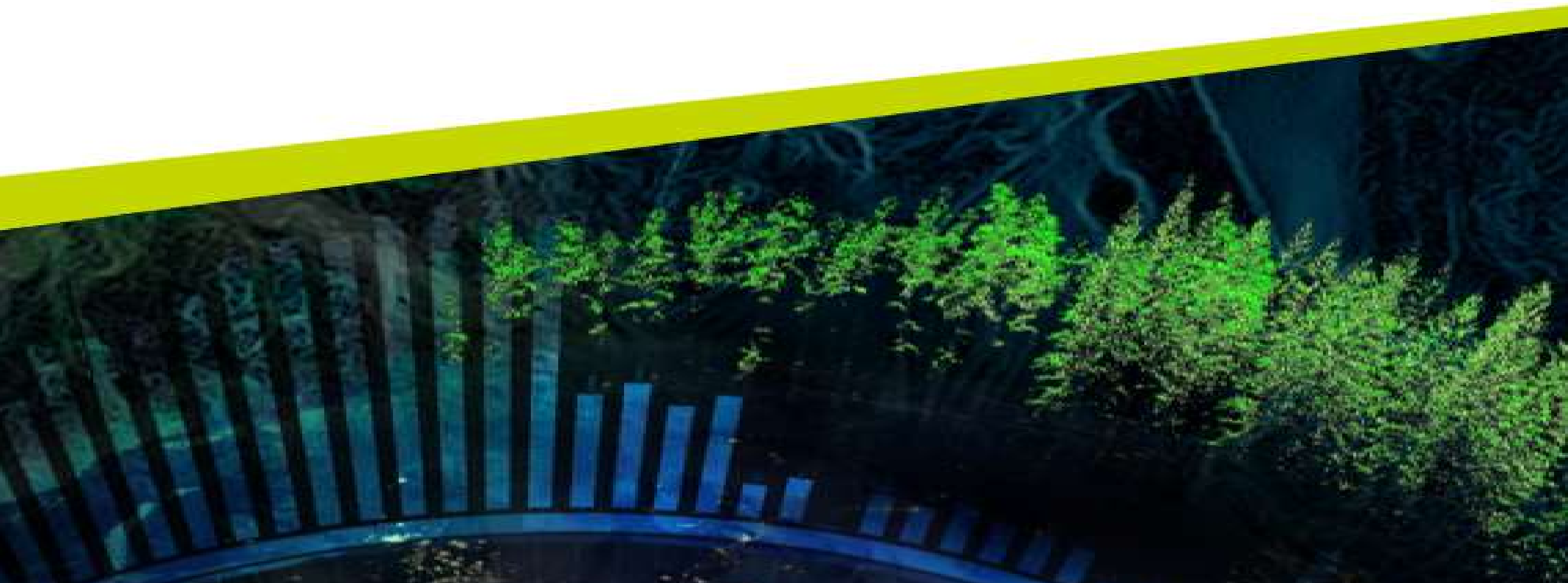
INFERENCIA Y MODELOS ESTADÍSTICOS

Jacqueline Köhler C. y José Luis Jara V.



12.2 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 paramétrico**, basándonos en las ideas descritas por Amat Rodrigo (2016) y Hesterberg et al. (2003).

12.2.1 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.

En general, si queremos inferir el valor de un parámetro de la población θ, hasta ahora lo hemos hecho a partir de un estimador puntual $\hat{\theta}$ calculado desde una muestra. Aplicar bootstrapping en este proceso de inferencia, en términos generales, sigue los siguientes pasos:

1. Crear una gran cantidad B de nuevas muestras (cientos, miles, decenas de miles y hasta cientos de miles) a partir de la muestra original, a las que se les suele llamar **remuestras**. Cada remuestra 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 el estadístico de interés $\hat{\theta}^*$ para cada una de las remuestras; aquí se usa "*" para indicar que corresponde a un **estadístico bootstrap**, es decir, obtenido desde una remuestra generada con bootstrapping. Estos estadísticos bootstrap producen una distribución empírica del estadístico $\hat{\theta}$, la que se conoce como **distribución bootstrap**.
3. Usar la distribución bootstrap para obtener información útil acerca de la forma, el centro y la variabilidad de la distribución muestral del estadístico de interés $\hat{\theta}$.

De esta forma, podemos obtener estadísticos bootstrap desde la distribución bootstrap. Por ejemplo, podemos obtener su media y error estándar por medio de las ecuaciones 12.7 y 12.8, respectivamente.

$$\overline{x}_{(\hat{\theta}^*,B)} = \frac{1}{B}\sum_{i=1}^{B}\hat{\theta}_i^* \tag{12.7}$$

$$SE_{(\hat{\theta}^*,B)} = \sqrt{\frac{\sum_{i=1}^{B}\left(\hat{\theta}_i^* - \overline{x}_{(\hat{\theta}^*,B)}\right)^2}{B-1}} \tag{12.8}$$

También es posible obtener fácilmente un intervalo de confianza para $\hat{\theta}$, simplemente se considera el rango de valores en torno al centro de la distribución muestral que cumple con el $100(1-\alpha)\,\%$ de confianza deseado.

Es importante remarcar que la distribución bootstrap se centra en el **estadístico observado** $\hat{\theta}$, y no, como desearíamos, en el parámetro θ. Este resultado teórico origina otro estadístico útil, llamado **sesgo**, *bias* en inglés, que corresponde al desplazamiento del estadístico $\hat{\theta}$ de la media de la distribución bootstrap que genera, y que está dado por la ecuación 12.9.

$$\delta_{(\hat{\theta}^*,B)} = \overline{x}_{(\hat{\theta}^*,B)} - \hat{\theta} \tag{12.9}$$

El mayor uso de bootstrapping apunta a construir intervalos de confianza más precisos para el parámetro θ a partir de la distribución bootstrap de $\hat{\theta}$, y no solo de la estimación puntual (un valor) que $\hat{\theta}$ entrega. A partir de allí, la técnica nos permite contrastar hipótesis.

Si bien, como sugiere esta introducción, la técnica de bootstrapping es útil para prácticamente **cualquier estadístico**, revisaremos su aplicación con la media, es decir cuando $\theta = \mu$ y $\hat{\theta} = \overline{x}$. Queda pendiende (como ejercicio propuesto) aplicarla a proporciones ($\theta = p$ y $\hat{\theta} = \hat{p}$).

12.2.2 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 12.9 muestra la distribución del tiempo de ejecución para la muestra.

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.

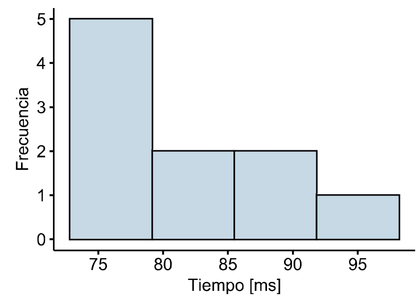


Figura 12.9: distribución del tiempo de ejecución para la muestra ejemplo.

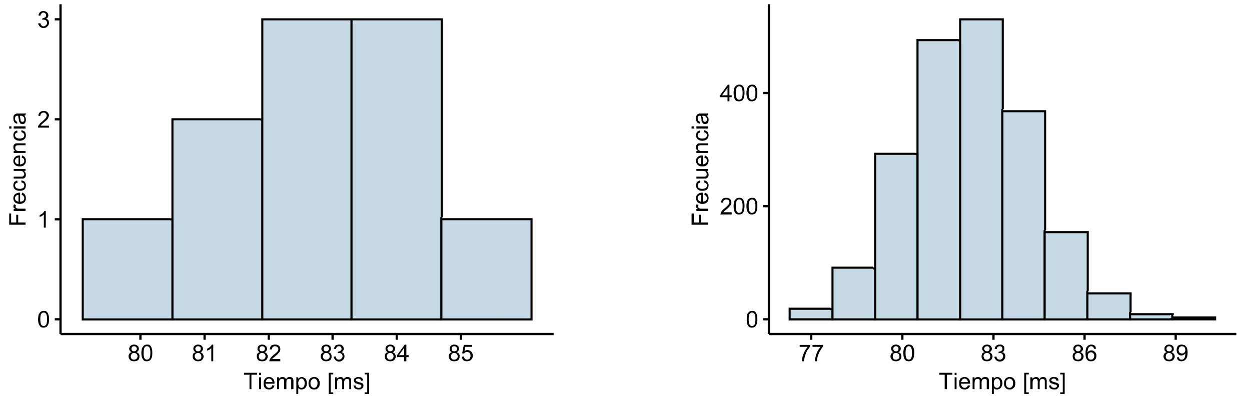
Evidentemente, la muestra es pequeña ($n = 10$) y su distribución exhibe asimetría hacia la derecha, por lo que Chufe ha decidido emplear bootstrapping como alternativa para enfrentar estos datos problemáticos.

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

Original	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
79	90	88	90	90	79	79	79	79	79	76
75	84	76	76	84	79	82	79	75	90	79
84	94	76	94	82	79	76	84	84	75	88
75	88	94	84	79	75	79	90	94	88	84
94	82	79	84	90	84	76	94	94	79	88
82	82	82	94	88	94	84	76	79	90	94
76	75	90	75	82	75	82	94	82	75	82
90	90	79	75	76	79	90	79	84	90	82
79	84	79	79	76	90	79	82	79	79	75
88	79	75	84	75	79	75	88	82	79	94
82,2	**84,8**	**81,8**	**83,5**	**82,2**	**81,3**	**80,2**	**84,5**	**83,2**	**82,4**	**84,2**

Tabla 12.2: muestra original y remuestreos de bootstrap

La figura 12.10 muestra la distribución bootstrap de la media para los 10 remuestreos del ejemplo (figura 12.10a) y para 2.000 remuestreos (figura 12.10b). 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.



(a) 10 remuestreos.

(b) 2.000 remuestreos.

Figura 12.10: distribuciones bootstrap de la media.

En la tabla 12.2 vemos que $\overline{x} = 82{,}2$, y que el promedio bootstrap es:

$$\overline{x}_{(\overline{x}^*,10)} = \frac{1}{10}\sum_{i=1}^{10} \overline{x}_i^* = \frac{84{,}8 + 81{,}8 + 83{,}5 + 82{,}2 + 81{,}3 + 80{,}2 + 84{,}5 + 83{,}2 + 82{,}4 + 84{,}2}{10} = 82{,}81$$

La figura 12.10b también nos da una idea acerca de la variabilidad de los promedios de las diferentes remuestras que la muestra original genera. Para el ejemplo con 10 remuestreos, la suma de las desviaciones cuadradas es:

$$\begin{aligned}\sum_{i=1}^{10} \left(\overline{x}_i^* - 82{,}81\right)^2 &= (84{,}8 - 82{,}81)^2 + (81{,}8 - 82{,}81)^2 + (83{,}5 - 82{,}81)^2 + \\ &\quad (82{,}2 - 82{,}81)^2 + (81{,}3 - 82{,}81)^2 + (80{,}2 - 82{,}81)^2 + \\ &\quad (84{,}5 - 82{,}81)^2 + (83{,}2 - 82{,}81)^2 + (82{,}4 - 82{,}81)^2 + \\ &\quad (84{,}2 - 82{,}81)^2 \\ &= 20{,}029\end{aligned}$$

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

$$SE_{(\overline{x}^*,10)} = \sqrt{\frac{20{,}029}{10-1}} = 2{,}225$$

Y el sesgo en este caso es:

$$\delta_{(\overline{x}^*,10)} = \overline{x}_{(\overline{x}^*,10)} - \overline{x} = 82{,}81 - 82{,}20 = 0{,}61$$

Ahora que ya conocemos toda esta información de la distribución bootstrap para la media, podemos **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, ya que $0{,}61 \ll 82{,}20$), podemos construir un intervalo de confianza del mismo modo que hicimos en el capítulo 4, teniendo el cuidado de corregir el sesgo detectado, como muestra la ecuación 12.10, donde $z^\star$ es el valor crítico para el nivel de confianza requerido.

$$(\overline{x} - \delta_{(\overline{x}^*,10)}) \pm z^\star \cdot SE_{(\overline{x}^*,10)} \tag{12.10}$$

Así, si consideramos para este ejemplo un nivel de significación $\alpha = 0{,}01$, el valor crítico (bilateral) es $z^* = z_{(1-\alpha/2)} = 2{,}576$. En consecuencia, el intervalo de 99 % confianza resultante para la media de la población es [75,859; 87,321].

Otra alternativa cuando la distribución bootstrap se asemeja a la normal, y que tiene en cuenta posibles asimetrías, es construir el intervalo de confianza en base a cuantiles críticos. En este caso, para $\alpha = 0{,}01$, los límites del intervalo están dados por los percentiles 1 y 99 de la distribución bootstrap, que para el ejemplo son: $[80{,}250;\ 84{,}786]$.

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. En R existen varias funciones para "replicar" operaciones sobre vectores o matrices, por lo que implementar bootstrapping, y remuestreo en general, no se dificulta demasiado. Pero además existen muchos paquetes que implementan numerosas funciones *wrapper* para entregar interfaces específicas, en teoría, más simples para usos comunes. Uno de los más usados es el paquete `boot`, que ofrece 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.frame`, se considera cada fila como una observación con múltiples variables (columnas).
- `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 (es decir, 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 Z, `"perc"` para el basado en los percentiles y `"bca"` para el método BCa).

Debemos mencionar que la función `boot()` puede recibir otros muchos argumentos, los cuales escapan al alcance de los contenidos aquí expuestos.

El script 12.6 construye intervalos de confianza mediante bootstrapping para el ejemplo, con $B = 2.000$ y manteniendo el nivel de significación $\alpha = 0{,}01$. En las líneas 13–14 se construye la función para el estadístico de interés (en este caso el promedio), que luego usa la función `boot()` para generar la distribución bootstrap[2] (línea 19). La líneas 22–23 muestran las estadísticas de la distribución bootstrap obtenida como se presentan en la figura 12.11 bajo el título "`ORDINARY NONPARAMETRIC BOOTSTRAP`".

Podemos ver gráficamente la distribución bootstrap obtenida mediante un histograma y un gráfico Q-Q (figura 12.12), gracias a la llamada a la función `plot()` con el resultado entregado por la función `boot()` como argumento (línea 24 del script). Vemos que esta distribución sigue un comportamiento aproximadamente normal, tal vez con una pequeña asimetría positiva.

En las líneas 27–28 se muestra el uso de `boot.ci()` para construir los intervalos de confianza usando diferentes métodos, obteniéndose los resultados mostrados en la figura 12.11 bajo el título "`BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS`", es decir $[77{,}03; 87{,}25]$ usando aproximación normal, $[77{,}40; 87{,}70]$ usando percentiles y $[77{,}48; 87{,}90]$ al usar el método BCa.

Las líneas 36–37 muestran otra alternativa para construir la distribución bootstrap, esta vez por medio del paquete `bootES`, que ofrece la función `bootES(data, R, ci.type, ci.conf, plot, ...)`. Esta función es una *wrapper*, que internamente realiza una llamada a la función `boot()` descrita en los párrafos precedentes, que no requiere implementar previamente la función para el cálculo de las medias de las remuestras. 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`).

[2]Notemos que la línea 18 inicializa la semilla para la generación de números pseudoaleatorios de manera de garantizar la reproducibilidad de los resultados.

```
*** Paquete 'boot' ***

ORDINARY NONPARAMETRIC BOOTSTRAP


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


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


BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL :
boot.ci(boot.out = distribucion_b, conf = 1 - alfa, type = c("norm",
    "perc", "bca"))

Intervals :
Level      Normal             Percentile            BCa
99%   (77.03, 87.25 )   (77.40, 87.70 )   (77.48, 87.90 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable


*** Paquete 'bootES' ***

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 12.11: salida del script 12.6 que ejemplifica el uso de las funciones `boot()`, `boot.ci()` y `bootES()` para el ejemplo.

- `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.

La llamada de las líneas 36–37 genera los mismos gráficos mostrados en la figura 12.12. El resultado obtenido con la función `bootES()` puede verse en la parte baja de a figura 12.11 bajo el título "`*** Paquete 'bootES' ***`".

Con estos resultados podemos concluir que tenemos 99 % de confianza de que el algoritmo tarda entre 77,48 ms y 87,90 ms en ejecutar las instancias del tamaño seleccionado.

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]

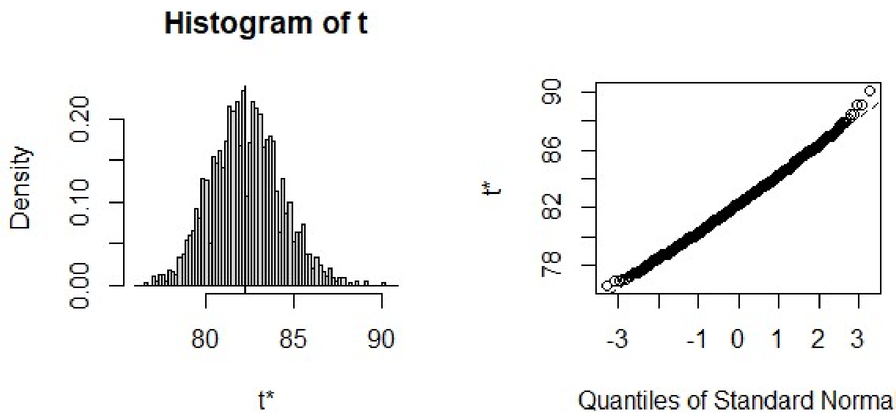


Figura 12.12: histograma y gráfico Q-Q de la distribución bootstrap generada para el ejemplo.

Script 12.6: construcción de un intervalo de confianza para la media poblacional mediante bootstrapping.

```
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

# Función para calcular el estadístico: media de la remuestra.
media <- function(valores, i) {
  mean(valores[i])
}

# Construir la distribución bootstrap usando el paquete boot.
set.seed(432)
distribucion_b <- boot(muestra, statistic = media, R = B)

# Mostrar y graficar la distribución bootstrap.
cat("*** Paquete 'boot' ***\n")
print(distribucion_b)
plot(distribucion_b)

# Construir y mostrar los intervalos de confianza.
ics <- boot.ci(distribucion_b, conf = 1 - alfa,
               type = c("norm", "perc", "bca"))
cat("\n\n")
print(ics)

# Construir distribución bootstrap usando el paquete bootES.
# Esta llamada además calcula (solo) un intervalo de confianza
# y grafica la distribución bootstrap.
set.seed(432)
distribucion_bES <- bootES(muestra, R = B, ci.type = "bca",
                           ci.conf = 1 - alfa, plot = TRUE)

```

```
# Mostrar bootstrap obtenida con bootES.
cat("\n\n*** Paquete 'bootES' ***\n")
print(distribucion_bES)
```

El contraste de hipótesis clásico se basa en una distribución muestral 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 que represente la hipótesis nula. 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 12.11, 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.

$$p = \frac{r+1}{B+1} \tag{12.11}$$

Tras hacer la prueba (script 12.7), 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 % de confianza que el tiempo de ejecución promedio del algoritmo para instancias del tamaño seleccionado supera los 75 milisegundos. Por supuesto, si Helen supiera un poco más de estadística inferencial, no le sorprendería este resultado puesto que el valor nulo no está contenido en el intervalo de confianza obtenido ($75 \notin [77{,}48; 87{,}90]$), por lo que su procedimiento solo sirve para obtener un valor p que reportar.

Script 12.7: inferencia sobre la media de una muestra con bootstrapping.

```
library(boot)

# 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])
}

set.seed(432)
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)
```

12.2.3 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:
   a) hacer un remuestreo con reposición de tamaño n_A a partir de la muestra A
   b) hacer un remuestreo con reposición de tamaño n_B a partir de la muestra B.
   c) calcular el estadístico de interés con las remuestras conseguidas
3. Construir el intervalo de confianza para el estadístico de interés a partir de la distribución bootstrap generada.

El paquete `simpleboot` facilita la construcción de distribuciones bootstrap para la **diferencia** de dos parámetros por medio de la función *wrapper* `two.boot(sample1, sample2, FUN, R)`, donde:

- `sample1, sample2`: muestras originales.
- `FUN`: función que calcula el estadístico de interés para cada remuestra.
- `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 (FUN(resample1) - FUN(resample2)).

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, los investigadores han comprobado que las notas de los varones no siguen una distribución normal ($W = 0{,}884$, $p = 0{,}006$), por lo que han decidido usar bootstrapping para la prueba de hipótesis, con un nivel de significación $\alpha = 0{,}05$ y $B = 9.999$ repeticiones.

El script 12.8 muestra el desarrollo de este ejemplo en R. 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 12.13), con media $\overline{x} = -1{,}628$ y desviación estándar $s = 0{,}377$. Notemos que debemos construir estos gráficos de forma manual (líneas 36–42 del script) al utilizar la función `two.boot()`.

Al construir el intervalo de confianza mediante el método BCa para la distribución bootstrap (líneas 53–54), R nos entrega como resultado el intervalo $[-2{,}372; -0{,}894)]$.

Supongamos ahora que el estudio del ejemplo pretende determinar, con un nivel de significación $\alpha = 0{,}05$, si la diferencia entre las calificaciones finales entre hombres y mujeres es mayor a 5 décimas, en favor de las mujeres. Las hipótesis serían:

Sean μ_h y μ_m las medias de las calificaciones finales de hombres y mujeres, respectivamente, en una asignatura inicial de programación al rendirla por primera vez en la Universidad en estudio, entonces:
H_0: $\mu_h - \mu_m = -0{,}5$
H_A: $\mu_h - \mu_m < -0{,}5$

Tras conseguir la distribución bootstrap para la hipótesis nula (líneas 59–61 del script 12.8) y determinar la proporción de remuestras con valores tanto o más extremos que el observado en la muestra original (línea 64), obtenemos $p = 0{,}001$, inferior al nivel de significación, por lo que rechazamos la hipótesis nula en favor de la alternativa. En consecuencia, concluimos con 95 % de confianza que la diferencia en la calificación final entre hombres y mujeres es mayor a 5 décimas en favor de las mujeres.

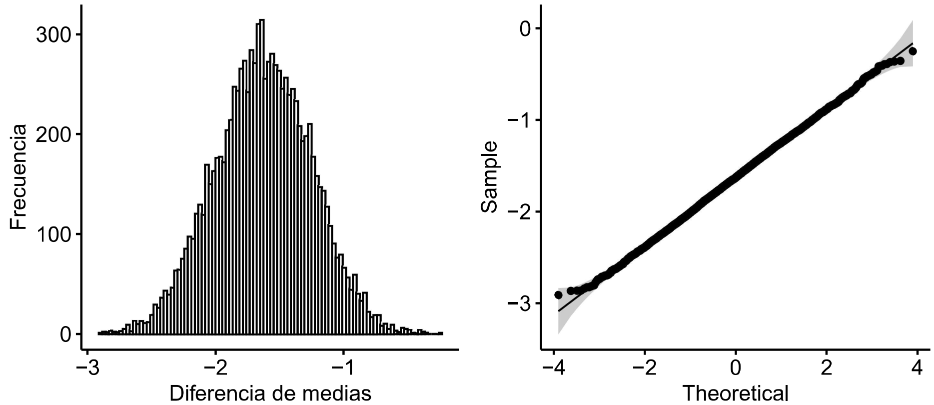


Figura 12.13: histograma y gráfico Q-Q de la distribución bootstrap generada para el ejemplo de la diferencia de dos medias.

Script 12.8: bootstraping para la diferencia de dos medias del ejemplo.

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

# 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)

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

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

cat("Media hombres:", round(media_hombres,3), "\n")
cat("Media mujeres:", round(media_mujeres,3), "\n")
cat("Diferencia observada:", round(diferencia_obs, 3), "\n")
cat("\n")

# Crear la distribución bootstrap.
B <- 9999
set.seed(432)
distribucion_b <- two.boot(hombres, mujeres, FUN = mean, R = B)

# Examinar la distribución bootstrap.
datos <- data.frame(diferencias = distribucion_b[["t"]])
g_hist <- gghistogram(datos, x = "diferencias", bins = 100,
                      xlab = "Diferencia de medias",
                      ylab = "Frecuencia")
g_qq <- ggqqplot(datos, x = "diferencias")
```

```
g <- ggarrange(g_hist, g_qq)
print(g)

media_b <- mean(datos[["diferencias"]])
sd_b <- sd(datos[["diferencias"]])

cat("Distribución bootstrap:\n")
cat("\tMedia:", round(media_b, 3), "\n")
cat("\tDesviación estándar:", round(sd_b, 3), "\n\n")

# Construir y mostrar los intervalos de confianza.
alfa <- 0.05
intervalo_bca <- boot.ci(distribucion_b, conf = 1 - alfa,
                         type = "bca")

print(intervalo_bca)

# Desplazar la distribución bootstrap para reflejar la hipótesis nula.
valor_nulo <- -0.5
desplazamiento <- media_b - valor_nulo
distribucion_nula <- datos[["diferencias"]] - desplazamiento

# Determinar el valor p.
p <- (sum(distribucion_nula < diferencia_obs) + 1) / (B + 1)
cat("\nValor p:", p, "\n")
```

12.2.4 Bootstrapping para dos muestras apareadas

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 para el caso de una única muestra que ya conocimos.

Supongamos ahora, que la Universidad del ejemplo anterior 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. Para ello, dispone de las siguientes calificaciones (en escala de 1,0 a 7,0) obtenidas en ambas pruebas para una muestra de 20 estudiantes:

Prueba 1: 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: 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

La Universidad ha decidido llevar a cabo el estudio mediante bootstrapping con $B = 3.999$ repeticiones y un nivel de significación $\alpha = 0{,}05$. Así, el equipo de investigación ha formulado las siguientes hipótesis:

Sean P_i^1 y P_i^2 las calificaciones obtenidas por el/la i-ésimo/a estudiante en la primera y la segunda prueba, respectivamente, de un curso inicial de programación. Sea $D_i = P_i^2 - P_i^1$ las diferencias de estas calificaciones para cada estudiante, con media μ_D. Entonces:
H_0: $\mu_D = 0{,}5$
H_0: $\mu_D \neq 0{,}5$

Para realizar el análisis inferencial propuesta, el equipo a cargo del estudio ha creado en R el script 12.9, obteniendo los resultados que se presentan en la figura 12.14.

Vemos que fallamos en rechazar la hipótesis nula. Concluimos entonces, con 95 % de confianza, que no es posible descartar que, en promedio, la diferencia de las calificaciones entre la primera y la segunda evaluación de un curso inicial de programación en la Universidad en estudio sea de 5 décimas ($p = 0{,}689$; 95 % IC: $(-0{,}656; 1{,}439)$).

```
Media de las diferencia observada: 0.325

Distribución bootstrap e intervalo de confianza:

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

Valor p: 0.689
```

Figura 12.14: intervalo de confianza BCa y valor p para la media de las diferencias.

Script 12.9: bootstraping para inferir acerca de la media de las diferencias.

```r
library(bootES)

set.seed(432)

# Ingresar datos originales.
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)

# 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
alfa <- 0.05

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

# Desplazar la distribución bootstrap para reflejar la hipótesis nula.
valor_nulo <- 0.5
desplazamiento <- mean(distribucion_bES[["t"]]) - valor_nulo
distribucion_nula <- distribucion_bES[["t"]] - desplazamiento

# Determinar el valor p.
p <- (sum(abs(distribucion_nula) > abs(valor_observado)) + 1) / (B + 1)

# Mostrar los resultados
cat("Media de las diferencia observada:", round(valor_observado, 3), "\n\n")
cat("Distribución bootstrap e intervalo de confianza:\n")
print(distribucion_bES)
cat("Valor p:", round(p, 3), "\n")
```

12.2.5 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 y considerar únicamente aquellas permutaciones que ocurren con igual o menor probabilidad que la obtenida para los datos del estudio. Esta prueba pertenece al grupo conocido como **pruebas exactas 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 método puede emplearse para diversos estadísticos, tales como la proporción, la media y la varianza.

En términos generales, las pruebas exactas de permutaciones para la diferencia entre dos grupos A y B 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 formas de separar la muestra combinada en dos grupos de tamaños n_A y n_B.
4. Construir la distribución de las diferencias 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**, y a partir de ella calcular un **valor p aproximado** dado por la ecuación 12.11. Las pruebas que siguen este procedimiento se denominan **pruebas de permutaciones** o pruebas de permutaciones de Monte Carlo. Podemos ver que en la ecuación 12.11 se suma 1 tanto al numerador como al denominador, una corrección necesaria 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 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[3] de permutaciones de las muestras originales, usando muestreo sin reposición sobre la muestra combinada, y obtener el estadístico θ para cada permutación.
3. Generar la distribución del estadístico θ (bajo el supuesto que la hipótesis nula es cierta).
4. Determinar la probabilidad de encontrar en la distribución generada un valor de θ al menos tan extremo como el observado en las muestras originales.

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 separación de la muestra combinada sería igualmente probable. Así, lo que se hace en cada repetición es **reordenar** las observaciones y asignarlas ordenadamente a uno de los grupos, respetando los tamaños n_A y n_B de las muestras originales.

12.2.6 Prueba de permutaciones para dos muestras independientes

Consideremos el siguiente ejemplo: 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 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 una prueba de permutaciones con $P = 5.999$ 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

[3] Tradicionalmente se usa un número terminado en 9, pues solían simplificar los cómputos cuando se hacían manualmente, sin software

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 12.15), y el valor p obtenido para el contraste de hipótesis es $p = 0{,}969$, por lo que concluye con 95 % de confianza que no hay evidencia suficiente para creer que existe diferencia entre los promedios de las calificaciones finales de ambos grupos de estudiantes.

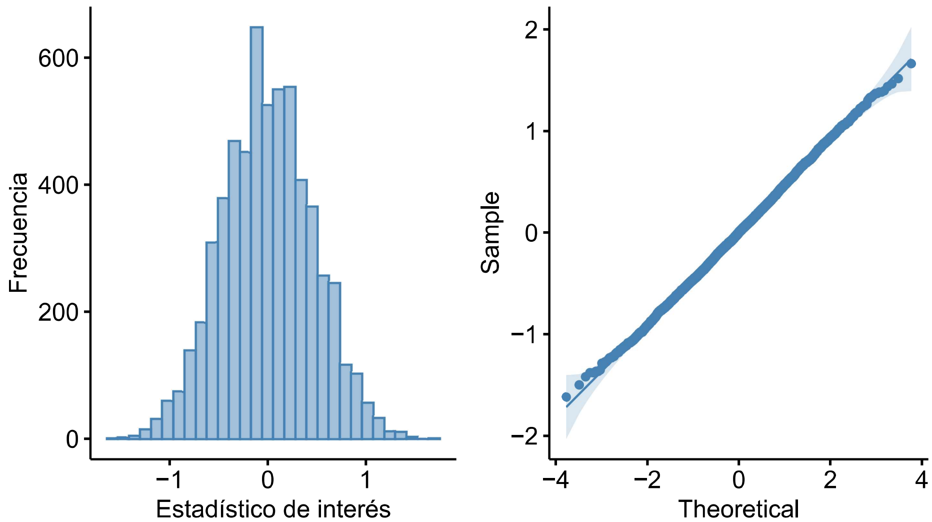


Figura 12.15: histograma y gráfico Q-Q de la distribución para la diferencia de medias generada mediante permutaciones.

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. Luego, 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 **??**. A pesar de que existen varios paquetes de R para realizar pruebas de permutaciones, él ha decidido realizar una implementación propia del procedimiento creando la función `contrastar_hipotesis_permutaciones()`, 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 entre dos remuestras y que la función que calcula dicho estadístico (para una muestra de datos) se entrega como argumento.

Script 12.10: pruebas de permutaciones para variables numéricas.

```
library(ggpubr)

```

```
# 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)

# Establecer semilla y cantidad de repeticiones.
R = 5999
set.seed(432)

# Función para obtener una permutación.
# Argumentos:
# - i: iterador (para llamadas posteriores).
# - muestra_1, muestra_2: muestras.
# Valor:
# - lista con las muestras resultantes tras la permutación.
obtiene_permutacion <- function(i, muestra_1, muestra_2) {
  n_1 <- length(muestra_1)
  combinada <- c(muestra_1, muestra_2)
  n <- length(combinada)
  permutacion <- sample(combinada, n, replace = FALSE)
  nueva_1 <- permutacion[1:n_1]
  nueva_2 <- permutacion[(n_1+1):n]
  return(list(nueva_1, nueva_2))
}

# Función para calcular la diferencia de un estadístico de interés entre las
# dos muestras.
# Argumentos:
# - muestras: lista con las muestras.
# - FUN: nombre de la función que calcula el estadístico de interés.
# Valor:
# - diferencia de un estadístico para dos muestras.
calcular_diferencia <- function(muestras, FUN) {
  muestra_1 <- muestras[[1]]
  muestra_2 <- muestras[[2]]
  diferencia <- FUN(muestra_1) - FUN(muestra_2)
  return(diferencia)
}

# 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", bins = 30, ...)

  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.
# - 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(list(muestra_1, muestra_2), FUN)
  cat("Valor observado:", observado, "\n")

  n_1 <- length(muestra_1)

  # Generar permutaciones.
  permutaciones <- lapply(1:repeticiones, obtiene_permutacion, muestra_1,
                          muestra_2)

  # Generar la distribución.
  distribucion <- sapply(permutaciones, calcular_diferencia, FUN)

  # Graficar la distribución.
  if(plot) {
    graficar_distribucion(distribucion, ...)
  }

```

```
  # Calcular el valor p.
  valor_p <- calcular_valor_p(distribucion, observado, repeticiones,
                              alternative)

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



# Hacer pruebas de permutaciones para la media y la varianza.
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)
```

12.2.7 Prueba de permutaciones para comparar más de dos muestras correlacionadas

Supongamos ahora que una 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 **??**.

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

Las hipótesis contrastadas son:

Sean Q_i, B_i y M_i los tiempos requeridos por los algoritmos de ordenamiento Quicksort, Bubblesort y Mergesort, respectivamente, para ordenar un arreglo i. Denotamos $(Y-X)$ al conjunto $\{Y_i-X_i\}$ de las diferencias en los tiempos de ejecución requeridos por los algoritmos X e Y. Entonces:

H_0: en promedio, no hay diferencias en el tiempo de ejecución necesitado por cada algoritmo de ordenamiento para ordenar las mismas instancias. Matemáticamente:: $\mu_{(B-Q)} = \mu_{(M-Q)} = \mu_{(M-B)} = 0$.

H_A: la media de las diferencias en el tiempo de ejecución necesitado para ordenar las mismas instancias es diferente para al menos un par de algoritmos. Matemáticamente: $\exists X, Y \in \{Q, B, M\}, \mid \mu_{(Y-X)} \neq 0$.

Tras comprobar mediante la figura 12.16 que no se cumple la condición de normalidad, la estudiante ha decidido usar permutaciones para resolver su problema. Para ello, ha considerado un nivel de significación $\alpha = 0{,}01$ y un total de 2.999 repeticiones, obteniendo como resultado un valor $p < 0,001$, 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.

A fin de determinar qué algoritmos difieren en su tiempo promedio de ejecución, ha decidido llevar a cabo un procedimiento post-hoc, calculando y ajustando 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 12.17. 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.

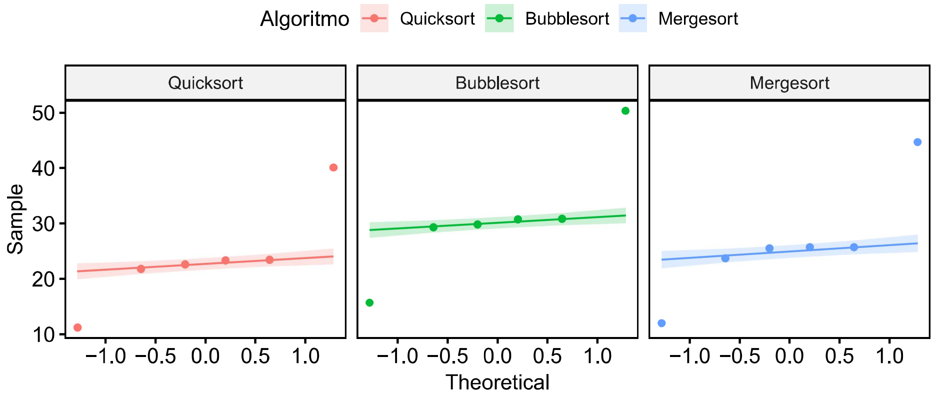


Figura 12.16: gráfico Q-Q para comprobar el supuesto de normalidad para el ejemplo.

```
ANOVA de una vía para muestras pareadas con permutaciones:
Valor p ómnibus: 0.0003333333


Análisis post-hoc (permutaciones) para la diferencia de las medias
-------------------------------------------------------------------
Valores p ajustados:
Quicksort - Bubblesort: 0.001
 Quicksort - Mergesort: 0.266
Bubblesort - Mergesort: 0.032

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

Figura 12.17: resultado del procedimiento post-hoc.

El script 12.11 corresponde a la solución desarrollada por la estudiante.

Script 12.11: prueba de permutaciones para muestras correlacionadas.

```r
library(ez)
library(ggpubr)
library(tidyr)

# Crear la matriz de datos.
Algoritmos <- c("Quicksort", "Bubblesort", "Mergesort")
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(all_of(Algoritmos),
                             names_to = "Algoritmo",
                             values_to = "Tiempo")
datos_largos[["Algoritmo"]] <- factor(datos_largos[["Algoritmo"]],
                                      levels = Algoritmos)

```

```r
# Verificar la 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)
valor_observado <- anova[["ANOVA"]][["F"]]

# Función para obtener una permutación.
# Devuelve una matriz de datos con formato ancho.
obtiene_permutacion <- function(i, df_ancho) {
  df_ancho[, 2:4] <- t(apply(df_ancho[, 2:4], 1, sample))
  return(df_ancho)
}

# Obtiene permutaciones
R = 2999
set.seed(432)
permutaciones <- lapply(1:R, obtiene_permutacion, datos_anchos)

# Función para obtener el estadístico F para una matriz de datos con
# formato ancho.
obtiene_F <- function(df_ancho) {
  df_largo <- df_ancho |>
              pivot_longer(c("Quicksort", "Bubblesort", "Mergesort"),
                           names_to = "Algoritmo",
                           values_to = "Tiempo")
  df_largo[["Algoritmo"]] <- factor(df_largo[["Algoritmo"]])

  anova <- ezANOVA(df_largo, dv = Tiempo, within = Algoritmo,
                   wid = Instancia)
  return(anova[["ANOVA"]][["F"]])
}

# Genera distribución de estadísticos F con las permutaciones.
distribucion <- sapply(permutaciones, obtiene_F)

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

# Análisis post-hoc.

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

# Obtiene las las medias de las diferencias observadas
dif_obs_Q_B <- obtiene_media_difs(datos_anchos, "Quicksort", "Bubblesort")
dif_obs_Q_M <- obtiene_media_difs(datos_anchos, "Quicksort", "Mergesort")
```

```r
dif_obs_B_M <- obtiene_media_difs(datos_anchos, "Bubblesort", "Mergesort")

# Obtiene las distribuciones de las medias de las diferencias permutadas
dist_medias_difs_Q_B <- sapply(permutaciones, obtiene_media_difs,
                               "Quicksort", "Bubblesort")
dist_medias_difs_Q_M <- sapply(permutaciones, obtiene_media_difs,
                               "Quicksort", "Mergesort")
dist_medias_difs_B_M <- sapply(permutaciones, obtiene_media_difs,
                               "Bubblesort", "Mergesort")

# Obtener valores p.
num <- sum(abs(dist_medias_difs_Q_B) > abs(dif_obs_Q_B)) + 1
den <- R + 1
p_Q_B <- num / den

num <- sum(abs(dist_medias_difs_Q_M) > abs(dif_obs_Q_M)) + 1
den <- R + 1
p_Q_M <- num / den

num <- sum(abs(dist_medias_difs_B_M) > abs(dif_obs_B_M)) + 1
den <- R + 1
p_B_M <- num / den

valores_p <- c(p_Q_B, p_Q_M, p_B_M)

# Ajustar y mostrar valores p
valores_p_adj <- p.adjust(valores_p, method = "BH")

cat("\n\n")
cat("Análisis post-hoc (permutaciones) para la diferencia de las medias\n")
cat("------------------------------------------------------------------\n")
cat("Valores p ajustados:\n")
cat(sprintf("Quicksort - Bubblesort: %.3f\n", valores_p_adj[1]))
cat(sprintf(" Quicksort - Mergesort: %.3f\n", valores_p_adj[2]))
cat(sprintf("Bubblesort - Mergesort: %.3f\n", valores_p_adj[3]))

cat("\nDiferencias observadas:\n")
cat(sprintf("Quicksort - Bubblesort: %6.3f\n", dif_obs_Q_B))
cat(sprintf(" Quicksort - Mergesort: %6.3f\n", dif_obs_Q_M))
cat(sprintf("Bubblesort - Mergesort: %6.3f\n", dif_obs_B_M))
```

12.2.8 Ejercicios propuestos para la sección 12.2

12.8 El conjunto de datos `diet` del paquete `WRS2` contiene datos de la pérdida de peso conseguida por tres tipos de dietas. Usando bootstrapping, determina si la pérdida de peso conseguida por las mujeres con las dietas A y C es la misma.

12.9 Considera el conjunto de datos `essays` descrito en la pregunta 12.2. Determina si una de las formas de retroalimentación estudiadas (directa o indirecta) es mejor que la otra (considera el ensayo 3 realizado al finalizar la intervención para este análisis) utilizando permutaciones.

12.10 Considera el conjunto de datos `essays` descrito en la pregunta 12.2. Determina, a través de remuestreo con bootstrapping, si las y los estudiantes del grupo de control pudieron mejorar la tasa de errores cometidos en el tercer ensayo respecto del segundo.

12.11 Considera el conjunto de datos `essays` descrito en la pregunta 12.2. Determina, usando remuestreo con permutaciones, si las y los estudiantes que recibieron retroalimentación directa mantuvieron la misma tasa de errores en el tercer y cuarto ensayo.

12.12 Considera el conjunto de datos `bush` descrito en la pregunta 12.5 Determina si el tiempo que se tarda una persona en tener arcadas al comer estas cosas es el mismo usando bootstrapping (considera que el procedimiento óminus y el post-hoc deben utilizar las mismas remuestras).

12.13 Considera el conjunto de datos `essays` descrito en la pregunta 12.2. Determina, utilizando remuestreo con permutaciones, si la tasa de errores cometidos en el tercer ensayo son las mismas para cada uno de los tres grupos de estudiantes.

12.14 Considera el conjunto de datos `essays` descrito en la pregunta 12.2. Determina con la técnica de bootstrapping si las tasas de errores mejoran al aplicar la retroalimentación indirecta y, si existe, esta se mantiene un mes después de la intervención.

12.15 Considera el conjunto de datos `essays` descrito en la pregunta 12.2. Determina usando bootstrapping un intervalo con 95 % de confianza para la probabilidad que un/a estudiante del grupo de control obtenga una tasa de errores menor a 1,5.

12.16 Considera el conjunto de datos `essays` descrito en la pregunta 12.2. Determina usando permutaciones si ambas formas de retroalimentación llevan a la misma probabilidad de obtener una tasa de errores menor a 1 en el cuarto ensayo.

12.3 BIBLIOGRAFÍA DEL CAPÍTULO

Amat Rodrigo, J. (2016). *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

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

Maechler, M. (2014). *CRAN task view: Robust statistical methods.* https://cran.r-project.org/web/views/Robust.html

Mair, P., & Wilcox, R. (2020). Robust statistical methods in R using the WRS2 package. *Behavior Research Methods*, *52*(2), 464-488.

Wilcox, R. R. (2012). *Introduction to robust estimation and hypothesis testing* (3.ª ed.). Academic Press.