para ilustrar mejor la utilidad de este gráfico, la figura 15.2 muestra las curvas ROC para dos modelos diferentes, además de la diagonal. Esta figura indica que el clasificador representado por la curva morada es mejor que el representado por la curva azul, pues se aleja más de la diagonal.

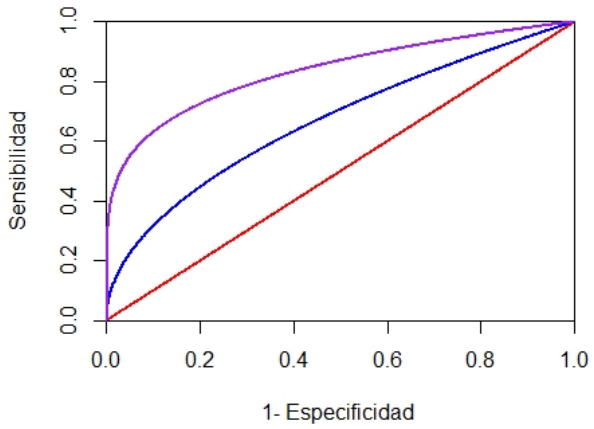


Figura 15.2: dos curvas ROC. Fuente: Ayala (2020).

15.2 BONDAD DE AJUSTE DEL MODELO

Al igual que en el caso de la regresión lineal, existen diversos mecanismos para evaluar el ajuste de un modelo de regresión logística. El estadístico de **log-verosimilitud** ($\ln \mathcal{L}$), dado por la ecuación 15.7, nos permite cuantificar la diferencia entre las probabilidades predichas y las observadas. Este estadístico se asemeja a la suma de los residuos cuadrados de la regresión lineal en el sentido de que cuantifica la cantidad de información que carece de explicación tras el ajuste del modelo. Así, mientras menor sea su valor, mejor es el ajuste del modelo.

La **desviación** (en inglés *deviance*), a menudo denotada por $-2LL$ y en pocas ocasiones llamada *devianza*, suele usarse en lugar de la log-verosimilitud porque sigue una distribución χ^2, lo que facilita calcular el nivel de significación del valor. Está dada por la ecuación 15.14.

$$-2LL = -2 \cdot \ln \mathcal{L} \tag{15.14}$$

Los criterios de evaluación de modelos basados, en el principio de parsimonia que estudiamos en el capítulo 14, también están definidos para la regresión logística. El más sencillo es el criterio de información de Akaike (AIC), dado por la ecuación 15.15, donde k corresponde a la cantidad de predictores en el modelo.

$$AIC = -2LL + 2k \tag{15.15}$$

Similar al AIC, el criterio bayesiano de Schwarz (BIC) ajusta la penalización a la complejidad del modelo según el tamaño de la muestra, como muestra la ecuación 15.16.

$$BIC = -2LL + 2k \cdot \ln n \tag{15.16}$$

15.3 REGRESIÓN LOGÍSTICA EN R

En R, podemos ajustar un modelo de regresión logística mediante la función `glm(formula, family = binomial(link = "logit"), data)`, donde:

- `formula` tiene la forma <variable de respuesta>∼<variable predictora>.
- `data`: matriz de datos.

Puesto que existen otros modelos generalizados de regresión lineal, el argumento `family = binomial(link = "logit")` indica que asumiremos una distribución binomial para la variable de respuesta y que usaremos la función logística.

Las líneas 11–19 del script 15.1 ilustran el uso de la función `glm()` para ajustar un modelo de regresión logística que prediga el tipo de transmisión de un automóvil (0 = automática, 1 = manual) a partir de su peso, usando para ello el ya conocido conjunto de datos `mtcars` disponible en R (recordemos que podemos consultar la descripción de las variables en la tabla 13.1). Para ello, consideramos un conjunto de entrenamiento con 80 % de las instancias. El modelo resultante se muestra en la figura 15.3, donde podemos apreciar que el AIC es bastante bajo ($AIC = 16,23$) y que la desviación del modelo con una variable (23 grados de libertad) es de $12,23$.

Las líneas 22–33 evalúan el modelo ajustado usando las herramientas descritas en la sección anterior y el conjunto de entrenamiento. La función `roc(response, predictor)` del paquete `pROC`, donde los argumentos corresponden, respectivamente, a las respuestas observadas y las respuestas predichas, nos permite obtener la curva ROC de la figura 15.4. La curva se aleja bastante de la diagonal, por lo que al parecer se trata de un buen modelo. A su vez, la función `confusionMatrix(data, reference)` del paquete `caret`, donde `data` corresponde a la respuesta predicha y `reference` a la observada, genera la matriz de confusión y obtiene las medidas de evaluación descritas anteriormente, como muestra la figura 15.5. Podemos ver que el modelo tiene una exactitud de 92,0 %. La sensibilidad de 100 % y la especificidad de 83,33 % muestran que el modelo se desempeña un poco mejor identificando elementos de la clase positiva, correspondiente en este caso a los vehículos de transmisión automática.

Pero, como ya hemos estudiado en capítulos anteriores, debemos evaluar el modelo con un conjunto de datos diferente al que usamos para su construcción. Así, las líneas 36–46 obtienen la curva ROC (figura 15.6) y la matriz de confusión (figura 15.7) para el conjunto de prueba, donde observamos un resultado con menor exactitud que con el conjunto de entrenamiento. Esto es una indicación de que el modelo podría estar un poco sobreajustado para el conjunto de entrenamiento, pero también de que el conjunto de prueba puede ser muy pequeño para obtener una evaluación confiable.

```
Call:
glm(formula = am ~ wt, family = binomial(link = "logit"), data = entrenamiento)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.17498  -0.40172  -0.00176   0.12321   2.26151

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   18.525      8.504   2.178   0.0294 *
wt            -5.883      2.645  -2.224   0.0261 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34.617  on 24  degrees of freedom
Residual deviance: 12.230  on 23  degrees of freedom
AIC: 16.23

Number of Fisher Scoring iterations: 7
```

Figura 15.3: ajuste de un modelo de regresión logística.

Script 15.1: ajuste de un modelo de regresión logística en R.

```r
library(pROC)
library(caret)

set.seed(1313)

# Cargar los datos.
datos <- mtcars
datos$am <- factor(datos$am)

# Separar conjuntos de entrenamiento y prueba.
n <- nrow(datos)
n_entrenamiento <- floor(0.8 * n)
muestra <- sample.int(n = n, size = n_entrenamiento, replace = FALSE)
entrenamiento <- datos[muestra, ]
prueba  <- datos[-muestra, ]

# Ajustar modelo.
modelo <- glm(am ~ wt, family = binomial(link = "logit"), data = entrenamiento)
print(summary(modelo))

# Evaluar el modelo con el conjunto de entrenamiento.
cat("Evaluación del modelo a partir del conjunto de entrenamiento:\n")
probs_e <- predict(modelo, entrenamiento, type = "response")

umbral <- 0.5
preds_e <- sapply(probs_e, function(p) ifelse(p >= umbral, "1", "0"))
preds_e <- factor(preds_e, levels = levels(datos[["am"]]))

ROC_e <- roc(entrenamiento[["am"]], probs_e)
plot(ROC_e)
```

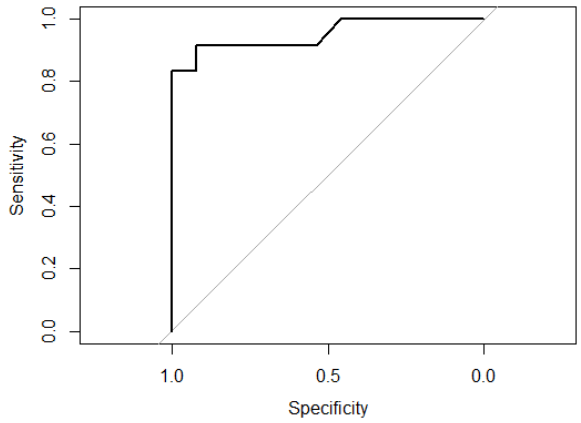


Figura 15.4: curva ROC obtenida al evaluar el modelo con el conjunto de entrenamiento.

```
31
32 matriz_e <- confusionMatrix(preds_e, entrenamiento[["am"]])
33 print(matriz_e)
34
35 # Evaluar el modelo con el conjunto de prueba.
36 cat("Evaluación del modelo a partir del conjunto de prueba:\n")
37 probs_p <- predict(modelo, prueba, type = "response")
38
39 preds_p <- sapply(probs_p, function(p) ifelse(p >= umbral, "1", "0"))
40 preds_p <- factor(preds_p, levels = levels(datos[["am"]]))
41
42 ROC_p <- roc(entrenamiento[["am"]], probs_e)
43 plot(ROC_p)
44
45 matriz_p <- confusionMatrix(preds_p, prueba[["am"]])
46 print(matriz_p)
```

15.4 CONDICIONES PARA USAR REGRESIÓN LOGÍSTICA

Desde luego, no basta con evaluar el desempeño del clasificador, sino que también necesitamos verificar el cumplimiento de ciertas condiciones para que un modelo de regresión logística sea válido:

1. Debe existir una relación lineal entre los predictores y la respuesta transformada.
2. Los residuos deben ser independientes entre sí.

Además de las condiciones anteriores, existen otras situaciones en que puede ocurrir que el método de optimización no converja:

1. Multicolinealidad entre los predictores, que en este caso se aborda del mismo modo que para RLM (por ejemplo, mediante el factor de inflación de la varianza o la tolerancia).
2. Información incompleta, que se produce cuando no contamos con observaciones suficientes para todas las posibles combinaciones de predictores.
3. Separación perfecta, que ocurre cuando no hay superposición entre las clases, es decir, ¡cuando los predictores separan ambas clases completamente!

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 13  2
         1  0 10

               Accuracy : 0.92
                 95% CI : (0.7397, 0.9902)
    No Information Rate : 0.52
    P-Value [Acc > NIR] : 2.222e-05

                  Kappa : 0.8387

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 1.0000
            Specificity : 0.8333
         Pos Pred Value : 0.8667
         Neg Pred Value : 1.0000
             Prevalence : 0.5200
         Detection Rate : 0.5200
   Detection Prevalence : 0.6000
      Balanced Accuracy : 0.9167

       'Positive' Class : 0
```

Figura 15.5: matriz de confusión y medidas de evaluación con el conjunto de entrenamiento para el modelo ajustado.

15.5 GENERALIZACIÓN DEL MODELO

En capítulos anteriores conocimos la validación cruzada como herramienta para mejorar la estimación del error, la cual podemos usar de manera análoga para regresión logística. El script 15.2 mejora el ejercicio realizado en el script 15.1, incorporando el uso de validación cruzada de 5 pliegues. Notemos que la llamada a la función `train()` también solicita que "se guarden" los valores predichos, lo que nos permite estimar el rendimiento promedio del modelo como si se repitiera el script 15.2, seleccionando aleatoriamente un conjunto de entrenamiento y otro de prueba, cinco veces.

Debemos fijarnos en que el modelo obtenido es idéntico al anterior (por lo que no se muestra aquí), ya que la función `train()` reentrena el modelo del pliegue que obtuvo mejor rendimiento con todos los datos disponibles. En el caso de la regresión logística (como con la regresión lineal), los pliegues solo se diferencian en los datos que utilizan, por lo que siempre se llega al mismo modelo. Esto no sería así si la validación cruzada se usara, por ejemplo, para seleccionar las variables predictoras a incluir en el modelo.

Script 15.2: ajuste de un modelo de regresión logística usando validación cruzada.

```
1 library(caret)
2
3 set.seed(1313)
4
5 # Cargar los datos.
6 datos <- mtcars
7 datos$am <- factor(datos$am)
```

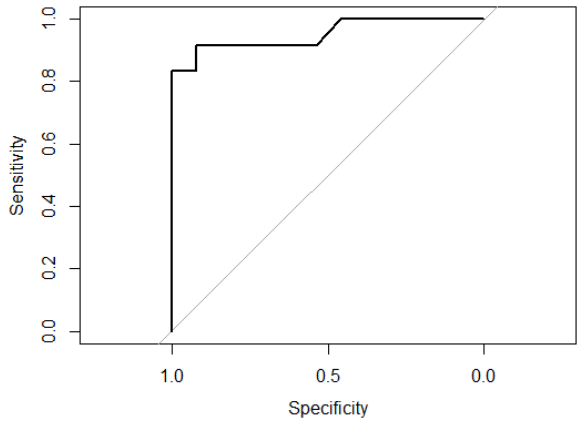


Figura 15.6: curva ROC obtenida al evaluar el modelo con el conjunto de prueba.

```
8
9  # Ajustar modelo usando validación cruzada de 5 pliegues.
10 modelo <- train(am ~ wt, data = entrenamiento, method = "glm",
11                  family = binomial(link = "logit"),
12                  trControl = trainControl(method = "cv", number = 5,
13                                           savePredictions = TRUE))
14
15 print(summary(modelo))
16
17 # Evaluar el modelo
18 cat("Evaluación del modelo basada en validación cruzada:\n")
19 matriz <- confusionMatrix(modelo$pred$pred, modelo$pred$obs)
20 print(matriz)
```

15.6 SELECCIÓN DE PREDICTORES

Cuando tenemos múltiples predictores potenciales, debemos decidir cuáles de ellos incorporar en el modelo. Una vez más, y tal como detallamos en el capítulo 14, el ideal es usar la regresión jerárquica para escoger los predictores de acuerdo a evidencia disponible en la literatura. Sin embargo, al explorar los datos, podemos emplear los demás métodos ya descritos: selección hacia adelante, eliminación hacia atrás, regresión escalonada o todos los subconjuntos. Se usan para ello las mismas funciones de R descritas en el capítulo 14.

15.7 COMPARACIÓN DE MODELOS

Al igual que con los modelos de regresión lineal, podemos comparar modelos de regresión logística mediante la función `anova()`, aunque ahora la prueba F resulta inapropiada. En cambio, una prueba muy utilizada en este caso es el *Likelihood Ratio Test* (LRT), el cual compara qué tanto más "probables" son los datos con un

```
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 5 0
         1 1 1

               Accuracy : 0.8571
                 95% CI : (0.4213, 0.9964)
    No Information Rate : 0.8571
    P-Value [Acc > NIR] : 0.7365

                  Kappa : 0.5882

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.8333
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.5000
             Prevalence : 0.8571
         Detection Rate : 0.7143
   Detection Prevalence : 0.7143
      Balanced Accuracy : 0.9167

       'Positive' Class : 0
```

Figura 15.7: matriz de confusión y medidas de evaluación con el conjunto de prueba para el modelo ajustado.

modelo que con el otro. Podemos ver un ejemplo de esta comparación más adelante en el script 15.3.

15.8 REGRESIÓN LOGÍSTICA EN R CON SELECCIÓN DE PREDICTORES

En páginas previas ajustamos un modelo de regresión logística para determinar el tipo de transmisión de un automóvil a partir de su peso. Sin embargo, el predictor fue seleccionado de manera aleatoria, simplemente para ilustrar el proceso, por lo que podríamos encontrar un mejor modelo usando algún método de selección de predictores. Las líneas 19–32 del script 15.3 llevan a cabo esta tarea usando regresión escalonada, obteniéndose como resultado el modelo presentado en la figura 15.8.

Sin embargo, al ajustar este modelo R emite algunas advertencias, como muestra la figura 15.9. Estas ocurren cuando los predictores separan completamente las clases, o bien cuando existen problemas de colinealidad. Al verificar los factores de inflación de la varianza de los predictores (script 15.3, líneas 35–41) podemos apreciar que, si bien ninguna de las variables presenta un *VIF* superior a 10, el promedio es bastante superior a 1 (figura 15.10), lo que confirma que el modelo puede tener problemas. En consecuencia, no es recomendable usar este modelo.

Por regla general, se recomienda eliminar la variable con mayor *VIF*, pero en este caso ambos son iguales. En consecuencia, en las líneas 44–57 del script 15.3 se ajustan los dos modelos posibles (figuras 15.11 y 15.12) y

```
Call:
glm(formula = am ~ wt + hp, family = binomial(link = "logit"),
    data = entrenamiento)

Deviance Residuals:
      Min          1Q      Median          3Q         Max
-3.378e-05  -2.100e-08  -2.100e-08   2.100e-08   2.597e-05

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.165e+02  3.478e+05   0.001    0.999
wt          -1.555e+02  1.287e+05  -0.001    0.999
hp           4.788e-01  4.620e+02   0.001    0.999

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3.4617e+01  on 24  degrees of freedom
Residual deviance: 2.2982e-09  on 22  degrees of freedom
AIC: 6

Number of Fisher Scoring iterations: 25
```

Figura 15.8: modelo de regresión logística obtenido mediante regresión escalonada.

```
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Figura 15.9: modelo de regresión logística obtenido mediante regresión escalonada.

luego se comparan (figura 15.13). Pero en este caso la prueba ANOVA no sirve, pues ambos modelos tienen igual cantidad de predictores y no entrega un valor p. Sin embargo, podemos ver que el *VIF* del modelo con la potencia como predictor ($VIF = 37,444$) es más alto que para el modelo con el peso como predictor ($VIF = 16,23$). En consecuencia, este último parece ser mejor.

A modo de ejercicio, a pesar de que lo descartamos por tener problemas de colinealidad, comparamos el modelo obtenido mediante regresión escalonada con el que tiene al peso como variable predictora (script 15.3, línea 68), obteniendo como resultado un valor $p < 0,001$ (figura 15.14). Puesto que el valor p obtenido es significativo, la prueba arroja que el modelo más complejo (es decir, el que tiene dos predictores) reduce la varianza de los residuos de forma significativa (¡llegando a cero!).

Dado que el mejor modelo es el mismo que habíamos usado en secciones previas, no repetiremos la evaluación

```
Verificación de colinealidad
-------------------------------------

VIF:
     wt      hp
4.14191 4.14191

Promedio VIF: [1] 4.14191
```

Figura 15.10: factores de inflación de la varianza para los modelos.

```
Call:
glm(formula = am ~ wt, family = binomial(link = "logit"), data = entrenamiento)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.17498  -0.40172  -0.00176   0.12321   2.26151

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   18.525      8.504   2.178   0.0294 *
wt            -5.883      2.645  -2.224   0.0261 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34.617  on 24  degrees of freedom
Residual deviance: 12.230  on 23  degrees of freedom
AIC: 16.23

Number of Fisher Scoring iterations: 7
```

Figura 15.11: modelo de regresión logística con el peso como predictor.

con el conjunto de prueba, pues ya presentamos el resultado en la figura 15.7.

Sin embargo, aún resta verificar el cumplimiento de la condición de independencia de los residuos, para lo cual, al igual que con modelos de regresión lineal, empleamos la prueba de Durbin-Watson (script 15.3, línea 75), cuyo resultado mostramos en la figura 15.15, donde podemos notar que, aunque cerca del borde para $\alpha = 0,05$, los residuos son independientes.

Una vez verificadas las condiciones, podemos concluir que el modelo es adecuado y puede ser generalizado. No obstante, aún falta determinar si su ajuste se ve afectado por la presencia de valores atípicos (script 15.3, líneas 78–137). La figura 15.16 muestra los gráficos asociados al modelo. El gráfico de la figura 15.16a muestra una única instancia cuyo residuo se aleja muchísimo de los demás, correspondiente al Maserati Bora, el cual también se aleja bastante de la recta esperada en el gráfico Q-Q de los residuos (figura 15.16b). A su vez, la figura 15.16d muestra claramente que esa misma instancia ejerce un importante apalancamiento.

Usando herramientas más precisas para replicar el análisis, parte de cuyos resultados presentamos en la figura 15.17[1], podemos determinar que la única observación cuyo residuo estandarizado escapa a la normalidad es el Maserati Bora. Asimismo, esta instancia presenta la mayor distancia de Cook y los mayores DFBeta, por lo que es la única que resulta preocupante y podría ser útil eliminarla para el ajuste del modelo. También queda como ejercicio reentrenar y evaluar el modelo sin considerar esta observación.

Script 15.3: ajuste y evaluación del mejor modelo para predecir el tipo de transmisión de un automóvil.

```
1 library(car)
2
3 set.seed(1313)
4
5 # Cargar los datos.
6 datos <- mtcars
7 am <- factor(datos$am)
8 datos$am <- NULL
```

[1]Por una cuestión de espacio, queda como ejercicio para el lector ejecutar el script 15.3 y ver el detalle de los valores obtenidos para las distintas métricas evaluadas para las observaciones sospechosas.

```
Call:
glm(formula = am ~ hp, family = binomial(link = "logit"), data = entrenamiento)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3663  -1.0648  -0.8953   1.1182   1.7415

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.820490   0.941997   0.871    0.384
hp          -0.006236   0.005965  -1.045    0.296

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34.617  on 24  degrees of freedom
Residual deviance: 33.444  on 23  degrees of freedom
AIC: 37.444

Number of Fisher Scoring iterations: 4
```

Figura 15.12: modelo de regresión logística con la potencia como predictor.

```
Analysis of Deviance Table

Model 1: am ~ wt
Model 2: am ~ hp
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        23     12.230
2        23     33.444  0  -21.214
```

Figura 15.13: comparación de los modelos con un único predictor.

```r
datos <- cbind(am, datos)

# Separar conjuntos de entrenamiento y prueba.
n <- nrow(datos)
n_entrenamiento <- floor(0.8 * n)
muestra <- sample.int(n = n, size = n_entrenamiento, replace = FALSE)
entrenamiento <- datos[muestra, ]
prueba  <- datos[-muestra, ]

# Ajustar modelo nulo.
nulo <- glm(am ~ 1, family = binomial(link = "logit"), data = entrenamiento)

# Ajustar modelo completo.
cat("\n\n")
completo <- glm(am ~ ., family = binomial(link = "logit"),
                data = entrenamiento)

# Ajustar modelo con regresión escalonada.
cat("Modelo con regresión escalonada\n")
cat("-------------------------------------\n")
mejor <- step(nulo, scope = list(lower = nulo, upper = completo),
              direction = "both", trace = 0)

```

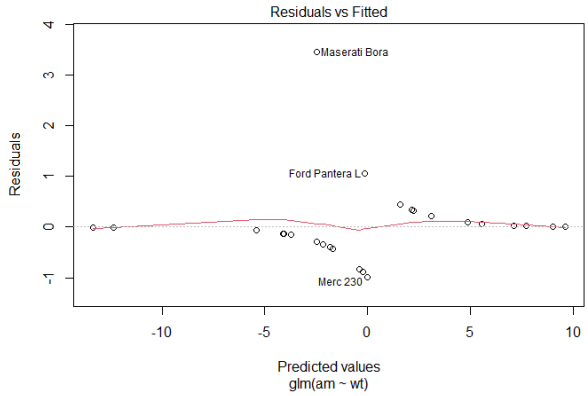
```
Analysis of Deviance Table

Model 1: am ~ wt
Model 2: am ~ wt + hp
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1        23      12.23
2        22       0.00  1    12.23 0.0004704 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 15.14: comparación del modelo con dos predictores y el que solo tiene el peso como predictor.

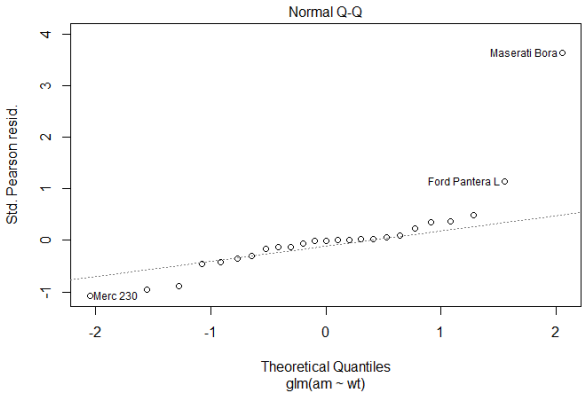
```
 lag Autocorrelation D-W Statistic p-value
   1     -0.30715746      2.583329   0.084
Alternative hypothesis: rho[lag] != 0
```

Figura 15.15: resultado de la prueba de Durbin-Watson para verificar la independencia de los residuos del modelo que solo tiene el peso como predictor.

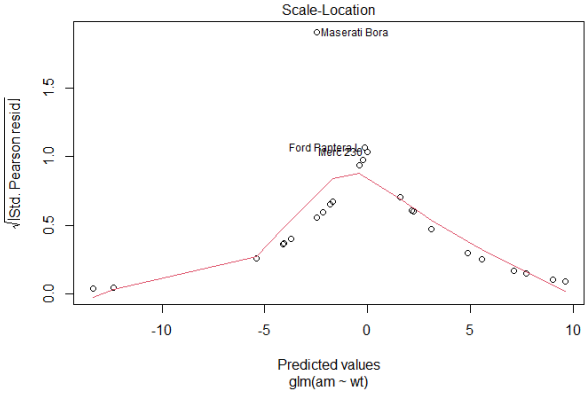
```
print(summary(mejor))

# Verificación de multicolinealidad.
cat("Verificación de colinealidad\n")
cat("-----------------------------------\n")
cat("\nVIF:\n")
vifs <- vif(mejor)
print(vifs)
cat("\nPromedio VIF: ")
print(mean(vifs))

# Ajustar modelo con el peso como predictor.
cat("Modelo con el peso como predictor\n")
cat("-----------------------------------\n")
modelo_peso <- glm(am ~ wt, family = binomial(link = "logit"),
                   data = entrenamiento)

print(summary(modelo_peso))

# Ajustar modelo con la potencia como predictor.
cat("Modelo con la potencia como predictor\n")
cat("-----------------------------------\n")
modelo_potencia <- glm(am ~ hp, family = binomial(link = "logit"),
                       data = entrenamiento)

print(summary(modelo_potencia))

# Comparar los modelos con el peso y la potencia como predictores.
cat("\n\n")
cat("Likelihood Ratio Test para los modelos\n")
cat("-----------------------------------\n")
print(anova(modelo_peso, modelo_potencia, test = "LRT"))

# A modo de ejercicio, comparar el modelo obtenido mediante
# regresión escalonada con el que solo tiene el peso como predictor.
cat("\n\n")
cat("Likelihood Ratio Test para los modelos\n")
```



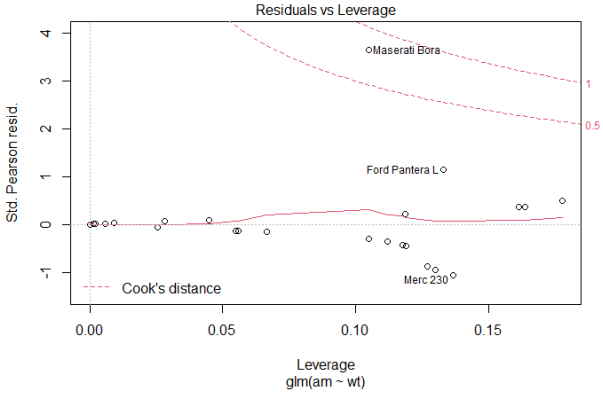
(a) residuos.



(b) distribución de los residuos.



(c) residuos estandarizados.



(d) apalancamiento.

Figura 15.16: gráficos para evaluar el modelo de regresión logística.

```r
cat("--------------------------------------\n")
print(anova(modelo_peso, mejor, test = "LRT"))

# Independencia de los residuos.
cat("Verificación de independencia de los residuos\n")
cat("--------------------------------------\n")
print(durbinWatsonTest(modelo_peso, max.lag = 5))

# Detectar posibles valores atípicos.
cat("Identificación de posibles valores atípicos\n")
cat("--------------------------------------\n")
plot(mejor)

# Obtener los residuos y las estadísticas.
output <- data.frame(predicted.probabilities = fitted(modelo_peso))
output[["standardized.residuals"]] <- rstandard(modelo_peso)
output[["studentized.residuals"]] <- rstudent(modelo_peso)
output[["cooks.distance"]] <- cooks.distance(modelo_peso)
output[["dfbeta"]] <- dfbeta(modelo_peso)
output[["dffit"]] <- dffits(modelo_peso)
output[["leverage"]] <- hatvalues(modelo_peso)

```

```
Residuales estandarizados fuera del 95% esperado
------------------------------------------------
[1] "Maserati Bora"

Residuales con una distancia de Cook alta
-----------------------------------------
character(0)

Residuales con levarage fuera de rango (> 0.344)
------------------------------------
character(0)

Residuales con DFBeta sobre 1
-----------------------------
[1] "Dodge Challenger" "Maserati Bora"    "Merc 280"
[4] "Valiant"          "Ferrari Dino"     "Volvo 142E"
[7] "Mazda RX4 Wag"

Casos sospechosos
-----------------
                 am  mpg cyl  disp  hp drat    wt  qsec vs gear carb
Dodge Challenger  0 15.5   8 318.0 150 2.76 3.520 16.87  0    3    2
Maserati Bora     1 15.0   8 301.0 335 3.54 3.570 14.60  0    5    8
Merc 280          0 19.2   6 167.6 123 3.92 3.440 18.30  1    4    4
Valiant           0 18.1   6 225.0 105 2.76 3.460 20.22  1    3    1
Ferrari Dino      1 19.7   6 145.0 175 3.62 2.770 15.50  0    5    6
Volvo 142E        1 21.4   4 121.0 109 4.11 2.780 18.60  1    4    2
Mazda RX4 Wag     1 21.0   6 160.0 110 3.90 2.875 17.02  0    4    4
```

Figura 15.17: identificación de posibles valores atípicos.

```
# Evaluar residuos estandarizados que escapen a la normalidad.
# 95% de los residuos estandarizados deberían estar entre
# -1.96 y 1.96, y 99% entre -2.58 y 2.58.
sospechosos1 <- which(abs(output[["standardized.residuals"]]) > 1.96)
sospechosos1 <- sort(sospechosos1)
cat("\n\n")
cat("Residuos estandarizados fuera del 95% esperado\n")
cat("-----------------------------------------------\n")
print(rownames(entrenamiento[sospechosos1, ]))

# Revisar casos con distancia de Cook mayor a uno.
sospechosos2 <- which(output[["cooks.distance"]] > 1)
sospechosos2 <- sort(sospechosos2)
cat("\n\n")
cat("Residuales con una distancia de Cook alta\n")
cat("-----------------------------------------\n")
print(rownames(entrenamiento[sospechosos2, ]))

# Revisar casos cuyo apalancamiento sea más del doble
# o triple del apalancamiento promedio.
leverage.promedio <- ncol(entrenamiento) / nrow(datos)
sospechosos3 <- which(output[["leverage"]] > leverage.promedio)
sospechosos3 <- sort(sospechosos3)
cat("\n\n")
```

```
cat("Residuales con levarage fuera de rango (> ")
cat(round(leverage.promedio, 3), ")", "\n", sep = "")
cat("-------------------------------------\n")
print(rownames(entrenamiento[sospechosos3, ]))

# Revisar casos con DFBeta >= 1.
sospechosos4 <- which(apply(output[["dfbeta"]] >= 1, 1, any))
sospechosos4 <- sort(sospechosos4)
names(sospechosos4) <- NULL
cat("\n\n")
cat("Residuales con DFBeta sobre 1\n")
cat("-----------------------------\n")
print(rownames(entrenamiento[sospechosos4, ]))

# Detalle de las observaciones posiblemente atípicas.
sospechosos <- c(sospechosos1, sospechosos2, sospechosos3, sospechosos4)
sospechosos <- sort(unique(sospechosos))
cat("\n\n")
cat("Casos sospechosos\n")
cat("-----------------\n")
print(entrenamiento[sospechosos, ])
cat("\n\n")
print(output[sospechosos, ])
```

15.9 EJERCICIOS PROPUESTOS

1. ¿Qué es un modelo lineal generalizado?
2. ¿Qué modela una regresión logística?
3. Explica por qué este modelo lleva el apellido "logística".
4. ¿Por qué se considera un modelo lineal?
5. Menciona las condiciones necesarias para aplicar regresión logística.
6. Explica las evaluaciones que deben aplicarse a un modelo de regresión logística.
7. Investiga cuáles son las hipótesis que se contrastan al hacer inferencia con la regresión logística.
8. ¿Se podría buscar un buen modelo de regresión logística usando el paquete `caret`? Investigue.
9. Reconstruye el último modelo de regresión logística que conseguimos, pero ahora sin considerar el "Maserati Bora" y evalúa si cumple las condiciones para ser usado.

REFERENCIAS

Amat Rodrigo, J. (2016a). *Resampling: test de permutación, simulación de Monte Carlo y Bootstrapping.* Consultado el 31 de mayo de 2021, desde https://www.cienciadedatos.net/documentos/23_resampling_test_permutacion_simulacion_de_monte_carlo_bootstrapping

Amat Rodrigo, J. (2016b). *Test de Friedman.* Consultado el 29 de mayo de 2021, desde https://www.cienciadedatos.net/documentos/21_friedman_test

Amat Rodrigo, J. (2016c). *Test Kruskal-Wallis.* Consultado el 29 de mayo de 2021, desde https://www.cienciadedatos.net/documentos/20_kruskal-wallis_test

Amat Rodrigo, J. (2016d). *Validación de modelos predictivos: Cross-validation, OneLeaveOut, Bootstraping.* Consultado el 23 de diciembre de 2021, desde https://www.cienciadedatos.net/documentos/30_cross-validation_oneleaveout_bootstrap#K-Fold_Cross-Validation

Ayala, J. (2020). *Minería de datos.* Consultado el 23 de junio de 2021, desde https://rpubs.com/JairoAyala/592802

Baguley, T. (2012). *Beware the Friedman test!* Consultado el 13 de diciembre de 2021, desde https://seriousstats.wordpress.com/2012/02/14/friedman/

Berman, H. (2000). *Scheffé's Test for Multiple Comparisons.* Consultado el 7 de mayo de 2021, desde https://stattrek.com/anova/follow-up-tests/scheffe.aspx

Cerda, J., Vera, C. & Rada, G. (2013). Odds ratio: aspectos teóricos y prácticos. *Revista médica de Chile*, *141*, 1329-1335.

Diez, D., Barr, C. D. & Çetinkaya-Rundel, M. (2017). *OpenIntro Statistics* (3.ª ed.). https://www.openintro.org/book/os/.

Durbin, J. & Watson, G. S. (1950). Testing for serial correlation in least squares regression: I. *Biometrika*, *37*(3/4), 409-428.

Durbin, J. & Watson, G. S. (1951). Testing for serial correlation in least squares regression: II. *Biometrika*, *38*(1/2), 159-179.

Field, A., Miles, J. & Field, Z. (2012). *Discovering statistics using R.* SAGE Publications Ltd.

Frost, J. (2021). *Variance Inflation Factors (VIFs).* Consultado el 17 de junio de 2021, desde https://statisticsbyjim.com/regression/variance-inflation-factors/

Glen, S. (2016). *Breusch-Pagan-Godfrey Test: Definition.* Consultado el 16 de junio de 2021, desde https://www.statisticshowto.com/breusch-pagan-godfrey-test/

Glen, S. (2017). *Receiver Operating Characteristic (ROC) Curve: Definition, Example.* Consultado el 23 de junio de 2021, desde https://www.statisticshowto.com/receiver-operating-characteristic-roc-curve/

Glen, S. (2021a). *Coefficient of Determination (R Squared).* Consultado el 10 de junio de 2021, desde https://www.statisticshowto.com/probability-and-statistics/coefficient-of-determination-r-squared/

Glen, S. (2021b). *Kruskal Wallis H Test: Definition, Examples & Assumptions.* Consultado el 5 de junio de 2021, desde https://www.statisticshowto.com/kruskal-wallis/

Glen, S. (2021c). *Post-Hoc Definition and Types of Post Hoc Tests.* Consultado el 7 de mayo de 2021, desde https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/post-hoc/#PHscheffes

Hesterberg, T., Monaghan, S., Moore, D. S., Clipson, A. & Epstein, R. (2003). *Bootstrap Methods and Permutation Tests.* Consultado el 3 de junio de 2021, desde https://statweb.stanford.edu/~tibs/stat315a/Supplements/bootstrap.pdf

Horn, R. A. (2008). *Sphericity in repeated measures analysis.* Consultado el 11 de mayo de 2021, desde http://oak.ucc.nau.edu/rh232/courses/EPS625/Handouts/RM-ANOVA/Sphericity.pdf

IBM. (1989). *ANOVA de un factor: Contrastes post hoc.* Consultado el 30 de abril de 2021, desde https://www.ibm.com/docs/es/spss-statistics/25.0.0?topic=anova-one-way-post-hoc-tests

Irizarry, R. A. (2019). *Introduction to Data Science.* https://rafalab.github.io/dsbook/.

Karadimitriou, S. M. & Marshall, E. (2016).
Repeated measures ANOVA in R [statstutor community project]. Consultado el 12 de mayo de 2021, desde https://www.sheffield.ac.uk/polopoly_fs/1.885219!/file/105_RepeatedANOVA.pdf

Lærd Statistics. (2020a). *Friedman Test in SPSS Statistics* [Lund Research Ltd.]. Consultado el 5 de junio de 2021, desde https://statistics.laerd.com/spss-tutorials/friedman-test-using-spss-statistics.php

Lærd Statistics. (2020b). *Sphericity* [Lund Research Ltd.]. Consultado el 11 de mayo de 2021, desde https://statistics.laerd.com/statistical-guides/sphericity-statistical-guide.php

Lowry, R. (1999). *Concepts & Applications of Inferential Statistics.*
Consultado el 3 de mayo de 2021, desde http://vassarstats.net/textbook/

Meier, L. (2021). *ANOVA: A Short Intro Using R.*
Consultado el 7 de mayo de 2021, desde https://stat.ethz.ch/~meier/teaching/anova/

Montero Muñoz, J., Solla Suárez, P. E. & Gutiérrez Rodríguez, J. (2012). Estimación del filtrado glomerular en el paciente anciano. Implicaciones clínicas en el uso de antibióticos.
Revista Española de Geriatría y Gerontología, *56*(5), 268-271-.

NIST/SEMATECH. (2013). *e-Handbook of Statistical Methods.*
Consultado el 29 de abril de 2021, desde http://www.itl.nist.gov/div898/handbook/

Pardoe, I., Simon, L. & Young, D. (2018). *Residuals vs. Fits Plot.*
Consultado el 21 de diciembre de 2021, desde https://online.stat.psu.edu/stat462/node/117/

Real Statistics Using Excel. (s.f.). *Mann-Whitney Table.* Consultado el 28 de mayo de 2021, desde https://www.real-statistics.com/statistics-tables/mann-whitney-table/

Winner, L. (2021). *Simple Linear Regression I — Least Squares Estimation.*
Consultado el 8 de junio de 2021, desde http://users.stat.ufl.edu/~winner/qmb3250/notespart2.pdf

Zelada, C. (2017). *Evaluación de modelos de clasificación.*
Consultado el 23 de junio de 2021, desde https://rpubs.com/chzelada/275494