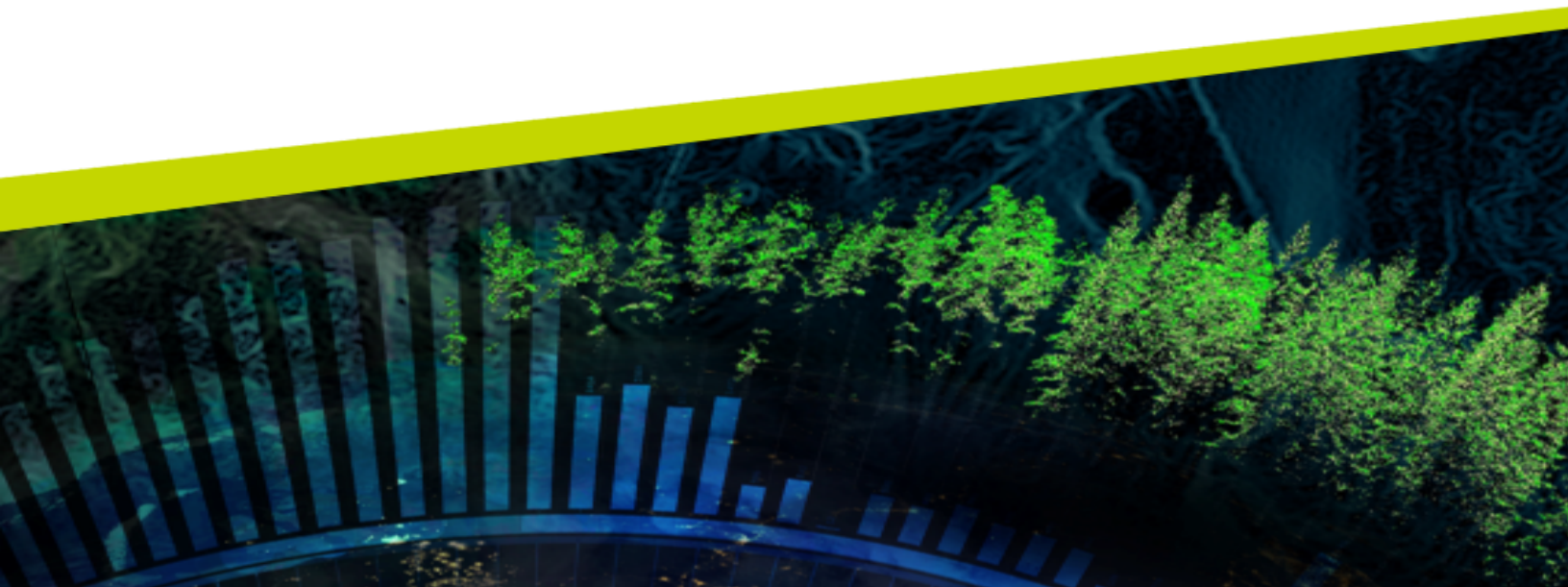
INFERENCIA Y MODELOS ESTADÍSTICOS

Jacqueline Köhler C. y José Luis Jara V.



CAPÍTULO 12. ALTERNATIVAS MÁS RECIENTES PARA ANALIZAR DATOS PROBLEMÁTICOS

Como ya sabemos, muchos procedimientos estadísticos requieren que los datos cumplan con ciertas propiedades o condiciones, lo que no siempre ocurre. En el capítulos anteriores hemos visto que, ante este escenario, podemos intentar analizar datos transformados (sección 11.1) o usar como alternativa algún métodos no paramétrico (capítulo 8 y secciones 11.2 y 11.3). Pero estas no son las únicas opciones como veremos en este capítulo, donde abordaremos otras estrategias, que podemos usar cuando necesitamos analizar datos problemáticos, que son más recientes y que requieren cierto poder de cómputo.

12.1 MÉTODOS ROBUSTOS

Pensemos como ejemplo en la prueba t de Student (capitulo 5), que se usa para inferir acerca de la media de una población. Sin embargo, como mencionamos en el capítulo 2, esta medida de tendencia central tiene el problema de ser sensible a la presencia de valores atípicos, a distribuciones asimétricas o a muestras muy pequeñas. En términos generales, el incumplimiento de las condiciones del supuesto de normalidad puede causar diversos problemas:

- Resultados sesgados.
- Intervalos de confianza sub o sobreestimados.
- Reducción del poder estadístico de la prueba.

En el capítulo 2 mencionamos la existencia de estimadores robustos, poco sensibles a asimetrías muestrales o valores atípicos. No obstante, el paradigma estadístico tradicional no suele considerarlos. Esta sección, basada en las ideas expuestas por Mair y Wilcox (2020), aborda pruebas alternativas para muchas de las pruebas estudiadas hasta ahora, disponibles en el paquete `WRS2` de R, basadas en **estimadores robustos**.

12.1.1 Alternativas robustas a la media

En el capítulo 2 conocimos distintas medidas de tendencia central. Entre ellas vimos que la **mediana**, correspondiente al valor central (o el promedio de los dos valores centrales) de la muestra ordenada, es una alternativa robusta a la media. No obstante, existen otras opciones que nos pueden ser útiles.

La **media truncada** es bastante similar a la media aritmética que ya conocemos, con la diferencia de que se calcula **descartando** un determinado porcentaje (γ) de los valores en ambos extremos del conjunto de datos. Tomemos como ejemplo una muestra X con 10 elementos, los cuales han sido ordenados por simplicidad:

$$X = \{5, 20, 37, 38, 40, 43, 43, 45, 87, 91\}$$

Si calculamos la media para la muestra anterior, tenemos que es $\overline{x} = 44{,}9$. No obstante, si observamos la muestra del ejemplo con detención, podemos darnos cuenta de que los valores extremos parecen ser atípicos y, en consecuencia, pueden tener una gran influencia en el valor resultante para la media. Así, podría ser más adecuado calcular la media truncada con $\gamma = 0{,}2$, es decir, podando el 20 % de los valores más pequeños y el 20 % de los valores más grandes, con lo que obtendremos:

$$\overline{x}^t = \frac{37 + 38 + 40 + 43 + 43 + 45}{6} = 41$$

En R, podemos calcular la media truncada mediante la ya conocida función `mean()` del paquete `base`[1], agregando el argumento adicional `trim` con la proporción γ de los datos extremos a descartar, esto es `mean(x, trim = 0.2)` para el ejemplo. Notemos que si $\gamma = 0{,}5$ (`trim = 0.5`), se obtiene la mediana del conjunto de datos.

Un problema de la media truncada es que, al usarla, podríamos estar descartando muchos datos, lo que puede causar problemas cuando la muestra es pequeña. Otra opción puede ser, en lugar de descartar los valores extremos en cada cola, reemplazarlos por los valores extremos que no serían descartados al usar la media truncada y luego calcular la media con la muestra modificada. A esta medida se le conoce como **media Winsorizada**. Si retomamos nuestro ejemplo para la media truncada, los valores extremos tras la operación de truncado son 37 y 45. Así, reemplazamos los valores truncados en la muestra original por estos nuevos extremos, con lo que nuestra muestra Winsorizada es:

$$X^w = \{37, 37, 37, 38, 40, 43, 43, 45, 45, 45\},$$

y la media Winsorizada entonces sería:

$$\overline{x}^w = \frac{37 + 37 + 37 + 38 + 40 + +43 + 45 + 45 + 45}{10} = 41$$

En R, podemos hacer este cálculo mediante la función `winmean(x, tr)` del paquete `WRS2`, donde:

- `x`: vector con los datos originales.
- `tr`: proporción de los datos a Winsorizar en cada extremo.

Así, la llamada para el ejemplo sería `winmean(x, tr = 0.2)`.

12.1.2 Prueba de Yuen para dos muestras independientes

La **prueba de Yuen** es una buena alternativa a la prueba t de Student para muestras independientes que trabaja con las medias truncadas y Winsorizadas en vez de las medias aritméticas originales. De este modo, son una buena alternativa para comparar dos medias independientes cuando los datos no cumplen la condición de normalidad, presentan datos atípicos, las varianzas son muy diferentes o los tamaños de las muestras son muy dispares.

La prueba de Yuen para dos muestras independientes puede aplicarse si se cumplen las siguientes condiciones:

- Las observaciones en una muestra son independientes, esto significa que la elección de una observación no influye en la selección de otra para esa muestra.
- Las muestras son independientes, es decir que las observaciones de una muestra no están relacionadas con ninguna de las observaciones de la otra.
- La(s) variable(s) estudiada(s) tiene(n) al menos escala de intervalos iguales.

Sin embargo, hay otras condiciones que si bien no son obligatorias para aplicar el procedimiento, sí influyen en la calidad de las interpretaciones que podemos obtener:

- Las poblaciones de origen no son extremadamente diferentes (por ejemplo, una es muy sesgada a la derecha y la otra a la izquierda), por lo que comparar sus medias recortadas tiene sentido.
- El impacto de los valores extremos no es de interés de la investigación, pues la prueba de Yuen esencialmente ignora la información que estos valores entregan.
- El nivel de poda no está cerca del nivel de la mediana, siendo $\gamma \approx 0{,}2$ un valor frecuente.
- Las muestras no son demasiado reducidas, cuando la poda puede tener efectos perjudiciales. Como es usual, no existe un número fijo, pero algunos autores mencionan 5, 6 o 10 observaciones por cada muestra luego de la poda.

La prueba entonces se basa en la estimación de la **diferencia de las medias truncadas**: $d^t = \overline{x}_1^t - \overline{x}_2^t$, donde $\overline{x}_1^t$ y $\overline{x}_2^t$ son las medias truncadas de cada una de las muestras. El error estándar para este estadístico

[1] Que se carga automáticamente al iniciar una sesión en R.

está dado por la ecuación 12.1:

$$SE_{d^t} = \sqrt{s^2_{X_1^t} + s^2_{X_2^t}} = \sqrt{\frac{(n_1 - 1)\, s^2_{X_1^w}}{n_1^t\,(n_1^t - 1)} + \frac{(n_2 - 1)\, s^2_{X_2^w}}{n_2^t\,(n_2^t - 1)}} \tag{12.1}$$

siendo n_i el tamaño de la muestra i original, n_i^t el tamaño de la muestra i truncada, y $s_{X_i^w}$ la desviación estándar de la muestra i Winsorizada.

Así, el estadístico de prueba está dado por la ecuación 12.2:

$$T_y = \frac{d^t}{SE_{d^t}} = \frac{\overline{x}_1^t - \overline{x}_2^t}{\sqrt{s^2_{X_1^t} + s^2_{X_2^t}}} \tag{12.2}$$

El estadístico T_y sigue una distribución t cuyos grados de libertad se calculan mediante la ecuación 12.3:

$$\nu_y = \frac{(s^2_{X_1^t} + s^2_{X_2^t})^2}{\frac{s^4_{X_1^t}}{n_1^t - 1} + \frac{s^4_{X_2^t}}{n_2^t - 1}} \tag{12.3}$$

Luego, es posible construir un intervalo de $100 \cdot (1 - \alpha)\,\%$ confianza como muestra la ecuación 12.4:

$$(\overline{x}_1^t - \overline{x}_2^t) \pm t^* \sqrt{s^2_{X_1^t} + s^2_{X_2^t}}\,, \tag{12.4}$$

donde t^* corresponde al cuantil crítico $1 - \alpha/2$ de la distribución t con ν_y grados de libertad.

Como sugieren estas ecuaciones, las hipótesis contrastadas por la prueba de Yuen para dos muestras independientes son que las medias truncadas de las poblaciones de origen son iguales:

H_0: $\mu_1^t = \mu_2^t$
H_A: $\mu_1^t \neq \mu_2^t$

En R, podemos aplicar la prueba de Yuen para muestras independientes mediante una llamada a la función `yuen(formula, data, tr)` del paquete `WRS2`, donde:

- `formula`: tiene la forma `<variable_dependiente> ~ <variable_independiente>`. Note que la variable independiente debe tener dos niveles, a fin de determinar a qué muestra pertenece cada observación de la variable dependiente.
- `data`: matriz de datos.
- `tr`: parámetro γ de la poda.

Veamos un ejemplo. El script 12.1 muestra el código que define dos muestras independientes del tiempo promedio de ejecución (en milisegundos) de dos algoritmos de complejidad computacional similar para resolver aproximadamente problemas de la un conjunto de 70 instancias del problema de ruteamiento de vehículos eléctricos con estaciones de carga intermedia, todas de igual tamaño y complejidad, que fueron construidas aleatoriamente. Estas instancias fueron asignadas al azar a cada uno de los algoritmos, resultando que $n_A = 40$ de ellas fueran resueltas por el algoritmo A y las restantes $n_B = 30$ por el algoritmo B.

Las líneas 19–22 del script 12.1 construyen gráficos Q-Q para comprobar el supuesto de normalidad requerido por la prueba t de Student para dos muestras independientes, obteniéndose como resultado la figura 12.1, donde podemos observar que las muestras obtenidas presentan desviaciones importantes de una distribución normal, especialmente para el caso del algoritmo A.

Por esta razón, **no podemos** utilizar una prueba t de Student para inferir sobre la diferencia del rendimiento medio de los algoritmos A y B para determinar si tienen igual desempeño o alguno de ellos es más eficiente.

Como alternativa para analizar estos datos usaremos la prueba de Yuen. Por la descripción del experimento, podemos confiar en que las condiciones de independencia al interior y entre las muestras se cumple, al igual que una escala adecuada. Por otro lado, como ambos algoritmos tienen complejidades computacionales parecidas, podríamos esperar que las distribuciones de sus tiempos de ejecución no sean muy diferentes. Además, las muestras son lo suficientemente grandes como para poder aplicar una poda razonable. Si usamos $\gamma = 0.2$, las muestras truncadas tendrían, respectivamente, $n_1^t = 24$ y $n_2^t = 24$ observaciones.

Para ilustrar los conceptos asociados a la prueba de Yuen, el script 12.2 trunca ambas muestras considerando $\gamma = 0{,}2$ y construyen gráficos Q-Q para las muestras podadas, los que pueden verse en la figura 12.2. Podemos apreciar que, tras la poda, los datos siguen distribuciones que se aproxima más a la normal.

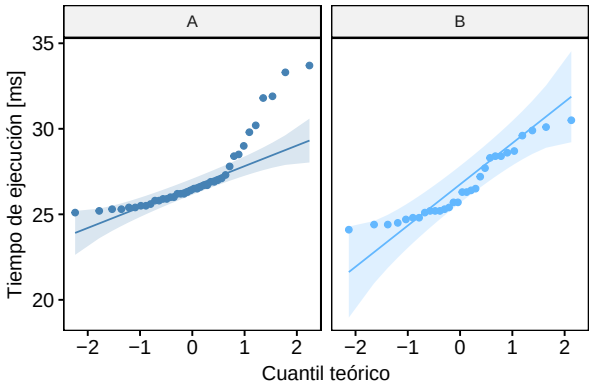


Figura 12.1: gráfico Q-Q de las muestras originales.

Script 12.1: datos del tiempo de ejecución registrado por dos algoritmos en diferentes instancias de igual tamaño y complejidad.

```r
library(ggpubr)
library(WRS2)

# Construir la matriz de datos
a <- c(25.1, 25.2, 25.3, 25.3, 25.4, 25.4, 25.5, 25.5, 25.6, 25.8, 25.8,
       25.9, 25.9, 26.0, 26.0, 26.2, 26.2, 26.2, 26.3, 26.4, 26.5, 26.5,
       26.6, 26.7, 26.7, 26.9, 26.9, 27.0, 27.1, 27.3, 27.8, 28.4, 28.5,
       29.0, 29.8, 30.2, 31.8, 31.9, 33.3, 33.7)

b <- c(24.1, 24.4, 24.4, 24.5, 24.7, 24.8, 24.8, 25.1, 25.2, 25.2, 25.2,
       25.3, 25.4, 25.7, 25.7, 26.3, 26.3, 26.4, 26.5, 27.2, 27.7, 28.3,
       28.4, 28.4, 28.6, 28.7, 29.6, 29.9, 30.1, 30.5)

Tiempo <- c(a, b)
Algoritmo <- c(rep("A", length(a)), rep("B", length(b)))
datos <- data.frame(Tiempo, Algoritmo)

# Comprobar normalidad
qq <- ggqqplot(datos, x = "Tiempo", facet.by = "Algoritmo",
               palette = c("steelblue", "steelblue1"), color = "Algoritmo",
               xlab = "Cuantil teórico", ylab = "Tiempo de ejecución [ms]")
qq <- qq + theme(legend.position = "none")
print(qq)
```

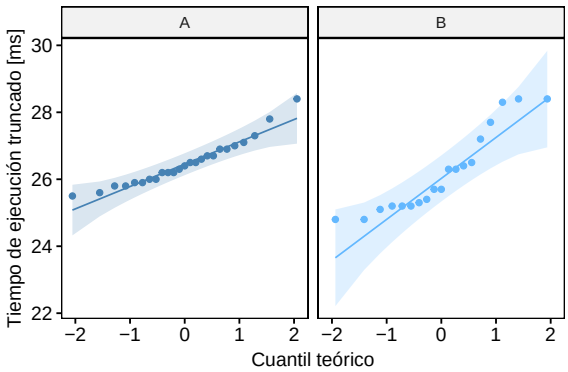


Figura 12.2: gráfico Q-Q de las muestras truncadas.

Script 12.2: (continuación del script 12.1) poda de los datos del ejemplo.

```
25 # Aplicar una poda del 20% a las muestras
26 gamma <- 0.2
27 n_a <- length(a)
28 n_b <- length(b)
29 poda_a <- floor(n_a * gamma)
30 poda_b <- floor(n_b * gamma)
31
32 a_trunc <- a[poda_a:(n_a - poda_a)]
33 b_trunc <- b[poda_b:(n_b - poda_b)]
34
35 Tiempo_t <- c(a_trunc, b_trunc)
36 Algoritmo_t <- c(rep("A", length(a_trunc)), rep("B", length(b_trunc)))
37 datos_t <- data.frame(Tiempo_t, Algoritmo_t)
38
39 qq_t <- ggqqplot(datos_t, x = "Tiempo_t", facet.by = "Algoritmo_t",
40                  palette = c("steelblue", "steelblue1"), color = "Algoritmo_t",
41                  xlab = "Cuantil teórico",
42                  ylab = "Tiempo de ejecución truncado [ms]")
43 qq_t <- qq_t + theme(legend.position = "none")
44 print(qq_t)
```

Por supuesto, la aplicación del procedimiento es más simple utilizando la función `yuen()`, como muestra la línea 48 del script 12.3, obteniéndose como resultado (figura 12.3) una diferencia entre las medias truncadas de 0,246, con intervalo de 95 % de confianza ($-0{,}859;\ 1{,}351$) y tamaño del efecto de 0,090. La prueba no resulta significativa ($T_y(29{,}05) = 0{,}455;\ p = 0{,}653$) al nivel de significación $\alpha = 0{,}05$, por lo que concluimos con 95 % de confianza que no es posible descartar que ambos algoritmos tienen, en promedio, igual tiempo de ejecución.

```
Prueba de Yuen para dos muestras independientes
-----------------------------------------------
Call:
yuen(formula = Tiempo ~ Algoritmo, data = datos, tr = gamma)

Test statistic: 0.455 (df = 29.05), p-value = 0.65252

Trimmed mean difference:  0.24583
95 percent confidence interval:
-0.8592     1.3509

Explanatory measure of effect size: 0.09
```

Figura 12.3: resultado de la prueba de Yuen para el ejemplo.

El paquete `WRS2` incluye también la función `pb2gen(formula, data, est, nboot)`, que usa bootstrapping para aplicar la prueba de Yuen usando otras medidas robustas de tendencia central, donde:

- `formula`: tiene la misma forma descrita para la prueba de Yuen.
- `data`: matriz de datos.
- `est`: medida a emplear. Puede tomar las opciones `"mean"` para la media y `"median"` para la mediana, entre otras opciones que escapan a los alcances de este curso.
- `nboot`: cantidad de repeticiones bootstrap.

Como estudiaremos más adelante, en cada iteración bootstrap se estima la diferencia de las medias truncadas de dos **remuestras** de las muestras originales. Esto permite obtener una especie de "promedio" del estadístico, que es más confiable que la única estimación que se obtiene de las muestras originales.

El script 12.3 muestra cómo usar la función `pb2gen()` en el ejemplo, usando como estimadores la media (línea 58) y la mediana (línea 62), obteniéndose los resultados de la figura 12.4. Podemos ver que, en ambos casos, las pruebas tampoco resultan significativas ($T_y \geq 0{,}61;\ p - value \geq 0{,}213$, con 999 iteraciones bootstrap).

Script 12.3: (continuación del script 12.2) prueba de Yuen para dos muestras independientes asintótica y usando bootstrapping.

```r
# Aplicar y mostrar la prueba de Yuen asintótica
prueba <- yuen(Tiempo ~ Algoritmo, data = datos, tr = gamma)
cat("\nPrueba de Yuen para dos muestras independientes\n")
cat("-----------------------------------------------\n")
print(prueba)

# Establecer cantidad de repeticiones con bootstrapping
B <- 999

# Aplicar la prueba de Yuen con bootstrapping y la media
set.seed(135)
prueba_media <- pb2gen(Tiempo ~ Algoritmo, data=datos, est="mean", nboot=B)

# Aplicar la prueba de Yuen con bootstrapping y la mediana
set.seed(135)
prueba_mediana <- pb2gen(Tiempo ~ Algoritmo, data=datos, est="median", nboot=B)

# Mostrar los resultados
cat("\nPrueba de Yuen - implemetación con bootstrapping\n")
cat("================================================\n")

cat("\nResultado al usar bootstrapping y la media como estimador\n")
cat("----------------------------------------------------------\n")
print(prueba_media)
cat("\nResultado al usar bootstrapping y la mediana como estimador\n")
cat("------------------------------------------------------------\n")
print(prueba_mediana)
```

```
Prueba de Yuen - implemetación con bootstrapping
================================================

Resultado al usar la media como estimador
-----------------------------------------
Call:
pb2gen(formula = Tiempo ~ Algoritmo, data = datos, est = "mean",
    nboot = bootstrap)

Test statistic: 0.61, p-value = 0.21321
95% confidence interval:
-0.3008    1.5617


Resultado al usar la mediana como estimador
-------------------------------------------
Call:
pb2gen(formula = Tiempo ~ Algoritmo, data = datos, est = "median",
    nboot = bootstrap)

Test statistic: 0.45, p-value = 0.47147
95% confidence interval:
-0.95    1.35
```

Figura 12.4: resultado de la prueba de Yuen con bootstrapping para el ejemplo, usando como estimadores la media y la mediana.

12.1.3 Prueba de Yuen para dos muestras pareadas

La prueba de Yuen, al igual que la prueba t de Student, fue adaptada para trabajar con dos muestras pareadas y es una alternativa cuando los datos no cumplen la condición de normalidad o existen valores atípicos. Las suposiciones de esta prueba son:

- Los pares de observaciones son independientes, es decir que la elección de un par no influye en la selección de otro.
- La variable medida tiene al menos escala de intervalos iguales.

Como cuando se trabaja con muestras independientes, hay otras condiciones que afectan la calidad de esta versión de la prueba:

- Las diferencias siguen una distribución relativamente simétrica.
- La poda que se aplica elimina valores atípicos extremos.
- El nivel de poda no es cercano al nivel de la mediana.
- Las muestras no son demasiado reducidas. Algunos autores mencionan que se deberían tener al menos 10 o 15 pares de observaciones.

Para el caso de dos muestras apareadas, como ha sido usual, debemos considerar el conjunto de las diferencias entre pares de observaciones:

$$D = \{d_i = x_{1i} - x_{2i}\},$$

siendo $(x_{1i},\, x_{2i})$ el i-ésimo par de observaciones en la muestra. Luego podemos aplicar la poda a este conjunto y calcular la media truncada de las diferencias: $\overline{D}^t$.

El error estándar de esta media truncada está dada por la ecuación 12.5:

$$SE_{\overline{D}^t} = \sqrt{\frac{(n-1)\, s^2_{D^w}}{n^t\,(n^t-1)}} \tag{12.5}$$

donde n es el número de pares en la muestra original D, n^t es la cantidad de pares disponibles luego de la poda, y s_{D^w} es la desviación estándar del conjunto de las diferencias Winsorizado.

Así, el estadístico de prueba para datos pareados sería como muestra la ecuación 12.6:

$$T_y = \frac{\overline{D}^t}{\sqrt{\frac{(n-1)\, s^2_{D^w}}{n^t\,(n^t-1)}}}\,, \tag{12.6}$$

que sigue una distribución t con $n^t - 1$ grados de libertad, lo que permite obtener valores p e intervalos de confianza.

Nuevamente estas ecuaciones sugieren que esta extensión de la prueba de Yuen contrasta hipótesis bilaterales, que en este caso corresponde a que la media truncada de las diferencias apareadas es cero:

H_0: $\mu^t_D = 0$
H_A: $\mu^t_D \neq 0$

En R, podemos aplicar la prueba de Yuen para muestras apareadas mediante una llamada a la función `yuend(x, y, tr)` del paquete `WRS2`, donde:

- `x`: vector numérico con la primera muestra.
- `y`: vector numérico con la segunda muestra.
- `tr`: parámetro γ de la poda.

Como es de esperar, la función falla si los largos de los vectores `x` e `y` no coinciden.

Consideremos un ejemplo. Supongamos ahora que queremos comparar el rendimiento de los algoritmos A y B para el problema de ruteamiento de vehículos eléctricos con estaciones de carga intermedia, descritos en la sección anterior, para lo cual esta vez hemos seleccionado aleatoriamente 25 instancias del problema y registrado el tiempo de ejecución en milisegundos que cada uno tarda en resolverlas.

El script 12.5 muestra los tiempos de ejecución observados para cada algoritmo (líneas 6–12). El resto del script construye gráficos Q-Q para el conjunto de las diferencias entre los pares de observaciones y para este

conjunto podado (obtenido en las líneas 17–21) aplicando un recorte $\gamma = 0.2$. Los gráficos resultantes se muestran en la figura 12.5.

En ella podemos ver que el conjunto de diferencias observadas (a la izquierda) exhibe desviaciones importantes de normalidad, por lo que no sería prudente asumir que esta proviene de una distribución normal. En cambio, el conjunto podado de estas observaciones (a la derecha) muestra un comportamiento más esperado para una muestra que proviene de una población normal. Así, es conveniente que usemos la prueba de Yuen para dos muestras pareadas en este caso, como alternativa a la prueba t pareada.

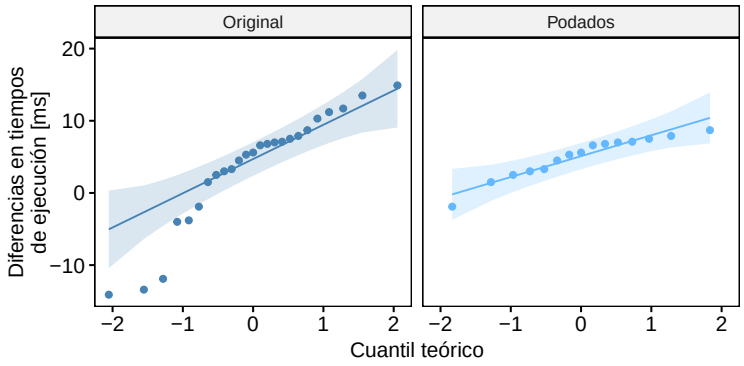


Figura 12.5: gráficos Q-Q de las diferencias originales y truncadas.

Script 12.4: datos del tiempo de ejecución registrado por los algoritmos en las mismas instancias.

```r
library(ggpubr)
library(WRS2)

# Construir las estructuras con los datos observados
a <- c(32.3, 32.0, 32.0, 36.0, 34.2, 32.7, 32.5, 32.0, 32.1, 33.4,
       32.3, 37.2, 32.1, 32.0, 33.9, 34.1, 36.6, 34.5, 32.7, 33.1,
       32.7, 32.1, 36.7, 32.2, 38.0)

b <- c(35.3, 20.1, 18.6, 46.3, 42.1, 39.3, 37.0, 28.0, 30.2, 40.4,
       35.6, 50.7, 33.6, 17.9, 41.0, 41.6, 47.8, 43.2, 38.3, 39.9,
       38.0, 28.3, 48.4, 34.7, 52.9)

dif <- a - b

# Aplicar una poda del 20% al conjunto de diferencias
gamma <- 0.2
n <- length(dif)
poda <- floor(n * gamma)
dif <- sort(dif)
dif_trunc <- dif[(poda + 1):(n - poda)]
n_t <- length(dif_trunc)

# Obtener gráficos Q-Q de las diferencias originales y podadas
datos <- data.frame(Diferencia = c(dif, dif_trunc),
                    Muestra = c(rep("Original", n),rep("Podados", n_t)))
qq <- ggqqplot(datos, x = "Diferencia", facet.by = "Muestra",
               palette = c("steelblue", "steelblue1"), color = "Muestra",
               xlab = "Cuantil teórico",
               ylab = "Diferencias en tiempos\nde ejecución [ms]")
qq <- qq + theme(legend.position = "none")
print(qq)
```

El enunciado del ejemplo sugiere que las condiciones de independencia entre los pares de observaciones se estaría cumpliendo. Por otro lado, los gráficos Q-Q de las diferencias en tiempos de ejecución (figura 12.5) muestran que existe bastante asimetría hacia la cola inferior en la muestra original, aunque se resuelve bien al considerar la muestra truncada, las que no tampoco incluyen valores atípicos extremos. Si bien el nivel de poda usado en este gráfico es razonable, el tamaño de la muestra es bastante reducida, con solo 15 pares

de observaciones. Por esta última razón y la asimetría que podría existir en la población de origen, seremos prudentes y usaremos un nivel de significación exigente: $\alpha = 0{,}01$.

Entonces, para el ejemplo vamos a contrastar las siguientes hipótesis:

H_0: Sin considerar casos extremos, en promedio, los algoritmos A y B tardan lo mismo en resolver las mismas instancias de prueba. Matemáticamente: si D es la distribución de las diferencias en el tiempo de ejecución que tardan los algoritmos en resolver las mismas instancias, entonces la media truncada de estas diferencias es nula: $\mu_D^t = 0$.

H_A: A pesar de no considerar los casos extremos, los algoritmos A y B, en promedio, no toman el mismo tiempo de ejecución para resolver las mismas instancias de prueba. Es decir, $\mu_D^t \neq 0$

El script 12.5 ilustra el uso de la función `yuend()` para el ejemplo, obteniéndose el resultado que muestra la figura 12.6.

```
Prueba de Yuen para dos muestras pareadas
----------------------------------------
Call:
yuend(x = a, y = b, tr = gamma)

Test statistic: -3.5843 (df = 14), p-value = 0.00299

Trimmed mean difference:  -5.02667
95 percent confidence interval:
-8.0346     -2.0188

Explanatory measure of effect size: 0.72
```

Figura 12.6: resultado de la prueba de Yuen para las muestras apareadas del ejemplo.

Podemos ver que la prueba resulta significativa ($T_y(14) = -3{,}584$; $p = 0{,}003$; $\gamma = 0{,}2$) para el nivel de significación establecido. Así, debemos concluir con 99 % confianza que existe una diferencia estadísticamente significativa en el desempeño de ambos algoritmos, siendo el algoritmo A el más eficiente (puesto que la diferencia estimada entre las medias tiene signo negativo).

Script 12.5: (continuación del script 12.4) prueba de Yuen para dos muestras pareadas.

```r
# Aplicar y mostrar la prueba de Yuen para muestras apareadas
gamma <- 0.2
prueba <- yuend(x = a, y = b, tr = gamma)
cat("Prueba de Yuen para dos muestras pareadas\n")
cat("----------------------------------------\n")
print(prueba)
```

12.1.4 Análisis robusto de una vía para muestras independientes

El paquete `WRS2` ofrece diferentes alternativas a ANOVA de una vía para muestras independientes que podemos usar cuando los tamaños muestrales son muy diferentes o no se cumple la condición de homocedasticidad. Todas ellas asumen que:

- Las observaciones en una muestra son independientes, es decir que las observación se eligen sin considerar ninguna otra.
- Las muestras son independientes, es decir que las observaciones de una muestra no tienen ninguna relación con alguna de las observaciones en las otras muestras.
- La(s) variable(s) estudiada(s) tiene(n) al menos escala de intervalos iguales.

Sin embargo, también debemos tener en cuenta condiciones que afectan la calidad de la prueba, que son análogas a las mencionadas para la prueba de Yuen.

La función `t1way(formula, data, tr, alpha)` efectúa un procedimiento similar a ANOVA usando medias truncadas. A su vez, la función `lincon(formula, data, tr, alpha)` permite realizar el procedimiento post-hoc correspondiente.

De manera similar, `t1waybt(formula, data, tr, nboot)` realiza un procedimiento análogo al anterior incorporando bootstrapping. En este caso, el procedimiento post-hoc puede realizarse mediante la función `mcppb20(formula, data, tr, nboot)`.

Una tercera opción es la función `med1way(formula, data, iter)`, que emplea la mediana y sigue un proceso iterativo. No obstante, en este caso el paquete no ofrece funciones que permitan realizar el procedimiento post-hoc.

Los argumentos asociados a las funciones mencionadas en los párrafos anteriores son:

- `formula`: de la forma `<variable_dependiente> ~ <variable_independiente>`.
- `data`: matriz de datos.
- `tr`: parámetro γ de la poda.
- `alpha`: nivel de significación.
- `nboot`: cantidad de repeticiones bootstrapping.
- `iter`: cantidad de iteraciones a realizar.

Para ejemplificar pensemos que ahora deseamos comparar el tiempo promedio de ejecución (en milisegundos) de tres algoritmos que resuelven instancias distintas del problema de ruteamiento de vehículos eléctricos con estaciones de carga intermedia, contando con $n_A = 40$ observaciones para el algoritmo A, $n_B = 30$ observaciones para el B y $n_C = 35$ observaciones para el C. El script 12.6 presenta estas observaciones (líneas 5–17). Considerando que esta comparación podría realizarse por medio de un procedimiento ANOVA para muestras independientes, el script también produce gráficos Q-Q para las muestras (líneas 25–29), los que son presentados en la figura 12.7, donde se observa que las observaciones recogidas presentan desviaciones importantes del comportamiento esperado para datos que provienen de distribuciones normales. De esta forma, por lo que no podríamos utilizar el procedimiento ANOVA, al menos de manera directa.

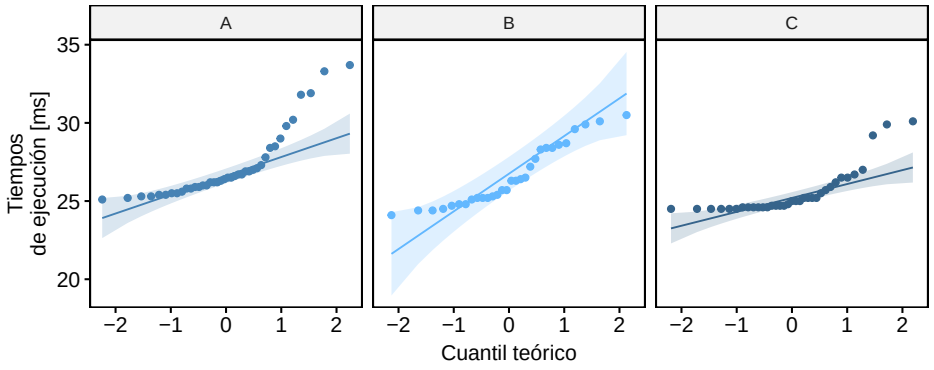


Figura 12.7: gráficos Q-Q de las observaciones del ejemplo.

Sin embargo, la figura 12.7 también sugiere que las desviaciones ocurren en los extremos de las muestras, por lo que razonablemente podríamos suponer que comparar medias truncadas podría ser más adecuado. Como la variable dependiente es una medición física (tiempo), ya sabemos que cumple con la escala de intervalos iguales. Además la descripción del ejemplo sugiere que las muestras también cumplen las condiciones de independencia en cada muestra y entre ellas.

Script 12.6: observaciones del tiempo de ejecución registrado por tres algoritmos en instancias diferentes.

```
1  library(ggpubr)
2  library(WRS2)
3
4  # Construir las estructuras con los datos
5  A <- c(25.1, 25.2, 25.3, 25.3, 25.4, 25.4, 25.5, 25.5, 25.6, 25.8, 25.8,
6         25.9, 25.9, 26.0, 26.0, 26.2, 26.2, 26.2, 26.3, 26.4, 26.5, 26.5,
7         26.6, 26.7, 26.7, 26.9, 26.9, 27.0, 27.1, 27.3, 27.8, 28.4, 28.5,
8         29.0, 29.8, 30.2, 31.8, 31.9, 33.3, 33.7)
9
10 B <- c(24.1, 24.4, 24.4, 24.5, 24.7, 24.8, 24.8, 25.1, 25.2, 25.2, 25.2,
11        25.3, 25.4, 25.7, 25.7, 26.3, 26.3, 26.4, 26.5, 27.2, 27.7, 28.3,
12        28.4, 28.4, 28.6, 28.7, 29.6, 29.9, 30.1, 30.5)
13
```

```
C <- c(24.5, 24.5, 24.5, 24.5, 24.5, 24.5, 24.6, 24.6, 24.6, 24.6, 24.6,
       24.6, 24.7, 24.7, 24.7, 24.7, 24.8, 25.0, 25.0, 25.0, 25.2, 25.2,
       25.2, 25.2, 25.5, 25.7, 25.9, 26.2, 26.5, 26.5, 26.7, 27.0, 29.2,
       29.9, 30.1)

Tiempo <- c(A, B, C)
Algoritmo <- c(rep("A", length(A)), rep("B", length(B)), rep("C", length(C)))
Algoritmo <- factor(Algoritmo)
datos <- data.frame(Tiempo, Algoritmo)

# Obtener gráficos Q-Q de las muestras
qq <- ggqqplot(datos, x = "Tiempo", facet.by = "Algoritmo", color = "Algoritmo",
               palette = c("steelblue", "steelblue1", "steelblue4"),
               xlab = "Cuantil teórico", ylab = "Tiempos\nde ejecución [ms]")
qq <- qq + theme(legend.position = "none")
print(qq)
```

De esta forma, podemos aplicar el análisis robusto de una vía para muestras independientes, con las siguientes hipótesis:

H_0 : el tiempo de ejecución promedio necesitado para resolver instancias de igual tamaño es la misma para los tres algoritmos. Matemáticamente: $\mu_A = \mu_B = \mu_C$.

H_A : el tiempo de ejecución promedio necesitado para resolver instancias de igual tamaño es diferente para al menos un algoritmo. Matemáticamente: $\exists i, j \in \{A, B, C\}, i \neq j \mid \mu_i \neq \mu_j$.

Supondremos un nivel de significación $\alpha = 0{,}05$ para este estudio.

El script 12.7 ilustra el funcionamiento de las funciones `t1way()` y `lincon()` del paquete `WRS2` que aplican, respectivamente, las pruebas robustas ómnibus y post-hoc para el análisis de las muestras independientes del ejemplo. Alguien prestando atención habrá notado que antes de la llamada a la prueba ómnibus hemos fijado una semilla para la reproducibilidad de números aleatorios y que, en la llamada, hemos indicado el argumento `nboot = B`. Debemos aclarar que esta prueba es asintótica, como la gran mayoría de las pruebas que hemos estudiado, pero los procedimientos para estimar el tamaño del efecto y el intervalo de confianza que reporta sí hacen uso de bootstrapping. Esta es la razón por la que hemos definido una semilla para números aleatorios fija y un número de iteraciones bootstrap. También debemos observar que no hemos indicado un método para ajustar los valores p de las comparaciones post-hoc, por lo que la función `lincon()` utiliza el procedimiento de Benjamini y Hochberg (por omisión).

Script 12.7: (continuación del script 12.6) análisis robusto de una vía para comparar muestras independientes.

```
# Fijar nivel de significación, nivel de poda y nro. de iteraciones bootstrap
alfa <- 0.05
gamma <- 0.2
nboot <- 999

# Comparar los diferentes algoritmos usando medias truncadas
set.seed(666)
una_via <- t1way(Tiempo ~ Algoritmo, data = datos,
                 tr = gamma, alpha = alfa, nboot = nboot)

cat("Análisis de una vía para muestras independientes (asimpótico)\n")
cat("-----------------------------------------------------------\n")
print(una_via)

if(una_via[["p.value"]] < alfa) {
  una_via_ph <- lincon(Tiempo ~ Algoritmo, data = datos,
                       tr = gamma, alpha = alfa)

  cat("Análisis post-hoc para muestras independientes (asimpótico)\n")
  cat("-----------------------------------------------------------\n")
  print(una_via_ph)
}
```

El resultado de ejecutar este trozo de código puede verse en la figura 12.8. Vemos que la prueba robusta ómnibus resulta significativa ($F(2;\ 34{,}39) = 10{,}981;\ p < 0{,}001$) al nivel establecido, por lo que debemos

rechazar la hipótesis nula. Concluimos, entonces, con 95 % confianza, que existe una diferencia estadísticamente significativa entre los tiempos promedio de ejecución de los algoritmos cuando no se consideran casos extremos. Al efectuar el procedimiento post-hoc, podemos concluir que, sin considerar los casos extremos, el algoritmo C presenta un tiempo de ejecución promedio más alto que los algoritmos A ($\overline{x}^t_A - \overline{x}^t_C = 1{,}496$ [ms]; 95 %CI: [0,731; 2,444] [ms]; $p < 0{,}001$) y B ($\overline{x}^t_B - \overline{x}^t_C = 1{,}250$ [ms]; 95 %CI: [0,163; 2,342] [ms]; $p = 0{,}008$).

```
Análisis de una vía para muestras independientes (asimpótico)
----------------------------------------------------------------
Call:
t1way(formula = Tiempo ~ Algoritmo, data = datos, tr = gamma,
    alpha = alfa, nboot = B)

Test statistic: F = 10.9813
Degrees of freedom 1: 2
Degrees of freedom 2: 34.39
p-value: 0.00021

Explanatory measure of effect size: 0.46
Bootstrap CI: [0.27; 0.67]

Análisis post-hoc para muestras independientes (asimpótico)
----------------------------------------------------------------
Call:
lincon(formula = Tiempo ~ Algoritmo, data = datos, tr = gamma,
    alpha = alfa)

         psihat ci.lower ci.upper p.value
A vs. B 0.24583 -1.11867  1.61033 0.65252
A vs. C 1.49583  0.65571  2.33596 0.00022
B vs. C 1.25000 -0.02757  2.52757 0.03909
```

Figura 12.8: resultado de la prueba de Yuen para las muestras apareadas del ejemplo.

Por otro lado, el script 12.8 muestra el uso de las funciones `t1waybt()` y `mcppb20()` que aplican bootstrapping para el análisis robusto ómnibus y post-hoc a las muestras independientes del ejemplo. Nuevamente debemos hacer una observación: no hemos definido un nivel de significación para el cálculo de intervalos de confianza. Esto se debe a que la función para aplicar el procedimiento ómnibus no estima tal intervalo, mientras que la función que implementa el procedimiento post-hoc ¡siempre utiliza $\alpha = 0{,}05$!. Además, debemos decir que la función `mcppb20()` ajusta los múltiples valores p con un método ad-hoc para controlar la tasa de error por familia.

Script 12.8: (continuación del script 12.7) análisis robusto de una vía para muestras independientes aplicando bootstrapping.

```
# Comparar los diferentes algoritmos usando medias truncadas y bootstrapping
set.seed(666)
una_via_bt <- t1waybt(Tiempo ~ Algoritmo, data = datos,
                      tr = gamma, nboot = nboot)

cat("Análisis de una vía para muestras independientes (bootstrapped)\n")
cat("----------------------------------------------------------------\n")
print(una_via_bt)

if(una_via_bt[["p.value"]] < alfa) {
  set.seed(666)
  una_via_bt_ph <- mcppb20(Tiempo ~ Algoritmo, data = datos,
                           tr = gamma, nboot = nboot)

  cat("Análisis post-hoc para muestras independientes (bootstrapped)\n")
  cat("----------------------------------------------------------------\n")
  print(una_via_bt_ph)
}
```

```
Análisis de una vía para muestras independientes (bootstrapped)
------------------------------------------------------------
Call:
t1waybt(formula = Tiempo ~ Algoritmo, data = datos, tr = gamma,
    nboot = B)

Effective number of bootstrap samples was 999.

Test statistic: 10.9813
p-value: 0.001
Variance explained: 0.179
Effect size: 0.424

Análisis post-hoc para muestras independientes (bootstrapped)
------------------------------------------------------------
Call:
mcppb20(formula = Tiempo ~ Algoritmo, data = datos, tr = gamma,
    nboot = B)

         psihat ci.lower ci.upper p-value
A vs. B 0.24583 -0.91667  1.61389 0.55656
A vs. C 1.49583  0.73690  2.44464 0.00000
B vs. C 1.25000  0.16270  2.34206 0.00801
```

Figura 12.9: resultado de la prueba de Yuen para las muestras apareadas del ejemplo.

El resultado que entrega el script 12.8 es mostrado en en la figura 12.9, donde vemos que la prueba robusta ómnibus resulta significativa, reportando el mismo estadístico que el procedimiento asintótico, con un valor p estimado vía bootstrapping algo mayor ($p = 0{,}001$), confirmando la existencia de diferencias estadísticamente significativas en los tiempos de ejecución requeridos por los algoritmos cuando no se consideran casos extremos.

El procedimiento post-hoc con bootstrapping también confirma que, con 95 % confianza, sin considerar los casos extremos, el algoritmo C presenta un tiempo de ejecución promedio más alto que los algoritmos A y B, aunque con intervalos de confianza y valores p, estimados con 999 repeticiones bootstrap aleatorias, son un tanto distintos.

12.1.5 Análisis robusto de una vía para muestras correlacionadas

Desde luego, el paquete `WRS2` también ofrece opciones robustas para reemplazar el procedimiento ANOVA de una vía para muestras correlacionadas, que podemos usar cuando los datos disponibles violan la condición de normalidad o de esfericidad. Estos procedimientos robustos asumen las siguientes condiciones:

- Los casos o bloques medidos son independientes entre sí.
- Se tiene un conjunto de mediciones (usualmente mayor a dos) para cada caso o bloque.
- Las variable medida tiene al menos escala de intervalos iguales.

También debemos tener en cuenta las condiciones que afectan la calidad de la prueba, que son análogas a las mencionadas para la prueba de Yuen con muestras apareadas.

La función `rmanova(y, groups, blocks, tr)` efectúa un procedimiento similar a ANOVA usando medias truncadas, mientras que la función `rmmcp(y, groups, blocks, tr, alpha)` implementa el procedimiento post-hoc para dicha prueba. Por otra parte, `rmanovab(y, groups, blocks, tr, nboot)` realiza la misma tarea que `rmanova()`, incorporando bootstrapping. En este caso, el procedimiento post-hoc está dado por la función `pairdepb(y, groups, blocks, tr, nboot)`. Los argumentos para esta familia de funciones son:

- `formula`: de la forma `<variable_dependiente> ~ <variable_independiente>`.
- `y`: vector con la variable dependiente.
- `groups`: vector que indica las medidas repetidas.
- `blocks`: vector que identifica los casos o bloques.

- `tr`: parámetro γ de la poda.
- `alpha`: nivel de significación.
- `nboot`: cantidad de repeticiones bootstrapping.

Consideremos, una vez más, la comparación del desempeño de los tres algoritmos implementados para el problema de ruteamiento de vehículos eléctricos con estaciones de carga intermedia (A, B y C), pero esta vez utilizando las mismas 25 instancias de igual tamaño y complejidad construidas aleatoriamente. El script 12.9 (líneas 5–15) muestra los tiempos de ejecución, en milisegundos, registrados para cada algoritmo al resolver cada presenta de estas instancias de prueba.

Considerando que esta comparación podría llevarse a cabo aplicando un procedimiento ANOVA para medidas repetidas, el script obtiene las diferencias pareadas entre cada par de algoritmos y construye gráficos Q-Q para estas diferencias (líneas 30–33), en los que se observa la presencia de valores atípicos y desviaciones del comportamiento esperado (figura 12.10) si las distribuciones de origen de estas diferencias fueran normales. Luego, parece más aconsejable utilizar las alternativas robustas para analizar estos datos como alternativa al procedimiento ANOVA para muestras correlacionadas.

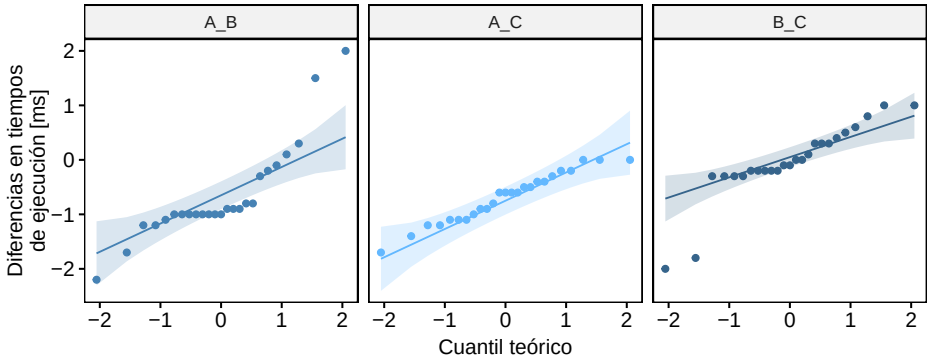


Figura 12.10: gráficos Q-Q de diferencias en los tiempos de ejecución requerido por los algoritmos del ejemplo.

Como hemos discutido en los ejemplos anteriores, los tiempos de ejecución cumplen con tener escala de intervalos iguales y el experimento asegura una selección aleatoria de los casos de prueba (instancias), que no afectan en el tiempo requerido por otras instancias, garantizando así su mutua independencia.

Script 12.9: diferencias en los tiempo de ejecución registrados por tres algoritmos en las mismas instancias.

```
library(dplyr)
library(ggpubr)
library(tidyr)
library(WRS2)

# Construir las estructuras con los datos
A <- c(32.0, 32.0, 32.0, 32.0, 32.1, 32.1, 32.1, 32.2, 32.3, 32.3, 32.5,
       32.7, 32.7, 32.7, 33.1, 33.4, 33.9, 34.1, 34.2, 34.5, 36.0, 36.6,
       36.7, 37.2, 38.0)

B <- c(33.0, 33.0, 33.0, 33.0, 33.0, 33.0, 33.3, 33.3, 33.3, 33.3, 33.5,
       33.6, 33.7, 33.9, 33.9, 34.2, 34.2, 34.3, 34.3, 34.4, 34.5, 34.6,
       36.4, 38.9, 40.2)

C <- c(32.0, 32.2, 32.5, 32.6, 32.7, 32.7, 32.7, 33.0, 33.2, 33.4, 33.6,
       33.6, 33.9, 34.1, 34.2, 34.4, 34.4, 34.5, 34.6, 34.7, 36.3, 36.6,
       36.7, 38.9, 39.2)

Instancia <- factor(1:length(A))
datos_anchos <- data.frame(Instancia, A, B, C)
dif_anchos <- data.frame(A_B = A - B, A_C = A - C, B_C = B - C)

# Llevar las matrices de datos a formato largo
datos <- datos_anchos |>
  pivot_longer(c("A", "B", "C"), names_to = "Algoritmo", values_to = "Tiempo") |>
```

```
  mutate(Algoritmo = factor(Algoritmo))

dif <- dif_anchos |>
  pivot_longer(everything(), names_to = "Algoritmos", values_to = "Diferencia") |>
  mutate(Algoritmos = factor(Algoritmos))

# Obtener gráficos Q-Q de las diferencias
qq <- ggqqplot(dif, x = "Diferencia", facet.by = "Algoritmos",
               color = "Algoritmos",
               palette = c("steelblue", "steelblue1", "steelblue4"),
               xlab = "Cuantil teórico",
               ylab = "Diferencias en tiempos\nde ejecución [ms]")
qq <- qq + theme(legend.position = "none")
print(qq)
```

Así, se cumplen las condiciones para aplicar un análisis robusto de una vía para muestras correlacionadas para contrastar, en el ejemplo, las siguientes hipótesis:

H_0: en promedio, no hay diferencias en el tiempo de ejecución necesitado por cada algoritmo en resolver las mismas instancias. Si $\mu_{(A-B)}$, $\mu_{(A-C)}$ y $\mu_{(B-C)}$ denotan las medias de las diferencias en tiempos de ejecución necesitado por cada par de algoritmos, entonces la hipótesis puede escribirse como: $\mu_{(A-B)} = \mu_{(A-C)} = \mu_{(B-C)} = 0$.

H_A: la media de las diferencias en el tiempo de ejecución necesitado para resolver las mismas instancias es diferente para al menos un par de algoritmos. Matemáticamente: $\exists A_i, A_j \in \{A, B, C\}, | \ \mu_{(A_i - A_j)} \neq 0$.

El script 12.10 ilustra el funcionamiento de las funciones `rmanova()` y `rmmcp()` del paquete `WRS2` que aplican las pruebas robustas ómnibus y post-hoc, respectivamente, para el análisis de las medidas repetidas del ejemplo con un nivel de significación $\alpha = 0{,}05$ y de recorte $\gamma = 0{,}2$. Como para las muestras independientes, es importante tener en cuenta que el procedimiento post-hoc utiliza el método de Benjamini y Hochberg para el ajuste de los valores p de las comparaciones post-hoc entre todos los pares de mediciones (aunque esto no es reportado por la función `rmmcp()`). Otro aspecto a notar es que a esta función también entregamos el nivel de significación establecido (`alpha = alfa`) para que pueda estimar los correspondientes intervalos de confianza.

Script 12.10: (continuación del script 12.9) análisis robusto de una vía para muestras correlacionadas.

```
# Fijar nivel de significación y nivel de poda
alfa <- 0.05
gamma <- 0.2

# Comparar los algoritmos usando medias truncadas de las diferencias
mr_rob <-rmanova(y = datos[["Tiempo"]], groups = datos[["Algoritmo"]],
                 blocks = datos[["Instancia"]], tr = gamma)

cat("Análisis de una vía para medidas repetidas (asimpótico)\n")
cat("-----------------------------------------------------\n")
print(mr_rob)

if(mr_rob[["p.value"]] < alfa) {
  mr_rob_ph <- rmmcp(y = datos[["Tiempo"]], groups = datos[["Algoritmo"]],
                     blocks = datos[["Instancia"]], tr = gamma, alpha = alfa)

  cat("Análisis post-hoc para medidas repetidas (asimpótico)\n")
  cat("-----------------------------------------------------\n")
  print(mr_rob_ph)
}
```

La figura 12.11 contiene el resultado de ejecutar el script 12.10, que indica que la prueba robusta ómnibus es significativa para el ejemplo ($F(1{,}50;\ 20{,}96) = 24{,}171;\ p < 0{,}001$) al nivel establecido, por lo que debemos rechazar la hipótesis nula. Concluimos, entonces, con 95 % confianza, que existe una diferencia estadísticamente significativa entre los tiempos promedio de ejecución de los algoritmos cuando no se consideran casos extremos.

Al efectuar el procedimiento post-hoc recortando los casos extremos podemos concluir que el algoritmo A es significativamente más eficiente que los algoritmos B ($\overline{x}^t_A - \overline{x}^t_B = -0{,}853$ [ms]; 95 %CI: $[-1{,}168;\ -0{,}538]$ [ms];

```
Análisis de una vía para medidas repetidas (asimpótico)
------------------------------------------------------
Call:
rmanova(y = datos[["Tiempo"]], groups = datos[["Algoritmo"]],
    blocks = datos[["Instancia"]], tr = gamma)

Test statistic: F = 24.1706
Degrees of freedom 1: 1.5
Degrees of freedom 2: 20.96
p-value: 1e-05

Análisis post-hoc para medidas repetidas (asimpótico)
------------------------------------------------------
Call:
rmmcp(y = datos[["Tiempo"]], groups = datos[["Algoritmo"]], blocks = datos[["Instancia"]],
    tr = gamma, alpha = alfa)

          psihat ci.lower ci.upper p.value p.crit   sig
A vs. B -0.85333 -1.16837 -0.53830 0.00000 0.0169  TRUE
A vs. C -0.68667 -0.98245 -0.39089 0.00002 0.0250  TRUE
B vs. C -0.00667 -0.26776  0.25443 0.94566 0.0500 FALSE
```

Figura 12.11: resultado del análisis robusto de una vía para las medidas repetidas del ejemplo.

$p < 0{,}001$) y C ($\overline{x}^t_A - \overline{x}^t_C = -0{,}687$ [ms]; 95 %CI: $[-0{,}982;\ -0{,}391]$ [ms]; $p < 0{,}001$). Sin embargo, debemos notar que la media recortada de las diferencia en los tiempos de ejecución requeridos por los algoritmos B y C ($\overline{x}^t_B - \overline{x}^t_C = -0{,}007$ [ms]; 95 %CI: $[-0{,}268;\ 0{,}254]$ [ms]) está al borde de ser significativamente distinta de cero ($p = 0{,}050$).

Una forma de comprobar los resultados anteriores es usar las funciones `rmanovab()` y `pairdepb()` que aplican bootstrapping para el análisis robusto ómnibus y post-hoc con las medidas repetidas del ejemplo, como muestra el script 12.11. Debemos comentar que nuevamente no indicamos un método de ajuste para pruebas múltiples ni el nivel de significación para el cálculo de intervalos de confianza en la llamada al procedimiento post-hoc, puesto que la función `pairdepb()` utiliza un método ad-hoc que considera el número de iteraciones bootstrap simuladas para controlar la tasa de error por familia y un nivel de significación constante $\alpha = 0{,}05$.

Script 12.11: (continuación del script 12.10) análisis robusto de una vía para muestras correlacionadas usando bootstrapping.

```r
# Fijar la cantidad de iteraciones bootstrap
nboot <- 999

# Comparar los algoritmos usando diferencias truncadas y bootstrapping
set.seed(666)
mr_bt <- rmanovab(y = datos[["Tiempo"]], groups = datos[["Algoritmo"]],
                  blocks = datos[["Instancia"]], tr = gamma, nboot = nboot)

cat("Análisis de una vía para medidas repetidas (bootstrapped)\n")
cat("------------------------------------------------------------\n")
print(mr_bt)

if(mr_bt[["test"]] > mr_bt[["crit"]]) {
  set.seed(666)
  mr_bt_ph <- pairdepb(y = datos[["Tiempo"]], groups = datos[["Algoritmo"]],
                       blocks = datos[["Instancia"]], tr = gamma, nboot = nboot)

  cat("Análisis post-hoc para medidas repetidas (bootstrapped)\n")
  cat("------------------------------------------------------------\n")
  print(mr_bt_ph)
}
```

La figura 12.12 presenta la salida que genera el script 12.11, donde es posible observar que el valor crítico estimado con bootstrapping $F^*_{\text{boot}} = 4{,}426$ es mucho menor que el estadístico de prueba observado $F = 24{,}171$;

```
Análisis de una vía para medidas repetidas (bootstrapped)
----------------------------------------------------------
Call:
rmanovab(y = datos[["Tiempo"]], groups = datos[["Algoritmo"]],
    blocks = datos[["Instancia"]], tr = gamma, nboot = nboot)

Test statistic: 24.1706
Critical value: 4.4257
Significant:  TRUE

Análisis post-hoc para medidas repetidas (bootstrapped)
--------------------------------------------------------
Call:
pairdepb(y = datos[["Tiempo"]], groups = datos[["Algoritmo"]],
    blocks = datos[["Instancia"]], tr = gamma, nboot = nboot)

          psihat ci.lower ci.upper     test    crit   sig
A vs. B -0.76000 -1.58621  0.06621 -4.59149 4.99147 FALSE
A vs. C -0.81333 -1.42198 -0.20469 -6.67012 4.99147  TRUE
B vs. C -0.05333 -0.52991  0.42325 -0.55859 4.99147 FALSE
```

Figura 12.12: resultado de procedimientos robustos para el análisis de una vía para medidas repetidas.

confirmando la existencia de diferencias estadísticamente significativas en los tiempos de ejecución requeridos por los algoritmos cuando no se consideran casos extremos.

El procedimiento post-hoc con bootstrapping, sin embargo, entrega una visión distinta a la obtenida con el equivalente asintótico. Vemos que se establece un valor crítico $t^*_{\text{boot}} = 4{,}991$ considerando las repeticiones bootstrap, el que solo es superado por la media recortada de las diferencias entre los algoritmos A y C: $(\overline{x}^t_A - \overline{x}^t_C)_{\text{boot}} = -0{,}813$ [ms]; 95 %CI: $[-1{,}422;\ -0{,}205]$ [ms]; $t_{\text{boot}} = 6{,}670$).

Consideremos nuevamente el gráfico Q-Q de las diferencias mostrado en la figura 12.10. Vemos que particularmente para las diferencias entre los algoritmos A y B se observan ciertas desviaciones de normalidad y asimetría que probablemente no se resuelven podando el 20 % de los casos de cada extremo, lo que podría estar afectando la calidad de la prueba asintótica. Por esta razón, en este caso, y en general con métodos asintóticos, los resultados basados en simulaciones de bootstrapping son más confiables. Así, concluimos entonces con 95 % confianza que la evidencia es solamente suficiente para establecer que el algoritmo A es más eficiente que el algoritmo C cuando no se consideran casos extremos.

Nota importante

Debemos saber que el paquete `WRS2` implementa muchas otros estimadores robustos y pruebas de hipótesis basadas en estos estimadores. Entre los no mencionados en esta sección, hay medidas y procedimientos robustos para obtener intervalos de confianza a partir de una muestra, alternativas para ANOVA de más de una vía, para modelos mixtos y para analizar regresión. También hay una familia de funciones que permite comparar variables aleatorias discretas (proporciones).

No es posible discutir las bases teóricas de todos estos métodos en esta sección, pero los desarrollos matemáticos y teoremas asociados a sus distribuciones muestrales se presentan en detalle en Wilcox (2012). Este libro se ha convertido en un referente de la estadística robusta, y presenta muchos más conceptos y métodos que los implementados en `WRS2`.

R. R. Wilcox no es el único investigador en esta área, como puede verse en la página de R dedicada al tema (Maechler, 2014).

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 los supuestos sobre la distribución de los datos (como normalidad u homocedasticidad) o el conocimiento de parámetros poblacionales (como la varianza) que hacen las pruebas paramétricas estudiadas. También pueden ser útiles cuando las muestras son reducidas e insuficiente para asegurar que el Teorema del Límite Central está operando, lo que hace que las aproximaciones asintóticas en que se basan las pruebas clásicas puedan ser imprecisas.

La idea básica detrás del remuestreo es extraer **repetidamente** muestras desde un conjunto original de datos observados para obtener **información sobre la población** de la que provienen. A estas muestras de la muestra original se les conoce como **remuestras**.

En vez de depender de supuestos teóricos sobre la distribución de la población, el remuestreo utiliza las remuestras para **simular múltiples experimentos** y obtener un conjunto de estadísticas de ellos, desde donde se pueden obtener estimaciones de la variabilidad de un estadístico muestral, construir intervalos de confianza o docimar hipótesis.

Sin embargo, aunque el remuestreo relaja muchas condiciones paramétricas, no está libre de supuestos:

- La muestra original es **representativa de la población**, que es fundamental para que las estadísticas obtenidas con las remuestras sean válidas. Si la muestra original está sesgada, las remuestras también lo estarán, y las conclusiones que podamos obtener podrían ser equivocadas para la población en estudio.
- Las observaciones dentro de la muestra original son **independientes**. Si bien existen algunos métodos de remuestreo para datos con dependencias (como series de tiempo con autocorrelación o datos agrupados), las técnicas básicas, como las estudiadas en este capítulo, requieren la independencia entre los casos incluidos en la muestra.

La validez de las conclusiones obtenidas con la mayoría de las técnicas de remuestreo depende críticamente de que estos supuestos se cumplan razonablemente. También es necesario tener en cuenta que cada técnica de remuestreo puede hacer suposiciones propias, y que deben cumplirse para aplicarse con validez.

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. Por otro lado, aunque 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, deberíamos tener claro que en estadística el ideal es contar con varias muestras grandes. Pero muchas veces solo disponemos de una muestra relativamente 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 remuestreo **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 remuestras (cientos, miles, decenas de miles y hasta cientos de miles) a partir de la muestra original. Cada remuestra debe tener el **mismo tamaño** que la original y se construye mediante **muestreo con reposición**. Esto quiere decir que al seleccionar un caso desde la muestra original, este se devuelve a ella antes de tomar el siguiente, por lo que el caso 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 de la estimación puntual (un solo 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 12.13 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.

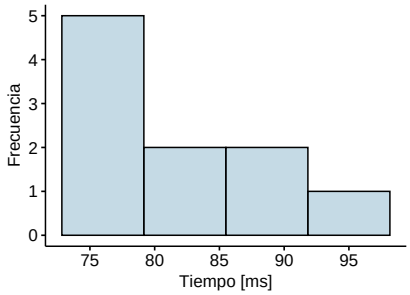


Figura 12.13: distribución del tiempo de ejecución para la muestra ejemplo.

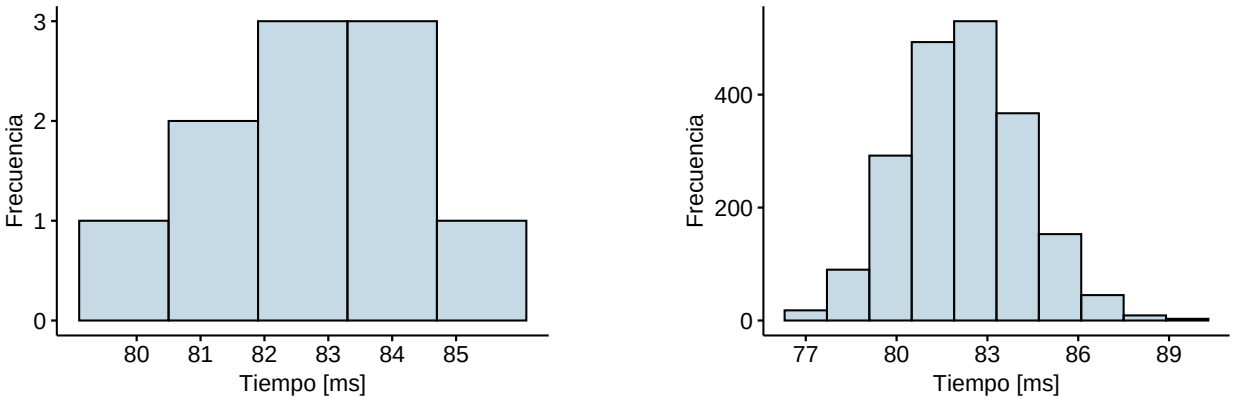
Evidentemente, la muestra es pequeña ($n = 10$) y su distribución exhibe asimetría hacia la derecha, por lo que Chufe ha decidido emplear bootstrapping como alternativa para enfrentar estos datos problemáticos.

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.

La figura 12.14 muestra la distribución bootstrap de la media para los 10 remuestreos del ejemplo (figura 12.14a) y para 2.000 remuestreos (figura 12.14b). En ella podemos ver claramente que, a medida que la

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



(a) 10 remuestreos.

(b) 2.000 remuestreos.

Figura 12.14: distribuciones bootstrap de la media.

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.

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 12.14b 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 12.12 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 12–13 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 (línea 18), luego de fijar una semilla para la generación de números pseudoaleatorios (línea 17) de manera de garantizar la reproducibilidad de los resultados.

Script 12.12: construcción de un intervalo de confianza para la media poblacional mediante bootstrapping.

```
1  library(boot)
2  library(bootES)
3
4  # Crear muestra inicial, mostrar su histograma y calcular la media
5  muestra <- c(79, 75, 84, 75, 94, 82, 76, 90, 79, 88)
6
7  # Establecer cantidad de remuestreos y nivel de significación
8  B <- 2000
9  alfa <- 0.01
10
```

```
11 # Función para calcular el estadístico: media de la remuestra
12 media <- function(valores, i) {
13   mean(valores[i])
14 }
15
16 # Construir la distribución bootstrap usando el paquete boot
17 set.seed(432)
18 distribucion_b <- boot(muestra, statistic = media, R = B)
19
20 # Mostrar y graficar la distribución bootstrap
21 print(distribucion_b)
22 plot(distribucion_b)
23
24 # Construir y mostrar los intervalos de confianza
25 ics <- boot.ci(distribucion_b, conf = 1 - alfa,
26                type = c("norm", "perc", "bca"))
27 cat("\n\n")
28 print(ics)
```

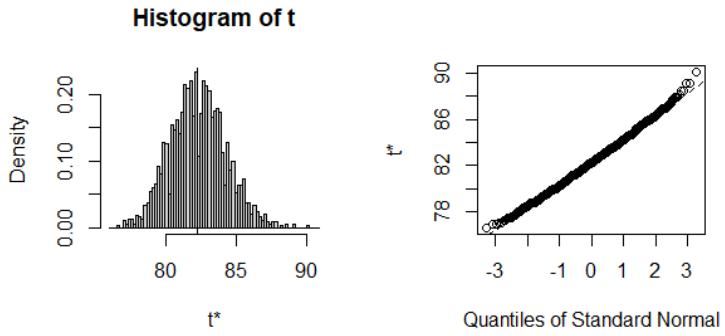


Figura 12.15: histograma y gráfico Q-Q de la distribución bootstrap generada para el ejemplo.

La línea 21 muestra las estadísticas de la distribución bootstrap obtenida, como se presentan en la figura 12.16, mientras que la figura 12.15 la muestra de forma gráfica mediante un histograma y un gráfico Q-Q que se obtiene invocando a la función `plot()` con el objeto entregado por la función `boot()` (línea 22).

Es importante revisar la distribución bootstrap para confirmar que tiene la forma esperada. Observando los gráficos de la figura 12.15, podemos notar que la distribución bootstrap obtenida sigue un comportamiento aproximadamente normal, tal vez con una pequeña asimetría positiva, que es lo esperado para medias muestrales.

En las líneas 25–26 se muestra el uso de `boot.ci()` para construir los intervalos de confianza usando diferentes métodos. Bajo el título `BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS` en la figura 12.16 se encuentran los resultados: [77,03; 87,25] usando aproximación normal, [77,40; 87,70] usando percentiles y [77,48; 87,90] al usar el método BCa.

El script 12.13 muestra 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 omisión `0.95`).
- `plot`: por omisión 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.

```
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
```

Figura 12.16: intervalos de confianza para el ejemplo obtenidos con bootstrapping.

El script 12.13 genera los mismos gráficos mostrados en la figura 12.15 y la siguiente salida a pantalla:

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

Podemos observar que la función `bootES()` entrega los mismos resultados que los obtenidos con `boot()`, donde el intervalo de confianza reportado corresponde al tipo Bca, con la excepción del número de decimales que usan.

Script 12.13: (continuación del script 12.12) uso de la función `bootES()` para aplicar bootstrapping al ejemplo.

```
# Construir la 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
print(distribucion_bES)
```

Considerando 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]

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.

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

El script 12.14 muestra cómo aplicar estas ideas al ejemplo, que entrega el siguiente resultado:

```
Media observada: 82.2
Valor p: 0.0009995002
```

es decir, 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.

Script 12.14: (continuación del script 12.13) obtención del valor p basado en bootstrapping para el ejemplo.

```
# 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 y mostrar la media observada y el valor p
valor_observado <- media(muestra, 1:length(muestra))
p <- (sum(distribucion_nula > valor_observado) + 1) / (B + 1)
cat("Media observada:", valor_observado, "\n")
cat("Valor p:", p, "\n")
```

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.

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 12.15 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 12.17), 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()`.

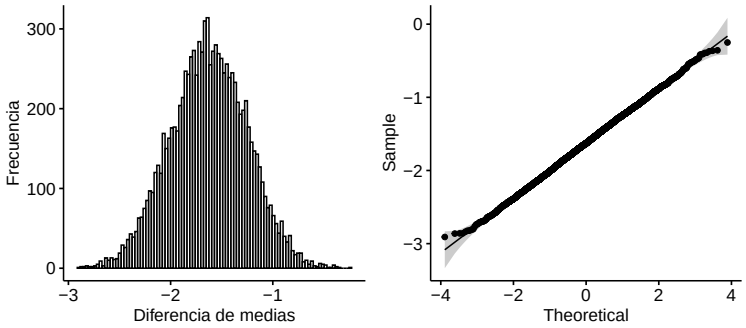


Figura 12.17: histograma y gráfico Q-Q de la distribución bootstrap generada para el ejemplo de la diferencia de dos medias.

Script 12.15: bootstraping para la diferencia de dos medias del ejemplo.

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

# Definir las muestras obtenidas
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 la 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\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 y mostrar el valor p
p <- (sum(distribucion_nula < diferencia_obs) + 1) / (B + 1)
cat("\nValor p:", p, "\n")
```

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 12.15) 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.

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 12.16, obteniendo los resultados que se presentan en la figura 12.18.

```
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 12.18: intervalo de confianza BCa y valor p para la media de las diferencias.

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: $(-0{,}656;\ 1{,}439)$.

Script 12.16: 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[2] 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 curso inicial de programación bajo estudio, y como μ_B al promedio de calificaciones finales de estudiantes

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

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 12.19), 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.

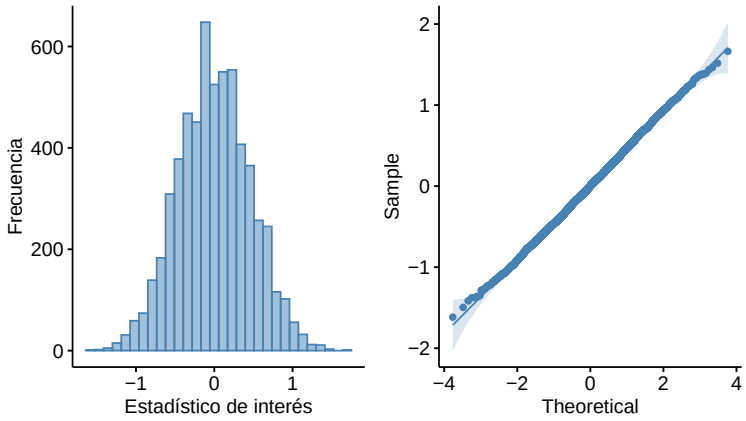


Figura 12.19: histograma y gráfico Q-Q de la distribución para la diferencia de medias generada mediante permutaciones.

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 12.17. 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 12.17: pruebas de permutaciones para variables numéricas.

```r
library(ggpubr)

# Definir las 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)
```

```r

# 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.
```

```r
# - ...: 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")

  # Generar permutaciones
  n_1 <- length(muestra_1)
  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 y mostrar el valor p
  valor_p <- calcular_valor_p(distribucion, observado, repeticiones,
                              alternative)

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


# ----- Bloque principal -----

# 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 12.3.

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 12.20 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.

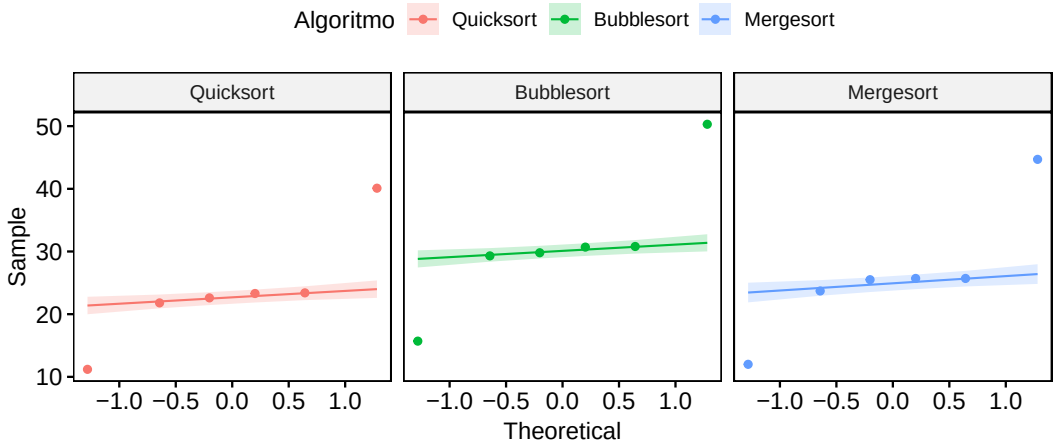


Figura 12.20: gráfico Q-Q para comprobar el supuesto de normalidad para el ejemplo.

A fin de determinar qué algoritmos difieren en su tiempo promedio de ejecución, ha decidido llevar a cabo un procedimiento post-hoc, calculando 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 12.21. 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 12.18 corresponde a la solución desarrollada por la estudiante.

```
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 12.21: resultado del procedimiento post-hoc.

Script 12.18: 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)

# 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)
```

```r
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")
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))
```

12.3 EJERCICIOS PROPUESTOS

12.3.1 Métodos robustos (sección 12.1)

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

Considera el siguiente enunciado:

> El conjunto de datos `essays` del paquete `WRS2` se compone de datos recolectados por un estudio de los efectos de dos formas de retroalimentación sobre la calidad de la escritura académica producida por estudiantes de inglés como lengua extranjera en educación superior. Tres grupos de estudiantes, dos de tratamiento (las dos formas de retroalimentación) y uno de control, se formaron de forma aleatoria. Cada estudiante escribió cuatro ensayos: uno antes del tratamiento, uno de práctica durante el tratamiento, uno terminado el tratamiento y el último un mes después del tratamiento. Obviamente, estudiantes del grupo de control realizaron las tareas de escritura pero no recibieron retroalimentación.

y responde las siguientes preguntas:

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).

12.3 Determina si las y los estudiantes del grupo de control pudieron mejorar la tasa de errores cometidos en el tercer ensayo respecto del segundo.

12.4 Determina si las y los estudiantes que recibieron retroalimentación directa mantuvieron la misma tasa de errores en el tercer y cuarto ensayo.

12.5 Determina si la tasa de errores cometidos en el tercer ensayo son las mismas para cada uno de los tres grupos de estudiantes.

12.6 Determina si las tasas de errores mejoran al aplicar la retroalimentación indirecta y, si es así, si esta se mantiene un mes después de la intervención.

Considera el siguiente enunciado:

> El conjunto de datos `bush` del paquete `WRS2` contiene datos de un "reality" australiano en que los participantes debieron comer palotes, ojos de pescado, testículos de canguro y larvas de una polilla. Específicamente se registró el tiempo que tardaron ocho participantes del programa en tener arcadas después de consumir cada uno de los tipos de alimentos.

y responde las siguiente pregunta:

12.7 Determina si el tiempo que se tarda una persona en tener arcadas al comer estas cosas es el mismo usando métodos robustos.

12.3.2 Técnicas de remuestreo (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 12.3.1. 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 12.3.1. 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 12.3.1. 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 12.3.1 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 12.3.1. 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 12.3.1. 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 12.3.1. 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 12.3.1. 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.

12.4 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.