MonteCarlo

Toolbox de Matlab

Herramientas para un laboratorio de estadística fundamentado en técnicas Monte Carlo

Josep Maria LOSILLA VIDAL

Tesis doctoral dirigida
por el Dr. Josep Maria Domènech i Massons

Departament de Psicologia de la Salut
Facultat de Psicologia
Universitat Autònoma de Barcelona
1994

MonteCarlo
Toolbox de Matlab

Herramientas para un laboratorio de estadística fundamentado en técnicas Monte Carlo

Josep Maria LOSILLA VIDAL

Tesis doctoral dirigida
por el Dr. Josep Maria Domènech i Massons

Departament de Psicologia de la Salut
Facultat de Psicologia
Universitat Autònoma de Barcelona
1994

A Mireia.

Este trabajo ha sido desarrollado en el marco del proyecto de investigación *"Programes del Laboratori d'Estadística Aplicada i de Modelització"*.

INDICE

1

INTRODUCCIÓN Y OBJETIVOS

«The history of science exhibs a steady tendency to eliminate intellectual effort in the solution of individual problems, by developing comprehensive formulas which can resolve by rote a whole class of them.»

Ernest Nagel, 1955

Las denominadas *técnicas de remuestreo* constituyen, sin duda, uno de los avances más importantes en la investigación y el desarrollo de nuevos procedimientos estadísticos. Estas técnicas permiten resolver de forma automática una clase más general de problemas de análisis de datos que las técnicas estadísticas clásicas, ofreciendo soluciones de especial interés en muchas situaciones en las que la estadística clásica es poco eficaz.

La finalidad básica de las técnicas de remuestreo es la extracción de la máxima información posible de un conjunto de datos para dar respuesta a las preguntas que se formulan los investigadores; el punto en común a todas ellas es la utilización del procedimiento Monte Carlo para generar aleatoriamente, a partir de un modelo de distribución teórica o de una muestra de datos observados, muchas muestras de datos y, a partir de las muestras simuladas, evaluar el error estadístico, estimar parámetros, u obtener el grado de significación de una prueba de contraste de hipótesis.

Desde la exposición en 1979 por parte de Bradley Efron de una nueva y prometedora técnica de remuestreo denominada *bootstrap* (descrita ya, aunque informalmente, en 1969 por Julian L. Simon), y a medida que ha ido aumentando la potencia y accesibilidad de los ordenadores personales, se han intensificado las investigaciones sobre este tipo de técnicas, entre las que se encuentran también, entre otras, las pruebas de aleatorización (o tests de permutación), el *jackknife* y la validación cruzada. Todas ellas se pueden considerar derivaciones del muestreo Monte Carlo (Manly, 1991); sin embargo, autores como Simon y Bruce (1991b), no están de acuerdo en incluir al *jackknife* dentro de este grupo.

Completando el aspecto terminológico, hay que señalar que, además de la denominación "técnicas de remuestreo" (Simon y Bruce, 1991a; Westfall y Young, 1993), con frecuencia se utiliza la denominación "técnicas de computación intensiva" (Diaconis y Efron, 1983; Noreen, 1989) para referirse a estos procedimientos, si bien este último término también engloba otro tipo de técnicas, como los algorítmicas (por ejemplo, el algoritmo EM; Dempster, Laird y Rubin, 1977) o el "Gibss sampler" (Geman y Geman, 1984; Gelfand y Smith, 1990).

Uno de los aspectos que ha despertado más el interés por las técnicas de remuestreo (o de computación intensiva en general) es, tal vez, el hecho de que se presentan como *"general-purpose tools"*, es decir, como herramientas de utilización general. Como comentábamos más arriba, el término "general" hace referencia aquí a que estos procedimientos tienen campos de aplicación más amplios que las técnicas estadísticas clásicas, al no requerir el cumplimiento de supuestos teóricos, en ocasiones difíciles de comprobar, sobre la forma de las distribuciones de los datos en las poblaciones y sobre las condiciones que deben cumplir algunos parámetros. En esta línea, Micceri (1989) demuestra que la mayor parte de variables investigadas en las ciencias del comportamiento no se distribuyen normalmente. Cuando estos supuestos no se cumplen, los métodos que se fundamentan en ellos ofrecen resultados menos fiables, y a pesar de ello éstos siguen aplicándose. En opinión de Efron (1982b, p. 173), «*la honesta respuesta a "¿Por qué asumir normalidad?", la mayoría de las veces es "Porque entonces podemos calcular el resultado"*».

Por otro lado, las técnicas estadísticas de computación intensiva permiten utilizar índices cuyas distribuciones muestrales no son matemáticamente conocidas. Actualmente nada impide al estadístico utilizar el ordenador para derivar, mediante métodos de simulación Monte Carlo, la forma de la distribución muestral de cualquier índice. De hecho, la mayoría de nosotros no recurre ya a tablas impresas de las distribuciones más conocidas, como las de z, t, F o χ^2, porque es más fácil para nuestras calculadoras de bolsillo realizar el cálculo del

valor de la función en cada caso particular. «*Estos métodos reemplazarán la "teoría de libro", tipificada por las tablas de t o de F, por la "teoría de la improvisación", generada de nuevo por el ordenador para cada problema de análisis de datos. Los teóricos no se quedarán, no obstante, sin trabajo. Los métodos estadísticos por ordenador son en sí mismos objetos perfectamente dignos de atención matemática. Su análisis, comparación y refinamiento es una perspectiva formidable para el estadístico teórico de finales del siglo XX*» (Efron, 1982b, p. 181).

Sin embargo, a pesar de que los fundamentos teóricos de las técnicas de remuestreo son ya el objeto principal de estudio para muchos estadísticos, y que se han desarrollado procedimientos específicos que permiten su aplicación casi automática, su uso es todavía muy restringido debido a la falta de herramientas informáticas adecuadas. En este sentido, creemos que los grandes paquetes de análisis estadístico como SAS, SPSS, BMDP, SYSTAT, etc. deben ir incorporando plenamente estos nuevos procedimientos, ya que hasta ahora sólo los han introducido de forma parcial en algunos de sus algoritmos, especialmente en el caso del SAS (Carson, 1985; Bailey, 1992; Chant y Dalgleish, 1992; Chen y Dunlap, 1993; Westfall y Young, 1993).

El avance en esta dirección permitirá utilizar las técnicas de remuestreo en el análisis de datos, pero no evitará a los investigadores tener que aprender el sofisticado arte de la programación de ordenadores para poder emplearlas en el contexto de la investigación y de la docencia de la estadística, lentificando su generalización. El objetivo del presente trabajo es ayudar a paliar este problema mediante la construcción de un instrumento informático que, sobre una plataforma estándar y fiable, permita aplicar de forma correcta y fácil los procedimientos de muestreo Monte Carlo, ajustándose a las características específicas de los dos contextos mencionados.

La primera parte del trabajo contiene una exposición sistemática de los fundamentos de las técnicas de muestreo Monte Carlo, así como de los criterios comparativos y de selección de la técnica más adecuada para cada problema de inferencia estadística. La aplicación de estas técnicas se ilustra de forma práctica mediante sus algoritmos en pseudo-código. Esta revisión constituye el punto principal del análisis de requerimientos del instrumento informático objeto de esta tesis.

En la segunda parte se amplia el análisis de requerimientos, concluyéndose que los paquetes estadísticos estándar en el entorno PC no satisfacen algunos de los requisitos más importantes (aunque algunos de ellos permiten alcanzar una cierta flexibilidad mediante la implementación de macros de programación) y, por

tanto, no son la herramienta adecuada. La opción que se escoge es diseñar y construir una librería de funciones y macro-mandatos que bautizamos con el nombre *MonteCarlo*, para su uso en el sistema MATLAB/SIMULINK (en cuyo entorno se utiliza el término "caja de herramientas" -*toolbox*- para referirse a las extensiones del lenguaje base). Este sistema cumple plenamente con el análisis de requerimientos realizado, y constituye además, a nuestro juicio, uno de los entornos de programación específicos para análisis numérico y visualización más potentes hoy en día.

Por último, en la tercera parte del trabajo se valida el *toolbox MonteCarlo* comprobando el *correcto análisis* de sus mandatos, la *precisión* de los resultados que se obtienen, y la adecuación de MATLAB como plataforma para la simulación estadística. Esta comprobación se realiza a través de la réplica de diferentes experimentos clásicos de simulación publicados por relevantes investigadores que trabajan en el ámbito del remuestreo estadístico.

2

PRINCIPALES APLICACIONES DE LOS MÉTODOS MONTE CARLO EN LA INFERENCIA ESTADÍSTICA

«Throughout history people had learned about the odds in gambling games by experimental trial-and-error experience. To find the chance of a given hand occurring in a card game, a person would deal out a great many hands and count the proportion of times that the hand in question occurred. That was the resampling method, plain and simple.»

Julian L. Simon y Peter Bruce, 1991

2.1. INTRODUCCIÓN

Los métodos Monte Carlo se inician en experimentos sobre series de números aleatorios, como las generadas por el lanzamiento sucesivo de un dado o los resultados de una rueda de ruleta. El término *Monte Carlo* se generaliza por esta analogía con los juegos de azar, y empieza a utilizarse hacia 1944, casi al final de la Segunda Guerra Mundial, momento en que se inicia el desarrollo de estos métodos, al ser aplicado por Fermi, von Neumann y Metropolis para dar solución a problemas relacionados con la fisión nuclear, que aparecen en la construcción de la bomba atómica. Estos investigadores, ya utilizan técnicas de reducción de la variancia, concretamente la denominada "ruleta rusa" o la técnica de la "división en partes" (*splitting*), para disminuir el número de simulaciones y aumentar la eficiencia de las estimaciones (Hammersley y Handscomb, 1964). Al poco tiempo, los métodos Monte Carlo se utilizan para resolver ciertas ecuaciones

integrales que aparecen en la física y para las cuales no se puede obtener una solución analítica.

Como señala Gordon (1980), habitualmente se utiliza la denominación *Monte Carlo* para describir cualquier método de cálculo que utilice números aleatorios, pero autores como Rubinstein (1981), Ripley (1987) y Lewis y Orav (1989), subrayan la conveniencia de utilizar este término sólo para usos específicos como la "integración Monte Carlo" o el "muestreo Monte Carlo", y utilizar el término *"simulación estocástica"* para los experimentos que hacen uso de series de números aleatorios generadas por ordenador. Rubinstein destaca las siguientes diferencias entre el método Monte Carlo y la simulación estocástica:

- En el método Monte Carlo el tiempo no juega un rol tan importante como en la simulación estocástica.
- En el método Monte Carlo las observaciones, por norma, son independientes. En la simulación, sin embargo, se experimenta con el modelo en el tiempo y, como regla, las observaciones están correlacionadas serialmente.
- En el método Monte Carlo es posible expresar la respuesta como una función simple de las variables estocásticas de entrada. En la simulación la respuesta es habitualmente muy compleja y sólo puede ser expresada explícitamente por el propio programa informático.

J.E. Gentle (1985, p. 612), ofrece una definición general de los métodos Monte Carlo, que incluye su aplicación en el ámbito de la estadística:

> «*Los métodos Monte Carlo son aquéllos en los que las propiedades de las distribuciones de las variables aleatorias son investigadas mediante la simulación de números aleatorios. Estos métodos, dejando a un lado el origen de los datos, son similares a los métodos estadísticos habituales, en los cuales las muestras aleatorias se utilizan para realizar inferencias acerca de las poblaciones origen. Generalmente, en su aplicación estadística, se utiliza un modelo para simular un fenómeno que contiene algún componente aleatorio. En los métodos Monte Carlo, por otro lado, el objeto de la investigación es un modelo en sí mismo, y se utilizan sucesos aleatorios o pseudo-aleatorios para estudiarlo.*»

Una de las contribuciones más tempranas del muestreo Monte Carlo en el ámbito de la investigación en estadística fue llevada a cabo por "Student" (W.S.

Gosset), en 1908, al estudiar la distribución del coeficiente de correlación y del estadístico *t* que lleva su nombre. Para ello, utilizó los datos de 3000 criminales, obteniendo de forma empírica, mediante muestreo Monte Carlo, la distribución muestral del estadístico *t* antes de resolver el problema analíticamente (Stigler, 1978).

En 1963, G.A. Barnard aplica por vez primera el muestreo Monte Carlo para evaluar la significación en un contraste de hipótesis. Su objetivo es comprobar la hipótesis de que un conjunto de datos constituye una muestra aleatoria de una población específica cuya distribución tiene parámetros conocidos y está perfectamente definida. El procedimiento consiste en simular muestras a partir de dicha población, obtener el valor del estadístico de contraste para cada muestra simulada, y ubicar posteriormente en la distribución muestral de los valores simulados el valor del estadístico de contraste obtenido al aplicar la prueba en la muestra real.

Hoy en día, el muestreo Monte Carlo se utiliza fundamentalmente en la investigación teórica de los métodos estadísticos, ya que en el contexto aplicado el conocimiento de los parámetros poblacionales es inusual. En la investigación aplicada generalmente se dispone sólo de una muestra de datos extraídos de una población y desea conocer, a partir de estos datos, algunas características de la población origen. Tal y como se mostrará en los siguientes apartados, el procedimiento Monte Carlo se puede utilizar en estos casos para generar nuevas muestras de datos mediante técnicas de muestreo (con o sin reposición), a partir de los datos de la muestra original.

Entre las técnicas que hacen uso del procedimiento de muestreo Monte Carlo actualmente destacan por su aplicabilidad y generabilidad las pruebas de aleatorización (*randomization tests*), el *jackknife*, la *validación recíproca* y el *bootstrap*.

Simon y Bruce (1991b), entre otros, utilizan el término *técnicas de remuestreo* para referirse globalmente a este tipo de procedimientos. Estos autores ofrecen la siguiente definición del término *remuestreo*:

> *«Utilizando el conjunto completo de datos que se poseen, o utilizando un mecanismo generador (a modo de dado) que sea un modelo del proceso que se desea comprender, se producen nuevas muestras de datos simulados, y se examinan los resultados de esas muestras... En algunos casos también puede ser apropiado amplificar este procedimiento con suposiciones adicionales.»*

Entre las revisiones introductorias encaminadas a la aplicación de estas técnicas, cabe destacar a Noreen (1989), que presenta una exposición de todas ellas; Edgington (1980), que se centra exclusivamente en las pruebas de aleatorización; Manly (1991), que revisa las pruebas de aleatorización y los métodos Monte Carlo; Gray y Schucany (1972), Miller (1974), Frangos (1987) y Hinkley (1983), que realizan una revisión de las principales investigaciones realizadas acerca del *jackknife* y de sus aplicaciones; Bailey, Harding y Smith (1984), Picard y Berk (1990) y Camstra y Boomsma (1992), que revisan las aplicaciones de la validación cruzada; y Lunneborg (1987), Stine (1990), Westfall y Young (1993) y Mooeny y Duval (1993), que presentan los fundamentos teóricos y aplicaciones del *bootstrap*.

A modo de preámbulo, antes de realizar una descripción de las características y propiedades específicas de cada una de estas técnicas, realizaremos una breve exposición de las ideas generales que rigen la utilización de las técnicas de remuestreo en el ámbito de la inferencia estadística, concretamente para la estimación del error estadístico y de parámetros poblacionales, y para la estimación del grado de significación en las pruebas de contraste de hipótesis.

2.1.1. Estimación del error estadístico y de parámetros poblacionales

Para realizar estimaciones válidas del error estadístico -entendiendo como tal el sesgo, el error estándar y la probabilidad de error de predicción (Efron y Gong, 1983)- y estimaciones de parámetros poblacionales, las técnicas de remuestreo se basan en la generación nuevas muestras de datos a partir de los datos de la muestra original, y en el cálculo posterior de la variabilidad del estadístico de interés a partir de la variabilidad observada en la distribución de muestras generadas. Las diferencias entre estas técnicas reside principalmente en el tipo de muestreo (o remuestreo) utilizado para generar los nuevos conjuntos de datos.

La más antigua de estas técnicas, el *jackknife*, fue ideada por Maurice Quenouille (1949, 1956), y sistematizada más tarde por John W. Tuckey (1958), a quien debe también su nombre. Ha sido ampliamente estudiada desde entonces. En el *jackknife*, las estimaciones se realizan a partir de las muestras resultantes de la exclusión de una observación (o varias) del conjunto inicial de datos.

En la técnica de la validación recíproca ("*cross-validation*") los datos se dividen en dos grupos (dos mitades, $n-1$ y 1, etc.). Sobre el primer grupo se

realizan las estimaciones de los parámetros de interés y, por último, se verifican dichas estimaciones aplicándolas (por ejemplo mediante ecuaciones predictivas) al segundo grupo de datos, con lo cual se obtiene un indicador fiable de la validez de las estimaciones para nuevos conjuntos de datos.

Por último, la técnica *bootstrap*, expuesta por Bradley Efron en 1979, se basa en la generación aleatoria de nuevos conjuntos de datos mediante un muestreo aleatorio con reposición de los datos de la muestra original. Aunque esta técnica requiere mucha más potencia computacional que las anteriores, como veremos más adelante, parece ser más versátil y general.

Además, y atendiendo a los desarrollos teóricos recientes, las estimaciones *bootstrap* del sesgo y del error estándar de los estadísticos, y las variaciones que sobre el cálculo de estas estimaciones pueden realizarse, ofrecen globalmente resultados más fiables que sus homólogos obtenidos con base en el *jackknife* o la validación recíproca (Efron 1979b, 1982a; Efron y Gong, 1983; Diaconis y Efron, 1983). Sin embargo, en determinadas ocasiones, puede resultar ventajoso combinar estas técnicas para obtener resultados más óptimos, como, por ejemplo, al utilizar el *jackknife* para reducir la variancia de las estimaciones *bootstrap* (Beran, 1987; Frangos y Schucany, 1990; Efron, 1990b). En este trabajo, se realizará una introducción del *jackknife* y la validación recíproca basada únicamente en el uso que puede realizarse de estas técnicas en los procedimientos de estimación *bootstrap*, que junto al muestreo Monte Carlo y las pruebas de aleatorización, constituyen el objeto principal de este capítulo.

Existen otras técnicas que pueden incluirse en este grupo, como la técnica de *replicación repetida y equilibrada*, desarrollada por McCarthy (1965, 1976), que efectúa particiones sistemáticas de los datos con el objetivo de evaluar la variabilidad de encuestas y de muestras censales, o el *submuestreo aleatorio*, debido a Hartigan (1969), que lo diseñó inicialmente para obtener intervalos de confianza en situaciones particulares.

2.1.2. Contraste de hipótesis

Como se verá en los siguientes apartados, mediante técnicas de remuestreo se puede hallar el grado de significación de una prueba estadística de contraste de hipótesis en cualquier situación. Sin embargo, esto no es siempre cierto para el caso de las pruebas clásicas. De hecho, si para unos determinados datos se cumplen los supuestos requeridos, la prueba de contraste de hipótesis convencional ofrece virtualmente el mismo grado de significación que la técnica

de computación intensiva correspondiente. Noreen (1989), señala que cuando una técnica de remuestreo es apropiada no está claro por qué alguien desearía utilizar una prueba paramétrica convencional, ya que, como se ha comentado anteriormente, el descenso del coste de la computación posibilita a cualquier estadístico, mediante un ordenador personal, aplicar estos nuevos métodos de cálculo intensivo para contrastar hipótesis.

El propio muestreo Monte Carlo, las pruebas de aleatorización y el *bootstrap* son las principales técnicas que se aplican para obtener el grado de significación de una prueba de contraste de hipótesis, así como para estimar el error respecto al nivel nominal fijado "a priori" y estimar su potencia.

El muestreo Monte Carlo y el *bootstrap* se utilizan cuando la hipótesis concierne a la población de la cual se supone que procede la muestra, aunque sus procedimientos de aplicación son distintos. Las pruebas de aleatorización, desarrolladas en el contexto de los diseños experimentales y extendidas luego a otros diseños de investigación, se aplican cuando la hipótesis concierne únicamente a la relación entre variables, sin realizar inferencia alguna a la población origen de los datos.

2.2. MUESTREO MONTE CARLO

2.2.1. Obtención del grado de significación en el contraste de hipótesis

El fundamento del muestreo Monte Carlo para evaluar el grado de significación de una prueba de contraste de hipótesis es el siguiente. Sea x una variable aleatoria extraída de una población y, sea $t(x)$ una función que define el estadístico de contraste de una prueba sobre x. Dado que x es una variable aleatoria, $t(x)$ será también una variable aleatoria que tiene su propia función de probabilidad. La probabilidad de que $t(x)$ pueda ser mayor o igual a un valor determinado (h), definida por la distribución muestral de $t(x)$, se describe formalmente por:

$$\text{prob}[\ t(x) \geq h\ ]$$

Si $t(x_o)$ es el valor obtenido al aplicar la prueba estadística en la muestra original de datos, la prueba de significación consistirá en calcular cuán inusual es $t(x_o)$ en relación a la distribución muestral de $t(x)$. Es decir, el grado de significación de la prueba estadística es

$$\text{prob}[\ t(x) \geq t(x_o)\ ]$$

y la regla para rechazar la hipótesis nula (H_0) es

$$\text{Rechazar } H_0 \text{ si prob}[\ t(x) \geq t(x_o)\ ] \leq \alpha$$

siendo α el nivel de significación establecido "a priori" (habitualmente 0.05).

Por tanto, el cálculo del grado de significación se basa en estimar la distribución muestral del estadístico de contraste $t(x)$ bajo la hipótesis nula. De este modo, el grado de significación unilateral es, simplemente, la proporción de muestras simuladas en las cuales el valor de $t(x)$ es mayor o igual (o inferior o igual) al valor obtenido en la muestra original $t(x_o)$.

Sólo si la hipótesis nula puede ser completamente especificada con base en parámetros poblacionales, entonces la distribución muestral de $t(x)$ puede ser estimada utilizando el muestreo Monte Carlo. Como señala Noreen (1989, p.44), «*este método es particularmente valioso en situaciones en las cuales la distribución de la población es conocida, pero la distribución muestral del estadístico de contraste no ha sido derivada analíticamente*».

El algoritmo general es el siguiente:

1. Decidir cuál es la prueba estadística de interés (θ)
2. Definir exactamente la población bajo H_0
3. Calcular $\hat{\theta}_o$ para la muestra de datos original X_o
4. Inicializar el contador NSIG = 0
5. Fijar el nº de muestras (NSIM) a simular
6. Para i = 1 hasta NSIM:
   - 6.1. X_i^* = DISTRND(*n*, *parámetros*)
   - 6.2. Calcular el estadístico $\hat{\theta}_i^*$ para la muestra generada X_i^*
   - 6.3. Si $\hat{\theta}_i^* \geq \hat{\theta}_o$ (o, según el caso $\hat{\theta}_i^* \leq \hat{\theta}_o$): NSIG = NSIG + 1
7. Calcular el grado de significación: p = (NSIG+1) / (NSIM+1)

ALGORITMO 1. Muestreo Monte Carlo para evaluar el grado de significación en una prueba de contraste de hipótesis.

La función DISTRND(*n*, *parámetros*) que aparece en el punto 6.1 del algoritmo, es la encargada de generar una muestra con *n* realizaciones extraídas aleatoriamente y sin reposición (con objeto de disminuir el error estándar) a partir de una función de distribución de probabilidad teórica cuyos *parámetros* se especifican.

Como se expone en los apartados 2.9. y 2.10., una prueba de contraste de hipótesis basada en el grado de significación obtenido a partir del cociente

$$p = \frac{(NSIG + 1)}{(NSIM + 1)}$$

es válida, puesto que la probabilidad de rechazar la hipótesis nula cuando ésta es verdadera no es mayor que el nivel de significación α fijado "a priori" (Dwass, 1957).

La principal desventaja del uso muestreo Monte Carlo para obtener el grado de significación de una prueba estadística estriba en que debe conocerse la función de distribución de probabilidad de la población.

2.2.2. Evaluación de las probabilidades de error Tipo I y Tipo II

El muestreo Monte Carlo no sólo se utiliza en el campo de la investigación aplicada, sino que es también una potente herramienta para evaluar las consecuencias de la violación de determinados supuestos en los que se fundamentan las pruebas estadísticas, o la comparación de las probabilidades de error Tipo I y Tipo II entre diferentes estadísticos de contraste en estas situaciones.

En estos casos, generalmente los datos se extraen aleatoriamente a partir de distribuciones que tienen características distintas de las que asumen los tests estadísticos (por ejemplo, distribuciones que difieren de la normal, asimetrías, variancias o covariancias heterogéneas, variables con escalas de medida muy limitadas, muestras muy pequeñas, etc.). El test estadístico de interés se aplica a un gran número de muestras de datos simulados (típicamente entre 1000 y 10000 muestras), registrando para cada una de ellas si el resultado del test (o de los tests, si se está comparando la robustez o la potencia de varias pruebas) es significativo ($p \leq \alpha$) o no significativo ($p > \alpha$).

Si la distribución teórica se ha definido de forma que la hipótesis nula (H_0) es cierta (por ejemplo, $\mu_1 = \mu_2$ en una prueba t bilateral de comparación de dos medias), entonces podemos considerar la proporción de resultados significativos del test como una *estimación empírica* de la probabilidad de cometer un error Tipo I. Si las consecuencias de las violaciones de los supuestos son poco importantes, entonces esta probabilidad debería ser similar al nivel nominal α fijado "a priori" (ver *Figura 1*).

Si la distribución teórica se ha definido de forma que H_0 es falsa (por ejemplo, $\mu_1 \neq \mu_2$ en una prueba t bilateral de comparación de dos medias), entonces la proporción de tests significativos es una estimación empírica de la potencia de la prueba estadística, $1 - \beta$. Este valor puede compararse con el valor teórico de la potencia esperado para el test en esta situación (aunque, como señalan Glass, Peckham y Sanders (1972), cuando las distribuciones tienen variancias heterogéneas, los cálculos teóricos de la potencia son problemáticos). Si las consecuencias de la violación de los supuestos del test son poco importantes, entonces el valor empírico y el valor teórico deberían ser similares (ver *Figura 2*).

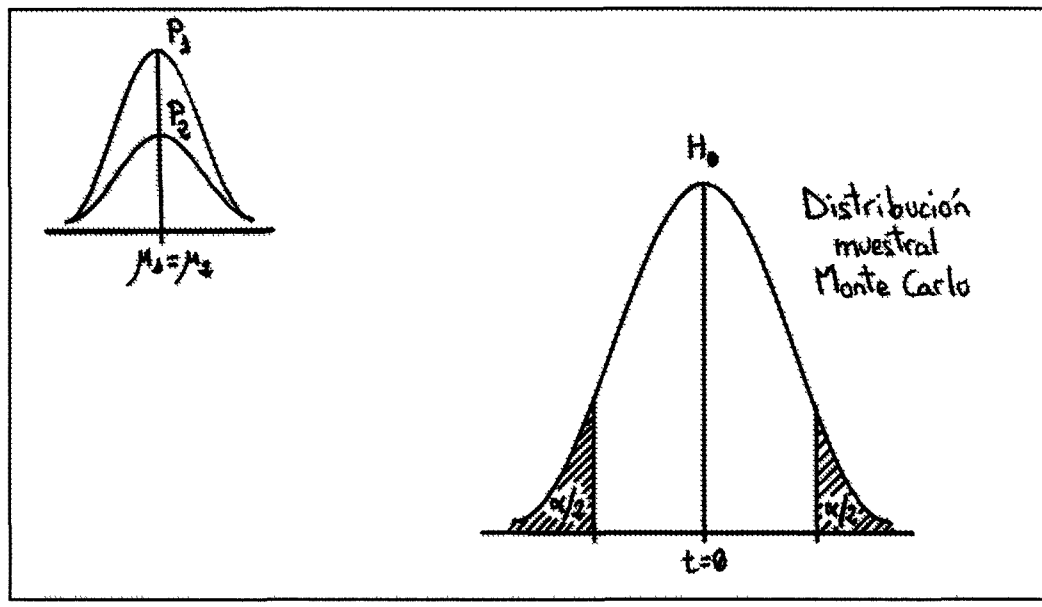


FIGURA 1. *Evaluación de la probabilidad de error Tipo I en un experimento Monte Carlo*

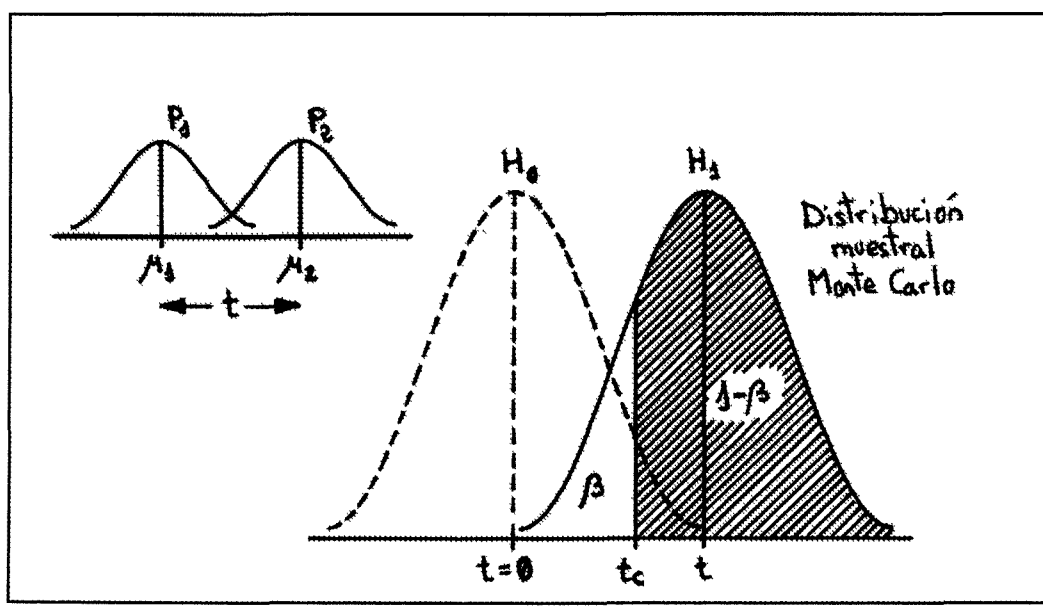


FIGURA 2. *Evaluación de la probabilidad de error Tipo II en un experimento Monte Carlo*

El algoritmo general en ambos casos es el siguiente:

1. Decidir cuál es la prueba estadística de interés (θ) y el nivel α
2. Definir exactamente la(s) distribución(es) que sigue(n) la(s) población(es), especificando la hipótesis nula H_0 en el caso de que se desee estudiar el nivel nominal α, o especificando la hipótesis alternativa H_1 si se desea estudiar la potencia de la prueba.
3. Inicializar el contador NSIG = 0
4. Fijar el nº de muestras (NSIM) a simular
5. Para i = 1 hasta NSIM:
   - 5.1. X_i^* = DISTRND([n_1, n_2, ...], [$parámetros_1$, $parámetros_2$,...])
   - 5.2. Calcular el pseudo-estadístico $\hat{\theta}_i^*$ para la muestra generada X_i^* y su grado de significación p_i^*
   - 5.3. Si $p_i^* \leq \alpha$:

     NSIG = NSIG + 1
6. Proporción de resultados significativos:

   p = NSIG / NSIM
7. Comparar p con el nivel α fijado a priori, o con el valor teórico $1 - \beta$, según se esté evaluando la robustez o la potencia del test.

ALGORITMO 2. *Muestreo Monte Carlo para evaluar los niveles de error Tipo I y Tipo II en pruebas de contraste de hipótesis.*

2.2.3. Aspectos computacionales

A nivel computacional, el aspecto principal en un muestreo Monte Carlo (así como en el resto de técnicas de computación intensiva) es el generador de secuencias de números pseudo-aleatorios, dado que supone la principal fuente de error en la simulación. Los números que produzca este generador deben ser *independientes* y mostrar una *distribución uniforme* en el intervalo [0,1]. Aplicando transformaciones sobre las series de números pseudo-aleatorios obtenidos se puede generar una variable aleatoria con una distribución determinada (Devroye, 1986; Bratley et al., 1987; Johnson, 1987; Lewis y Orav, 1989).

No se revisarán aquí los métodos para generar series de números aleatorios U(0,1), ni las técnicas para transformarlos en variables aleatorias de distribución conocida, dado que existen ya muchas subrutinas en diferentes lenguajes de programación que pueden utilizarse para realizar una aplicación basada en el muestreo Monte Carlo o en cualquier otra técnica de computación intensiva. De entre estas subrutinas, cabe destacar las que se encuentran en las librerías NAG, IMSL o Numerical Recipes.

Por otra parte, existen pruebas que permiten comprobar el carácter realmente casual o aleatorio de las series de números generadas, que siempre deberían ser aplicadas al ejecutar por primera vez las rutinas de generación de variables aleatorias. Ejemplos de este tipo de pruebas son, entre otras, la de χ^2 de bondad de ajuste, la de Kolmogorov-Smirnov, la de series, la de póker, la de corridas, etc. (Fishman, 1978; Knuth 1981).

En el capítulo 3 se presentan los algoritmos de generación de números pseudo-aleatorios según una distribución uniforme y de variables aleatorias no uniformes implementados en el instrumento informático que se desarrolla en la tesis.

2.3. PRUEBAS DE ALEATORIZACIÓN

Noreen (1989, p. 32), describe perfectamente el contexto histórico en el que surgieron las pruebas de aleatorización:

> «*Fisher originó el concepto de prueba de aleatorización en su libro "The Design of Experiments" publicado en 1935. En las dos décadas que siguieron a la publicación de este libro, las pruebas de aleatorización atrajeron la atención de estadísticos teóricos como Pitman (1937a, 1937b, 1937c), Pearson (1937), Scheffé (1943), Noether (1949), Lehmann y Stein (1949), y Hoeffding (1951, 1952). La aplicación de las pruebas de aleatorización para la comprobación de hipótesis reales fue impedida, sin embargo, por el alto coste que suponía recalcular manualmente la prueba estadística tantas veces. Los estadísticos acabaron con esta dificultad práctica construyendo tablas de probabilidad para datos genéricos (por ejemplo, rangos y categorías nominales). El resultado final de este esfuerzo es el amplio rango de pruebas no paramétricas fundadas en aquellas referencias estándar como Siegel (1956) y Hollander y Wolfe (1973). Genuinamente, todas las pruebas no paramétricas son casos especiales de pruebas de aleatorización exactas en las cuales las observaciones son, o han sido reemplazadas por, rangos o categorías nominales*».

Edgington (1980, p. 1) define las pruebas de aleatorización como «*procedimientos para determinar la significación estadística directamente desde los datos experimentales sin recurrir a tablas de significación*». En las pruebas de aleatorización la población de permutaciones representa el modelo de independencia. En este sentido, es fácil comprobar que estas pruebas son un caso especial de muestreo Monte Carlo en el que la población está formada por el conjunto de todas las permutaciones de una variable en relación a las otras, y las muestras que se obtienen son cada una de las posibles permutaciones.

En estos procedimientos, la aleatorización es utilizada para comprobar la hipótesis nula genérica de que una variable (o un conjunto de variables) no está relacionada con otra variable (u otro conjunto de variables).

La significación estadística se obtiene mediante la permutación de los valores de una variable (o variables) respecto a otra (u otras). Por tanto, la permutación de los datos conduce al rechazo o no rechazo de la hipótesis nula de independencia entre las variables. En efecto, cuando las variables están relacionadas, el valor que se obtiene al aplicar la prueba a los datos originales

será un valor poco frecuente respecto al conjunto de valores que se obtendrían al aplicar la misma prueba a las muestras de datos obtenidos a partir de la permutación de los datos originales.

La principal ventaja que ofrecen las pruebas aleatorización es, sin duda, su gran versatilidad y aplicabilidad, ya que pueden ser empleadas para hallar la significación tanto de pruebas simples como de pruebas más sofisticadas, incluso cuando las distribuciones muestrales de los estadísticos de contraste de dichas pruebas no están tabuladas, o en el caso de que no puedan utilizarse por incumplir alguno de los supuestos teóricos en que se basan (muestreo aleatorio, normalidad, homogeneidad de variancias, etc.). Como se expondrá más adelante, la validez de las pruebas de aleatorización depende sólo de la *asignación aleatoria* de los sujetos a cada uno de los niveles de la variable independiente.

2.3.1. Algoritmo general

Desde una perspectiva algorítmica, el término "prueba de aleatorización" ("*randomization test*") se refiere, concretamente, al procedimiento de determinar el grado de significación mediante la permutación de los datos experimentales y el cálculo repetido de una prueba estadística (en este sentido, según Noreen (1989), quizá sería más apropiado el término *"pruebas de permutación"*).

El algoritmo general de las pruebas de aleatorización es el siguiente:

1. Decidir cuál es la prueba estadística de interés (θ)
2. Calcular $\hat{\theta}_o$ para el conjunto de datos original X_o
3. Inicializar el contador NSIG = 0
4. Fijar el nº de permutaciones (NP) que se realizarán con los datos
5. Para i = 1 hasta NP:
   - 5.1. $X_i^* = PERM(X_o)$.
   - 5.2. Calcular el pseudo-estadístico $\hat{\theta}_i^*$ para el conjunto de datos generado X_i^*
   - 5.3. Si $\hat{\theta}_i^* \geq \hat{\theta}_o$ (o, según el caso $\hat{\theta}_i^* \leq \hat{\theta}_o$) : NSIG = NSIG + 1
6. Calcular el grado de significación: (NSIG+1) / (NP+1)

ALGORITMO 3. *Algoritmo general de las pruebas de aleatorización.*

En el paso 5.1. del algoritmo anterior, la función PERM(X_o) se encarga de generar las permutaciones del vector (o matriz) X_o que contiene los datos originales. Un algoritmo rápido propuesto por Knuth (1981) para permutar las filas de una matriz X de dimensión $(n \times m)$ es el siguiente:

1. Asignar n a j
2. Generar un número aleatorio U comprendido entre 0 y 1
3. Generar k = valor_entero(j U) + 1
4. Intercambiar los valores X_j y X_k
5. Restar 1 a j.
   - Si $j > 1$ → volver al paso 2
   - Si $j = 1$ → acabar

ALGORITMO 4. *Algoritmo para permutar los valores de un vector. (Tomado de Knuth, 1981, p. 139)*

Los datos pueden ser permutados de diferentes modos dependiendo de la hipótesis planteada y del diseño de la investigación. Un caso común implica una variable dependiente y una o más variables independientes o explicativas, y se plantea la hipótesis nula según la cual la variable dependiente no está relacionada con las variables independientes. En esta situación, los valores de la variable dependiente se permutan repetidamente respecto a los valores de las variables explicativas que se mantienen fijos. La permutación de este vector de datos genera muestras en las que la variable dependiente tiende a no estar relacionada con las variables predictoras. En otros casos, sin embargo, debe aplicarse otro tipo de permutaciones, como por ejemplo, permutaciones estratificadas. La permutación estratificada de los datos debe aplicarse, como señala Noreen (1989), siempre que haya una razón para pensar que el valor de la variable dependiente está afectado por el valor que toma una variable categórica que no es de interés primario para el contraste de hipótesis planteado. En el apartado 2.3.3. se revisarán los aspectos fundamentales que permitirían determinar el diseño más adecuado del tipo de permutaciones a realizar.

El valor del grado de significación obtenido a partir de una prueba de aleatorización es de fácil interpretación: no es más que la interpretación de una proporción observada, tal como ilustra la *Figura 3*. La distribución de probabilidad del estadístico utilizado (distribución pseudo-muestral) se genera a partir de la población de permutaciones. Por tanto, el grado de significación

obtenido en las pruebas de aleatorización se deriva directamente de las permutaciones de los datos experimentales.

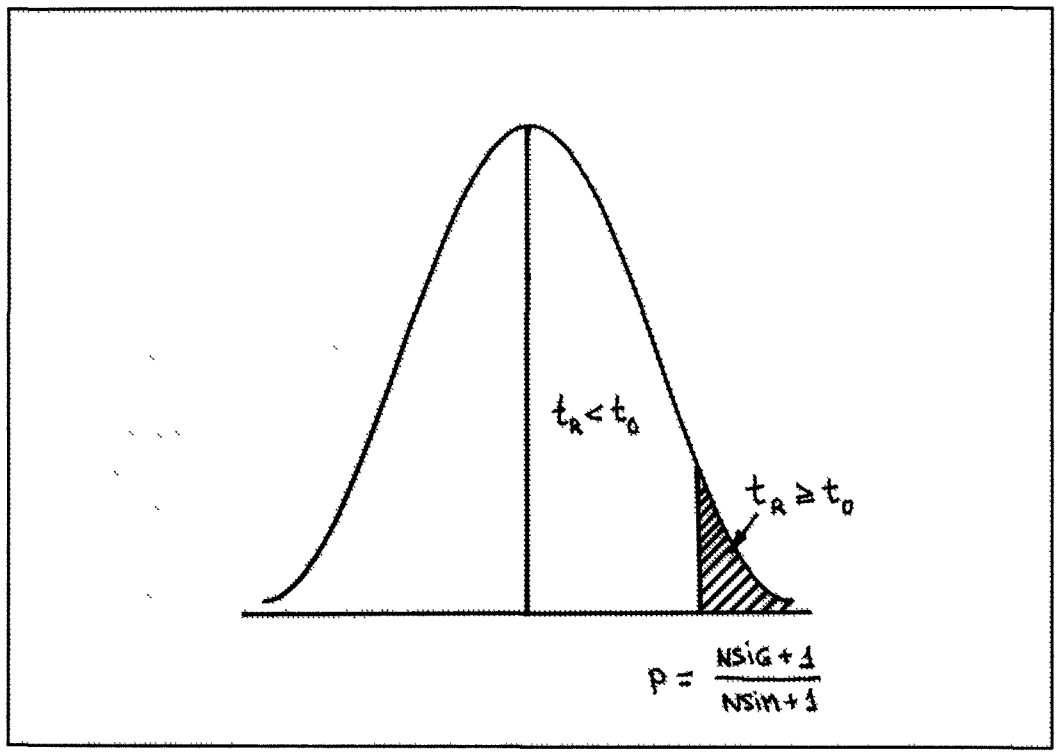


FIGURA 3. *Interpretación del grado de significación obtenido a partir de una prueba de aleatorización*

Es importante destacar que una prueba de aleatorización no es una prueba estadística en el sentido habitual del término, sino una forma de determinar el grado de significación estadística de una hipótesis. En otras palabras, las pruebas de aleatorización no son alternativas a las pruebas estadísticas convencionales, pero sí a las distribuciones muestrales de los estadísticos de contraste de dichas pruebas.

Edgington (1980) insiste en que la determinación de la significación por permutación de los datos es un procedimiento no paramétrico y, en consecuencia, el grado de significación de cualquier prueba estadística que se determine por este procedimiento también será no paramétrico.

Por otra parte, un aspecto fundamental que caracteriza a las pruebas de aleatorización, es su validez para cualquier tipo de muestra, tanto si ha sido

extraída aleatoriamente de una población como si no. Esta es una de las propiedades más importantes de estos procedimientos, debido al uso común de muestras no aleatorias en la experimentación, y a que las pruebas basadas en las distribuciones clásicas (*t*, *F*, ...) pueden no ser válidas para este tipo de muestras.

Siguiendo a Edgington (1980, p. 3):

> «*pocos experimentos en biología, educación, medicina, psicología o cualquier otro campo aplican un muestreo aleatorio de los sujetos, y aquellos casos en los que se aplica, generalmente conciernen a poblaciones tan específicas como de bajo interés. (...) En muchos experimentos, el concepto de población aparece en el análisis estadístico porque el investigador ha aprendido a interpretar los resultados de una prueba estadística en términos de inferencias a la población, no porque este investigador haya muestreado aleatoriamente una población a la cual desee generalizar*».

El supuesto que subyace en las pruebas estadísticas convencionales es que un procedimiento de muestreo aleatorio otorga la misma probabilidad a todas las posibles muestras de *n* individuos que se extraigan de la población, y "asegura" que la forma de la distribución de probabilidad empírica (obtenida a partir de los datos de la muestra extraída) será una buena aproximación de la distribución de la población. Así pues, si no se ha aplicado un muestreo aleatorio no es lícito realizar inferencias sobre la población de interés a partir del resultado obtenido con una prueba de significación paramétrica clásica. Sin embargo, este tipo de inferencias "no estadísticas" son práctica común en nuestra comunidad científica. Los investigadores generalizan a partir de sus sujetos experimentales a otros sujetos que son muy similares en aquellas características que consideran relevantes.

Una de las desventajas que presentan las pruebas de aleatorización es que no permiten, en sentido estricto, realizar inferencias a la población origen de la muestra, aunque ésta haya sido extraída por procedimientos de muestreo aleatorio. Sobre este inconveniente, Noreen (1989) apunta, sin embargo, que si se rechaza una hipótesis nula con base en una prueba de aleatorización aplicada a una muestra *realmente representativa* de la población, es probable que también pudiera rechazarse la hipótesis en dicha población. Es decir, que los procedimientos de extracción o elección de los sujetos y/o el diseño de la investigación pueden utilizarse como soportes "casi-formales" para realizar inferencias a la población.

Por otro lado, como señala Edgington (1966), en muchos casos, la hipótesis que se desea contrastar no implica inferencias a la población origen de los datos, sino simplemente la relación entre variables para un conjunto de sujetos experimentales. Las pruebas de aleatorización, que permiten realizar inferencias a partir de la distribución obtenida al aplicar el estadístico de contraste a permutaciones de los datos originales de los sujetos, sólo necesitan que la asignación de los sujetos a los diferentes niveles de las variables independientes se haya realizado aleatoriamente.

Por último, se puede rodear esta limitación aplicando la denominada prueba de aleatorización de Fisher (1924), en los casos en que es razonable suponer que las observaciones de una muestra son una muestra aleatoria extraída de una distribución simétrica alrededor de la media poblacional. En esta situación, si se resta a cada una de las observaciones el hipotético valor de la media poblacional y se asume que el número de signos positivos de las diferencias obtenidas tienen igual probabilidad al de signos negativos, la media observada en la muestra se puede entonces comparar con la distribución de medias que se obtiene asignando aleatoriamente los signos positivos y negativos a los valores de las diferencias observadas.

2.3.2. Determinación del numero de permutaciones necesarias

Las pruebas de aleatorización se pueden aplicar realizando todas las posibles permutaciones de los datos, o a partir de una muestra aleatoria de permutaciones. Cuando se realizan todas las permutaciones posibles de los datos de unas variables relativas a las otras *(permutación sistemática)*, la prueba se denomina *prueba de aleatorización exacta.* Sin embargo, en situaciones en las que el número de permutaciones posibles es tan grande que imposibilita la aplicación de la prueba de aleatorización exacta, se realizan *permutaciones aleatorias* de los datos y, en este caso, la prueba se denomina *prueba de aleatorización aproximada* (Noreen, 1989).

En las pruebas de aleatorización aproximadas, el procedimiento consiste en calcular el estadístico de interés para el conjunto original de datos y para una muestra aleatoria de permutaciones de estos datos. Esto supone generar un primer conjunto de datos por permutación aleatoria de los datos originales, y a continuación ir generando, también por permutaciones aleatorias, nuevos conjuntos de datos, siempre a partir del último conjunto de datos generado.

Cuando se realizan pruebas de aleatorización aproximadas, es interesante conocer entre qué valores oscilarán las estimaciones realizadas por aleatorización aproximada para un determinado nivel de significación p obtenido mediante la realización de todas las posibles permutaciones (test de aleatorización exacto), y un determinado número de permutaciones *NP* (Manly, 1991, p. 34). El nivel de significación estimado se distribuirá aproximadamente según una ley normal alrededor de p con variancia $p(1-p)/N$ cuando del número de permutaciones aleatorias *NP* sea grande:

$$p \pm z_{(1-\alpha/2)}\sqrt{\frac{p(1-p)}{NP}}$$

Aplicando la fórmula anterior, se obtiene que serán necesarias unas 5000 permutaciones aleatorias de los datos para que el intervalo del 99% de las significaciones alrededor de un valor $p = 0.05$ oscile entre 0.042 y 0.058. Con 10000 permutaciones aleatorias y un valor $p = 0.01$, el intervalo que se obtiene es [0.007 - 0.013].

Por otro lado, el número de permutaciones posibles (NP+1) de los datos se determina con base en las fórmulas de combinatoria presentadas en la *Tabla 1*, donde n simboliza el número de casos de la muestra, y n_i el número de efectivos en cada uno de los k niveles de la variable independiente (k simboliza también el número de variables en el diseño de datos apareados):

TABLA 1. Número de permutaciones en función del diseño de la investigación.

Tipo de diseño	**Nº de permutaciones posibles (NP+1)**
Datos independientes	$\frac{n!}{n_1!\ n_2!\ \ldots\ n_k!}$
Datos apareados	$(k!)^n$ ó $n!$

Supongamos a continuación la siguiente matriz con los datos de 5 casos para un diseño con 2 grupos independientes A_1 y A_2, con $n_1=2$ y $n_2=3$ respectivamente, y una variable B registrada en dos momentos distintos B_1 y B_2:

		Variable B	
Caso	**Variable A**	B_1	B_2
1 2 3 4 5	A_1	2 4	4 8
	A_2	7 8 9	14 16 18

A partir de la matriz de datos anterior se pueden plantear, entre otras, las siguientes hipótesis:

Hipótesis 1. ¿Existen diferencias entre las respuestas a B_1 de los dos grupos de sujetos A_1 y A_2?.
H_0: $A_1 = A_2$

Hipótesis 2. ¿Existen diferencias entre las respuestas a B_1 y las respuestas a B_2?
H_0: $B_1 = B_2$

Hipótesis 3. ¿Las respuestas a B_1 están relacionadas con las respuestas a B_2?
H_0: $\rho_{12} = 0$ (B_1 independiente de B_2)

El número total de permutaciones de los datos originales que se deben realizar para contrastar cada una de las tres hipótesis planteadas es el siguiente:

Hipótesis 1. Debido a que la hipótesis implica la comparación de los dos grupos independientes A_1 y A_2, el número total de permutaciones diferentes viene dado por la fórmula de las *permutaciones con repetición*:

$$NP = \frac{n!}{n_{A_1}!\, n_{A_2}!} = \frac{5!}{2!\ 3!} = 10$$

Hipótesis 2. Debido a que la hipótesis implica la comparación de las respuestas apareadas B_1 y B_2, el número total de permutaciones

diferentes viene dado por las $k!$ *permutaciones sin repetición* de las respuestas para cada sujeto asociadas con las $k!$ posibles permutaciones para cada uno de los n−1 sujetos restantes:

$$NP = (k_B!)^n = (2!)^5 = 32$$

Hipótesis 3. Debido a que la hipótesis implica la relación entre las respuestas apareadas B_1 y B_2, el número total de permutaciones diferentes viene dado por las *n! permutaciones sin repetición*, que representan todas las formas en que cada uno de los n datos B_1 pueden asociarse con los n datos B_2:

$$NP = n! = 5! = 120$$

Formalmente, todas las permutaciones de una variable respecto a las otras son equiprobables y, consecuentemente, los datos representan muestras de tamaño uno de la población de posibles permutaciones. Como sucede en el muestreo Monte Carlo, una prueba de contraste de hipótesis basada en el grado de significación obtenido a partir del cociente:

$$p = \frac{(NSIG + 1)}{(NP + 1)}$$

es válida, dado que la probabilidad de rechazar la hipótesis nula cuando es verdadera no es mayor que el nivel de significación α fijado "a priori" -ver apartados 2.9 y 2.10- (Dwass, 1957).

2.3.3. Consideraciones sobre la aplicación de las permutaciones

La aplicación de las permutaciones para generar nuevos conjuntos de datos requiere que, previamente, se consideren dos aspectos fundamentales:

1. Cuál es la *naturaleza de la hipótesis nula* planteada, es decir, qué variables y/o niveles de las mismas están implicadas en la hipótesis.
2. Cuáles han sido la *unidades de asignación*, es decir, cuáles han sido las unidades que se han asignado aleatoriamente a cada nivel de las variables independientes (tratamientos, momento de aplicación de cada tratamiento, etc.).

La idea importante aquí es que en una prueba de aleatorización las permutaciones se aplican sobre las unidades de asignación, no sobre los sujetos (generalmente denominados *unidades de muestreo*, las cuales, como ya se ha señalado en el apartado anterior, no son relevantes para la aplicación de una prueba de aleatorización). Así, por ejemplo, en un experimento clásico realizado bajo un diseño de datos independientes, el investigador asigna aleatoriamente los sujetos a los diferentes tratamientos. En este caso, cada sujeto (unidad de muestreo) es una unidad de asignación.

Sin embargo, en los diseños de medidas repetidas, aunque puede parecer que los sujetos se asignan a todos los tratamientos, en realidad, la unidad de asignación es el momento en que se aplica cada tratamiento a cada sujeto. Por tanto, conviene determinar precisamente cuáles son las unidades de asignación (especialmente en el caso de diseños complejos), para así garantizar la correcta aplicación de las permutaciones.

Como ya se ha destacado anteriormente, las pruebas de aleatorización pueden aplicarse para evaluar el grado de significación de cualquier contraste de hipótesis. En esta línea, y considerando a los diseños de caso único como un caso particular de los diseños de medidas repetidas, las pruebas de aleatorización también permiten obtener el grado de significación a partir de las diferentes observaciones realizadas en un solo sujeto (Edgington, 1980).

En las siguientes páginas se presenta un conjunto de tablas que ejemplifican la aplicación de las permutaciones con base en la naturaleza de la hipótesis nula y el tipo de diseño de la investigación, para el cálculo del estadístico de contraste *F* en un ANOVA. En estas tablas los diseños sólo incluyen una variable dependiente (Y).

TABLA 2. Aplicación de las permutaciones para el cálculo de la F en un ANOVA unifactorial con un diseño de datos independientes.

Datos Independientes (1 Var. Indep.)		
A / Y: A_1 : : : ; A_2 : : :	Hipótesis nula	H_0: $\theta_{A1} = \theta_{A2}$
	Unidad de Asignación	Sujetos
	Valores a permutar	Variable Y
	Nº permutaciones posibles	$\frac{n!}{n_{A_1}!\ n_{A_2}!}$

TABLA 3. Aplicación de las permutaciones para el cálculo de la F en un ANOVA unifactorial con un diseño de datos apareados.

Datos Apareados (1 Var. Indep.)		
A_1 | A_2	Hipótesis nula	H_0: $\theta_{A1} = \theta_{A2}$
	Unidad de Asignación	Tratamiento para cada sujeto
	Valores a permutar	Valores A_1 y A_2 para cada sujeto
	Nº permutaciones posibles	$(k!)^n$

TABLA 4. Aplicación de las permutaciones para el cálculo de la F en un ANOVA de dos factores con un diseño de datos independientes

<table>
<tr>
<td rowspan="4">Datos Independientes (2 Var. Indep.)

A B Y
A_1 B_1 : : :
B_2 : : :
A_2 B_1 : : :
B_2 : : :</td>
<td>Hipótesis nula</td>
<td>Efectos principales de A:
H_0: $\theta_{A1} = \theta_{A2}$
Interacción AxB:
H_0: $\theta_{AB11}+\theta_{AB12} = \theta_{AB21}+\theta_{AB22}$
Efectos simples B en A_1:
H_0: $\theta_{AB11} = \theta_{AB12}$</td>
</tr>
<tr>
<td>Unidad de Asignación</td>
<td>Sujetos</td>
</tr>
<tr>
<td>Valores a permutar</td>
<td>Variable Y</td>
</tr>
<tr>
<td>Nº permutaciones posibles</td>
<td>Efectos principales de A:
$\frac{n!}{n_{A_1}!\ n_{A_2}!}$
Interacción AxB:
$\frac{n!}{n_{AB_{11}}!\ n_{AB_{12}}!\ n_{AB_{21}}!\ n_{AB_{22}}!}$
Efectos simples B en A_1:
$\frac{n_{A_1}!}{n_{AB_{11}}!\ n_{AB_{12}}!}$</td>
</tr>
</table>

TABLA 5. Aplicación de las permutaciones para el cálculo de la F en un ANOVA de dos factores con un diseño de datos apareados.

Datos Apareados (2 Var. Indep.) A_1 | A_2 B_1 | B_2 | B_1 | B_2 : | : | : | : : | : | : | : : | : | : | : : | : | : | :	Hipótesis nula	Efectos principales de A: H_0: $\theta_{A1} = \theta_{A2}$ Interacción AxB: H_0: $\theta_{AB11}+\theta_{AB12} = \theta_{AB21}+\theta_{AB22}$ Efectos simples B en A_1: H_0: $\theta_{AB11} = \theta_{AB12}$
	Unidad de Asignación	Sujeto-tratamiento
	Valores a permutar	Efectos principales de A: Valores $(AB_{11}+AB_{12})/2$ y $(AB_{21}+AB_{22})/2$ para cada sujeto Interacción AxB: Valores AB_{11}, AB_{12}, AB_{21} y AB_{22} para cada sujeto Efectos simples B en A_1: Valores AB_{11} y AB_{12} para cada sujeto
	Nº permutaciones posibles	Efectos principales de A: $(K_A!)^n$ Interacción AxB: $(k_A!\ k_B!)^n$ Efectos simples B en A_1: $(k_B!)^n$

En el caso de un diseño multivariable de datos independientes, la unidad de asignación son los sujetos y, por lo tanto, se permutarán los vectores fila completos de la matriz Y, manteniendo constantes sus valores (es decir, sin deshacer el caso). En el caso de un diseño multivariable de medidas repetidas, la unidad de asignación son las variables Y_i, y la permutación se realizará entre los valores de cada vector fila de la matriz Y.

Edgington (1980) y Manly (1991) presentan una revisión de las permutaciones que deben aplicarse para otros tipos de pruebas estadísticas (correlación, regresión, series temporales, etc.) y otros tipos de diseños (mixtos, diseños de caso único, etc.).

2.3.4. Aspectos computacionales

Lógicamente, a nivel computacional, el aspecto fundamental para la aplicación de las pruebas de aleatorización son los generadores de permutaciones (sistemáticas o aleatorias). No se profundizará aquí sobre los fundamentos y los métodos de generación de este tipo de secuencias numéricas. Se pueden encontrar subrutinas desarrolladas en varios lenguajes de programación en librerías tales como NAG, IMSL o Numerical Recipes. En el capítulo 3 se presentan los algoritmos de generación de permutaciones aleatorias implementados en el instrumento informático que se desarrolla en la tesis.

Un tema de investigación relevante en todas las técnicas de computación intensiva es el que se encamina a la búsqueda de algoritmos y procedimientos que permiten optimizar el tiempo y la calidad de la propia computación. En este sentido, a continuación se describen dos estrategias que permiten reducir el tiempo de computación de las pruebas de aleatorización, así como dos métodos para comprobar la calidad de las permutaciones que se realizan.

La primera estrategia que permite reducir el tiempo de computación necesario para llevar a cabo una prueba de aleatorización se basa en el concepto de *pruebas equivalentes*. Edgington (1980, p.44) define estas pruebas del siguiente modo:

> *«Dos pruebas estadísticas que conducen al mismo grado de significación en una prueba de aleatorización se denominan "pruebas estadísticas equivalentes". (...) Dos pruebas estadísticas son equivalentes si y sólo si están perfecta y monótonamente correlacionadas para todas las posibles permutaciones de los datos».*

Es decir, podemos reducir el tiempo total necesario para evaluar el grado de significación implementando pruebas estadísticas equivalentes a las pruebas clásicas, pero más simples en cuanto a su cálculo, puesto que pequeñas diferencias, como la multiplicación por una constante, pueden llegar a ser importantes cuando el estadístico utilizado tiene que calcularse miles de veces.

Cabe destacar que las pruebas estadísticas equivalentes sólo afectan al cálculo, no a la interpretación y, por tanto, no es necesario especificar la utilización de las mismas en la interpretación del resultado. Esta idea puede generalizarse válidamente para otras técnicas de remuestreo o de computación intensiva.

Asimismo, esta estrategia presenta una ventaja adicional que consiste en permitir al investigador ensayar una prueba que esté más acorde con la hipótesis que realmente desea comprobar y que, simultáneamente, le proporcione un mayor grado de comprensión en el momento de interpretar los resultados.

A continuación, y a modo de ejemplo, se presentan las pruebas equivalentes para evaluar el grado de significación de la prueba F en un ANOVA con un factor y de la prueba de correlación r unilateral entre dos variables (Edgington, 1980):

$$\sum_{k=1}^{k}\left(\frac{T_k^2}{n_k}\right) \quad \textit{prueba } F \textit{ equivalente a } \quad \frac{\left(\sum_{k=1}^{k}\left(\frac{T_k^2}{n_k}\right)-\frac{T_G^2}{N}\right)}{(k-1)} \Bigg/ \frac{\left(\sum_{i=1}^{N}x_i^2-\sum_{k=1}^{k}\left(\frac{T_k^2}{n_k}\right)\right)}{(N-k)}$$

$$\sum_{i=1}^{n}x_iy_i \quad \textit{prueba } r \textit{ equivalente a } \quad \frac{\sum_{i=1}^{n}x_iy_i-\frac{\left(\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i\right)}{n}}{\sqrt{\left(\sum_{i=1}^{n}x_i^2-\frac{\left(\sum_{i=1}^{n}x_i\right)^2}{n}\right)\left(\sum_{i=1}^{n}y_i^2-\frac{\left(\sum_{i=1}^{n}y_i\right)^2}{n}\right)}}$$

La segunda estrategia para reducir el tiempo de computación sólo se puede utilizar en el caso de que se aplique una prueba de aleatorización exacta para un diseño de grupos independientes equilibrado en cuanto al número de sujetos de cada grupo. Esta estrategia consiste en eliminar todas aquellas permutaciones de los datos que sean redundantes, es decir, que ofrecen el mismo valor del índice estadístico. En general, se demuestra que para k tratamientos aplicados al mismo número de sujetos, sólo una $1/k!$ parte de todas las permutaciones posibles son necesarias para determinar el grado de significación exacto en pruebas bilaterales (Edgington, 1980). Por otro lado, la presencia de grupos equilibrados presenta la ventaja de proporcionar un mayor número de permutaciones posibles de los mismos, con lo cual aumenta la potencia estadística de la prueba de aleatorización.

Es importante comprobar la calidad computacional cuando se aplican estas pruebas. En el caso de que se realicen permutaciones sistemáticas de los datos, para comprobar si la generación de las permutaciones es correcta basta con calcular la probabilidad teórica de obtener el máximo valor posible para la prueba estadística aplicada, y verificar si esta probabilidad es la que aparece una vez realizadas todas las permutaciones posibles.

En el caso en que se esté aplicando una prueba de aleatorización aproximada, no puede realizarse una estimación exacta de la probabilidad en que aparecerá un cierto valor de la prueba estadística, pero puede evaluarse esta probabilidad mediante la construcción de intervalos de probabilidad que son función únicamente del número de permutaciones aleatorias que se realizan NP, y de la probabilidad teórica p -que puede obtenerse mediante una prueba de aleatorización exacta- en que debería aparecer el valor de la prueba estadística aplicada (Edgington, 1969b):

$$\left[\frac{NP \times p + \left(z_{(1-\alpha/2)} \times \sqrt{(NP \times p \times (1-p))} \right) + 1}{NP + 1} , \right.$$

$$\left. \frac{NP \times p - \left(z_{(1-\alpha/2)} \times \sqrt{(NP \times p \times (1-p))} \right) + 1}{NP + 1} \right]$$

2.4. *JACKKNIFE*

Como ya se ha comentado en la introducción, el *jackknife* (literalmente, "navaja de usos múltiples") es una técnica ideada por Maurice Quenouille (1949, 1956) y perfeccionada por John W. Tukey (1958), quien la denominó con este nombre por su carácter de aplicabilidad general para la estimación del sesgo y del error estándar de un estadístico.

Este método consiste básicamente en, definida una muestra de observaciones de tamaño n, suprimir cada vez un conjunto de observaciones g y calcular sobre el conjunto restante de datos ($n = g$) el estadístico de interés. La aplicación más generalizada se basa, empero, en excluir cada vez una única observación, debido a que, como indica Miller (1974), por un lado evita la arbitrariedad en la formación de los subgrupos y, por otro lado, parece haberse mostrado como la mejor forma de aplicación del *jackknife* para cualquier problema. En este trabajo nos referiremos a esta modalidad en que, a partir de una muestra de tamaño n, se generan n grupos con $n-1$ observaciones cada uno.

Una vez obtenidas las n estimaciones del estadístico para cada una de las submuestras *jackknife*, se puede calcular una estimación del sesgo asociado a la muestra original, así como una estimación no paramétrica del error estándar del estadístico. Dicho cálculo se efectúa a partir de las submuestras *jackknife* y de la estimación realizada con base en los n datos de la muestra original.

A pesar de que esta técnica también ha sido aplicada en la estimación por intervalo de parámetros, en modelos de correlación canónica, de análisis discriminante, etc. (Gray y Schucany, 1972; Miller, 1974; Parr y Schucany, 1980; Hinkley, 1983), se expone aquí únicamente su aplicación a la estimación del sesgo y del error estándar, dado que estas aplicaciones pueden combinarse en ciertas situaciones con las estimaciones *bootstrap* que se revisarán en el apartado 2.6.

2.4.1. Estimación *jackknife* del sesgo

Sea $Y_1, \ldots, Y_n$ un conjunto de n variables aleatorias i.i.d. Sea $\hat{\theta}_o$ un estimador del parámetro θ, basado en la muestra de tamaño n; y sea $\hat{\theta}_{-i}$ el estimador correspondiente a la submuestra de tamaño $n-1$ de la cual se ha excluido el *i-ésimo* dato. Se define el estadístico:

$$\hat{\theta}_i^* = n\hat{\theta}_o - (n-1)\hat{\theta}_{-i} \qquad (i = 1,\ldots,n)$$

El estimador *jackknife* del parámetro θ queda definido por:

$$\hat{\theta}^* = \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_i^* = n\hat{\theta}_o - (n-1)\frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{-i}$$

Teniendo en cuenta que el sesgo de un estimador $\hat{\theta}_o$ se define como:

$$SESGO(\hat{\theta}_o) = E(\hat{\theta}_o) - \theta$$

se demuestra (Quenouille, 1956), que el estimador *jackknife* tiene la propiedad de eliminar el sesgo correspondiente al término de orden $1/n$, en la expresión de la esperanza matemática y de la variancia del estadístico desarrollada en las siguientes expansiones (Efron, 1982a; Hinkley, 1983):

$$E(\hat{\theta}) = \theta + \frac{a_1(\theta)}{n} + \frac{a_2(\theta)}{n^2} + \ldots$$

$$var(\hat{\theta}) = \frac{\sigma_1^2(\theta)}{n} + \frac{\sigma_2^2(\theta)}{n^2} + \ldots$$

La corrección *jackknife* del sesgo permite establecer que:

$$\theta^* = \hat{\theta}_o - SESGO(\hat{\theta}_o)$$

y que la estimación *jackknife* del sesgo de $\hat{\theta}_o$ es:

$$SESGO(\hat{\theta}_o) = (n-1)\left(\frac{1}{n}\sum_{i=1}^{n}(\hat{\theta}_i - \hat{\theta}_o)\right)$$

Schucany, Gray y Owen (1971), Gray, Watkins y Adams (1972) y Bissell (1977) describen métodos más refinados y complejos para la estimación jackknife del sesgo en estadísticos de orden y en situaciones especiales.

2.4.2. Estimación *jackknife* del error estándar

Aunque Quenouille y Tukey desarrollan inicialmente la técnica *jackknife* para la estimación del sesgo, posteriormente se generaliza para la estimación del

error estándar. Tal como demuestran Efron y Gong (1983), la estimación *jackknife* del error estándar viene dada por:

$$\hat{\sigma}_J = \sqrt{\frac{(n-1)}{n} \sum_{i=1}^{n} \left(\hat{\theta}_{-i} - \left(\frac{1}{n} \sum_{i=1}^{n} \hat{\theta}_{-i} \right) \right)^2}$$

Esta expresión presenta la ventaja de que es aplicable para estimar el error estándar de cualquier estadístico. Coincide con el error estándar de la media:

$$EE(\hat{\mu}) = \sqrt{\frac{\sigma^2}{n}}$$

cuando $\hat{\theta}_o = \hat{\mu}$.

2.4.3. Algoritmo general

De acuerdo con lo expuesto en los dos apartados anteriores, el algoritmo general de la estimación *jakknife* del sesgo y el error estándar de $\hat{\theta}$ es:

1. Definir el estadístico de interés θ
2. Calcular $\hat{\theta}_o$ para la muestra de datos original X_o
3. Para i = 1 hasta *n* (*n* es el tamaño de la muestra X_o):
   3.1. $X_{-i} = X_o - x_i$ (exclusión del i-ésimo dato)
   3.2. Calcular el estimador $\hat{\theta}_{-i}$ para la submuestra X_{-i}
4. Calcular la media de $\hat{\theta}_{-i}$:
   $\bar{\theta}^* = SUM(\hat{\theta}_{-i})/n$
5. Estimación *jackknife* del parámetro θ:
   $\hat{\theta}^* = n\hat{\theta}_o - (n-1)\bar{\theta}^*$
6. Estimación *jackknife* del sesgo de $\hat{\theta}$:
   $SESGO(\hat{\theta}_o) = (n-1)\, SUM(\hat{\theta}_{-i} - \hat{\theta}_o)/n$
7. Estimación *jackknife* del error estándar de $\hat{\theta}$:
   $\hat{\sigma}_J = SQRT((n-1)\, SUM((\hat{\theta}_{-i} - \bar{\theta}^*)^2)/n)$

ALGORITMO 5. Estimación jackknife del sesgo y del error estándar de un estadístico.

2.5. VALIDACION RECIPROCA

Según Efron (1982a), la *validación recíproca* ("*cross validation*") parte de una idea antigua (Larson, 1931) que ha sido retomada actualmente gracias al advenimiento de las nuevas generaciones de ordenadores. Su principal aplicación es la estimación de la probabilidad de error de predicción para aquellas técnicas estadísticas que permiten predecir valores de variables aleatorias. Otra aplicación frecuente de la validación recíproca es la detección de observaciones con valores extremos (*outliers*) (Sprent, 1989).

La técnica de validación recíproca se basa en la división aleatoria de los datos muestrales en dos grupos. Aunque puede realizarse cualquier tipo de división de los datos, se han generalizado los dos tipos de divisiones siguientes:

a) *División en dos mitades.* En este caso, se utiliza la primera mitad de los datos para ajustar y estimar los parámetros del modelo de predicción (por ejemplo, regresión múltiple).

   Seguidamente, el procedimiento consiste en estimar la probabilidad de error de predicción a partir de la aplicación de la ecuación predictiva, obtenida en el primer paso para cada observación de la segunda mitad de la muestra (Efron 1979a, 1982a).

b) *División en n−1 y 1* (siendo *n* el tamaño muestral). En opinión de Efron (1982a), este tipo de división es la más frecuente. Como en la división en dos mitades, en este caso se utilizan los *n−1* datos para estimar los parámetros del modelo, y posteriormente se aplica la ecuación obtenida para probar si predice correctamente el i-ésimo que había sido excluido de la estimación de la ecuación.

   Comparativamente, esta forma de validación recíproca se asemeja al *jackknife* únicamente en el sentido de que se omite un dato cada vez. Sin embargo, Efron (1982a) señala que en este caso no se calcula un estimador especial y, por otro lado, las posibles conexiones propuestas entre ambas técnicas han sido rechazadas en la literatura (Sprent, 1989).

2.5.1. Estimación de la probabilidad de error de predicción

En la exposición de los conceptos básicos de la técnica de validación cruzada para la estimación de la probabilidad error de predicción, tomaremos como referente su aplicación en un problema de regresión (Efron, 1982a, 1983).

En un modelo de regresión múltiple cada caso consiste en parejas de vectores $x_i = (t_i, y_i)$ que son realizaciones de un vector aleatorio $X = (T, Y)$ con distribución F en $\mathbb{R}^{p+1}$, donde T es un vector de variables aleatorias explicativas de dimensión p e Y es la expresión vectorial de una variable aleatoria dependiente. Formalmente:

$$X_1, X_2, \ldots, X_n \overset{iid}{\sim} F$$

Habiendo observado $X = x$, un método para ajustar una regla de predicción que se pueda usar para predecir futuros valores de la variable dependiente sería:

$$\text{valor predicho de } y_o = \eta(t_o)$$

donde el subíndice "o" indica un nuevo punto-dato $x_o = (t_o, y_o)$ distinto de los valores x_i originales, y η es la función de predicción.

Por ejemplo, en un modelo de regresión lineal la regla de predicción es:

$$\eta_x(t_o) = t_o\hat{\beta}$$

donde:

$$\hat{\beta} = (t' t)^{-1} t' y$$

Si se define un indicador de error $Q(y, \eta)$ tal que otorgue la siguiente regla:

$$Q(y,\eta) = \begin{cases} 0 & si \quad y=\eta \\ 1 & si \quad y\neq\eta \end{cases}$$

en este caso, puede definirse:

$$\Sigma Q(y_i, \eta_i)$$

como el número de errores de predicción, y la *probabilidad aparente de error* $\overline{err}$:

$$\overline{err} = \hat{E}\{Q[Y_o, \eta(T_o; x)]\} = \frac{1}{n}\sum_{i=1}^{n} Q[y_i, \eta(t_i; x)]$$

como un estimador de la verdadera probabilidad de error de predicción *err*, donde se otorga masa $1/n$ a cada observación x_i respecto a la distribución empírica $\hat{F}_n$.

Siguiendo a Efron (1982a), esta estimación es optimista, dado que es menor que la verdadera probabilidad de error *err*. Por ello es interesante definir la *esperanza del exceso de error* o *sesgo de err* como la variable aleatoria:

$$R(X, F) = err - \overline{err} = E\{Q[Y_o; \eta(T_o)]\} - \hat{E}\{Q[Y_o; \eta(T_o)]\}$$

La estimación de la probabilidad de error que se obtiene mediante validación recíproca viene dada por:

$$err_{VR} = \frac{1}{n} \sum_{i=1}^{n} Q[y_i, \eta(t_i, x_{-i})]$$

donde x_{-i} indica que ha sido suprimido el i-ésimo dato.

Es decir err_{VR} es la estimación del verdadero error para los datos observados sin que el i-ésimo dato participe de la estimación de la regla de predicción para él mismo.

Como señala Efron (1982a), err_{VR} es un estimador con menos sesgo que *err*, y la estimación de la *esperanza del exceso de error* o *sesgo de err* vale:

$$\omega_{VR} = R(X, F) = err_{VC} - \overline{err}$$

Stone (1974, 1977) señala que la aplicación de validación cruzada en la regresión es, en ocasiones, peligrosa, debido a la desproporcionada influencia que pueden tener los datos extremos. Bailey et al. (1984) apuntan también que, en diseños experimentales altamente estructurados, la eliminación de una observación puede destruir esta estructura, incrementándose el esfuerzo computacional.

2.5.2. Algoritmo general

El algoritmo general de la estimación de la probabilidad de error de predicción aplicando la técnica de validación recíproca es el siguiente:

1. Definir el modelo de predicción $y = \eta[\, t(x) \,]$
2. Definir el indicador de error $Q(y, \eta)$
3. Inicializar el contador $NE = 0$
4. Estimar la probabilidad aparente de error:
   - 4.1. Estimar los parámetros del modelo de predicción a partir de los n datos de la muestra original X_o
   - 4.2. Para $i = 1$ hasta n:
     - a. Aplicar el indicador de error para el sujeto i:
       $$NE = NE + Q\{y_i, \eta[\, t(x_i) \,]\}$$
   - 4.3. Estimar la probabilidad aparente de error $\overline{err}$:
     $$\overline{err} = NE / n$$
5. Inicializar el contador $NE = 0$
6. Para $i = 1$ hasta n:
   - 6.1. $X_i = X_o - x_i$ (exclusión del i-ésimo dato)
   - 6.2. Estimar los parámetros del modelo de predicción a partir de los $(n - 1)$ datos de la muestra X_i
   - 6.3. Aplicar el indicador de error para la predicción del sujeto i excluido de X_o:
     $$NE = NE + Q\{y_i, \eta[\, t(x_i) \,]\}$$
7. Obtener la estimación de la probabilidad de error de predicción:
   $$err_{VR} = NE / n$$
8. Obtener la estimación del sesgo del error de predicción:
   $$\omega_{VR} = err_{VR} - \overline{err}$$

ALGORITMO 6. Estimación por validación recíproca del error de predicción y del sesgo del mismo

2.6. REMUESTREO *BOOTSTRAP*

En el año 1979, y a partir de un profundo y reflexivo análisis sobre el origen y evolución de los métodos estadísticos, tanto a nivel teórico como aplicado, Bradley Efron desarrolla la técnica *bootstrap*. El término *bootstrap*, expresión inglesa que significa "levantarse tirando hacia arriba de las propias correas de las botas", refleja el aspecto fundamental de esta técnica, su autosuficiencia.

Concretamente, la idea básica sugerida por Efron (1979b) es la siguiente. Si una muestra aleatoria contiene la máxima información disponible sobre la población, ¿por qué no proceder como si la muestra fuese la población y, entonces, estimar la distribución muestral de un estadístico generando nuevas muestras mediante un muestreo aleatorio con reposición, a partir de los datos de la muestra original?. Esta técnica, también denominada "remuestreo *bootstrap*" por la utilización de la muestra como mecanismo generador de nuevas muestras (Noreen, 1989), permite ampliar el uso del muestreo Monte Carlo en la investigación aplicada, al no precisar el conocimiento de los parámetros poblacionales, ni realizar supuestos sobre éstos.

Inicialmente Efron describe el *bootstrap* como una técnica útil para evaluar el error estadístico (error estándar, sesgo y probabilidad de error de predicción) y para realizar estimaciones de parámetros, no obstante, rápidamente se describen diversas adaptaciones y desarrollos para obtener el grado de significación en las pruebas de hipótesis (Lunneborg, 1985, 1987; Noreen, 1989; Holms, 1990; Westfall y Young, 1993).

Uno de los atractivos más importantes del *bootstrap* es que para su uso, el usuario sólo debe ser consciente de *que funciona* y saber cómo llevar a cabo el remuestreo de una forma adecuada a su problema de estimación, sin necesidad de efectuar sofisticados cálculos matemáticos.

2.6.1. Estimación *bootstrap* del error estadístico y de parámetros poblacionales

En los apartados siguientes se exponen las técnicas clásicas de estimación *bootstrap* introducidas por Efron.

a. Bootstrap no paramétrico

La estimación *bootstrap* se basa en el supuesto de que la función de distribución empírica $\hat{F}_n$, obtenida a partir de una muestra correspondiente a extracciones aleatorias $\{ x_1, x_2, \ldots, x_n \}$ de n variables aleatorias independientes X_i con idéntica función de distribución F:

$$X_1, X_2, \ldots, X_n \overset{iid}{\sim} F$$

es la estimación máximo verosímil no paramétrica de F, fundamentada en la asignación de probabilidad $1/n$ a todos y cada uno de los datos muestrales (Efron 1979b, 1982a). En este sentido, Bickel y Freedman (1981), en su estudio sobre las propiedades asintóticas del *bootstrap* demuestran el postulado anterior a partir del teorema de Glivenko-Cantelli, principio que permite establecer una convergencia casi segura, cuando n→∞, entre las distribuciones F y $\hat{F}_n$,

$$|F - \hat{F}_n| \rightarrow 0 \text{ c.s.}$$

Consecuentemente, se puede considerar que $F = \hat{F}_n$, proposición que sustenta al *bootstrap* no paramétrico.

Siguiendo a Sánchez (1990a), la forma habitual de estudiar la distribución muestral (asintótica) de un estadístico se realiza en dos etapas:

1. Encontrar una aproximación (*large sample approximation*) para la distribución muestral cuando $n \rightarrow \infty$.
2. Sustituir F por $\hat{F}_n$ en la distribución asintótica.

El problema que presenta esta aproximación reside en la complejidad de la distribución asintótica. El *bootstrap* soluciona este problema al sustituir F por $\hat{F}_n$ antes de hallar la distribución muestral, la cual es estimada a través del repetido muestreo con reposición a partir de los datos de la muestra original.

Considerando lo anterior, el algoritmo general de la estimación *bootstrap* no paramétrica se basará en la extracción aleatoria y con reposición de un conjunto de muestras X_b^* partir de los datos contenidos en la muestra original X_o, calculando el estadístico o estadísticos $\hat{\theta}_b^*$ de interés para cada una de las muestras *bootstrap* (el supraíndice asterisco "*" indica que se trata de una muestra extraída a partir de X_o, y el subíndice "b" indica que se trata de la b muestra extraída de X_o).

Se dispone de múltiples demostraciones de la consistencia y exactitud asintótica del *bootstrap* (Singh, 1981; Babu y Singh, 1984; Bickel y Freedman, 1981; Sánchez, 1990a; Hall, 1992). Sin embargo, algunos de estos autores señalan diversos casos de inconsistencia de las estimaciones *bootstrap*.

1. Definir el estadístico de interés θ
2. Calcular $\hat{\theta}_o$ para la muestra de datos original X_o
3. Fijar el número B de remuestras a efectuar
4. Para b = 1 hasta B:
   - 4.1. Para i = 1 hasta n (tamaño de la muestra):
     - a) x_i = elemento con posición en X_o = UNIFORM(1,n)
     - b) Añadir x_i a X_b^* (muestra *bootstrap*)
   - 4.2. Calcular el estadístico $\hat{\theta}_b^*$ para los datos de X_b^*
5. Calcular la estimación *bootstrap* del parámetro: $\hat{\theta}^{*} = \sum\hat{\theta}_b^*/B$

ALGORITMO 7. Algoritmo general del bootstrap no paramétrico.

b. Bootstrap paramétrico

En el caso de que la función de distribución F_θ a partir de la cual se han extraído los datos sea conocida excepto en su parámetro θ, puede recurrirse a la estimación de este parámetro considerando que $\hat{\theta}$ es una buena estimación de θ obtenida a partir de los datos muestrales y, por tanto, que
$F_\theta \approx F_{\hat{\theta}}$.

En este caso, en lugar de extraer muestras aleatorias a partir de la muestra original X_o, pueden extraerse directamente por muestreo Monte Carlo con base en el modelo de distribución $F_{\hat{\theta}}$. Este proceso se corresponde con el denominado *bootstrap* paramétrico.

El algoritmo general del *bootstrap* paramétrico es igual al del *bootstrap* no paramétrico, excepto en el paso en que se extrae la muestra *bootstrap* X_b^*:

1. Definir el estadístico de interés θ
2. Calcular $\hat{\theta}_o$ para la muestra de datos original X_o
3. Fijar el número B de remuestras a efectuar
4. Para b = 1 hasta B:
   4.1. Extraer una muestra *bootstrap* X_b^* a partir del modelo de distribución $F_{\hat{\theta}}$: $X_b^* = \text{DISTRND}(n, \hat{\theta}_o)$
   4.2. Calcular el estadístico $\hat{\theta}_b^*$ para los datos de X_b^*
5. Calcular la estimación *bootstrap* del parámetro: $\hat{\theta}^{*\cdot} = \sum \hat{\theta}_b^* / B$

ALGORITMO 8. *Algoritmo general del bootstrap paramétrico.*

c. Bootstrap suavizado (smoothed bootstrap)

Una solución intermedia entre el *bootstrap* no paramétrico y el *bootstrap* paramétrico es el *bootstrap suavizado*. La idea consiste en sustituir la distribución empírica $F_{\hat{\theta}}$ por otra distribución $F^s_{\hat{\theta}}$ previamente suavizada. El proceso de suavizado puede realizarse mediante la "perturbación" (término utilizado por Holmes, 1990) de cada observación de la muestra original añadiéndole una variable aleatoria con función de distribución suave, como por ejemplo, la normal o la uniforme (Silverman y Young, 1987).

Siguiendo a Efron (1979b), esta "perturbación" de los datos se efectuaría con base en la siguiente combinación lineal de ponderación de dos datos x^o y x^s con funciones de distribución $F_{\hat{\theta}}$ y $F^s_{\hat{\theta}}$ respectivamente:

$$X = \sqrt{(1-c^2)}\, x^o + cx^s, \qquad 0 \le c \le 1$$

correspondiendo los valores intermedios del coeficiente "c" a distintas combinaciones entre ambas funciones de distribución.

Efron (1982a) muestra un ejemplo de suavizado con base en una distribución normal, para la estimación *bootstrap* del coeficiente de correlación.

Una vez suavizadas las observaciones contenidas en la muestra original X_o, el proceso es idéntico al del *bootstrap* no paramétrico, siendo el algoritmo general del *bootstrap* suavizado el siguiente:

1. Definir el estadístico de interés θ
2. Calcular $\hat{\theta}_o$ para la muestra de datos original X_o
3. Fijar el número B de remuestras a efectuar
4. Para i = 1 hasta n (tamaño de la muestra):
   4.1. x_i = elemento con posición en X_o = UNIFORM(1,n)
   4.2. x_i^s = SUAVE(x_i)
5. Para b = 1 hasta B:
   5.1. Para i = 1 hasta n (tamaño de la muestra):
      a) x_i^s = RAN(1,n)
      b) Añadir x_i^s a X_b^* (muestra *bootstrap*)
   5.2. Calcular el estadístico $\hat{\theta}_b^*$ para los datos de X_b^*
6. Calcular la estimación *bootstrap* del parámetro: $\hat{\theta}^* = \sum \hat{\theta}_b^* / B$

ALGORITMO 9. Algoritmo general del bootstrap suavizado.

d. Estimación bootstrap del sesgo

Atendiendo a la definición de sesgo de un estimador $\hat{\theta}$ de un parámetro θ:

$$SESGO(\hat{\theta}) = E(\hat{\theta}) - \theta$$

y siendo $\hat{\theta}^*$ la estimación *bootstrap* de un parámetro, el sesgo se calcula a partir de $\hat{\theta}^* - \hat{\theta}$. En la práctica, la estimación del sesgo se aproxima a partir del promedio de las diferencias entre las estimaciones realizadas en cada muestra *bootstrap* y la estimación $\hat{\theta}$ obtenida a partir de la muestra original; esto es:

$$SESGO(\hat{\theta}) = \frac{1}{B}\sum_{b=1}^{B}(\hat{\theta}_b^* - \hat{\theta}) = \bar{\theta}^* - \hat{\theta}$$

Efron (1982a) demuestra que la estimación *bootstrap* del sesgo es asintóticamente igual a *(n−1)/n* multiplicado por la estimación *jackknife* del sesgo que, como se mostró en el apartado 2.4.1. viene dada por:

$$SESGO(\hat{\theta}) = (n-1)\left(\frac{1}{n}\sum_{i=1}^{n}(\hat{\theta}_{-i} - \hat{\theta})\right)$$

e. Estimación boostrap del error estándar

La estimación *bootstrap* del error estándar se realiza a partir de la desviación estándar de la distribución muestral generada empíricamente a partir del remuestreo Monte Carlo, constituyendo, como indica Efron (1982a), la estimación máximo verosímil no paramétrica del verdadero error estándar $\sigma(F)$:

$$\hat{\sigma}_{BOOT} = \sqrt{\frac{\sum_{b=1}^{B} (\hat{\theta}_b^* - \bar{\theta}^*)^2}{(B - 1)}}$$

siendo $\bar{\theta}^*$:

$$\bar{\theta}^* = \frac{\sum_{b=1}^{B} \hat{\theta}_b^*}{B}$$

Al igual que sucedía en el caso de la estimación *jackknife* del error estándar, la estimación *bootstrap* es independiente de $\hat{\theta}$, y los estudios teóricos demuestran que coincide con el verdadero error estándar σ cuando $B \rightarrow \infty$ (Efron 1982a; Efron y Tibshirani 1986). Efron y Tibshirani (1986) señalan que, en la mayor parte de casos, el número B de muestras *bootstrap* adecuado para la estimación del error estándar puede oscilar entre 50 y 200. Paralelamente, estos autores también remarcan que si el tamaño n de las muestras *bootstrap* difiere del tamaño de la muestra original, entonces $\hat{\sigma}_{BOOT}$ no converge en $\hat{\sigma}$.

Efron (1979b), sugiere otro método para estimar el error estándar a partir de la distribución *bootstrap* del estadístico, consistente en escoger dos valores θ^*_{inf} y θ^*_{sup} de dicha distribución correspondientes a los percentiles 16 y 84. De este modo obtenemos el intervalo $[\theta^*_{inf}, \theta^*_{sup}]$ que encierra el 68% central de los valores del estadístico obtenidos a partir de las muestras *bootstrap*. A partir de estos límites se puede definir una estimación robusta del error estándar de la siguiente forma:

$$\hat{\sigma}_{BOOT} = \frac{(\hat{\theta}^*_{sup} - \hat{\theta}^*_{inf})}{2}$$

f. Intervalos de confianza bootstrap

La construcción de intervalos de confianza para un parámetro a través de los métodos *bootstrap* requiere la participación de dos conceptos importantes, el estadístico pivote y la precisión, que seguidamente presentamos.

En general, resulta difícil obtener intervalos de confianza exactos, es decir, que recubran exactamente la región paramétrica fijada "a priori" para ellos (por ejemplo, $1-\alpha$). Esto se consigue sólo en aquellos casos en que se dispone de un *estadístico pivote.*

Un estadístico pivote es una función continua y monótona del parámetro θ y de la muestra, que presenta una distribución conocida independiente de dicho parámetro, de la muestra, y de cualquier otro parámetro desconocido. Simbolizaremos estas funciones por $g(\hat{\theta}, X)$. En esta situación, cuando puede definirse una función de este tipo, entonces, fijado el nivel $1-\alpha$ para el intervalo, siempre es posible encontrar dos valores θ_1 y θ_2 que cumplan la condición de que el $100(1-\alpha)\%$ de las veces contengan el verdadero parámetro θ. Por ejemplo, las cantidades pivote para los parámetro μ y σ^2 de una variable normal $N(\mu;\sigma^2)$ son:

$$\mu \rightarrow \frac{\bar{x}-\mu}{\sqrt{s^2/n}} \sim t_{(n-1)} \qquad \sigma^2 \rightarrow \frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{(n-1)}$$

y los intervalos $1-\alpha$ que pueden construirse sobre estos parámetros son:

$$\mu \rightarrow \left[\bar{x} - t_{(n-1;1-\alpha/2)}\, s/\sqrt{n} \;,\; \bar{x} + t_{(n-1;1-\alpha/2)}\, s/\sqrt{n} \right]$$

$$\sigma^2 \rightarrow \left[(n-1)\, s^2 / \chi^2_{(n-1;\alpha/2)} \;,\; (n-1)\, s^2 / \chi^2_{(n-1;1-\alpha/2)} \right]$$

Por otra parte, existe un modo de evaluar la *precisión* de los intervalos de confianza a partir del desarrollo asintótico de los puntos críticos o de la función de distribución de un estadístico (generalmente mediante desarrollos en series de Edgeworth; Hall, 1992).

Según Efron (1987) y Hall (1988a, 1988b, 1992), los límites superior e inferior de los intervalos de confianza, calculados en una muestra de tamaño n, pueden expresarse en la siguiente forma:

$$\hat{\theta} + \hat{\sigma}\left(z^{(\alpha)} + \frac{A_n^{(\alpha)}}{\sqrt{n}} + \frac{B_n^{(\alpha)}}{n} + \ldots \right)$$

A partir de este desarrollo puede establecerse el grado de precisión de los intervalos, en función del número de términos que coinciden. Por ejemplo, si coinciden sólo en el primer término:

$$\hat{\sigma}\left(z^{(\alpha)}\right)$$

se denominarán *precisos de primer orden* (*"first order accurate"* o también *"first order correct"*), si coinciden en los dos primeros términos:

$$\hat{\sigma}\left(z^{(\alpha)} + \frac{A_n^{(\alpha)}}{\sqrt{n}} \right)$$

se denominarán *precisos de segundo orden*, y así sucesivamente.

Estos autores también señalan que los *intervalos estándar* del tipo $\hat{\theta} \pm z_{(1-\alpha/2)}\hat{\sigma}$ son precisos de primer orden, y que el segundo orden es el que tiene mayor efecto en el caso de muestras pequeñas.

En diversas publicaciones de Efron (sintetizadas en Efron, 1987), se han desarrollado tres procedimientos para construir intervalos de confianza aproximados, con base en la estimación de la distribución muestral *bootstrap* de un estadístico, denominados *método percentil*, *método percentil con corrección del sesgo* y *método percentil con corrección acelerada del sesgo.*

Cada uno de estos procedimientos presenta un grado de generalidad mayor que el anterior y, además, el tercero de ellos aparece como preciso de segundo orden, frente a los otros dos, que ofrecen precisiones de primer orden en el sentido comentado anteriormente. Además, debe tenerse en cuenta que estos métodos pueden aplicarse siempre en situaciones *bootstrap* no paramétricas, paramétricas o suavizadas. En general, según Efron (1987) será necesario simular entre 1000 y 2000 muestras para obtener estimaciones óptimas de los límites de los intervalos.

Empezaremos por revisar las características de los intervalos de confianza estándar y el denominado *bootstrap-t*, para, seguidamente proceder a la exposición de los conceptos y procedimientos que fundamentan los tres métodos percentil de Efron.

f.1. Intervalos de confianza estándar

Se denominan así los intervalos de la forma:

$$\hat{\theta} \pm z_{1-\alpha/2}\,\hat{\sigma}$$

donde $z_{(1-\alpha/2)}$ es el valor del percentil $100(1-\alpha/2)$ de la distribución normal estandarizada *N(0;1)*, y $\hat{\sigma}$ es un estimador (como, por ejemplo, el estimador bootstrap $\hat{\sigma}^*$) del error estándar de $\hat{\theta}$. Noreen (1989, p. 69) denomina a este procedimiento "método de la aproximación normal".

Estos intervalos son de gran utilidad dado que tienen la ventaja de ser automáticos, fáciles de calcular, y requieren un número de remuestreos entre 50 y 200 para conseguir que $\hat{\sigma}^*$ aproxime el verdadero valor $\hat{\sigma}$ (Efron y Tibshirani, 1986). Contrariamente, presentan el inconveniente de que, como ya se ha comentado antes, sólo son precisos de primer orden y se basan en el supuesto de normalidad de la distribución de $\hat{\theta}$.

Para solucionar los casos en que no es lícito aceptar el supuesto de normalidad, puede recurrirse a un estadístico pivote mediante una transformación de los datos en otra escala, calcular a continuación los límites del intervalo y, finalmente, aplicar la transformación inversa para obtener estos límites en la escala original.

Pero esta solución, a pesar de que es satisfactoria, requiere conocer el estadístico pivote y además, y este es el punto más importante, despoja a los intervalos de confianza estándar de su carácter automático.

Como se verá en los siguientes apartados, la técnica *bootstrap* permite definir intervalos de confianza con la misma o mayor precisión que los intervalos estándar, y de aplicabilidad más general y automática, dado que el usuario no debe recurrir a estadísticos pivote si éstos no son conocidos.

f.2. Método percentil-t (o bootstrap estudentizado)

Este método se basa en el supuesto de que existe un estadístico pivote tal como:

$$|\hat{\theta} - \theta| / \hat{\sigma}$$

que permite construir un intervalo de confianza $1-\alpha$ con base en:

$$\hat{\theta} \pm a(F)\,\hat{\sigma}$$

siendo $a(F)$ el valor del percentil $100(1-\alpha/2)$ de la distribución muestral del pivote.

Si se desconoce θ puede realizarse una buena aproximación a partir de la distribución *bootstrap* del pivote:

$$|\hat{\theta}^* - \hat{\theta}| \,/\, \hat{\sigma}^*$$

El intervalo $\hat{\theta} \pm a(F)\,\hat{\sigma}$ que se obtiene es simétrico y por ello recibe también el nombre de "*método percentil simetrizado*".

Este método también puede aplicarse para estimar los percentiles $100(\alpha/2)$ y $100(1-\alpha/2)$, correspondiendo entonces al "*método percentil de colas iguales*". Hall (1988b) demuestra que ambos métodos ofrecen intervalos de confianza precisos de segundo y cuarto orden respectivamente y, en su opinión, el *percentil-t* representa la técnica más eficiente de todas. El *Algoritmo 10* presenta el proceso del cálculo del método percentil de colas iguales, dado que éste incluye al percentil simetrizado.

1. Definir el estadístico de interés θ
2. Calcular $\hat{\theta}_o$ y $\hat{\sigma}_o$ para la muestra de datos original X_o
3. Fijar el número B de remuestras a efectuar
4. Para b = 1 hasta B:
   - 4.1. Para i = 1 hasta n (tamaño de la muestra):
     - a) x_i = elemento con posición en X_o = UNIFORM(1,n)
     - b) Añadir x_i a X_b^* (muestra *bootstrap*)
   - 4.2. Calcular el estadístico $\hat{\theta}_b^*$ para los datos de X_b^*
   - 4.3. Calcular la desviación estándar: $\hat{\sigma}_b^* = \sigma(X_b^*)$
   - 4.4. Calcular el pivote: $a(X_b^*) = (\hat{\theta}_b^* - \hat{\theta}_o) \,/\, \hat{\sigma}_b^*$
   - 4.5. Añadir $a(X_b^*)$ a $A(X_b^*)$
5. Calcular los percentiles $q(\alpha/2)$ y $q(1-\alpha/2)$ de $A(X_b^*)$
6. Calcular el límite inferior del intervalo: $l_i = \hat{\theta}_o + q_{\alpha/2} \times \hat{\sigma}_o$
7. Calcular el límite superior del intervalo: $l_s = \hat{\theta}_o + q_{1-\alpha/2} \times \hat{\sigma}_o$

ALGORITMO 10. Método percentil-t o bootstrap estudentizado.

Sin embargo, a pesar de la gran precisión de este método, tiene el gran inconveniente de requerir un estimador estable y consistente de $\sigma(F)$. Una posibilidad cuando no se posee este estimador, como señalan Holmes (1990) y Sánchez (1990c), es utilizar en el paso 4.3. del *Algoritmo 10* el estimador *bootstrap* del error estándar (lo cual requeriría un doble nivel *bootstrap*, es decir, B^B remuestreos), o el estimador *jackknife* como solución intermedia que tiene un menor coste computacional. Tibshirani (1988b) presenta otra vía para mejorar las estimaciones del procedimiento *bootstrap-t*, con base a un algoritmo más complejo que permitiría encontrar de forma automática una transformación estabilizadora de la variancia del estimador, y requeriría menos iteraciones que el doble nivel *bootstrap* o *jackknife*.

De todos modos, Sánchez (1990c) apunta que en este caso podría subsistir el problema de la inconsistencia de estos estimadores para algunos casos, y además, este tipo de intervalos no son invariantes frente a cambios de escala, es decir, que si se construye el intervalo sobre una transformación de los datos, este intervalo puede resultar luego incorrecto al aplicar la transformación inversa.

f.3. Métodos percentil de Efron

Se revisan en este punto los tres métodos desarrollados por Efron en publicaciones sucesivas, que se encuentran sintetizadas en Efron (1987).

f.3.1. Método percentil

Este método es el que mayor repercusión ha tenido entre los investigadores, debido en gran medida a su sencillez y a su carácter intuitivo una vez asumida la idea básica del bootstrap (Hall, 1988b). El concepto fundamental del método percentil es simple: los límites inferior y superior del intervalo de confianza $1-\alpha$ serán los percentiles $100(1-\alpha/2)$ y $100(\alpha/2)$ de la distribución muestral *bootstrap* del estadístico $\hat{\theta}$.

El algoritmo de cálculo es el siguiente:

1. Definir el estadístico de interés θ
2. Fijar el nivel α para el intervalo de confianza
3. Fijar el número B de remuestras a efectuar
4. Para b = 1 hasta B:
   - 4.1. Para i = 1 hasta n (tamaño muestral):
     - a) x_i = elemento con posición en X_o = UNIFORM(1,n)
     - b) Añadir x_i a X_b^*
   - 4.2. Calcular $\hat{\theta}_b^* = \theta(X_b^*)$
   - 4.3. Añadir $\hat{\theta}_b^*$ a $\hat{\theta}^*$
5. Calcular los percentiles $q_{100(\alpha/2)}$ y $q_{100(1-\alpha/2)}$ de $\hat{\theta}^*$

ALGORITMO 11. Método percentil de Efron.

Este método presenta pues la gran ventaja de que es sencillo de ejecutar. No se requieren fórmulas analíticas complejas para estimar el parámetro $\hat{\theta}$ de la distribución muestral, ni tampoco tablas de valores de probabilidad para los puntos de una distribución muestral estandarizada. Sólo se requiere calcular los $\hat{\theta}_b^*$ (sobre las 1000 veces), ordenarlos, y recontar por encima y por debajo de los percentiles definidos.

Efron (1987) justifica la validez de este procedimiento si se cumple el supuesto de que existe una transformación monótona $g(\hat{\theta}, X)$, que no es necesario conocer, que normalice el parámetro.

Como señala este autor, los intervalos que se obtienen por este procedimiento son invariantes frente a transformaciones monótonas de los datos. A pesar de su aplicabilidad y sencillez, los intervalos obtenidos sólo son precisos de primer orden y a menudo resultan muy inexactos y sesgados, especialmente en muestras pequeñas, debido a la gran importancia de las colas de la distribución muestral en este método (DiCiccio, 1986; DiCiccio y Romano, 1988). Para solucionar estos problemas Efron introduce el *método percentil con corrección del sesgo.*

f.3.2. Método percentil con corrección del sesgo

Este método se basa en el método percentil expuesto anteriormente, pero introduciendo una corrección para el sesgo de $\hat{\theta}$. Generalmente se le denomina "método BC".

Como ya se ha indicado, la transformación monótona $g(\hat{\theta}, X)$ sobre la que se basa el método percentil presenta un sesgo de orden $1/\sqrt{n}$. Para corregir este sesgo, Efron (1982a, 1987), en lugar de suponer que las diferencias $\hat{\theta}^* - \hat{\theta}$ y $\hat{\theta} - \theta$ están centradas en el valor 0 (es decir, que $\hat{\theta}^*$ y $\hat{\theta}$ son estimadores insesgados de $\hat{\theta}$ y θ, respectivamente), asume que estas diferencias están distribuidas alrededor de una constante, $z_0\tau$, donde τ es la desviación estándar de la distribución respectiva. Esta cantidad z_0 es una constante de sesgo para la que debe ajustarse la distribución *bootstrap* de $\hat{\theta}$.

Siguiendo a Efron, para realizar este ajuste sólo es necesario asumir la existencia de ciertas transformaciones $\hat{\phi} = g(\hat{\theta})$ y $\phi = g(\theta)$, monótonas crecientes y estabilizadoras de la variancia, cuyas diferencias estén normalmente distribuidas y centradas en $z_0\tau$:

$$\frac{(\hat{\phi} - \phi)}{\tau} + z_0 \sim \mathbb{N}(0,1)$$

donde τ y z_0 son constantes para todo ϕ.

La diferencia $\hat{\phi} - \phi$ es una cantidad pivote, con la misma distribución bajo $F(\hat{\theta})$ y $\hat{F}^*(\hat{\theta}^*)$ -ver apartado 2.6.1.f-. Asumiendo la existencia de estas transformaciones normalizantes (aunque pueden utilizarse otras funciones de distribución conocidas; DiCiccio y Romano, 1988), la constante z_0 será el resultado de aplicar la función inversa de la función de distribución acumulada normal para el cuantil correspondiente al valor de $\hat{\theta}$ en la distribución *bootstrap*. Lógicamente, en el caso de que este último cuantil sea igual a 0.5, lo cual indica que no hay sesgo, entonces $z_0 = 0$.

El método percentil con corrección del sesgo tiene pues como límites $\alpha/2$ y $1-\alpha/2$ los percentiles de la distribución *bootstrap* correspondientes a los valores z:

$$2z_0 \pm z_{1-\alpha/2}$$

Como puede observarse, si $z_0 = 0$, entonces las estimaciones obtenidas por este método y el método percentil coincidirán. Al igual que en el caso del método percentil, no es necesario conocer la transformación monótona ϕ, aunque si se conociera, entonces el intervalo de confianza construido sería exacto, y en caso de no conocerse, se obtiene un intervalo preciso de primer orden (Efron, 1987; Hall, 1988a, 1988b).

El procedimiento de cálculo del método BCa es el siguiente:

1. Definir el estadístico de interés θ
2. Calcular $\hat{\theta}_o$ para la muestra original de datos X_o
3. Fijar el nivel α para el intervalo de confianza
4. Fijar el número B de remuestras a efectuar
5. Para b = 1 hasta B:
   - 5.1. Para i = 1 hasta n (tamaño muestral):
     - a) x_i = elemento con posición en X_o = UNIFORM(1,n)
     - b) Añadir x_i a X_b^*
   - 5.2. Calcular $\hat{\theta}_b^* = \theta(X_b^*)$
   - 5.3. Añadir $\hat{\theta}_b^*$ a $\hat{\theta}^*$
6. Hallar $p(\hat{\theta}_o)$ = pecentil de rango $\hat{\theta}_o$ en la distribución $\hat{\theta}^*$
7. Hallar $z_0 = z_{p(\hat{\theta}_o)} \sim N(0,1)$
8. Calcular $[\, z(BC_{inf}) \;,\; z(BC_{sup}) \,] = [\, 2z_0 + z_{(\alpha/2)} \;,\; 2z_0 + z_{(1-\alpha/2)} \,]$
9. Hallar $[\, p(z(BC_{inf})) \;,\; p(z(BC_{sup})) \,] \sim N(0,1)$
10. Hallar en $\hat{\theta}^*$ (dsitr. *bootstrap* de $\hat{\theta}$) el límite inferior del intervalo: l_{inf} = percentil ($100 \times p(\, z(BC_{inf}) \,)$)
11. Hallar en $\hat{\theta}^*$ (dsitr. *bootstrap* de $\hat{\theta}$) el límite superior del intervalo: l_{sup} = percentil ($100 \times p(\, z(BC_{sup}) \,)$)

ALGORITMO 12. *Método percentil con corrección del sesgo.*

f.3.3. Método percentil con corrección acelerada del sesgo

Este método es introducido por Efron en 1987, con el objeto de generalizar el método percentil con corrección del sesgo, dado que existen transformaciones que, como la de Wilson-Hilferty (Schenker, 1985), a pesar de que consiguen aproximaciones a la normal, no constituyen cantidades pivote porque su distribución depende del parámetro θ.

Al método percentil con corrección acelerada del sesgo se le denomina, de forma abreviada, "método BC_a". En este caso, la validez se basa en la existencia de una transformación $g(\hat{\theta}, X)$ monótona creciente, pero no necesariamente estabilizadora de la variancia, y de dos constantes z_0 y a que satisfacen:

$$\frac{(\hat{\phi} - \phi)}{\tau} + z_0 \sim N(0,1)$$

donde $\tau = 1 + a\phi$.

Como puede observarse, la distribución depende del valor de la constante a (denominada "constante de aceleración"), lo cual implica que $\hat{\phi} - \phi$ no es una cantidad pivote.

Esta transformación tiene por objeto principal ajustar la asimetría de la distribución $\hat{F}^*(\hat{\theta}^*)$ mediante la constante a basada en esta asimetría, que permite ajustar los percentiles de $\hat{F}^*(\hat{\theta}^*)$ y aumentar de este modo la precisión de la estimación de los límites del intervalo de confianza *bootstrap*.

Los límites de dicho intervalo se obtienen de la misma forma que en el método BC, aunque en este caso los límites $\alpha/2$ y $1-\alpha/2$ son los valores de los percentiles de rango $100(\alpha/2)$ y $100(1-\alpha/2)$ de la distribución *bootstrap* correspondientes a los valores z :

$$\left[z_0 - \frac{(z_{\alpha/2} + z_0)}{1 - a(z_0+z_{\alpha/2})} \quad , \quad z_0 + \frac{(z_{\alpha/2} + z_0)}{1 - a(z_0+z_{\alpha/2})} \right]$$

Como puede observarse, si $a = 0$, y por tanto no existe asimetría, entonces los métodos BC y BC_a coinciden. Si, además, $z_0 = 0$, porque no existe sesgo, entonces estos métodos coincidirán también con el método percentil.

Al igual que en el caso del método percentil y el método BC, en el método BC_a no es necesario conocer la transformación monótona ϕ, aunque sí es necesario determinar la constante de aceleración "a".

El principal problema que presenta el método BC_a reside en el hecho de que todavía no se dispone de un procedimiento automático de hallar el valor de la constante a únicamente a partir de los datos (DiCiccio y Romano, 1988; Hall, 1988b). Efron (1987) sugiere estimar a, por ejemplo, por funciones basadas en el coeficiente de asimetría (en el caso del *bootstrap* paramétrico) o en la función de influencia empírica (en el caso del *bootstrap* no paramétrico). Frangos y Schucany (1990) sugieren utilizar una estimación *jackknife* de la constante a, aunque este procedimiento requiere una mayor potencia computacional y no es todavía automático.

Así pues, los intervalos construidos mediante el método BC_a son más exactos que los obtenidos mediante los otros dos métodos percentil desarrollados por Efron, ya que tienen precisión de segundo orden. No obstante, en la práctica no pueden ser utilizados hasta que no se establezca un método automático para

estimar la constante de aceleración *a*. El *Algoritmo 13* presenta el procedimiento de cálculo del método BC_a:

1. Definir el estadístico de interés θ
2. Calcular $\hat{\theta}_o$ para la muestra original de datos X_o
3. Fijar el nivel α para el intervalo de confianza
4. Fijar el número B de remuestras a efectuar
5. Para b = 1 hasta B:
   - 5.1. Para i = 1 hasta n (tamaño muestral):
     - a) x_i = elemento con posición en X_o = UNIFORM(1,n)
     - b) Añadir x_i a X_b^*
   - 5.2. Calcular $\hat{\theta}_b^* = \theta(X_b^*)$
   - 5.3. Añadir $\hat{\theta}_b^*$ a $\hat{\theta}^*$
6. Hallar $p(\hat{\theta}_o)$ = percentil de rango $\hat{\theta}_o$ en la distribución $\hat{\theta}^*$
7. Hallar $z_0 = z_{p(\hat{\theta}_o)} \sim N(0,1)$
8. Estimar la constante de aceleración *a*
9. Calcular $[\, z(BCa_{inf}) \;,\; z(BCa_{sup}) \,] =$

$$= \left[z_0 + \frac{z_0 + z_{(\alpha/2)}}{1 - a(z_0 + z_{(\alpha/2)})} \;,\; z_0 + \frac{z_0 + z_{(1-\alpha/2)}}{1 - a(z_0 + z_{(1-\alpha/2)})} \right]$$

10. Hallar $[\, p(z(BCa_{inf})) \;,\; p(z(BCa_{sup})) \,] \sim N(0,1)$
11. Hallar en $\hat{\theta}^*$ (dsitr. *bootstrap* de $\hat{\theta}$) el límite inferior del intervalo: l_{inf} = percentil ($100 \times p(\, z(BCa_{inf}) \,)$)
12. Hallar en $\hat{\theta}^*$ (dsitr. *bootstrap* de $\hat{\theta}$) el límite superior del intervalo: l_{sup} = percentil ($100 \times p(\, z(BCa_{sup}) \,)$)

ALGORITMO 13. Método percentil con corrección acelerada del sesgo.

En síntesis, los procedimientos de construcción de intervalos de confianza presentados (excepto el método percentil con corrección acelerada del sesgo), pueden implementarse de forma automática y relativamente sencilla para casi cualquier estadístico calculado a partir de una simple muestra aleatoria (Efron y Gong, 1983; Tibshirani, 1988a). Bickel y Freedman (1984) y Rao y Wu (1988), entre otros autores, ilustran diversos usos del *bootstrap* en situaciones con muestreos complejos.

Al final de este capítulo (apartado 2.8), en la *Tabla 6*, se resumen las características de los diferentes métodos de construcción de intervalos *bootstrap* revisados, para facilitar la elección del procedimiento más adecuado en cada caso.

2.6.2. Contraste de hipótesis *bootstrap*

Desde el enfoque *bootstrap*, los fundamentos para obtener el grado de significación de una prueba de hipótesis son los siguientes. Sea x_o una muestra de datos extraída aleatoriamente de una población. Sea x_b^* una muestra *bootstrap* extraída aleatoriamente y con reposición a partir de los datos contenidos en la muestra original (recuérdese que el supraíndice asterísco "*" indica que se trata de una muestra extraída a partir de x_o, y el subíndice "b" indica que se trata de la b muestra extraída de x_o). Sea $t(x_b^*)$ el valor de un estadístico aplicado a la muestra *bootstrap* x_b^*. La probabilidad de que $t(x^*)$ sea mayor o igual a un valor determinado (h), dada por la distribución muestral de $t(x^*)$, se describe formalmente por prob($t(x^*) \geq h$). Esta distribución muestral puede utilizarse de distintos modos para estimar la distribución muestral real de $t(x)$.

Una forma directa de obtener el grado de significación a partir de la distribución muestral *bootstrap* de $t(x^*)$ es la siguiente. Si $t(x_o)$ es el valor obtenido al aplicar la prueba estadística en la muestra original de datos, la prueba de significación consistirá en calcular cuán inusual es $t(x_o)$ en relación a la distribución muestral *bootstrap* de $t(x^*)$. Es decir, el grado de significación de la prueba estadística es

$$\text{prob}[\ t(x^*) \geq t(x_o)\ ]$$

y la regla de decisión para rechazar la hipótesis nula (H_0) es

$$\text{Rechazar } H_0 \text{ si } \text{prob}[\ t(x^*) \geq t(x_o)\ ] \leq \alpha$$

siendo α el nivel de significación establecido "a priori" (habitualmente 0.05).

La principal ventaja que ofrece el remuestreo *bootstrap* frente al muestreo Monte Carlo reside en su generalidad. El remuestreo *bootstrap* puede aplicarse para cualquier prueba estadística, aunque no se conozca la distribución muestral ni se tenga tampoco información sobre la forma y los parámetros de la población origen de los datos. Respecto a las pruebas de aleatorización aproximadas ofrece también ventajas, puesto que permite realizar *inferencias válidas* sobre la

población, si bien este es un punto no falto de controversia (Schenker, 1985; Hinkley, 1986; Rasmussen, 1987a, 1987b, 1988, 1989; Efron, 1988; Strube, 1988). En el apartado 2.6.2.f se revisa un método basado en la combinación de las pruebas de aleatorización aproximadas y el remuestreo *bootstrap*.

a. Algoritmo general

Se han desarrollado diferentes variaciones de la técnica *bootstrap* para evaluar el grado de significación de una prueba estadística. El algoritmo general que subyace en todas ellas corresponde al denominado *método directo* (Noreen, 1989). Este método es la aplicación más intuitiva de la técnica de remuestreo *bootstrap* en el contraste de hipótesis, aunque, también es el procedimiento que ofrece una menor potencia y validez.

En el paso 5.1. del *Algoritmo 14* se aplica la función UNIFORM(1,*n*) para elegir los elementos de la nueva muestra de datos *bootstrap* (X_b^*). Este proceso se realiza mediante extracción aleatoria con reposición a partir de la muestra original X_o. Cabe señalar que X_b^* pueden ser varias muestras, dependiendo del tipo de diseño y el número de variables que estén involucradas en la hipótesis nula que se desee comprobar.

1. Decidir cuál es la prueba estadística de interés (θ)
2. Calcular $\hat{\theta}_o$ para la muestra de datos originales X_o
3. Inicializar el contador NSIG = 0
4. Fijar el número de B de remuestras a efectuar
5. Para b = 1 hasta B:
   - 5.1. Para i = 1 hasta n (tamaño de la muestra):
     - a) x_i = elemento con posición en X_o = UNIFORM(1,n)
     - b) Añadir x_i a X_b^*
   - 5.2. Calcular el pseudo-estadístico $\hat{\theta}_b^*$ para la muestra generada X_b^*
   - 5.3. Si $\hat{\theta}_b^* \geq \hat{\theta}_o$ (o, según el caso $\hat{\theta}_b^* \leq \hat{\theta}_o$)
     NSIG = NSIG + 1
6. Calcular el grado de significación unilateral: (NSIG+1) / (B+1)

ALGORITMO 14. *Remuestreo bootstrap: método directo*

Así, por ejemplo, si se desea comparar las medias de una variable para dos grupos de sujetos (por ejemplo, dos tratamientos experimentales), deberán extraerse aleatoriamente y por separado muestras partir de los datos de cada uno de los dos grupos de sujetos en la muestra original. En el caso de que la prueba estadística se refiera a dos variables cuantitativas en un diseño de medidas repetidas (por ejemplo, una correlación lineal, una comparación de dos medias, etc.), se extraerán las muestras *bootstrap* a partir de los vectores de datos de todos los sujetos de la muestra original. En el apartado 2.6.2.g. se revisarán con detalle los aspectos relativos al diseño del remuestreo *bootstrap*.

Como ya se ha comentado, existen diversas variaciones del algoritmo general de remuestreo *bootstrap* con objeto de utilizarlo en el contraste de hipótesis. El *método del desplazamiento* (con transformación pivotal para aumentar la validez), el *método de la aproximación Normal* y el *método de la aleatorización bootstrap* son las tres principales variantes. A continuación se revisan estas tres adaptaciones del método directo que permiten aumentar la potencia y validez de las pruebas.

Antes de pasar a exponer estos procedimientos, revisaremos los principios generales o directivas que deben tenerse en cuenta para aplicar el algoritmo *bootstrap* en las pruebas de contraste de hipótesis.

b. Reglas para la aplicación del contraste de hipótesis bootstrap

A raíz de los polémicos artículos de Wahrendorf, Becher y Brown (1987) y Becher (1993), Hall y Wilson (1991, 1993) exponen dos reglas fundamentales que deben guiar la implementación de una prueba de hipótesis *bootstrap*:

> «*La primera regla dice que debe tenerse cautela para asegurar que incluso en el caso de que los datos hayan sido extraídos de una población que no se ajusta a la hipótesis nula (H_0), el remuestreo debe realizarse de modo que refleje la H_0. La segunda regla sostiene que la prueba de hipótesis bootstrap debería utilizar aquellos métodos de los que se conoce su buen funcionamiento en el problema estrechamente relacionado de la construcción de intervalos de confianza. Esto conlleva a menudo utilizar cantidades pivote, lo cual significa usualmente corregir la escala.*» (Hall y Wilson, 1991 p. 757)

Siguiendo a estos autores, la primera directriz puede tener un profundo efecto en la potencia de la prueba, ya que si no se aplica, entonces el resultado del test *bootstrap* puede tener una potencia realmente muy baja, incluso en los casos en que la hipótesis alternativa mantenga una considerable distancia respecto a la hipótesis nula. En la práctica, esta regla implica lo siguiente. Supongamos, por simplicidad, que la hipótesis que se desea probar es H_0: $\theta = \theta_0$, en contra de la hipótesis alternativa bilateral H_1: $\theta \neq \theta_0$. Sea $\hat{\theta}$ el valor de una función de la muestra original $X_1, \ldots, X_n$, que constituye el estimador del parámetro θ desconocido, y sean $\hat{\theta}_b^*$ los valores de $\hat{\theta}$ calculados en las B muestras $X_1^*, \ldots, X_n^*$ extraídas aleatoriamente y con reposición de la muestra original. Siendo θ_0 el hipotético valor del parámetro bajo el supuesto de que se cumple H_0, la regla implica que, en lugar de comparar las diferencias $(\hat{\theta}_b^* - \theta_0)$ con $(\hat{\theta} - \theta_0)$, se deben comparar las diferencias $(\hat{\theta}_b^* - \hat{\theta})$ con $(\hat{\theta} - \theta_0)$. De este modo, incluso si es cierta la hipótesis alternativa, la distribución muestral *bootstrap* $(\hat{\theta}_b^* - \hat{\theta})$ será similar a la verdadera distribución bajo la hipótesis nula $(\hat{\theta} - \theta_0)$ en el sentido de que ambas estarán centradas en el valor 0.

La segunda regla no tiene una influencia directa sobre la potencia, pero sí reduce el error en el nivel de significación, es decir, la diferencia entre el nivel de significación actual de la prueba *bootstrap* y el nivel nominal α fijado "a priori". Esta regla implica que, siendo $\hat{\sigma}$ un estimador de la escala de $\hat{\theta}$, con la propiedad de que la distribución de $(\hat{\theta} - \theta_0)/\hat{\sigma}$ es virtualmente libre de desconocidos cuando H_0 es cierta (por ejemplo, podría ajustarse aproximadamente a una ley normal $N(0,1)$), y siendo $\hat{\sigma}_b^*$ el valor de $\hat{\theta}$ calculado para la b remuestra, entonces, la distribución *bootstrap* de $(\hat{\theta}_b^* - \hat{\theta})/\hat{\sigma}_b^*$ será un buen estimador de la distribución de $(\hat{\theta} - \theta_0)/\hat{\sigma}$ bajo H_0.

c. Método del desplazamiento

A partir de la primera regla, Noreen (1989) presenta el método del desplazamiento (*shift method*), que asume que la distribución muestral *bootstrap* y la distribución muestral bajo la hipótesis nula tienen la misma forma, pero diferente esperanza matemática (diferente localización). Para obtener el grado de significación del valor de la prueba estadística, la esperanza matemática de la distribución muestral *bootstrap* $\hat{\theta}^*$ debe corregirse desplazándola hasta que coincida con la esperanza matemática de la distribución de $\hat{\theta}$. De este modo, el proceso de remuestreo refleja realmente la y permite detectar H_0 la H_1.

Este método representa, pues, una mejora respecto al *método directo* en términos de potencia, si bien todavía se puede conseguir que su actual nivel de significación esté más próximo al nivel nominal α fijado "a priori" (mejorando de este modo su validez). Esto puede conseguirse aplicando la segunda regla propuesta por Hall y Wilson, esto es, transformando la escala para conseguir que el estadístico de contraste sea una cantidad *pivote*.

1. Decidir cuál es la prueba estadística de interés θ y cuál es la esperanza matemática $\bar{\theta}$ de su distribución muestral bajo H_0
2. Calcular $\hat{\theta}_o$ para el conjunto de datos original X_o
3. Inicializar el contador NSIG = 0
4. Fijar el número B de remuestras a efectuar
5. Para b = 1 hasta B:
   - 5.1. Para i = 1 hasta n (tamaño de la muestra):
     - a) x_i = elemento con posición en X_o = UNIFORM(1,n)
     - b) Añadir x_i a X_b^*
   - 5.2. Calcular el pseudo-estadístico $\hat{\theta}_b^*$ para la remuestra X_b^*
   - 5.3. Si $\hat{\theta}_b^* - \hat{\theta}_o \geq \hat{\theta}_o - \bar{\theta}$ (o, según el caso $\hat{\theta}_b^* - \hat{\theta}_o \leq \hat{\theta}_o - \bar{\theta}$): NSIG = NSIG + 1
6. Calcular el grado de significación unilateral: (NSIG+1) / (B+1)

ALGORITMO 16. *Método del desplazamiento ("shift method")*

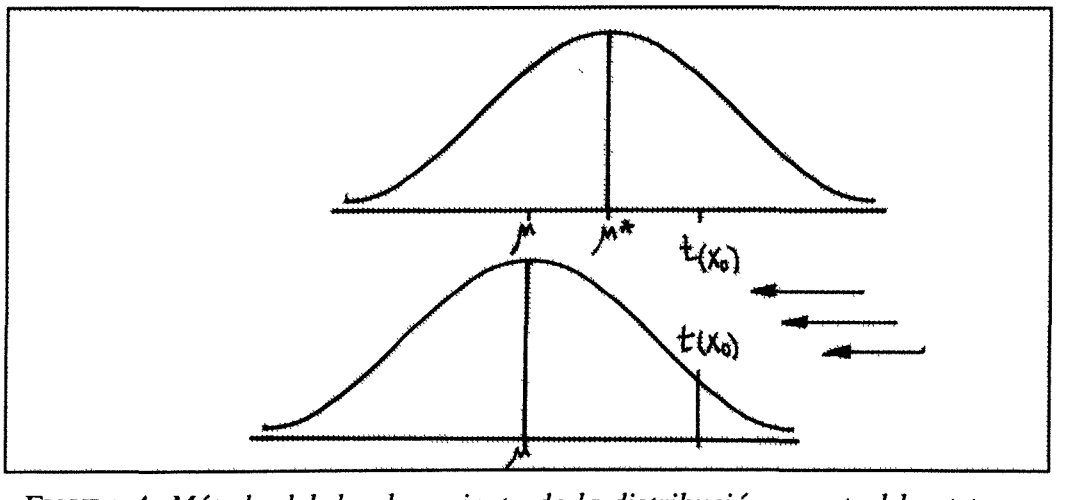


FIGURA 4. *Método del desplazamiento de la distribución muestral bootstrap. (Adaptada de Noreen, 1989)*

d. Método del desplazamiento con transformación pivotal

La aplicación de la segunda regla de Hall y Wilson procede como sigue. La regla enuncia que el estadístico de contraste $\hat{\theta}$ debe ser un pivote. Esto es, su distribución muestral, bajo la hipótesis nula en este caso, debe ser independiente del parámetro θ, de la muestra, y de cualquier otro parámetro desconocido.

Como se ha visto en el apartado 2.6.1.f., por ejemplo, las cantidades pivote para los parámetros μ y σ^2 de una variable normal $N(\mu;\sigma^2)$ son:

$$\mu \rightarrow \frac{\bar{x}-\mu}{\sqrt{s^2/n}} \sim t_{n-1} \qquad \sigma^2 \rightarrow \frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}$$

A partir del ejemplo, si los datos son una muestra aleatoria de una población no normal con media μ y variancia σ^2, entonces el estadístico es *asintóticamente pivotal*, es decir, sin reparar en μ, σ, o la distribución poblacional actual, la distribución del estadístico pivote aproxima $N(0,1)$ a medida que aumenta n, consecuencia directa del teorema central del límite. Siguiendo el ejemplo, $(\bar{x}-\mu)/\sqrt{n}$ no sería pivotal ya que su distribución asintótica es $N(0,\sigma^2)$.

Como demuestran Babu y Singh (1984), Abramovitch y Singh (1985) y Hall y Wilson (1991, 1993), la aplicación del remuestreo *bootstrap* para estadísticos pivote tiene mejores propiedades asintóticas, y la velocidad de convergencia del nivel de significación actual hacia el nivel de significación nominal es más rápida cuando se remuestrea este tipo de estadísticos.

El *Algoritmo 17* es una modificación del *Algoritmo 16* incluyendo el uso de un estadístico pivote.

A continuación se presentan dos métodos propuestos por Noreen (1989), que permiten obtener buenos resultados de la aplicación del *bootstrap* para obtener el grado de significación de una prueba de hipótesis, y cuya aplicación es automática (no es necesario, por ejemplo, conocer el estadístico pivote) si se cumplen los supuestos: el método de la aproximación normal y el método de la aleatorización *bootstrap*.

1. Decidir cuál es la prueba estadística de interés θ y cuál es la esperanza matemática $\overline{\theta}$ de su distribución muestral bajo H_0
2. Calcular $\hat{\theta}_o$ y $\hat{\sigma}_o$ para el conjunto de datos original X_o
3. Inicializar el contador NSIG = 0
4. Fijar el número B de remuestras a efectuar
5. Para b = 1 hasta B:
   - 5.1. Para i = 1 hasta n (tamaño de la muestra):
     - a) x_i = elemento con posición en X_o = UNIFORM(1,n)
     - b) Añadir x_i a X_b^*
   - 5.2. Calcular el estadístico $\hat{\theta}_b^*$ y σ_b^* para X_b^*
   - 5.3. Si $(\hat{\theta}_b^* - \hat{\theta}_o) / \sigma_b^* \geq (\hat{\theta}_o - \overline{\theta}) / \hat{\sigma}_o$ (ó $\leq$ según el caso): NSIG = NSIG + 1
6. Calcular el grado de significación unilateral: (NSIG+1) / (B+1)

ALGORITMO 17. *Método del desplazamiento con transformación pivotal.*

e. Método de la aproximación Normal

Noreen (1989) señala que si la muestra de datos es suficientemente grande, y se aceptan los supuestos de que la distribución muestral de un estadístico de contraste bajo H_0 sigue una ley normal y que su desviación estándar σ puede aproximarse por la desviación estándar σ^* de la distribución muestral *bootstrap* (ver apartado 2.6.1.e.), puede aplicarse la siguiente regla:

$$\text{Rechazar } H_0 \text{ si } \text{prob}[z \geq z_o] \leq \alpha$$

donde:

$$z_o = (\hat{\theta}_o - \mu) / \sigma^*$$

El algoritmo para implementar este método es el siguiente:

1. Decidir cuál es la prueba estadística de interés θ y cuál es la esperanza matemática $\bar{\theta}$ de su distribución muestral H_0
2. Calcular $\hat{\theta}_o$ para el conjunto de datos original X_o
3. Fijar el número B de remuestras a efectuar
4. Para b = 1 hasta B:
   4.1. Para i = 1 hasta n (tamaño de la muestra):
      a) x_i = elemento con posición en X_o = UNIFORM(1,n)
      b) Añadir x_i a X_b^*
   4.2. Calcular el pseudo-estadístico $\hat{\theta}_b^*$ para la muestra generada X_b^*
   4.3. Añadir $\hat{\theta}_b^*$ a θ^* (distribución muestral *bootstrap* de θ)
5. Calcular la desviación estándar σ^* de θ^* (distr. muestral *bootstrap*)
6. Calcular $z_0 = (\hat{\theta}_o - \bar{\theta}) / \sigma^*$
7. Hallar el grado de significación unilateral: $p(z \geq z_0) \sim N(0,1)$

ALGORITMO 18. *Método de la aproximación Normal.*

f. Método de la aleatorización bootstrap

Noreen (1989) y Holmes (1990), describen este método como una combinación de las pruebas de aleatorización y el remuestreo *bootstrap*. La aplicación de este método permite ampliar el alcance de las conclusiones de las pruebas de aleatorización. Como se ha revisado en el apartado 2.3., en las pruebas de aleatorización no se realiza ningún supuesto sobre el hecho de que las observaciones sean una muestra aleatoria de alguna población, por lo que no pueden realizarse inferencias formales sobre la población a partir de los resultados de estas pruebas. Por el contrario, el supuesto que subyace en el remuestreo *bootstrap* es que los datos son una muestra extraída aleatoriamente de alguna población y, por tanto, aplicando el remuestreo *bootstrap* se pueden realizar inferencias sobre la población. La combinación de ambas técnicas debe permitir, entonces, contrastar la hipótesis nula de que, por ejemplo, dos variables son independientes en la población origen.

El algoritmo de la aleatorización *bootstrap* es similar al de una prueba de aleatorización aproximada. Sin embargo, en lugar de permutar los valores de una variable (o grupo de variables) respecto a otra(s), se extraen muestras *bootstrap* para una o todas las variables. Si una variable (o grupo de variables) es controlada (variable independiente), entonces las muestras *bootstrap* se extraen a partir de

la(s) otra(s) variable(s) (por ejemplo, prueba *t*, ANOVA, etc.). Si ambas variables (o grupos de variables) no son controladas (por ejemplo, correlación), entonces las muestras *bootstrap* se extraen a partir de cada una de las dos variables (o grupos de variables) de forma independiente. En este caso, la independencia de la extracción de muestras *bootstrap* constituye la premisa principal que debe tomarse en consideración.

Noreen (1989) destaca que al aplicar el método de la aleatorización *bootstrap*, además de la hipótesis de independencia entre variables en la población, se está probando también otro supuesto: que las distribuciones marginales de las variables en la muestra aproximan satisfactoriamente las de las variables en la población. Por tanto, la prueba de aleatorización *bootstrap* será válida para contrastar la hipótesis conjunta de que los datos son una muestra aleatoria de una población en la cual las variables son estocásticamente independientes, y de que las distribuciones marginales de las variables en la muestra aproximan satisfactoriamente las distribuciones marginales de las variables en la población.

1. Decidir cuál es la prueba estadística de interés (θ)
2. Calcular $\hat{\theta}_o$ para el conjunto de datos original X_o
3. Inicializar el contador NSIG = 0
4. Fijar el número B de remuestras a efectuar
5. Para b = 1 hasta B:
   - 5.1. Para cada variable *j* que interviene en H_0:
     - a) Para i = 1 hasta n (tamaño de la muestra):
       - a.1. x_i = elemento con posición en X_o = UNIFORM(1,n)
       - a.2. Añadir x_i a X_b^*
   - 5.2. Calcular el pseudo-estadístico $\hat{\theta}_b^*$ para la muestra generada X_b^*
   - 5.3. Si $\hat{\theta}_b^* \geq \hat{\theta}_o$ (o, según el caso $\hat{\theta}_b^* \leq \hat{\theta}_o$): NSIG = NSIG + 1
6. Calcular el grado de significación unilateral: (NSIG+1) / (B+1)

ALGORITMO 19. *Método de la aleatorización bootstrap.*

g. Elección del esquema de remuestreo bootstrap

Se revisan en este apartado las consideraciones previas a la aplicación de un diseño de remuestreo *bootstrap*. Los tres aspectos fundamentales a tener en cuenta antes de iniciar el procedimiento de remuestreo son los siguientes (Lunneborg, 1987):

- ¿Cuál es la *unidad de muestreo* según la hipótesis nula planteada, los valores observados o los errores asociados a cada uno de estos valores?.

  Se trata de reproducir en las muestras *bootstrap* los mismos supuestos que se asumen para los datos de la muestra original.

  Wu (1986) y Doss y Chiang (1994) utilizan las denominaciones "método libre del modelo" (*model-free method*) y "método basado en el modelo" (*model-based method*) para identificar estos dos tipos de remuestreo. Estos autores también realizan análisis comparativos de la aplicación de estos dos métodos en situaciones complejas, aunque sin llegar a establecer reglas generales que permitan decidir de forma automática cuál de ellos hay que aplicar en cada caso.

- ¿Cuál es el *número de poblaciones* que han sido muestreadas?.

  La respuesta a esta pregunta puede depender de la respuesta a la pregunta anterior. El remuestreo *bootstrap* deberá llevarse a cabo de forma independiente para los datos de cada una de las poblaciones muestreadas.

  Así, por ejemplo, en un diseño factorial de análisis de la variancia con grupos independientes, el remuestreo *bootstrap* se realiza de forma independiente para cada submuestra de observaciones asociadas a cada nivel de los factores (o interacción de factores), puesto que cada uno de ellos representa una población distinta.

  Si, por el contrario, el diseño factorial es de medidas repetidas, el remuestreo se aplica sobre todo el conjunto de vectores fila de datos.

- ¿La hipótesis es univariante o multivariante?.

  El remuestreo *bootstrap* no deshace nunca la unidad de muestreo y, consecuentemente, en el caso de que la hipótesis incluya varias variables dependientes, entonces la unidad de remuestreo será el vector de valores Y_i para cada caso.

Por ejemplo, en un ANOVA unifactorial con un diseño de grupos independientes, cada observación se conceptualiza como un componente de *media* de grupo más un componente de *error* para cada sujeto:

$$Y_{ij} = \mu_i + \varepsilon_{ij}$$

El componente media es fijo, determinado por la asignación al grupo de tratamiento, mientras que el componente error se asume que es aleatorio y normalmente distribuido con variancia igual en cada grupo.

El supuesto de normalidad no interviene en el remuestreo *bootstrap*. El supuesto de homogeneidad de variancias en el remuestreo *bootstrap* es más flexible ya que puede afrontarse de dos formas (Lunneborg, 1987): suponiendo que realmente existe homogeneidad o suponiendo que los grupos son heterogéneos en cuanto a sus variancias.

En el caso de que se suponga homogeneidad, el remuestreo se realizará a partir de una única muestra (población) compuesta por los errores de los n sujetos de la muestra original (ε_i) -conceptualizando cada error como la distancia respecto a la media del grupo al cual pertenece cada sujeto-. Si por el contrario, se supone heterogeneidad de variancias, el remuestreo se realizará a partir de k muestras compuestas por los errores de los n_k sujetos de cada uno de los k grupos.

La única excepción a las reglas mencionadas, concretamente a las dos primeras, se produce al aplicar el método de aleatorización *bootstrap* revisado en el apartado anterior. Este método se rige por las reglas expuestas en el apartado 2.3.3. para la aplicación de las permutaciones en las pruebas de aleatorización. Esto es así, debido a que el principio de generación de la distribución que refleja la hipótesis nula de independencia entre variables se basa en el mismo procedimiento, es decir, en "mezclar" los datos de una(s) variable(s) respecto a la(s) otra(s) variable(s).

La diferencia entre las pruebas de aleatorización y el método de la aleatorización *bootstrap*, estriba únicamente en que en las primeras la población está formada por todas las posibles permutaciones de una variable respecto a la otra, mientras que en el método de la aleatorización *bootstrap*, la población está constituida por todas las posibles mezclas por remuestreo *bootstrap* independientemente para cada variable.

Así, por ejemplo, para hallar el grado de significación de la correlación entre dos variables, el procedimiento de aleatorización *bootstrap* consistirá en extraer cada vez dos muestras *bootstrap*, una para la primera variable y otra para la segunda variable, y emparejar luego los valores de cada una de ellas atendiendo para ello al orden en que fueron extraídos dichos valores de la muestra original.

2.7. CRITERIOS COMPARATIVOS Y DE SELECCIÓN DE LA TÉCNICA MÁS ADECUADA

Después de haber revisado los fundamentos de las principales técnicas estadísticas de remuestreo aplicadas a la estimación de parámetros y al contraste de hipótesis, quedan todavía diversas cuestiones por abordar. ¿Cuándo debería aplicarse una técnica de remuestreo en lugar de una técnica clásica?. En caso de optar por la aplicación de una técnica de remuestreo, ¿cuáles son los criterios para decidir cuál es la más adecuada?. Las dos cuestiones planteadas son, en realidad, una sola, puesto que las técnicas de remuestreo y las técnicas clásicas no son opuestas, sino complementarias.

Las características de las distintas técnicas *bootstrap* revisadas para construir intervalos de confianza se resumen en la *Tabla 6*.

En la *Tabla 7* se resumen los siguientes aspectos que intervienen en la elección de la técnica más adecuada para aplicar en las pruebas de hipótesis:

- Cuál es la naturaleza exacta de la hipótesis nula que se desea comprobar.
- Cuál es el proceso de cálculo de la significación estadística en el que se basa cada técnica.
- Cuáles son las ventajas que ofrece cada técnica respecto a las otras (o en qué situaciones es más adecuada).
- Cuáles son los inconvenientes y los problemas en la aplicación de cada una de las técnicas (por ejemplo, incumplimiento de supuestos).

TABLA 6. *Características principales de los métodos bootstrap de construcción de intervalos de confianza de un parámetro.*

Método	Supuestos	Precisión	Ventajas	Inconvenientes
Estándar	✓ La distribución de θ sigue ley Normal. ✓ La estimación $\hat{\theta}$ de θ es insesgada. ✓ Existe un estadístico pivote conocido en los casos en que se vulnera el supuesto de normalidad.	1er orden	✓ Son fáciles de calcular. ✓ Su construcción es automática.	✓ En muchas ocasiones no se cumple el supuesto de normalidad. ✓ No corrigen el posible sesgo de $\hat{\theta}$. ✓ Puede no existir el estadístico pivote cuando éste es necesario.
Percentil-t	✓ Debe existir un estadístico pivote conocido tal como: $\lvert\hat{\theta} - \theta\rvert / \hat{\sigma}$. ✓ El estimador $\hat{\sigma}$ debe ser estable y consistente.	✓ "simetrizado: 2º orden ✓ "colas iguales": 4º orden	✓ Son los más precisos. ✓ Su construcción es automática. ✓ Puede estimarse θ a partir de la distribución *bootstrap* del pivote.	✓ Debe disponerse de un estimador estable y consistente de $\sigma(F)$. ✓ No son invariantes frente a cambios de escala.
Percentil	✓ Existe una transformación monótona g, que no es necesario conocer, que normaliza la distribución. ✓ La distribución muestral *bootstrap* de $\hat{\theta}$ debe ser insesgada en la mediana.	1er orden	✓ Su construcción es automática. ✓ No es necesario que la distribución muestral sea normal en su forma. ✓ Son invariantes frente a transformaciones monótonas de los datos.	✓ A menudo resultan muy inexactos y sesgados.
Percentil con corrección del sesgo	✓ Existe una transformación monótona creciente y estabilizadora de la variancia g y una constante z_0, que no es necesario conocer, que normaliza la distribución.	1er orden	✓ Su construcción es automática. ✓ No es necesario que la distribución muestral sea normal en su forma. ✓ Son invariantes frente a transformaciones monótonas de los datos. ✓ Corrigen el sesgo del estimador $\hat{\theta}$. ✓ Si se conoce la transformación g y la constante z_0 el intervalo es exacto.	✓ La estimación de los límites del intervalo se ve afectada por la asimetría de la distribución.
Percentil con corrección acelerada del sesgo	✓ Existe una transformación monótona creciente g y dos constantes z_0 y a, ésta última conocida.	2º orden	✓ No es necesario que la distribución muestral sea normal en su forma. ✓ La transformación g no necesariamente debe ser estabilizadora de la variancia. ✓ Corrigen el sesgo del estimador $\hat{\theta}$. ✓ Ajustan la asimetría de la distribución.	✓ No son completamente automáticos dado que se debe conocer el valor de la constante a.

TABLA 7. *Principales características de las técnicas de obtención del grado de significación en el contraste de hipótesis.*

TÉCNICA		Hipótesis Nula	Obtención del grado de significación de la prueba	Ventajas	Inconvenientes
Pruebas paramétricas clásicas		Los datos son una muestra aleatoria de una determinada población en la cual se supone independencia entre las variables.	Derivar matemáticamente la distribución muestral de la prueba estadística. Si ya ha sido derivada y existen tablas, simplemente buscar en ellas el grado de significación.	✓ Si los supuestos requeridos para derivar la distribución muestral son válidos, ningún otro método es mejor. ✓ Existe software excelente ✓ Requiere menos tiempo de cálculo que las técnicas Monte Carlo	✓ Los supuestos requeridos para derivar la distribución muestral pueden no ser válidos. ✓ La distribución muestral exacta puede no haber sido derivada para la prueba estadística de interés.
Pruebas de aleatorización (*randomization tests*)		Una variable(s) es independiente de la otra(s) variable(s).	Aproximar la distribución muestral bajo la H_0 mediante permutaciones repetidas (sistemáticas o aleatorias) de una(s) variable(s) respecto a la(s) otra(s), recalculando cada vez la prueba estadística.	✓ Puede usarse para cualquier prueba estadística (incluso en diseños de caso único) ✓ Los datos pueden haber sido extraídos de cualquier población. ✓ No es necesario que los datos hayan sido extraídos de forma aleatoria.	✓ Formalmente no pueden realizarse inferencias a la población origen de los datos. ✓ Debe asegurase que todas las posibles permutaciones son igualmente probables.
Muestreo Monte Carlo		Los datos son una muestra aleatoria de una determinada población cuya distribución está *perfectamente* especificada bajo H_0.	Aproximar la distribución muestral bajo H_0 por extracción aleatoria de muestras *sin reposición a partir de un modelo* poblacional perfectamente especificado, recalculando cada vez la prueba estadística.	✓ Puede usarse para cualquier prueba estadística (incluso en diseños de caso único). ✓ Puede usarse aún cuando la distribución muestral del estadístico no se haya derivado analíticamente.	✓ Deben conocerse los parámetros poblacionales. ✓ Los supuestos que se realicen en caso de desconocimiento de algún parámetro poblacional pueden no ser válidos.
Remuestreo Bootstrap	**Método del desplazamiento** (*shift method*) **-con transformación pivotal-**	Los datos son una muestra aleatoria, la esperanza matemática de la distribución muestral del estadístico es un valor determinado, y la forma de esta distribución puede aproximarse por la de la distribución muestral *bootstrap*.	Aproximar la forma de la distribución muestral bajo H_0 por remuestreo repetido con reposición a partir de la propia muestra, recalculando cada vez la prueba estadística.	✓ Puede usarse para cualquier prueba estadística (incluso en diseños de caso único). ✓ Puede usarse aún cuando la distribución muestral de la prueba estadística no haya sido derivada analíticamente. ✓ No es necesario definir parámetros poblacionales para determinar H_0 (la muestra aproxima la población). ✓ La aplicación es completamente automática una vez ha sido especificado la prueba estadística.	✓ El supuesto de que la forma de la distribución muestral del estadístico puede aproximarse por remuestreo *bootstrap* puede no ser válido.
	Método de la aproximación Normal	Los datos son una muestra aleatoria suficientemente grande, la esperanza matemática de la distribución muestral es un valor conocido, y su desviación estándar puede aproximarse por la distribución muestral *bootstrap*.	Aproximar la desviación estándar de la distribución muestral bajo H_0 por remuestreo repetido y con reposición a partir de la propia muestra, recalculando cada vez la prueba estadística.		✓ Los supuestos de que la distribución muestral del estadístico es Normal y que puede aproximarse su desviación estándar por remuestreo *bootstrap* pueden no ser válidos.
	Método de la aleatorización *bootstrap*	Los datos son una muestra aleatoria de una población en la cual una variable(s) es independiente de la otra(s), y las distribuciones marginales de las variables pueden aproximarse bien por las distribuciones *bootstrap*.	Aproximar distribución muestral bajo H_0 "mezclando" los valores de una(s) variable(s) respecto a la(s) otra(s), por remuestreo repetido con reposición e independiente para cada variable, recalculando la prueba cada vez.		✓ Las distribuciones marginales de las variables deben aproximarse bien por las distribuciones *bootstrap*.

La *Figura 4* ilustra el proceso de selección de la técnica más adecuada para obtener el grado de significación en una prueba de hipótesis, atendiendo a los cuatro aspectos mencionados anteriormente.

Como puede observarse, la primera distinción se basa en el tipo de hipótesis nula que se desea comprobar. Si se desea comprobar la relación entre variables, sin realizar inferencias a la población origen de los datos, entonces la prueba más adecuada es la de aleatorización.

Si por el contrario, la hipótesis nula se plantea en términos de si la muestra puede provenir de una población determinada, entonces debe plantearse una nueva cuestión: ¿se conoce la forma y los parámetros que definen la distribución de la hipótesis nula en la población?.

En caso de que no sea así, las técnicas de remuestreo *bootstrap* son las más adecuadas para obtener el grado de significación de la prueba estadística.

En el caso de que sí se conozca la forma de la distribución y los parámetros poblacionales, y existan tablas de la distribución de probabilidad muestral, entonces no existen técnicas más adecuadas que las paramétricas convencionales.

Por último, en el caso de que no haya sido derivada analíticamente la distribución muestral del estadístico de contraste para la población definida puede aplicarse el muestreo Monte Carlo.

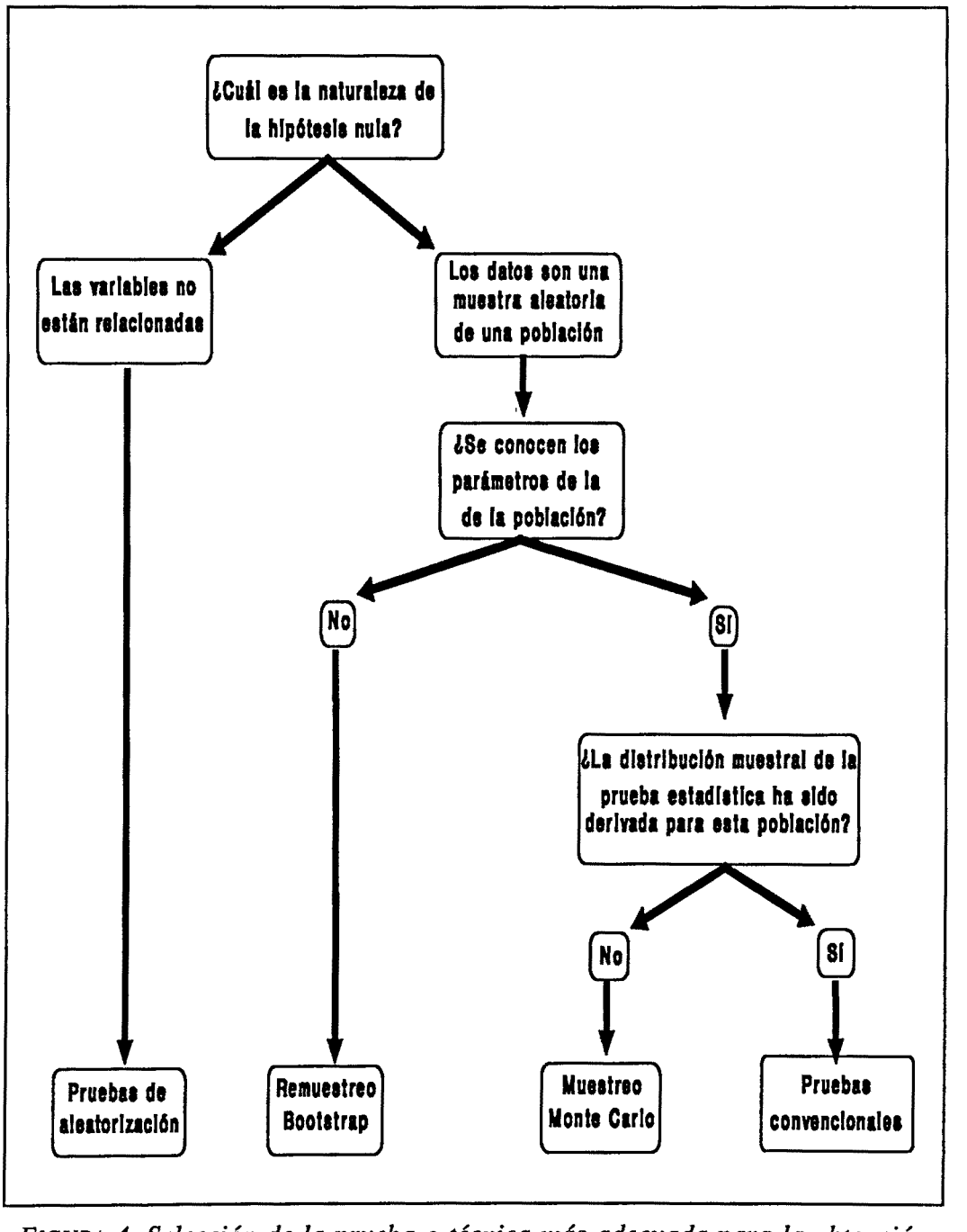


FIGURA 4. Selección de la prueba o técnica más adecuada para la obtención de la significación en el contraste de hipótesis. (Adaptada de Noreen, 1989).

2.8. AUMENTO DE LA EFICIENCIA DE LA SIMULACIÓN

De los apartados anteriores se desprende que las técnicas de remuestreo ofrecen soluciones para la estimación estadística en general, incluso en aquellas situaciones en las que no se poseen modelos paramétricos de referencia, aunque, como indican Davison, Hinkley y Schechtman (1986), estos procedimientos de simulación no están desprovistos de error y, por tanto, requieren todavía que se investiguen nuevos métodos para aumentar la eficiencia y la validez de las estimaciones, controlando las fuentes de error que introducen estas técnicas.

La primera fuente de error que se produce en los procedimientos de remuestreo se deriva del hecho de que el estimador θ^* obtenido no aproxima con precisión el verdadero valor θ, debido a que la función de distribución empírica no aproxima precisamente la distribución real ($F^* \neq F$). Los distintos refinamientos en la construcción de los intervalos de confianza (Efron, 1990; DiCiccio, 1992; DiCiccio y Efron, 1992; Shi, 1992), el uso de cantidades pivote (Hall y Wilson, 1991), y la correcta elección del esquema de remuestreo (Wu, 1986; Doss y Chang, 1994), constituyen ejemplos de procedimientos o estrategias para reducir este tipo de error.

La segunda fuente de error proviene de la propia simulación Monte Carlo, dado que los estimadores se obtienen mediante un número finito de remuestreos. Este tipo de error requiere soluciones a nivel de algoritmos más eficientes, es decir, que ofrezcan resultados más precisos (con menor variabilidad). Se trata de aumentar la velocidad de la *convergencia estocástica* sin tener que aumentar para ello el número de iteraciones (remuestreos) del algoritmo Monte Carlo, dado que el aumento del número de iteraciones constituye una solución "lenta", en el sentido que la precisión aumenta, generalmente, en un factor de $1/\sqrt{n}$ (Naylor et al., 1982) -en esta línea, si deseamos, por ejemplo, que el error aleatorio se reduzca a la mitad, es necesario cuadruplicar el tamaño n de la muestra-.

El uso de las técnicas clásicas de reducción de la variancia (también denominadas *Monte Carlo swindles*), como el "muestreo de importancia" (Therneau, 1983; Johns, 1988; Hinkley y Shi, 1989; Do y Hall, 1991), o las variables antitéticas, que pueden dividir por dos el número de iteraciones necesarias (Hall, 1989a, 1989b; Do, 1992), y el uso de esquemas de remuestreo más sofisticados, como el "remuestreo balanceado" (Davison, Hinkley y Schechtmann, 1986; Graham, Hinkley, Jhon y Shi, 1987; Gleason, 1988; Do, 1992), o la "simulación acelerada" (Ogbonwman y Wynn, 1986), que aceleran la convergencia de θ^* hacia θ, constituyen las principales vías para controlar el error Monte Carlo. Utilizando una analogía, el uso de estos métodos en la simulación

juega el mismo papel que el control de variables de confusión en los diseños de investigación.

Como señala Sánchez (1990b), en la práctica es difícil separar las dos fuentes de error descritas, y los métodos señalados a menudo las reducen de forma simultánea.

En el siguiente capítulo se presenta el uso del esquema de remuestreo balanceado aplicado en el instrumento informático que se desarrolla en la tesis.

2.9. VALIDEZ

Según el enfoque de la teoría estadística del contraste de hipótesis, una prueba es válida si la probabilidad de rechazar la hipótesis nula (H_0) cuando ésta es cierta, no es superior al nivel de significación α fijado "a priori". Así, una prueba de hipótesis es *exactamente válida* cuando la probabilidad de rechazar incorrectamente la H_0 es exactamente igual al valor α fijado para la prueba.

Muchos estadísticos teóricos han demostrado la validez de las diferentes técnicas de remuestreo para evaluar el grado de significación de una prueba de contraste de hipótesis (Hoeffding, 1952; Box y Andersen, 1955; Dwass, 1957; Edgnington, 1969, 1980; Foutz, 1980, 1981). Noreen (1989) señala que la demostración más general es la propuesta por Foutz (1980, 1981), puesto que, a diferencia de otros autores, no realiza supuestos sobre propiedades asintóticas y resuelve la posible presencia de empates en los valores de la prueba estadística mediante el empleo de una variable aleatoria auxiliar para deshacerlos. Además, esta demostración es general en el sentido de que demuestra la validez de todas las técnicas derivadas del muestreo Monte Carlo (como las pruebas de aleatorización y las técnicas de remuestreo *bootstrap*).

Noreen (1989) amplia esta demostración para el caso de que no se emplee ninguna variable para deshacer los posibles empates. Siguiendo a Noreen, la demostración de la validez de las técnicas estadísticas de remuestreo se desarrollaría de acuerdo con los pasos que se exponen a continuación.

Sea $t(X_0)$ un estadístico función de una matriz de datos X_0. Sean $t(X_1)$, $t(X_2)$, ... , $t(X_{NSIM})$ los resultados de esta función para las NSIM matrices de datos generadas a partir de la matriz original X_0. Si se ordenan los valores de $t(X_i)$ del siguiente modo:

$$t(X_1) \geq t(X_2) \geq \ldots \geq t(X_{NSIM})$$

puede definirse:

$$NSIG = \text{máx}[\, k \mid t(X_k) \geq t(X_0) \text{ para } k = 0, \ldots, NSIM \,]$$

La hipótesis nula (H_0) es rechazada al nivel α si:

$$(NSIG + 1) / (NSIM + 1) \leq \alpha$$

La posibilidad de empates entre los diferentes $t(X_i)$ dificulta la comprobación de que:

$$\text{prob}[\ (NSIG + 1) / (NSIM + 1)\ ] \leq \alpha$$

Para eliminar estos empates, se generan NSIM+1 variables aleatorias auxiliares $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{NSIM}$ uniformemente distribuidas en un intervalo $(-\delta, +\delta)$, tal que $\delta < \text{mín}[\,|t(X_i) - t(X_j)| > 0]$. Seguidamente se transforman los resultados de las pruebas estadísticas sumando a cada valor $t(X_i)$ el correspondiente valor aleatorio ε_i:

$$t'(X_i) = t(X_i) + \varepsilon_i \quad \text{para } i = 0, \ldots, NSIM$$

De esta forma quedan eliminados los posibles empates, y los $t'(X_i)$ ahora aparecen ordenados del siguiente modo:

$$t'(X_1) > t'(X_2) > \ldots > t'(X_{NSIM})$$

El valor transformado del estadístico para los datos originales $t'(X_0)$ estará comprendido en uno de los NSIM+1 intervalos:

$$(-\infty, t'(X_{NSIM})], \ldots, (t'(X_2), t'(X_1)], (t'(X_1), \infty)$$

Bajo H_0, el valor observado para $t'(X_0)$ es una muestra de tamaño 1 de una distribución i.i.d. a la distribución de $t'(X)$. Puede deducirse entonces, que la probabilidad de que $t'(X_0)$ esté comprendido en uno de los intervalos anteriormente especificados es 1/(NSIM+1). La variable aleatoria NSI' queda ahora definida por:

$$NSIG' = \text{máx}[\ k \mid t'(X_k) \geq t'(X_0) \quad \text{para } k = 0, \ldots, NSIM\ ]$$

o, NSIG' es el número de intervalos antes descritos que están comprendidos en el intervalo $[t'(X_0), \infty)$. Es decir, NSIG' puede ser cualquier valor entero comprendido entre 0 y NSIM, y cada uno de esos valores enteros tienen igual probabilidad bajo H_0. Por tanto, dado un nivel α, si se selecciona NSIM de tal modo que $\alpha(NSIM+1)$ sea un entero, entonces:

$$\text{prob}[\ (NSIG+1) \leq \alpha(NSIM+1)\ ] = \alpha(NSIM+1) / (NSIM+1) = \alpha$$

ó

$$\text{prob}[\ (\text{NSIG}+1) / (\text{NSIM}+1) \leq \alpha\ ] = \alpha$$

lo cual demuestra que si se selecciona un NSIM tal que α(NSIM+1) sea un entero, y si se lleva a cabo la aleatorización auxiliar para asegurar que no haya empates, entonces el grado de significación (NSIG+1)/(NSIM+1) es exactamente válido. Es decir, la probabilidad de rechazar la hipótesis nula H_0 cuando ésta es cierta es exactamente igual al nivel de rechazo α fijado para la prueba.

Si se establece NSIM = 100k−1, siendo k cualquier número entero positivo, se satisfará la condición de que α(NSIM+1) sea un entero. Por ejemplo, NSIM=99 ó 199 ó 999 satisfacen la condición para α = 0.01, 0.05 y 0.10.

Si no se hace uso de una variable aleatoria para eliminar empates, el grado de significación (NSIG+1)/(NSIM+1) no será necesariamente exactamente válido, pero puede asegurase que no superará en ningún caso el nivel de rechazo α fijado para la prueba.

El efecto de añadir una variable aleatoria auxiliar provoca dos tipos de desempates cuando $t(X_i) = t(X_0)$: aquellos en que $t(X_i)$ pasa a ser mayor que $t(X_0)$ y, aquellos en que $t(X_i)$ pasa a ser menor que $t(X_0)$. Dado que NSIG es igual al número de $t(X_i) \geq t(X_0)$, al calcular NSIG', los casos en que $t(X_i) = t(X_0)$ pasa a ser $t(X_i) < t(X_0)$ dan como resultado que NSIG' siempre sea superior a NSIG. Por tanto:

$$\text{prob}[(\text{NSIG}+1)/(\text{NSIM}+1) \leq \alpha] \leq \text{prob}[(\text{NSIG}'+1)/(\text{NSIM}+1) \leq \alpha] = \alpha$$

En conclusión, puede decirse que las técnicas estadísticas de remuestreo aplicadas al contraste de hipótesis, que se basan en el cociente (NSIG+1)/(NSIM+1) para calcular el grado de significación, son válidas.

2.10. POTENCIA

Con objeto de estudiar la potencia de las técnicas de remuestreo para el contraste de hipótesis, se han llevado a cabo muchas investigaciones en las que se han comparado los resultados obtenidos por estos procedimientos con los obtenidos al aplicar las pruebas paramétricas convencionales. En estos estudios se evalúa cuál es el procedimiento más potente atendiendo al criterio de máxima potencia (*optimalidad*) de Neyman-Pearson. Es decir, se considera más potente aquella prueba que, fijado un nivel α, es capaz de rechazar la hipótesis nula (H_0) un mayor número de veces cuando ésta es realmente falsa (en el apartado 2.2.2 se ha revisado la aplicación del muestreo Monte Carlo para investigar la potencia de las pruebas de hipótesis).

Los datos de estas investigaciones sobre potencia parecen indicar sin lugar a dudas que no existe pérdida de potencia cuando un procedimiento de remuestreo es utilizado en lugar de una prueba paramétrica convencional (Hoeffding, 1952; Kempthorne y Doerfler, 1969; Edgington, 1969a, 1980; Noreen, 1989; Romano, 1989).

Hope (1968) y Noreen (1989), entre otros, ofrecen demostraciones de que la potencia, para un determinado nivel α, es una función incremental del número de muestras simuladas que se generan. En otras palabras, cuanto mayor sea el número de permutaciones aleatorias en una prueba de aleatorización, o mayor el número de muestras simuladas a partir de los datos en un remuestreo *bootstrap*, o a partir del modelo poblacional en un muestreo Monte Carlo, mayor será la potencia de la prueba estadística. Otros autores, como Hall y Wilson (1991) -véase apartado 2.6.2.b-, presentan reglas específicas para aumentar la potencia a nivel del propio procedimiento.

3

CONSTRUCCIÓN DE UN INSTRUMENTO INFORMÁTICO PARA APLICAR LOS MÉTODOS MONTE CARLO: EL *TOOLBOX MONTECARLO*

«Fisher was able to provide a statistical theory that took full advantage of the computational facilities of the 1920's. The goal now is to do the same for the 1980's.»

Persi Diaconis y Bradley Efron, 1983

El contenido de lo expuesto en el capítulo anterior permite determinar cuáles son los principales procedimientos y algoritmos que intervienen en las técnicas de remuestreo. El objetivo de este capítulo es el análisis, diseño y desarrollo de una herramienta informática, de carácter universal, que constituya un laboratorio de investigación estadística con base en el muestreo Monte Carlo y en las técnicas de remuestreo que de él se derivan.

En primer lugar, se realiza un análisis de los aspectos que deben caracterizar el instrumento informático, así como los aspectos logísticos y de infraestructura que se requieren para su elaboración. Seguidamente, se revisan los diferentes sistemas informáticos que actualmente existen en el entorno de los ordenadores personales, y que permiten programar o utilizar las técnicas estadísticas de remuestreo; esta revisión está encaminada a la elección de la plataforma de programación más adecuada para la construcción de la herramienta final.

Por último, se diseñan y desarrollan los módulos necesarios para implementar las técnicas de muestreo Monte Carlo y de remuestreo estadístico expuestas en el capítulo anterior, presentándose la sintaxis de los mandatos creados, así como diversos algoritmos que ilustran su facilidad de uso.

El producto final pretende abarcar el conjunto de herramientas para llevar a cabo los experimentos básicos de muestreo Monte Carlo y para aplicar de un modo estandarizado las técnicas de aleatorización y las técnicas *bootstrap* revisadas en el capítulo anterior.

3.1. ANÁLISIS DE REQUERIMIENTOS

Antes de iniciar el desarrollo de un instrumento informático es preciso especificar claramente cuáles son las características que deberá poseer. Este análisis permitirá guiar el *estudio de la viabilidad* del proyecto, estableciendo un marco preciso para evaluar y comparar las diferentes plataformas existentes en la actualidad, con el objetivo de decidir cuál de ellas es la más adecuada para implementar la herramienta final (Hawryskiewycz, 1990; Bell et al., 1992).

La primera consideración importante que se plantea al analizar las características adecuadas para un entorno de simulación estadística es, sin duda, si dicho entorno debe consistir en un sistema totalmente abierto, en el que el usuario puede ir escribiendo los mandatos en un lenguaje de programación específico, o bien, si el entorno debe ser cerrado, ofreciendo las distintas opciones a través de la selección de comandos de menús y cuadros de diálogo.

La flexibilidad inherente a los sistemas basados en lenguajes de programación parece ser un argumento de peso para abogar por este tipo de entorno. Además, mediante el uso un lenguaje de programación, el investigador tiene pleno control sobre todos los factores que intervienen en el proceso de simulación. En este sentido, tampoco puede olvidarse que el uso de un lenguaje algorítmico es uno de los aspectos más relevantes de la aplicación didáctica de la tecnología de simulación Monte Carlo en la enseñanza de la estadística (Travers, 1981; Clements, 1986; Zimmerman y Shavilk, 1987; Holmes, 1990; Khamis, 1991). En este ámbito, la característica *interactiva* del lenguaje es un factor importante (Norusis, 1980; Moahmoud y Davidson, 1985; Rosebery y Rubin, 1990), por lo que son preferibles los lenguajes que funcionan en modo intérprete.

Por otro lado, la facilidad de uso de los sistemas guiados por menús es una característica deseable en cualquier herramienta informática aunque ello implica, generalmente, una menor libertad en el momento de especificar las características de las tareas a realizar, dado que sólo pueden manipularse aquellos parámetros que han previsto los analistas del sistema.

La solución idónea puede ser un entorno que aglutine los aspectos positivos tanto de los lenguajes de programación como de los programas cerrados. Esto se puede conseguir con los actuales lenguajes de programación de alto nivel que funcionan en modo interactivo. Algunos de estos sistemas están diseñados de forma que el usuario puede definir nuevos mandatos, ampliando el lenguaje a su conveniencia, y además, este lenguaje puede incorporar las funciones necesarias para la creación de programas completos que funcionen de forma independiente

bajo el entorno intérprete, con lo que se pueden programar aplicaciones completas para realizar familias de experimentos específicos.

Uno de los aspectos de los lenguajes intérprete que tradicionalmente han recibido más críticas es su lentitud respecto a los lenguajes compilados (Bishop, 1991). Esta crítica no tiene sentido actualmente, dado que la mayor parte de sistemas aparecidos en los dos últimos años realizan una pre-compilación de los grupos de mandatos que se ejecutan, lo cual tan sólo supone un pequeña demora en el momento de iniciar la ejecución (en realidad, esta tarea se realiza en menos tiempo del que habitualmente se necesita para realizar un proceso completo de compilación y enlace -*linkage*- de un programa ejecutable en cualquier lenguaje de programación).

El verdadero problema de los lenguajes intérprete estriba, en realidad, en el hecho de que acostumbran a ser de propósito general, mostrándose poco eficaces para tareas que requieran rutinas altamente específicas, como sucede en la simulación estadística. Por este motivo, el lenguaje debe estar diseñado para el análisis numérico, o bien poseer suficientes funciones de bajo nivel especializadas para este uso.

Por último, el entorno debe proporcionar funciones para la visualización gráfica del progreso de los procesos de simulación y de sus resultados, aspecto de especial relevancia en la aplicación didáctica de estas técnicas (Gentleman, 1977; Bland, 1984; Goodman, 1986; Gordon y Hunt, 1986).

En síntesis, las principales características que debería cumplir la herramienta informática más adecuada para implementar las técnicas de muestreo Monte Carlo son, en nuestra opinión, las siguientes:

- Debe tratarse de un lenguaje de programación y no de un programa cerrado.
- El lenguaje debe funcionar en modo intérprete (interactivo).
- El lenguaje debe estar diseñado de forma específica para el análisis numérico (o debe proveer las funciones de bajo nivel necesarias para ello).
- El lenguaje debe ser ampliable mediante la definición de nuevos mandatos.
- El lenguaje debe proporcionar las funciones necesarias para construir sub-programas que funcionen de forma independiente en el entorno.
- El lenguaje debe incorporar los mandatos y funciones para la visualización gráfica (a ser posible, el propio entorno de trabajo debería ser gráfico).

3.2. ESTUDIO DE LA VIABILIDAD

3.2.1. Plataformas existentes para el entorno PC

En general, la implementación de las técnicas de remuestreo se lleva a cabo mediante el desarrollo de programas escritos en uno de los lenguajes de programación estándar, como FORTRAN, PASCAL, C o BASIC, entre otros (Fishman, 1978; Gordon, 1980; Ripley, 1987; Bratley et al, 1987; Lewis y Orav, 1989). Los lenguajes de programación ofrecen la ventaja de ser muy adaptables y amplios permitiendo, a su vez, resolver cualquier problema de simulación estadística.

El esfuerzo y la cantidad de tiempo requeridos para construir los programas es bastante elevado (Naylor et al., 1982; Thisted, 1988; Kleignen y Groenendaal, 1992), incluso tomando rutinas estándar de las librerías más conocidas como NAG, IMSL o Numerical Recipes (Press et al., 1989). Cualquier programa que se construya debe ser revisado y probado con mucha atención antes de poder asegurar que los resultados que ofrece son correctos. Además, una vez escrito y revisado un programa, su alcance acostumbra a ser muy limitado, dado que generalmente se construye para resolver un problema propio de una investigación en particular.

Se han desarrollado algunos lenguajes de programación especializados para la construcción de programas de simulación estadística, como DATASIM (Bradley, 1988, 1989), GAUSS (1992), Resampling Stats (Simon y Bruce, 1991a, 1991b, 1992), SIMSTAT (Péladeau y Lacouture, 1993), StatLab (Hodges et al., 1975; Farnum, 1991), etc. Estos lenguajes reducen el tiempo de desarrollo a una fracción del necesario con lenguajes de propósito general. A pesar de ello, su uso no se ha generalizado debido, en parte, a que son sistemas cerrados que no poseen la flexibilidad y potencia de los lenguajes estándar, si bien están provistos de un gran número de funciones específicas para construir algoritmos de simulación estadística.

Como ya se ha comentado, los lenguajes de programación no son la única plataforma elegida por los investigadores para llevar a cabo sus experimentos de simulación. Entre las plataformas alternativas a los lenguajes de programación, pueden destacarse las siguientes:

- Los paquetes de análisis estadístico, como True Epistat (Goldstein, 1988), SAS, BMDP, MINITAB, SPSS, SYSTAT, etc.

- Las hojas de cálculo, como Excel, Lotus 1-2-3, Quattro-Pro, etc. (Lee y Soper, 1985, 1986; Coe, 1989; Hewett, 1988; Stent y McAlevey, 1991; Collier, 1992; Wood, 1992; Willeman, 1994), y las hojas de cálculo adaptadas para el análisis y programación estadística, como PREDICT (Coe, 1991) y SpreadStat (Roos, 1988).
- Los programas diseñados específicamente para realizar inferencias estadísticas con base en técnicas de remuestreo, como los desarrollados por Lunneborg (1987, 1988) y Edgington y Khuller (1992), el programa BOJA (Boomsma, 1990; Boomsma y Molenaar, 1991), PITMAN (Dallal, 1988), RANOVA, TESTR y TTESTS (Koslov y Enright, 1985), RT (Manly, 1991), etc.

  Goldstein (1990), realiza un análisis comparativo de algunos de estos programas.

3.2.2. MATLAB como plataforma para simulación en estadística

Después de revisar la mayor parte de los sistemas mencionados, se ha escogido MATLAB (1992a, 1992b), en su versión 4.2, como plataforma de desarrollo. MATLAB cumple la totalidad de los requisitos enunciados en el apartado anterior, y permite realizar cualquier experimento de simulación Monte Carlo con precisión y eficacia. Se trata de un completo lenguaje de programación que funciona en modo intérprete (interactivo).

MATLAB está diseñado específicamente para el análisis y visualización numérica, e incorpora un gran número de funciones matemáticas. El aspecto fundamental de MATLAB es el manejo de matrices de datos (su nombre es el acrónimo de "*matrix laboratory*"). En este sentido, una de las características más sobresalientes de MATLAB es su capacidad para dimensionar de forma dinámica las matrices de datos, lo cual permite trabajar de forma sencilla y rápida con estas estructuras. Por otra parte, el hecho de funcionar bajo el entorno MS-Windows, y de poseer un gran número de funciones de bajo y alto nivel para el control gráfico, lo hacen idóneo para un entorno de investigación.

Respecto a los otros lenguajes examinados (C, FORTRAN, DATASIM, GAUSS, PASCAL, Resampling Stats, SIMSTAT, SAS, SPSS y SYSTAT), MATLAB ofrece la ventaja de requerir unos conocimientos mínimos de programación y de ser totalmente abierto, admitiendo la extensión sin límite del lenguaje base mediante la definición de nuevos mandatos, que pueden ser

utilizados del mismo modo que los originales, de forma totalmente transparente. Esto permite, por ejemplo, crear un entorno de alto nivel específico para llevar a cabo los experimentos de simulación Monte Carlo.

Por otro lado, aunque MATLAB es una plataforma de programación completa, posee un potente sistema de extensión que permite utilizar rutinas y programas escritos en lenguaje C y FORTRAN, ya sea para aprovechar las inversiones realizadas en estos lenguajes, o bien para aumentar la velocidad de ejecución de procesos repetitivos clave de un experimento de simulación. Este sistema, denominado *external interface* (MATLAB, 1993a), es compatible con las versiones más conocidas de los dos lenguajes mencionados.

A través del *extenal interface* pueden utilizarse, pues, las rutinas estándar de librerías como NAG, IMSL o Numerical Recipes, que se convierten en archivos MEX que MATLAB maneja como funciones o mandatos propios. Estos archivos MEX pueden ser de tipo REX (*Relocatable Executable*), que utiliza código de 32 bits al igual que MATLAB, o de tipo DLL (*Dinamic Link Library*), que maneja código de 16 bits en el formato estándar de MS-Windows, permitiendo el acceso a todos los recursos de este sistema operativo (impresoras, tarjetas de expansión, protocolos de comunicación, etc.).

El *external interface* de MATLAB incluye también el protocolo de intercambio dinámico de datos DDE estándar en MS-Windows, a través del cual se puede establecer fácilmente una "conversación" (transmisión y recepción de datos) con cualquier aplicación de funcione bajo este sistema operativo como, por ejemplo, paquetes estadísticos y hojas de cálculo. En esta situación, MATLAB puede actuar como *cliente*, iniciando la conversación, o como *servidor*, proporcionando respuesta a las solicitudes de otros programas.

a. Desarrollo de cajas de herramientas (Toolboxes)

Los investigadores pueden implementar los mandatos que consideren oportunos para realizar un experimento concreto, y estos mandatos pueden utilizarse de nuevo en futuras investigaciones, ya sea directamente, o aumentando su alcance mediante reparametrización. En este sentido, MATLAB es un verdadero repositorio en el que se van acumulando algoritmos a medida que se utiliza.

Las estructuras que MATLAB incorpora para la ampliación de su lenguaje son las denominadas "cajas de herramientas" (*toolboxes* en terminología MATLAB). En realidad se trata simplemente de directorios que contienen

archivos de macros, funciones o programas completos, que una vez creados MATLAB reconoce automáticamente.

Las áreas para las que ya existen *toolboxes* específicos son, entre otras, el análisis de series temporales, el diseño de sistemas de control, la simulación del funcionamiento de redes neuronales, la simulación de sistemas dinámicos (denominado SIMULINK, y que se distribuye actualmente junto con MATLAB), el análisis estadístico descriptivo y las distribuciones de probabilidad (mediante el *toolbox* Statistics; Bradley, 1993). Existen varias publicaciones, como la de Stark y Woods (1994), que recogen las aplicaciones de MATLAB en diferentes disciplinas científicas. The MathWorks (1994) edita periódicamente un resumen de estas publicaciones.

El objetivo del presente trabajo se concreta, pues, en el análisis y desarrollo de un nuevo *toolbox*, denominado ***MonteCarlo***, que contenga los mandatos necesarios para la aplicación del muestreo Monte Carlo, y de las técnicas de remuestreo estadístico descritas en el capítulo 2. Con el diseño de este *toolbox* se pretende que MATLAB se convierta en un laboratorio de estadística con base en este tipo de técnicas. En el apartado 3.3. se describen los módulos que conforman el *toolbox* y su funcionamiento.

b. Elementos de programación

En este apartado se describen, de forma sucinta, las principales características de MATLAB como lenguaje de programación. El objetivo de esta exposición es ofrecer una visión panorámica que permita entender la sintaxis y facilite la lectura de los algoritmos de simulación Monte Carlo que se presentan posteriormente. Por otro lado, los conceptos que aquí se presentan constituyen la selección mínima de mandatos MATLAB, que junto con los mandatos implementados en el *toolbox MonteCarlo* conforman el cuerpo básico de instrucciones necesarias para construir los algoritmos de simulación estadística.

b.1. Manejo de matrices

MATLAB trabaja esencialmente con un único tipo de objeto, una matriz numérica rectangular, de forma tal que todas las operaciones y mandatos permiten indicar las operaciones matriciales de forma análoga a como se escriben en papel. Como ya se ha indicado antes, las matrices no requieren un dimensionado previo,

lo cual permite resolver muchos problemas numéricos en una fracción del tiempo necesario para escribir un programa en un lenguaje como FORTRAN, BASIC, o C (Jones, 1993). A continuación se indica brevemente la forma de operar con las matrices.

La forma habitual de crear una matriz consiste en escribir su nombre, el operador de asignación "igual" (=), y la lista de valores (o recuperar esta lista desde un fichero externo) separando las columnas mediante una coma o un espacio en blanco, y las filas mediante un punto y coma. Ejemplo:

$$X = [\ 1\ 2\ ;\ 3\ 4\ ;\ 5\ 6\ ] \Rightarrow X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Para referenciar los elementos de una matriz deben indicarse sus índices entre paréntesis, a continuación del nombre de la matriz. Ejemplo:

$$X(2,2) \Rightarrow 4$$

Para expresar una submatriz se utiliza el nombre de la matriz indicando entre paréntesis las listas de índices de fila y de columna separados por una coma. Estas listas se escriben con la expresión *ini : fin*, o bien dos puntos (:) para simbolizar los índices de toda la fila o la columna. Ejemplo:

$$X(\ 2{:}3\ ,\ :\ ) \Rightarrow \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Para eliminar un elemento (o una submatriz), debe realizarse la siguiente asignación nula. Ejemplo:

$$X(\ 2\ ,\ :\ ) = [\,] \Rightarrow X = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}$$

Por último, para añadir nuevos elementos a una matriz basta con realizar la asignación, indicando la lista de índices de fila y columna. La matriz se redimensiona automáticamente. Ejemplo:

$$X(\ 3\ ,\ :\ ) = [\ 7\ 8\ ] \Rightarrow X = \begin{bmatrix} 1 & 2 \\ 5 & 6 \\ 7 & 8 \end{bmatrix}$$

MATLAB distingue dos tipos de operaciones aritméticas: las definidas por las reglas del algebra lineal y las operaciones matemáticas que se realizan individualmente, elemento a elemento. Para indicar una operación elemento a elemento, debe escribirse un punto (.) antes del operador aritmético. La siguiente tabla resume los operadores aritméticos básicos:

TABLA 8. *Operadores aritméticos*

OPERADOR	SIGNIFICADO
A + B	Adición
A – B	Sustracción
A * B ó A .* B	Multiplicación
A / B ó A ./ B	División
A ^ B ó A .^ B	Potencia
A ' ó A .'	Transposición

Finalmente, MATLAB contiene una completa librería con más de 400 funciones, que permiten realizar operaciones lógicas, matemáticas (básicas y especializadas), de ordenación, de representación gráfica (en dos y tres dimensiones), etc.

b.2. Bifurcación condicional

Para indicar una bifurcación condicional MATLAB posee el mandato IF, cuya sintaxis es la siguiente:

IF *condición_1*
    mandatos
ELSEIF *condición_2*
    mandatos
[ELSEIF *condición_3*
    mandatos ...]
ELSE
    mandatos
END

Pueden utilizarse los siguientes operadores lógicos y relacionales:

TABLA 9. *Operadores lógicos y relacionales.*

	OPERADOR	SIGNIFICADO
Operadores lógicos	& | ~	Y O No
Operadores relacionales	< <= > >= == ~=	Inferior Inferior o igual Superior Superior o igual Igual Distinto

b.3. Estructuras de repetición

Las dos estructuras de repetición que pueden utilizarse en MATLAB son los bucles FOR y WHILE. La sintaxis de la iteración FOR es la siguiente:

```
FOR i = 1 : n ,
      mandatos que requieren ejecución iterativa
END
```

Para indicar un incremento (o decremento) determinado, debe indicarse un valor numérico positivo (o negativo) del siguiente modo:

```
FOR i = 1 : incremento : n ,
      mandatos que requieren ejecución reiterada
END
```

Por su parte, la sintaxis de la estructura WHILE es la siguiente:

```
WHILE condición
      mandatos que requieren ejecución reiterada
      mientras se cumple la condición indicada
END
```

Para finalizar la ejecución reiterada antes de que se cumpla la condición de salida, debe utilizarse el mandato BREAK (generalmente dentro de una bifurcación condicional):

```
FOR i = 1 : n ,
      mandatos que requieren ejecución reiterada
      IF condición_salida
            BREAK
      END
END
```

c. Cuadros de diálogo para representar familias de experimentos

Además de las estructuras de programación básicas expuestas en el apartado anterior, MATLAB contiene un conjunto de funciones y mandatos que permiten crear aplicaciones completas que funcionen con el interface de usuario gráfico (GUI) característico de MS-Windows, con base en menús y cuadros de diálogos (MATLAB, 1993b).

Para el diseño de los cuadros de diálogo pueden utilizarse los principales tipos de controles existentes en MS-Windows: botones de comando, casillas de verificación, botones de opción, listas de opciones, recuadros de texto y barras de desplazamiento y de selección.

Este tipo de funciones tienen un gran interés para el *toolbox MonteCarlo*, dado que permiten definir cuadros de diálogo que encierren familias concretas de experimentos Monte Carlo. En el apartado 3.3.6. se ilustran las posibilidades de este enfoque, a través de la presentación del programa DMUESTRAL, que permite estudiar la distribución muestral de un estimador manipulando diferentes aspectos, como la función de distribución de probabilidad de la población, el número de muestras aleatorias que se generan y el tamaño de las muestras. Los cambios en los parámetros se ven reflejados al instante en el histograma que muestra la forma de la distribución muestral.

3.3. ESPECIFICACIÓN Y DISEÑO DE MÓDULOS: EL *TOOLBOX MONTECARLO*

En este apartado se analizan y describen los diferentes módulos que conforman la "caja de herramientas" *MonteCarlo* para llevar a cabo experimentos de muestreo Monte Carlo. El análisis de los mandatos programados se realiza a partir de los algoritmos y procedimientos expuestos en el capítulo 2 y de la revisión de las instrucciones que incorporan los lenguajes de programación específicos para simulación como DATASIM (Bradley, 1988, 1989), GAUSS (1992), Resampling Stats (Simon y Bruce, 1991a, 1991b, 1992), SIMSTAT (Péladeau y Lacouture, 1993) y StatLab (Hodges et al., 1975; Farnum, 1991).

La exposición de cada módulo se realiza a dos niveles: (1) presentación de un algoritmo creado a partir de los mandatos básicos de *MonteCarlo*, que ilustra el proceso de resolución del problema y, (2) presentación del macro-mandato (que en este trabajo denominaremos "algoritmo encapsulado") que resuelve el problema mediante una única instrucción parametrizada.

En la última sección de este apartado se presentará el tercer nivel de actuación previsto en *MonteCarlo*, correspondiente a los subprogramas (cuadros de diálogo en terminología MS-Windows) que encierran las distintas familias de experimentos Monte Carlo.

La *Figura 5* muestra un organigrama de los módulos del *toolbox MonteCarlo*, así como la conexión de MATLAB con otras herramientas informáticas utilizadas para simulación.

3.3.1. Muestreo aleatorio

Los algoritmos que se construyen para llevar a cabo un experimento de muestreo Monte Carlo se inician con la especificación del mecanismo de generación de las nuevas muestras. Este mecanismo generador puede ser la definición teórica de una(s) distribución(es) de probabilidad con parámetros conocidos, o bien una muestra de datos observados. En este segundo caso, se deberá decidir si el *remuestreo* (nótese que utilizamos este término cuando el mecanismo generador es la propia muestra de datos original) se realiza sin o con reposición.

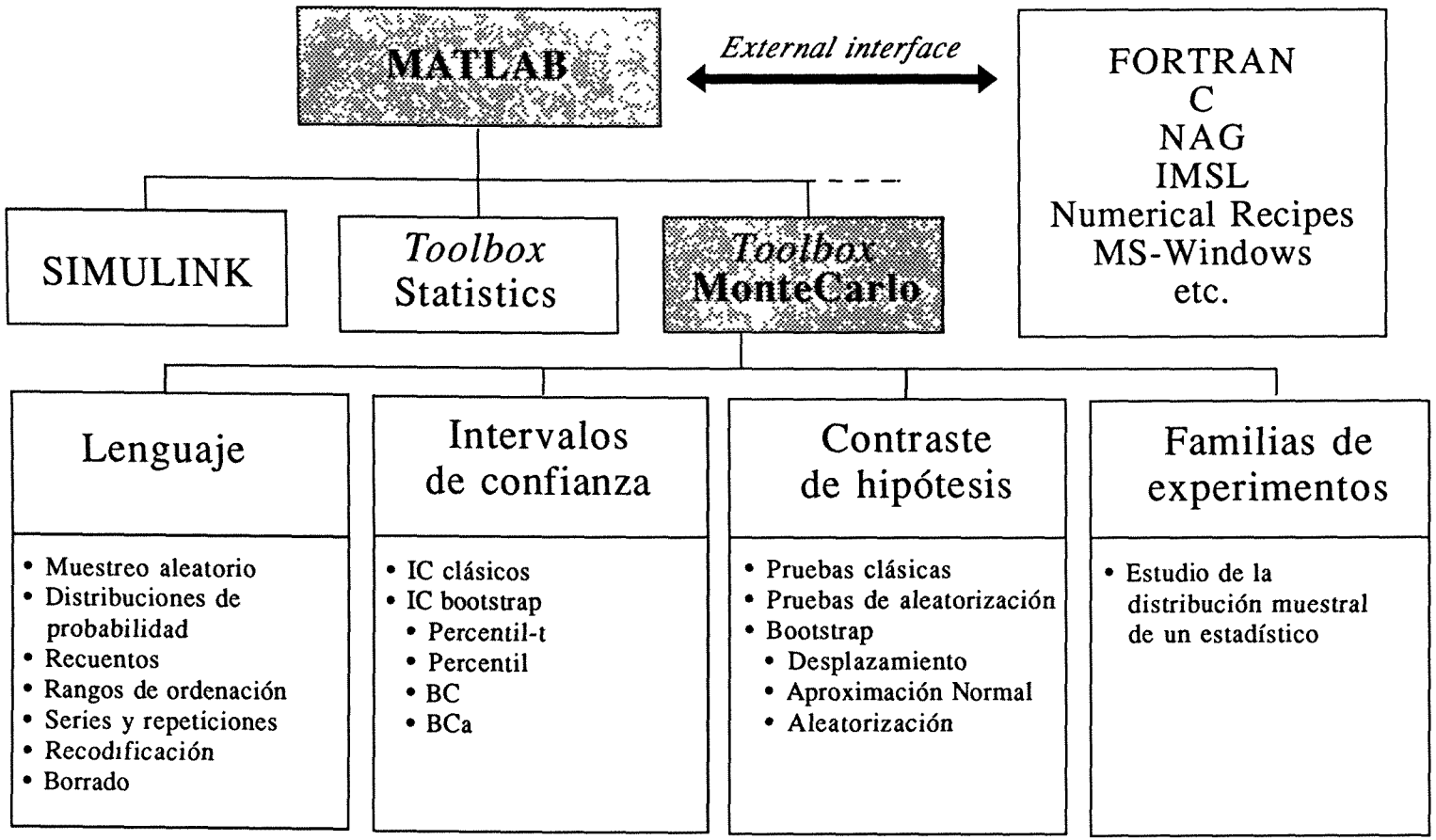


FIGURA 5. *MATLAB. Módulos del toolbox MonteCarlo.*

Por tanto, el *toolbox MonteCarlo* debe proveer tres mandatos distintos para realizar el muestreo: (1) el mandato DISTRND, que permite generar muestras de datos aleatorios con distribuciones de densidad de probabilidad conocidas, (2) el mandato RANDOMIZE, que genera una muestra de datos mediante la permutación aleatoria de los datos de la muestra original (remuestreo sin reposición) y, (3) el mandato RESAMPLE, que genera una nueva muestra de datos escogidos de forma aleatoria y con reposición a partir de la muestra original. Por otro lado, *MonteCarlo* también permite inicializar la semilla de los generadores de números aleatorios con el mandato SEED, con la finalidad de poder replicar exactamente los experimentos realizados.

a. Muestreo aleatorio a partir de una distribución de probabilidad conocida

El mandato **DISTRND** genera muestras de datos mediante la especificación del tipo de distribución de densidad de probabilidad de la variable y sus parámetros. Además, como último argumento, debe indicarse el tamaño de la muestra. Por ejemplo, la sintaxis para generar una muestra de 30 datos distribuidos aleatoriamente según una ley normal con media 100 y desviación estándar 15 es la siguiente:

DISTRND('Normal', 100, 15, 30)

Se han creado también los mandatos DISTCDF, DISTPDF y DISTINV, que permiten obtener la probabilidad acumulada, la densidad de probabilidad, y la inversa de la distribución acumulada, respectivamente. La sintaxis de los mandatos DISTCDF, DISTPDF y DISTINV es la siguiente:

DISTCDF('Distribución', valor, par1, par2, par3)

DISTPDF('Distribución', valor, par1, par2, par3)

DISTINV('Distribución', probabilidad, par1, par2, par3)

La *Tabla 10* muestra las etiquetas y los parámetros que se deben especificar para cada una de las funciones de distribución implementadas actualmente en estos mandatos.

TABLA 10. *Funciones de distribución de probabilidad implementadas en los mandatos DISTRND, DISTCDF, DISTPDF y DISTINV.*

DISTRIBUCIÓN	ETIQUETA	PARÁM. 1	PARÁM. 2	PARÁM. 3
Beta [1]	'beta'	A	B	----------
Binomial [1]	'binomial'	nº de ensayos	probabilidad	----------
χ^2 [1]	'chi2'	gl	----------	----------
Exponencial [1]	'exp'	lambda	----------	----------
F [1]	'f'	gl1	gl2	----------
Gamma [2]	'gamma'	A	B	----------
Geométrica [1]	'geom'	probabilidad	----------	----------
Hipergeométrica [1]	'hiperg'	tamaño población	nº ítem con característica	nº muestras extraídas
Normal [3]	'normal'	media	desviación estándar	----------
Poisson [1]	'poisson'	lambda	----------	----------
T [1]	't'	gl	----------	----------
Uniforme continua [4]	'unic'	mínimo	máximo	----------
Uniforme discreta [4]	'unid'	mínimo	máximo	----------
Weibull [1]	'weibull'	A	B	----------

[1] Devroye (1986).

[2] Cody (1975) y Devroye (1986).

[3] Forsythe, Malcom y Moler (1977).

[4] Park y Miller (1988).

b. Remuestreo aleatorio sin reposición

El mandato **RANDOMIZE** genera una nueva muestra aleatoria de datos mediante remuestreo sin reposición (por permutación) a partir de los datos de la muestra original.

La sintaxis completa del mandato RANDOMIZE es la siguiente:

RANDOMIZE(X [, nPerm [, n]])

El único argumento obligatorio que debe indicarse es el nombre del vector o matriz que contiene los datos originales. Si el argumento es un vector columna, la permutación se realiza a nivel de las filas (casos); si el argumento es una matriz (o un vector fila), la permutación se realiza a nivel de las columnas (variables) de cada vector fila, a no ser que se indique el valor *1* en el segundo argumento (que es opcional), en cuyo caso, la permutación se realizará a nivel de los vectores fila de la matriz indicada en el argumento 1. Como tercer argumento puede indicarse el tamaño de la nueva muestra (si se omite este argumento, se genera una muestra con el mismo tamaño que la muestra original).

La tabla siguiente muestra los posibles resultados de RANDOMIZE:

TABLA 11. *Posibles resultados del mandato RANDOMIZE.*

MANDATO	MATRIZ DE ENTRADA	RESULTADO
RANDOMIZE(X)	$X = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$	$R = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$
RANDOMIZE(X)	$X = [\ 1\ 2\ 3\ ]$	$R = [\ 2\ 3\ 1\ ]$
RANDOMIZE(X)	$X = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \end{bmatrix}$	$R = \begin{bmatrix} 5 & 1 \\ 2 & 6 \\ 7 & 3 \end{bmatrix}$
RANDOMIZE(X, 1)	$X = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \end{bmatrix}$	$R = \begin{bmatrix} 2 & 6 \\ 3 & 7 \\ 1 & 5 \end{bmatrix}$

c. Remuestreo aleatorio con reposición

El mandato **RESAMPLE** genera una nueva muestra de datos mediante remuestreo aleatorio con reposición a partir de los datos de la muestra original.

La sintaxis completa del mandato RESAMPLE es la siguiente:

RESAMPLE(X [, nRes [, n]])

El único argumento obligatorio que debe indicarse es el nombre del vector o matriz que contiene los datos originales. Si el argumento es un vector columna, el remuestreo se realiza a nivel de las filas (casos); si el argumento es una matriz (o un vector fila), el remuestreo se realiza a nivel de las columnas (variables) de cada vector fila, a no ser que se indique el valor 1 en el segundo argumento (que es opcional), en cuyo caso, el remuestreo se realizará a nivel de los vectores fila de la matriz indicada en el argumento 1. Como tercer argumento puede indicarse el tamaño de la nueva muestra (si se omite este argumento, se genera una muestra con el mismo tamaño que la muestra original).

La tabla siguiente muestra los posibles resultados de RESAMPLE:

TABLA 12. *Posibles resultados del mandato RESAMPLE*

MANDATO	MATRIZ DE ENTRADA	RESULTADO
RESAMPLE(X)	$X = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$	$R = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}$
RESAMPLE(X)	$X = [\ 1\ 2\ 3\ ]$	$R = [\ 2\ 3\ 2\ ]$
RESAMPLE(X)	$X = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \end{bmatrix}$	$R = \begin{bmatrix} 1 & 5 \\ 2 & 2 \\ 7 & 3 \end{bmatrix}$
RESAMPLE(X, 1)	$X = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \end{bmatrix}$	$R = \begin{bmatrix} 2 & 6 \\ 3 & 7 \\ 2 & 6 \end{bmatrix}$

d. Inicialización de la semilla para los generadores de números aleatorios

La semilla de los generadores de números aleatorios, al igual que los algoritmos concretos que utilizan estos generadores deben indicarse siempre en las publicaciones para poder replicar correctamente los experimentos. El mandato **SEED** permite inicializar la semilla de todos los generadores de secuencias de números aleatorios que maneja *MonteCarlo*. La sintaxis es la siguiente:

SEED(nº_entero_positivo)

3.3.2. Mandatos para construir algoritmos Monte Carlo

En este apartado se describen los mandatos de *MonteCarlo* que, junto con los mandatos de (re)muestreo expuestos en el apartado anterior y los mandatos propios de MATLAB para manejar matrices (y realizar cálculos sobre ellas), estructuras de repetición y bifurcaciones condicionales, completan el juego de instrucciones básicas para construir algoritmos que permitan llevar a cabo los experimentos Monte Carlo. Por motivos prácticos, sólo se describirá la finalidad y la sintaxis de los mandatos; en los apartados siguientes se ilustrará su utilización en los algoritmos de construcción de intervalos de confianza y de contraste de hipótesis.

a. Recuentos y significación estadística

Los mandatos **COUNT** y **SIGRES** permiten efectuar recuentos y obtener el grado de significación en las pruebas de contraste de hipótesis.

La sintaxis del mandato COUNT es la siguiente:

COUNT(X, mín [,máx])

El primer argumento es el vector o matriz sobre el que se desea realizar el recuento. El segundo y tercer argumentos indican el rango de valores a contar. El corchete "[ ]" indica que el tercer argumento es opcional. Si no se indica el argumento *máx*, entonces se recontarán sólo los valores iguales a *mín*. Se pueden utilizar los valores *−inf* y *+inf* para indicar los valores

mínimo y máximo, recontándose en este caso los valores inferiores o iguales o superiores o iguales a *mín* o *máx*, respectivamente.

El mandato SIGRES permite obtener el grado de significación en las pruebas de contraste de hipótesis por (re)muestreo Monte Carlo. La sintaxis es la siguiente:

SIGRES(X, valor, dirección)

donde *X* es el vector que habitualmente contiene los resultados de la aplicación del estadístico de contraste en cada una de las muestras simuladas, *valor* es el valor del estadístico en la muestra de datos original, y *dirección* es el tipo de hipótesis alternativa que se desea obtener (unilateral superior o inferior, o bilateral). El argumento *dirección* permite indicar tres valores distintos: *1* para recontar los valores superiores o iguales, *-1* para recontar los valores inferiores o iguales, o *0* para recontar los valores superiores o iguales en valor absoluto. Una vez realizado el recuento, SIGRES aplica el siguiente cálculo:

$$(NSI + 1) / (NSIM + 1)$$

donde NSI es el número de valores que cumplen la condición especificada en el argumento *dirección*, y NSIM el número total de valores en el vector especificado como primer argumento. Como se puede observar, tanto el numerador como el denominador se incrementan en 1 para incluir el valor del estadístico en la muestra original.

b. Recodificación de valores

El mandato **RECODE** permite cambiar unos valores por otros. La sintaxis de este mandato es la siguiente:

RECODE(X, nuevo_valor, mín [, máx])

El primer argumento es el vector o matriz sobre el que se desea realizar el recuento. El segundo argumento es el nuevo valor por el que se desea reemplazar los valores que se encuentren entre *mín* y *máx* (ambos incluidos). Si no se indica el argumento *máx*, entonces sólo se cambiarán

los valores iguales a *mín.* Se pueden utilizar los valores *−inf* y *+inf* para indicar los valores mínimo y máximo, recodificándose en este caso los valores inferiores o iguales o superiores o iguales a *mín* o *máx*, respectivamente.

c. Obtención de rangos de ordenación

Los mandatos **RANKS** y **TAGSORT** devuelven las posiciones de los valores ordenados. La diferencia entre estos dos mandatos estriba en el hecho de que RANKS devuelve un vector con las posiciones que ocuparían los valores del vector original si fueran ordenados, mientras que TAGSORT ordena los valores y devuelve un vector con las posiciones originales de los valores en el vector original. La ordenación se realiza siempre de forma ascendente, y los empates se resuelven por interpolación lineal simple en el mandato RANKS. El único argumento que debe indicarse en ambos casos es el nombre del vector de valores *X*.

La tabla siguiente ilustra el resultado de ambos mandatos:

TABLA 13. *Posibles resultados de los mandatos RANKS y TAGSORT.*

MANDATO	MATRIZ DE ENTRADA	RESULTADO
RANKS(X)	$X = \begin{bmatrix} 8 \\ 6 \\ 5 \\ 6 \end{bmatrix}$	$R = \begin{bmatrix} 4 \\ 2.5 \\ 1 \\ 2.5 \end{bmatrix}$
TAGSORT(X)	$X = \begin{bmatrix} 8 \\ 6 \\ 5 \\ 6 \end{bmatrix}$	$R = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 1 \end{bmatrix}$

d. Evaluación de series y repeticiones

Los mandatos **DUPLICATES** y **RUNS** devuelven el número de repeticiones de valores no consecutivas o consecutivas (series),

respectivamente:

DUPLICATES(X [, mín [, máx]])

RUNS(X [, mín [, máx]])

El primer argumento es el vector o matriz sobre el que se desea realizar el recuento. El segundo y tercer argumentos permiten indicar la longitud mínima y máxima de las repeticiones. Si no se indica el argumento *máx*, entonces se recontará sólo el número de repeticiones iguales a *mín*. Se pueden utilizar los valores *−inf* y *+inf* para indicar los valores mínimo y máximo, recontándose en este caso los valores con número de repeticiones inferiores o iguales o superiores o iguales a *mín* o *máx*, respectivamente.

La tabla siguiente ilustra el resultado de ambos mandatos:

TABLA 14. *Posibles resultados de los mandatos DUPLICATES y RUNS.*

MANDATO	MATRIZ DE ENTRADA	RESULTADO
DUPLICATES(X, 2, +inf)	$X = \begin{bmatrix} 8 \\ 6 \\ 5 \\ 6 \\ 8 \\ 8 \end{bmatrix}$	R = 2
RUNS(X, 2, +inf)	$X = \begin{bmatrix} 8 \\ 6 \\ 5 \\ 6 \\ 8 \\ 8 \end{bmatrix}$	R = 1

e. Borrado de valores

MATLAB permite eliminar un elemento (o un vector) de una matriz igualando su valor a "[ ]" (que indica el valor nulo). *MonteCarlo* incluye los

mandatos **DEDUP** y **WEED**, que permiten borrar grupos de vectores fila de una matriz. El mandato DEDUP borra las filas que contengan valores duplicados, mientras que el mandato WEED borra las filas que contengan algún valor dentro de un rango especificado.

La sintaxis de estos mandatos es:

DEDUP(X)

WEED(X, mín [,máx])

donde *X* es el vector o matriz del cual se desea borrar sus elementos, y el segundo y tercer argumentos indican el rango de valores a eliminar (ambos inclusive). Si no se indica el argumento *máx*, entonces se eliminarán sólo las filas con algún valor igual a *mín*. Se pueden utilizar los valores *−inf* y *+inf* para indicar los valores *mín* y *máx*, borrándose en este caso los valores inferiores o iguales o superiores o iguales a *mín* o *máx*, respectivamente.

3.3.3. Indices estadísticos y gráficos

MATLAB contiene los mandatos correspondientes a los gráficos e índices estadísticos básicos. *MonteCarlo* completa estos mandatos añadiendo, además, los gráficos e índices estadísticos descriptivos propios del análisis exploratorio de datos (EDA):

TABLA 15. *Mandatos para gráficos estadísticos.*

MANDATO	INDICE ESTADÍSTICO	ORIGEN
BAR(X [, aPos])	Gráfico de barras	Matlab
BOXPLOT(X [, ...])	Gráfico de caja	Statistics
HIST(X, [, aPos])	Histograma	Matlab
NORMPLOT(X)	Normal probability plot	Statistics
PLOT(X [, Y, [, char]])	Gráfico lineal 2-D	Matlab
PLOT3(X, Y, Z [, char])	Gráfico en 3-D	Matlab
QQPLOT(X, Y [, vprct])	Quantile-quantile plot	Statistics
STAIRS(X [, aPos])	Gráfico de escalera	Matlab
STEMLEAF(X [, nR ...])	Diagrama de tallo y hojas	MonteCarlo
TUKEYDISP(X)	Diagrama de Tukey	MonteCarlo

TABLA 16. *Mandatos para índices estadísticos.*

MANDATO	INDICE ESTADÍSTICO	ORIGEN
CORR(X [, Y])	Coef. de correlación r de Pearson	MonteCarlo
COV(X [, Y])	Covariancia	MonteCarlo
FSPREAD(X)	Rango intercuartil basado en cuartos	MonteCarlo
GEOMEAN(X)	Media geométrica	Statistics
HARMMEAN(X)	Media harmónica	Statistics
IQR(X)	Rango intercuartil	Statistics
KURTOSIS(X)	Apuntamiento	MonteCarlo
MAD(X)	Media de las desviaciones absolutas	Statistics
MANDREWS(X [, c])	M-estimador de loc. de Andrews	MonteCarlo
MEAN(X)	Media aritmética	Matlab
MEDIAN(X)	Mediana	Matlab
MHAMPEL(X [, A, B, C])	M-estimador de loc. de Hampel	MonteCarlo
MHUBER(X [, c])	M-estimador de loc. de Huber	MonteCarlo
MODE(X)	Moda	MonteCarlo
MTUKEY(X [, c])	M-estimador de loc. de Tukey	MonteCarlo
NPRCTILE((X, valor)	Percentil correspondiente a un valor	MonteCarlo
ODD(X, mín [, máx])	Odd	MonteCarlo
PRCTILE(X, valor)	Percentil	Statistics
PROP(X, mín [, máx])	Proporción	MonteCarlo
RANGE(X)	Rango	Statistics
RMAD(X)	Mediana de las desv. absolutas	MonteCarlo
RVARCOEF(X)	Coeficiente de variación robusto	MonteCarlo
SEMEDIAN(X)	Error estándar de la mediana	MonteCarlo
SKEWNESS(X)	Asimetría	MonteCarlo
STD(X)	Desviación estándar	Matlab
TRIMEAN(X)	Trimedia	MonteCarlo
TRIMMEAN(X, %)	Media recortada	Statistics
VAR(X)	Variancia	Statistics
VARCOEF(X)	Coeficiente de variación	MonteCarlo

3.3.4. Intervalos de confianza

Como se ha expuesto en el capítulo 2, una de las principales aplicaciones de las técnicas de remuestreo es la estimación por intervalo de parámetros poblacionales. A continuación se exponen los algoritmos y mandatos implementados en el *toolbox MonteCarlo* para realizar dichas estimaciones. Se han programado los mandatos que permiten obtener los intervalos de confianza clásicos y los intervalos de confianza *bootstrap*. Además, en estos últimos, se han implementado diferentes algoritmos, todos ellos revisados en el capítulo anterior, que permiten aumentar la precisión y la eficiencia de las estimaciones: la estimación del error estándar del estadístico mediante un doble nivel *bootstrap* o *jackknife*, y el remuestreo balanceado.

a. Intervalos de confianza clásicos

Los intervalos de confianza clásicos implementados en *MonteCarlo* son los siguientes:

TABLA 17. *Mandatos para estimación por intervalo clásicos.*

MANDATO	DESCRIPCIÓN
ICMEAN(X, α/2)	IC de una media (con base en la distribución t)
ICMEDIAN(X, α/2)	IC de una mediana (con base en la distribución t)
ICPROP(prop, n, α/2)	IC de una proporción (con base en distribución F)
ICVAR(X, α/2)	IC de una variancia (con base en la distribución χ^2)

b. Intervalos de confianza bootstrap

Se han programado cuatro macro-mandatos en *MonteCarlo* que realizan la estimación *bootstrap* de los límites de un intervalo de confianza, según los métodos percentil-t, percentil, percentil con corrección del sesgo

(BC) y percentil con corrección acelerada del sesgo (BCa). Estos mandatos permiten, por otro lado, visualizar gráficamente el proceso constructivo de la distribución muestral del estadístico durante el proceso de remuestreo, así como la variación que se produce en los límites del intervalo de confianza.

La tabla siguiente presenta los nombres y sintaxis de estos mandatos:

TABLA 18. *Mandatos para estimación por intervalo bootstrap.*

MANDATO	DESCRIPCIÓN
ICBT(X, B, α/2, 'Estadístico' [, xEE, nBalance, nDib, nHist, nTpo])	IC *bootstrap*: método percentil-t
ICBP(X, B, α/2, 'Estadístico' [, nBalance, nDib, nHist, nTpo])	IC *bootstrap*: método percentil
ICBBC(X, B, α/2, 'Estadístico' [, nBalance, nDib, nHist, nTpo])	IC *bootstrap*: método BC
ICBBCA(X, B, α/2, 'Estadístico' [, a, nBalance, nDib, nHist, nTpo])	IC *bootstrap*: método BCa

Los argumentos de estos mandatos indican lo siguiente:

X Es el nombre del vector columna o matriz que contiene los datos observados en la muestra.

B Es el número de remuestreos a realizar.

'Estadístico' Es el nombre de la función que calcula el estadístico. Se puede utilizar cualquiera de los índices estadísticos presentados en el apartado anterior, o los estadísticos de contraste que se detallan más adelante. El nombre de la función se debe escribir entre comillas simples (' '), utilizando el carácter ventana (#) para representar la matriz de datos sobre la cual se desea realizar los cálculos. Ejemplos: 'mean(#)', 'corr(#)', 'ttestg(#, 0.05)', 'mhampel(#, 1.7, 3.4, 8.5)', etc.

[α/2] Es la precisión unilateral del intervalo.

[*xEE*] Indica el tipo de estimación del error estándar que se desea, pudiéndose especificar una fórmula de cálculo, un entero positivo que especifique el número de remuestreos a realizar para obtener la estimación *bootstrap* del error estándar (lo cual representa un doble nivel *bootstrap*), ó *-1* en el caso de que se desee utilizar la estimación *jackknife* del error estándar para cada una de las remuestras.

[*nBalance*] El valor *0* indica que se realizará un remuestreo simple. El valor *1* indica que se aplicará el algoritmo de muestreo balanceado propuesto por Gleason (1988).

[*nDib*] Especifica cada cuántos remuestreos se desea actualizar el histograma que ilustra la construcción de la distribución muestral *bootstrap*, así como los valores de la estimación por intervalo. Indicando el valor *-1* se suprime la visualización gráfica y sólo se muestra el resultado final.

[*nHist*] Indica el número de intervalos de clase para el eje de abcisas del histograma. Por defecto se realizan *50* intervalos de clase.

[*nTpo*] El valor *1* especifica que se vaya mostrando el tiempo transcurrido en el proceso de obtención del intervalo.

Las *Figura 6* muestra el aspecto de la ventana que aparece en pantalla durante la ejecución de estos mandatos, concretamente en el caso del intervalo de confianza *bootstrap* por el método percentil para el coeficiente de correlación:

```
ICBP( X, 5000, 0.025, 'corr( # )', 1, 100, 99, 1 )
```

Con esta ventana se ilustra de forma progresiva la evolución de los remuestreos, la construcción del histograma de la distribución muestral *bootstrap*, el valor del estadístico y el tamaño de la muestra original, la estimación puntual y por intervalo, el sesgo del estimador, y el tipo de remuestreo que se utiliza.

En los siguientes apartados se presentan los algoritmos correspondientes a cada uno de estos procedimientos. Estos algoritmos están escritos utilizando los mandatos *MonteCarlo* y MATLAB descritos hasta el momento.

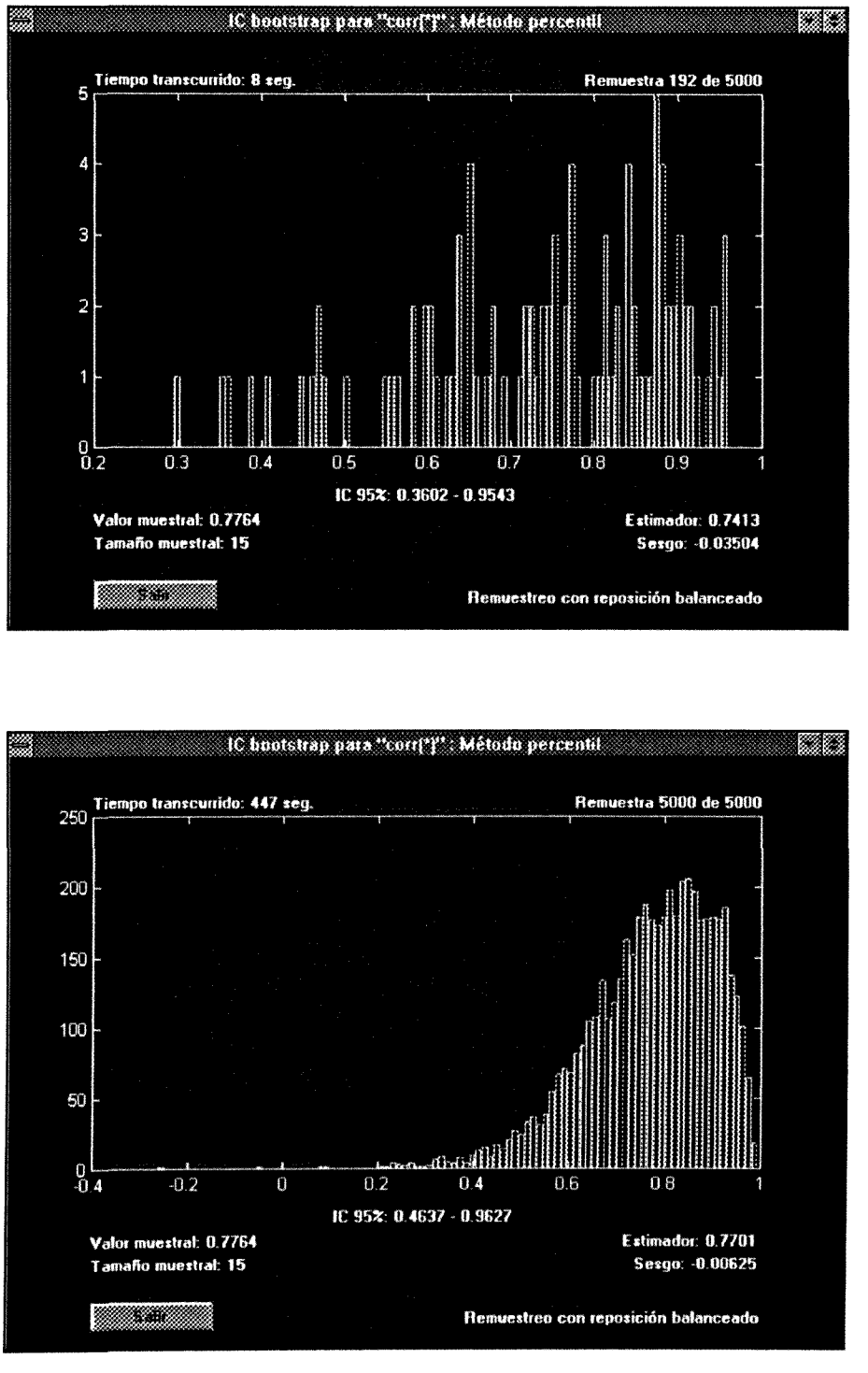


FIGURA 6. *Ventanas de los mandatos IC bootstrap.*

b.1. Método percentil-t

El algoritmo para obtener los límites del intervalo de confianza por el método percentil-t, por ejemplo, del parámetro media es el siguiente:

```
mo = mean( Xo );
stdo = std( Xo );
for b = 1 : B,
      Rb = resample( Xo );
      Xb = ( mean( Rb ) - mo ) / std( Rb );
end
prc = prctile( Xb, [2.5, 97.5] );
ic = [mo-(prc(2)*stdo), mo-(prc(1)*stdo) ]
hist( Xb .* stdo + meano )
```

FIGURA 7. *Intervalo de confianza bootstrap: método percentil-t*

b.1.1. Estimación bootstrap del error estándar

Cuando no se posee un estimador estable y consistente del error estándar del estadístico, se puede utilizar el estimador *bootstrap*, lo cual requiere un doble nivel de remuestreo. El siguiente algoritmo presenta este procedimiento para el caso de la estimación por intervalo del coeficiente de correlación de Pearson:

```
mo = mean( Xo );
for b = 1 : B,
      Rb = resample( Xo );
      for d = 1:50,
            Rd = resample( Rb );
            stdd( d ) = std( Rd );
      end
      stdb( b ) = mean( stdd );
      Xb( b ) = ( mean( Rb ) - mo ) / stdb( b );
end
stdb = mean( stdb );
prc = prctile( Xb, [2.5, 97.5] );
ic = [mo-(prc(2)*stdb), mo-(prc(1)*stdb) ]
hist( Xb .* stdb + meano )
```

FIGURA 8. *Método percentil-t: estimación bootstrap del error estándar.*

b.1.2. Estimación jackknife del error estándar

Otra posibilidad para estimar el error estándar del estadístico es utilizar la estimación *jackknife* como solución intermedia que tiene un menor coste computacional. El algoritmo es el siguiente:

```
n = length( Xo );
mo = mean( Xo );
for b = 1 : B,
      Rb = resample( Xo );
      for j = 1:n,
            Rj = Rb;
            Rj( j ) = [ ];
            stdj( j ) = std( Rj );
      end
      stdj( b ) = mean( stdj );
      Xb( b ) = ( mean( Rb ) - mo ) / stdj( b );
end
stdj = mean( stdj );
prc = prctile( Xb, [2.5, 97.5] );
ic = [mo-(prc(2)*stdj), mo-(prc(1)*stdj) ]
hist( Xb .* stdj + meano )
```

FIGURA 9. *Intervalo de confianza bootstrap: método percentil-t, con estimación jackknife del error estándar del estadístico.*

b.1.3. Remuestreo balanceado

A continuación se presenta el algoritmo propuesto por Gleason (1988) para realizar el remuestreo balanceado (Davison, Hinkley y Schechtmann, 1986). Este procedimiento permite aumentar la eficiencia de la estimación *bootstrap* equiparando la probabilidad de aparición de cada uno de los datos al final del proceso de remuestreo.

```
% Inicialización del algoritmo de balanceo
n = length( Xo );
c = zeros( B, 1 );
p = c;
M = 2^31 - 1;
for i = 1 : n,
      c( i ) = B;
      p( i ) = i;
end
k = n;
J = k;
C = B;
t = C / M;
% Estimación bootstrap del IC
mo = mean( Xo );
stdo = mean( Xo );
for b = 1 : B,
      % Iteración para remuestreo balanceado
      for i = 1 : n
            while 1,
                  s = ceil( distrnd('Uniform',0,1,1) * ( M - 1 ) );
                  j = 1 + rem( s, k );
                  if ( s * t < c( j ) ), break, end;
            end
            I = p( j );
            c( j ) = c( j ) - 1;
            if ( j == J )
                  if ( C - c( j ) > fix( B - b + k ) / k )
                        for ii = 1 : k,
                              if ( c( ii ) > c( J ) ), J = ii, end;
                        end
                        C = c( J );
                        t = C / M;
                  end
            end
            if ( c( j ) == 0 )
                  if ( J == k )
                        J = j;
                        p( j ) = p( k );
                        c( j ) = c( k );
                        k = k - 1;
                  end
            end
            % Se elige el elemento Xo( I, : )
            Rb( i, : ) = Xo( I, : );
      end
      Xb( b ) = ( mean( Rb ) - mo ) / std( Rb );
end
prc = prctile( Xb, [2.5, 97.5] );
ic = [mo-(prc(2)*stdo), mo-(prc(1)*stdo) ]
```

FIGURA 10. *Intervalo de confianza boostrap: percentil-t con remuestreo balanceado.*

El algoritmo anterior se puede utilizar en cualquiera de los métodos percentil que se presentan a continuación.

b.2. Método percentil

El algoritmo del método percentil para obtener los límites del intervalo de confianza, por ejemplo, del parámetro de localización M-estimador de Andrews (Andrews et al., 1972) es el siguiente:

```
mo = mandrews( Xo );
for b = 1 : B,
      Rb = resample( Xo );
      Xb( b ) = mandrews( Rb );
end
ic = prctile( Xb, [2.5, 97.5] );
hist( Xb .* stdo + meano )
```

FIGURA 11. *Intervalo de confianza bootstrap: método percentil.*

b.3. Método BC

El algoritmo del método percentil con corrección del sesgo (método BC) para obtener los límites del intervalo de confianza, por ejemplo, del parámetro variancia es el siguiente:

```
varo = var( Xo );
zo = distinv( 'Normal', 0, 1, 0.975 );
for b = 1 : B,
      Rb = resample( Xo );
      Xb( b ) = var( Rb );
end
zb = distinv( 'Normal', 0, 1, count( Xb, -inf, varo ) / B );
bcinf = distcdf( 'Normal', 0, 1, (2 * zb) - zo );
bcsup = distcdf( 'Normal', 0, 1, (2 * zb) + zo );
ic = prctile( Xb, [bcinf*100, bcsup*100] );
hist( Xb )
```

FIGURA 12. *Intervalo de confianza bootstrap: método BC.*

b.4. Método BCa

El algoritmo del método BCa de Efron para obtener los límites del intervalo de confianza, por ejemplo, de una proporción es el siguiente:

```
po = prop( Xo, 1 );
zo = distinv( 'Normal', 0, 1, 0.975 );
for b = 1 : B,
      Rb = resample( Xo );
      Xb( b ) = prop( Rb, 1 );
end
zb = distinv( 'Normal', 0, 1, count( Xb, po, +inf ) / B );
bcinf = distcdf( 'Normal', 0, 1, zb + ((zb-zo) / (1 - a*(zb-zo))));
bcsup = distcdf( 'Normal', 0, 1, zb + ((zb+zo) / (1 - a*(zb+zo))));
ic = prctile( Xb, [bcinf*100, bcsup*100] );
hist( Xb )
```

FIGURA 13. *Intervalo de confianza bootstrap: método BCa.*

3.3.5. Contraste de hipótesis

a. Pruebas clásicas

Las pruebas de contraste de hipótesis clásicas implementadas en el *toolbox MonteCarlo* son las siguientes:

TABLA 19. *Mandatos para pruebas clásicas de contraste de hipótesis.*

MANDATO	DESCRIPCIÓN
ZTEST(X, μ, σ, α/2)	Prueba z de conformidad de una media observada vs teórica
TTEST(X, μ, α/2)	Prueba t de conformidad de una media observada vs teórica
TTESTG(X, Y, α/2, h)	Prueba t de comparación de 2 medias: grupos independientes
TTESTP(X, Y, α/2, h)	Prueba t de comparación de 2 medias: datos apareados
ZCORR(X, Y, α/2, h)	Prueba de significación del coeficiente de correlación de Pearson
CHI2TEST(X, α/2, h)	Prueba χ^2 de conformidad de repartición observada vs teórica
CHI2TEST2(X ,α/2, h)	Prueba χ^2 de independencia (cálculo de PD, PR y OR si procede)
CHI2MN(X, α/2, h)	Prueba χ^2 de McNemar

Estos mandatos devuelven un vector con los siguientes resultados:

h Variable binaria con valor *0* ó *1* si el resultado es o no significativo al nivel α fijado "a priori", respectivamente.

sig Grado de significación.

ic Intervalo de confianza del estadístico de contraste.

b. Muestreo Monte Carlo

El mandato MTEST de *MonteCarlo* permite realizar una estimación del grado de significación de una prueba de conformidad entre el valor observado y el valor teórico de un parámetro. La sintaxis del mandato MTEST es la siguiente:

MTEST(X, S, 'Estadístico', 'Distribución', P1, P2, P3 [, nDib, nHist, nTpo])

Los argumentos del mandato MTEST (y del resto de mandatos para el contraste de hipótesis que se revisan más adelante) indican lo siguiente:

X — Es el nombre del vector columna que contiene los datos observados en la muestra para la variable *X*.

Y — Es el nombre del vector columna que contiene los datos observados en la muestra para la variable *Y*.

B ó R ó S — Es el número de remuestreos (*B* o *R*) o de muestreos (*S*) a realizar.

'Estadístico' — Es el nombre de la función o la expresión que calcula el estadístico. Se puede utilizar cualquiera de los índices estadísticos revisados anteriormente. El nombre de la función se debe escribir entre comillas simples (' '), utilizando el carácter ventana (#) para representar el vector de datos *X*, y el carácter arroba (@) para representar el vector de datos *Y*. Ejemplos: 'mean(#)', 'mean(#) – mean(@)', 'mean(# – @)', etc.

[α/2] — Es el nivel α unilateral fijado "a priori". El valor por defecto en caso de que no se indique el argumento es 0.025.

[*h*] — El valor *0*, que es el valor por defecto, indica que se realizará una prueba bilateral. El valor *1* indica que se trata de una prueba unilateral por la derecha (superior o igual). El valor *–1* indica una prueba unilateral por la izquierda (inferior o igual).

[*nDib*] — Especifica cada cuántos muestreos o remuestreos se desea actualizar el histograma que ilustra la construcción de la distribución muestral, así como los valores de la estimación del grado de significación y del intervalo de confianza del estadístico. Indicando el valor *–1* se suprime la visualización gráfica y sólo se muestra el resultado final. El valor por defecto es *0*, que indica que se realizará la visualización del histograma y de los resultados sólo al final del proceso.

[*nHist*] — Indica el número de intervalos de clase para el eje de abcisas del histograma. El valor por defecto es *50*.

[*nTpo*] — El valor *1* especifica que se vaya mostrando el tiempo transcurrido en el proceso de estimación.

En los mandatos de contraste de hipótesis de *MonteCarlo*, si se indica el nombre de una variable en el momento de la ejecución, sólo se devuelve el valor dicotómico *0* ó *1* que indica si el resultado es significativo o no significativo, respectivamente. Ejemplo:

h = MTEST(X, 'mean(#)', 'Normal', 100, 15)

Si se indica un vector de resultados del tipo [h, sig, ip, dm] (también se puede indicar [h, sig] o [h, sig, ip]) como, por ejemplo:

[h, sig, ip] = MTEST(X, 'mean(#)', 'Normal', 100, 15)

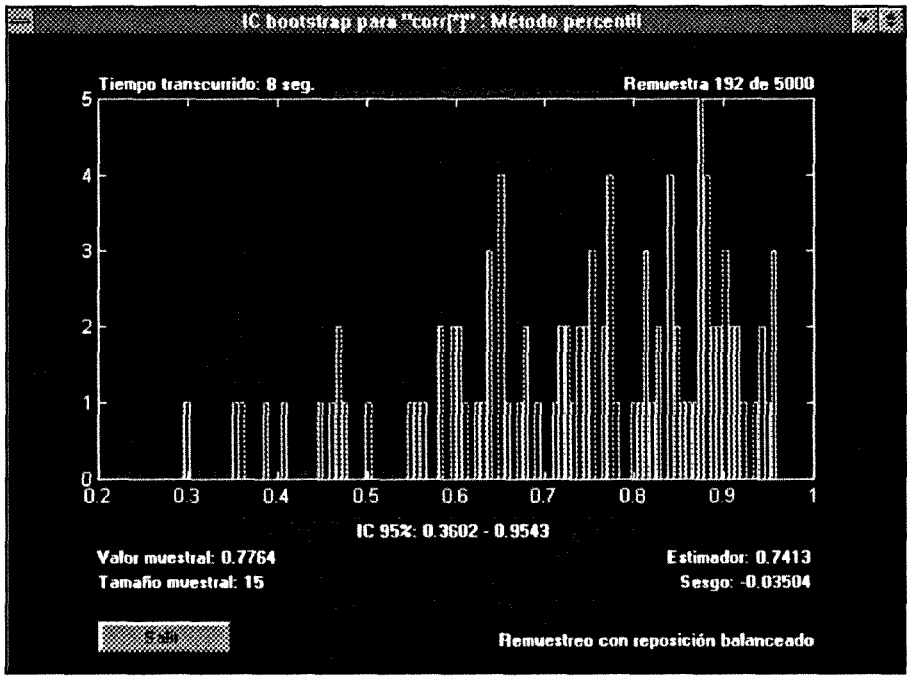
se obtienen los siguientes resultados:

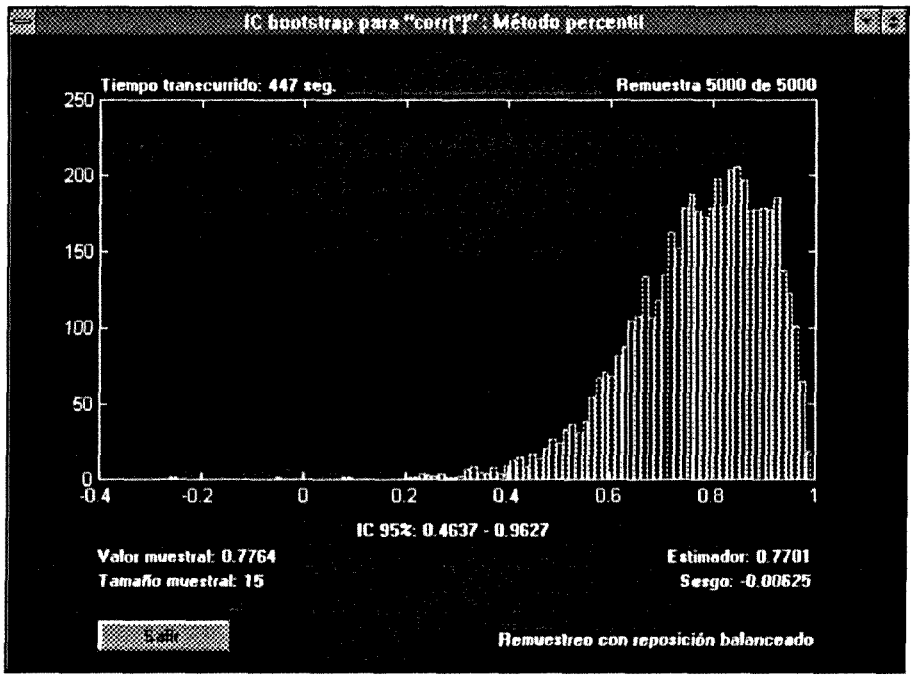
h	Variable binaria con valor *0* ó *1* si el resultado es o no significativo al nivel α fijado "a priori", respectivamente.
sig	Grado de significación.
ip o *ic*	Intervalo de predicción o de confianza del estadístico.
distm	Vector de la distribución muestral del estadístico.

Las siguientes figuras muestran el aspecto de la ventana que aparece en pantalla durante la ejecución de estos mandatos, concretamente de la prueba de conformidad entre una media observada y una teórica por muestreo Monte Carlo de una población Exponencial:

MTEST(X, 1000, 'Exp', 10, 100, 99, 1)

En esta ventana se muestra progresivamente la evolución de la simulación, la construcción del histograma de la distribución muestral del estadístico, el tamaño de la muestra original, el tipo de hipótesis (bilateral o unilateral), la estimación del grado de significación, y los límites del intervalo de predicción o de confianza.





FIGURAS 14-15. *Ejemplos de las ventanas de los mandatos MonteCarlo para el contraste de hipótesis.*

Se puede ejecutar la prueba de conformidad por muestreo Monte Carlo, por ejemplo, de una media observada respecto a una población con distribución normal y parámetros μ y σ conocidos, con el siguiente algoritmo:

```
mo = mean( Xo );
for sim = 1 : NSIM,
      Xs = distrnd( 'Normal', mu, sigma, 30 );
      msim( sim ) = mean( Xs );
end
sig = sigres( msim - mu, mo - mu );
ip = prctile( msim, [2.5, 97.5] );
hist( msim )
```

FIGURA 16. *Prueba de conformidad por muestreo Monte Carlo.*

Como se revisó en el capítulo 2, una de las principales aplicaciones del muestreo Monte Carlo es el estudio de la validez y potencia de las pruebas de contraste de hipótesis clásicas en situaciones en las que no se cumple alguno de sus supuestos. El siguiente algoritmo es un ejemplo de este tipo de experimentos, que permite estudiar la probabilidad de error Tipo I de la prueba *z* de conformidad de una proporción observada respecto a una teórica con probabilidad pequeña (0.03), en muestras de tamaño 100:

```
zcriterio = distinv( 'Normal', 0, 1, 0.975 );
for sim = 1 : 5000,
      Xs = distrnd( 'Binomial', 1, 0.03, 100 );
      ps = prop( Xs, 1 );
      zsim( sim ) = ( ps - 0.03 ) / sqrt( ps*(1-ps)/100 );
end
alphauni = prop( zsim', zcriterio, +inf );
alphabi = prop( abs(zsim'), zcriterio, +inf );
hist( zsim )
```

FIGURA 17. *Estudio de la probabilidad de error Tipo I en una prueba z de conformidad para una proporción con muestra pequeña.*

La ejecución de este experimento ofrece un resultado unilateral de 0.0038 y bilateral de 0.1920, muy alejados del nivel $\alpha = 0.05$ fijado "a priori". La siguiente figura muestra el histograma de las 5000 puntuaciones *z* obtenidas, en el cual se puede constatar la asimetría de la distribución muestral:

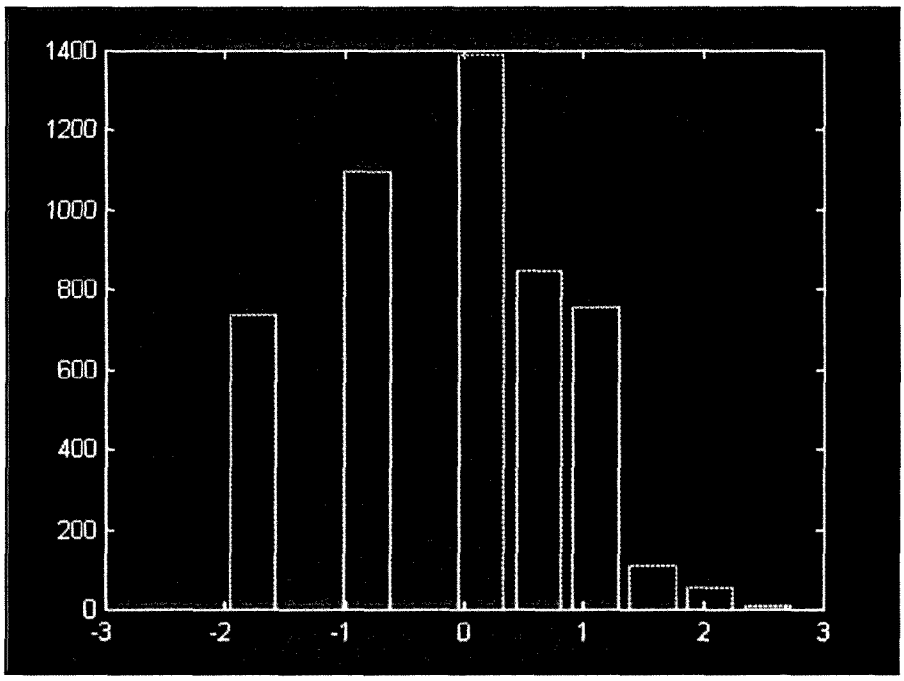


FIGURA 18. *Distribución muestral de las puntuaciones z.*

c. Pruebas de aleatorización aproximadas

Se han programado cuatro macro-mandatos en *MonteCarlo* que realizan una estimación del grado de significación en una prueba de contraste de hipótesis mediante pruebas de aleatorización aproximadas. Paralelamente, estos mandatos permiten visualizar cómo se va construyendo la distribución muestral del estadístico de contraste durante el proceso de remuestreo, así como la variación que se produce en la estimación del grado de significación y en los límites del intervalo de confianza del estadístico.

La siguiente tabla presenta los nombres y sintaxis de estos mandatos:

TABLA 20. *Mandatos para estimación del grado de significación mediante pruebas de aleatorización aproximadas*

MANDATO	DESCRIPCIÓN
RFISHER(X, R, 'Estadístico', μ [, α/2, h, nDib, nHist, nTpo])	Prueba de aleatorización de Fisher (prueba de conformidad).
RTESTG(X, Y, R, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de independencia entre dos variables: diseño de grupos independientes.
RTESTP(X, Y, R, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de independencia entre dos variables: diseño de datos apareados (o de medidas repetidas).
RASOCIA(X, B, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de asociación entre dos variables: diseño de medidas repetidas.

La *prueba de independencia para grupos independientes* se fundamenta en la permutación del vector completo de datos de forma tal que, a partir de [Xo ; Yo], el mandato RANDOMIZE genera un único vector a partir de los dos vectores de datos originales:

```
nx = length( Xo );
ny = length( Yo );
difo = mean( Xo ) - mean( Yo );
for r = 1 : R,
      Cr = randomize( [ Xo ; Yo ] );
      difr( r ) = mean( Cr( 1 : nx ) ) - mean( Cr( nx+1 : ny+nx ) );
end
sig = sigres( difr, difo );
hist( difr )
```

FIGURA 19. *Prueba de independencia por aleatorización: diseño de grupos independientes.*

Por otro lado, en la *prueba de independencia para datos apareados (o medidas repetidas)*, la permutación se realiza entre las parejas de valores de cada caso:

```
n = length( Xo );
difo = mean( Xo - Yo );
for r = 1 : R,
      Cr = randomize( [ Xo , Yo ] );
      difr( r ) = mean( Cr( :, 1 ) - Cr( :, 2 ) );
end
sig = sigres( difr, difo );
hist( difr )
```

FIGURA 20. *Prueba de independencia por aleatorización: diseño de datos apareados (o medidas repetidas).*

Para obtener el grado de significación para una *prueba de asociación entre dos variables en un diseño de medidas repetidas*, la permutación se aplica a uno de los vectores de datos, manteniendo fijo el otro, con lo cual se deshace la posible relación entre las variables, y se simula, por tanto, el modelo de independencia que permite calcular la proporción de resultados con valores superiores o iguales al obtenido al aplicar la prueba en la muestra original.

El algoritmo es el siguiente:

```
n = length( Xo );
asociao = corr( Xo, Yo );
for r = 1 : R,
      Xr = randomize( Xo );
      asociar( r ) = corr( Xr, Yo );
end
sig = sigres( asociar, asociao );
hist( asociar )
```

FIGURA 21. *Prueba de aleatorización para evaluar la relación entre dos variables: diseño de medidas repetidas.*

Por último, se presenta el algoritmo de la prueba clásica de aleatorización de Fisher (1924), que permite evaluar la conformidad entre una muestra y una población teórica con una distribución supuestamente simétrica:

```
Xf = Xo - mu;
mo = mean( Xf );
signos = Xf / abs( Xf );
Xf = abs( Xf );
for r = 1 : R,
      signos = randomize( signos );
      mr( r ) = mean( Xf * signos );
end
sig = sigres( mr, mo );
hist( mr )
```

FIGURA 22. *Prueba de aleatorización de Fisher.*

d. Contraste de hipótesis bootstrap

MonteCarlo contiene nueve macro-mandatos para realizar la estimación *bootstrap* del grado de significación en una prueba de contraste de hipótesis. Estos mandatos permiten aplicar los tres métodos *bootstrap* revisados en el capítulo 2, a saber, el método del desplazamiento de la distribución (con transformación pivotal), el método de la aproximación normal, y el método de la aleatorización *bootstrap*. Se presentan los mandatos y sus algoritmos agrupados para cada uno de los métodos.

d.1. Método del desplazamiento bootstrap

La siguiente tabla presenta los nombres y sintaxis de los tres mandatos *MonteCarlo* que permiten aplicar el método del desplazamiento de la distribución muestral *bootstrap*:

TABLA 21. *Mandatos para estimación del grado de significación por el método bootstrap del desplazamiento de la distribución.*

MANDATO	DESCRIPCIÓN
BTEST(X, B, 'Estadístico', μ [, α/2, h, nDib, nHist, nTpo])	Prueba de conformidad.
BTESTG(X, Y, R, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de independencia entre dos variables: diseño de grupos independientes.
BTESTP(X, Y, R, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de independencia o de asociación entre dos variables: diseño de datos apareados (o de m. repetidas).

El algoritmo de la *prueba de conformidad* por el método del desplazamiento de la distribución es el siguiente:

```
mo = mean( Xo );
for b = 1 : B,
      Xb = resample( Xo );
      mb( b ) = mean( Xb );
end
sig = sigres( mb - mo, mo - mu );
ic = prctile( mb, [2.5, 97.5] );
hist( mb )
```

FIGURA 23. *Prueba de conformidad bootstrap. Método del desplazamiento.*

Para aplicar la prueba de independencia en un diseño con dos grupos independientes, debe realizarse el remuestreo para cada uno de los dos vectores de datos de forma independiente:

```
difo = mean( Xo ) - mean( Yo );
for b = 1 : B,
      Xb = resample( Xo );
      Yb = resample( Yo );
      difb( b ) = mean( Xb ) - mean( Yb );
end
sig = sigres( difb - difo, difo );
ic = prctile( difb, [2.5, 97.5] );
hist( difb )
```

FIGURA 24. *Prueba de independencia bootstrap. Método del desplazamiento: diseño de grupos independientes.*

A diferencia de lo que sucede en las pruebas de aleatorización, el procedimiento *bootstrap* no requiere un esquema de remuestreo distinto para el caso de las pruebas de independencia y las pruebas de asociación entre dos variables en un diseño de datos apareados (o de medidas repetidas). En la técnica *bootstrap*, la unidad de remuestreo aleatorio es el vector fila, con lo que no se deshace el caso; esto se consigue indicando el valor *1* en el segundo argumento del mandato RESAMPLE:

```
difo = mean( Xo - Yo );
for b = 1 : B,
      Cb = resample( [ Xo , Yo ], 1 );
      difb( b ) = mean( Cb( :, 1 ) - Cb( :, 2 ) );
end
sig = sigres( difb - difo, difo );
ic = prctile( difb, [2.5, 97.5] );
hist( difb )
```

FIGURA 25. *Prueba de independencia o de asociación bootstrap. Método del desplazamiento: datos apareados (o medidas repetidas).*

d.2. Método de la aproximación normal bootstrap

La siguiente tabla presenta los nombres y sintaxis de los tres mandatos *MonteCarlo* que permiten aplicar el método de la aproximación normal *bootstrap*:

TABLA 22. *Mandatos para estimación del grado de significación por el método bootstrap de la aproximación normal.*

MANDATO	DESCRIPCIÓN
BNORMAL(X, B, 'Estadístico', μ [, α/2, h, nDib, nHist, nTpo])	Prueba de conformidad.
BNORMALG(X, Y, R, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de independencia entre dos variables: diseño de grupos independientes.
BNORMALP(X, Y, R, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de independencia o de asociación entre dos variables: diseño de datos apareados (o de medidas repetidas)

El algoritmo de la *prueba de conformidad* por el método de la aproximación normal es el siguiente:

```
mo = mean( Xo );
for b = 1 : B,
      Xb = resample( Xo );
      mb( b ) = mean( Xb );
end
deb = std( mb );
sig = 1 - distcdf( 'Normal', 0, 1, abs( mo - mu ) / deb );
ic = prctile( mb, [2.5, 97.5] );
hist( mb )
```

FIGURA 26. *Prueba de conformidad bootstrap. Método de la aproximación normal.*

Las siguientes figuras muestran los algoritmos *bootstrap* para las pruebas de independencia y de asociación en los diseños de grupos independientes y de datos apareados (o medidas repetidas).

```
difo = mean( Xo ) - mean( Yo );
for b = 1 : B,
      Xb = resample( Xo );
      Yb = resample( Yo );
      difb( b ) = mean( Xb ) - mean( Yb );
end
deb = std( difb );
sig = 1 - distcdf( 'Normal', 0, 1, abs( difo ) / deb );
ic = prctile( difb, [2.5, 97.5] );
hist( difb )
```

FIGURA 27. *Prueba de independencia bootstrap. Método de la aproximación normal: grupos independientes.*

```
difo = mean( Xo - Yo );
for b = 1 : B,
      Cb = resample( [ Xo , Yo ], 1 );
      difb( b ) = mean( Cb( :, 1 ) - Cb( :, 2 ) );
end
deb = std( difb );
sig = 1 - distcdf( 'Normal', 0, 1, abs( difo ) / deb );
ic = prctile( difb, [2.5, 97.5] );
hist( difb )
```

FIGURA 28. *Prueba de independencia o de asociación bootstrap. Método de la aproximación normal: datos apareados (o medidas repetidas).*

d.3. Método de la aleatorización bootstrap

La *Tabla 23* presenta los nombres y sintaxis de los tres mandatos *MonteCarlo* que permiten aplicar el método de la aleatorización *bootstrap*.

A diferencia de los dos métodos *bootstrap* anteriores, en el método de la aleatorización *bootstrap* para diseños de datos apareados (o de medidas repetidas), las pruebas de independencia y las pruebas de asociación tienen mandatos separados, puesto que requieren distintos esquemas de remuestreo (al igual que sucede en las pruebas de aleatorización).

Es importante destacar que no se ha programado el mandato para la prueba de conformidad por este método, dado que su algoritmo coincide con el método del desplazamiento de la distribución *bootstrap*.

TABLA 23. *Mandatos para estimación del grado de significación por el método de la aleatorización bootstrap.*

MANDATO	DESCRIPCIÓN
RBTESTG(X, Y, B, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de independencia entre dos variables: diseño de grupos independientes.
RBTESTP(X, Y, B, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de independencia entre dos variables: diseño de datos apareados (o de medidas repetidas).
RBASOCIA(X, Y, B, 'Estadístico' [, α/2, h, nDib, nHist, nTpo])	Prueba de asociación entre dos variables: diseño de medidas repetidas.

En la *prueba de independencia para un diseño de grupos independientes* el modelo de independencia se genera mediante el remuestreo aleatorio del vector total de datos (creado por la concatenación de los vectores originales Xo e Yo):

```
nx = length( Xo );
ny = length( Yo );
difo = mean( Xo ) - mean( Yo );
for rb = 1 : RB,
      Crb = resample( [ Xo ; Yo ] );
      difrb( rb ) = mean( Crb( 1 : nx ) ) - mean( Crb( nx+1 : ny+nx ) );
end
sig = sigres( difrb, difo );
hist( difrb )
```

FIGURA 29. *Prueba de independencia por aleatorización bootstrap: diseño de grupos independientes.*

En la *prueba de independencia para un diseño de datos apareados (o de medidas repetidas)*, el modelo de independencia se crea por el remuestreo aleatorio de los datos de cada caso, puesto que la unidad de asignación aleatoria es el momento de aplicación o registro de las variables apareadas.

El mandato RESAMPLE actúa de este modo cuando recibe como primer argumento una matriz.

Como se puede comprobar, el esquema de remuestreo coincide con el esquema de aplicación de las permutaciones en las pruebas de aleatorización.

```
n = length( Xo );
difo = mean( Xo - Yo );
for rb = 1 : RB,
      Crb = resample( [ Xo , Yo ] );
      difrb( rb ) = mean( Crb( :, 1 ) - Crb( :, 2 ) );
end
sig = sigres( difrb, difo );
hist( difrb )
```

FIGURA 30. *Prueba de independencia por aleatorización bootstrap: diseño de datos apareados (o de medidas repetidas).*

Finalmente, el algoritmo de la *prueba de asociación bootstrap entre dos variables para un diseño de medidas repetidas* es el siguiente:

```
n = length( Xo );
asociao = corr( Xo, Yo );
for rb = 1 : BR,
      Xrb = resample( Xo );
      Yrb = resample( Yo );
      asociarb( rb ) = corr( Xrb, Yrb );
end
sig = sigres( asociarb, asociao );
hist( asociarb )
```

FIGURA 31. *Prueba de relación entre dos variables por aleatorización bootstrap: diseño de medidas repetidas.*

3.3.6. Aplicaciones didácticas del *toolbox MonteCarlo*: estudio de la distribución muestral de un estadístico

En los cursos de introducción a los métodos estadísticos, muchos estudiantes tienen problemas para comprender conceptos teóricos como probabilidad, población, muestra, variable aleatoria, independencia de las observaciones, distribución muestral, error estándar, etc., que requieren un elevado grado de abstracción. Las dificultades también aparecen en el momento de seleccionar y aplicar la técnica de análisis más adecuada, debido, por un lado, a la gran diversidad de éstas y, por el otro, a la complejidad de su formulación matemática.

Ante esta situación, desgraciadamente, el alumno a menudo se conforma con intentar aprender la forma de solucionar correctamente cada uno de los problemas particulares, sin llegar realmente a entender los principios generales que los rigen.

En los últimos años, la incorporación de los ordenadores en la docencia de la estadística ha eliminado progresivamente los tediosos cálculos matemáticos, mejorando en parte esta situación, aunque paradójicamente, en ocasiones, el uso de los programas informáticos de análisis de datos ha añadido nuevas dificultades por la precipitación con la que se introducen, cayendo incluso en la tentación de

realizar análisis complejos sin un correcto conocimiento previo de la metodología estadística (Searle, 1989; Dallal, 1990; Paton, 1990). Pero a nuestro juicio, éste no es el papel más importante que pueden jugar los ordenadores en la didáctica de la estadística.

La estadística se fundamenta en el *modelo matemático* de la teoría de la Probabilidad, es decir, «*el modelo de las regularidades que se observan en las series de frecuencias correspondientes a los fenómenos aleatorios.*» (Ríos, 1967, p. 20).

Se trata del modelo matemático de un *proceso dinámico*, pero, como en todo modelo matemático, las predicciones se obtienen a través de la deducción lógico-matemática (Jañez, 1981), expresada generalmente con ***fórmulas totalmente alejadas del proceso subyacente*** que exigen al alumno un alto grado de abstracción. La rama izquierda de la *Figura 32* ilustra este camino.

Hoy, los ordenadores permiten utilizar el muestreo Monte Carlo para simular el proceso de muestreo (y de remuestreo) del cual se derivan conceptos como el Teorema Central del Límite, los intervalos de confianza y las pruebas de hipótesis, sustituyendo las fórmulas matemáticas por algoritmos que reflejan los procesos estadísticos y probabilísticos subyacentes.

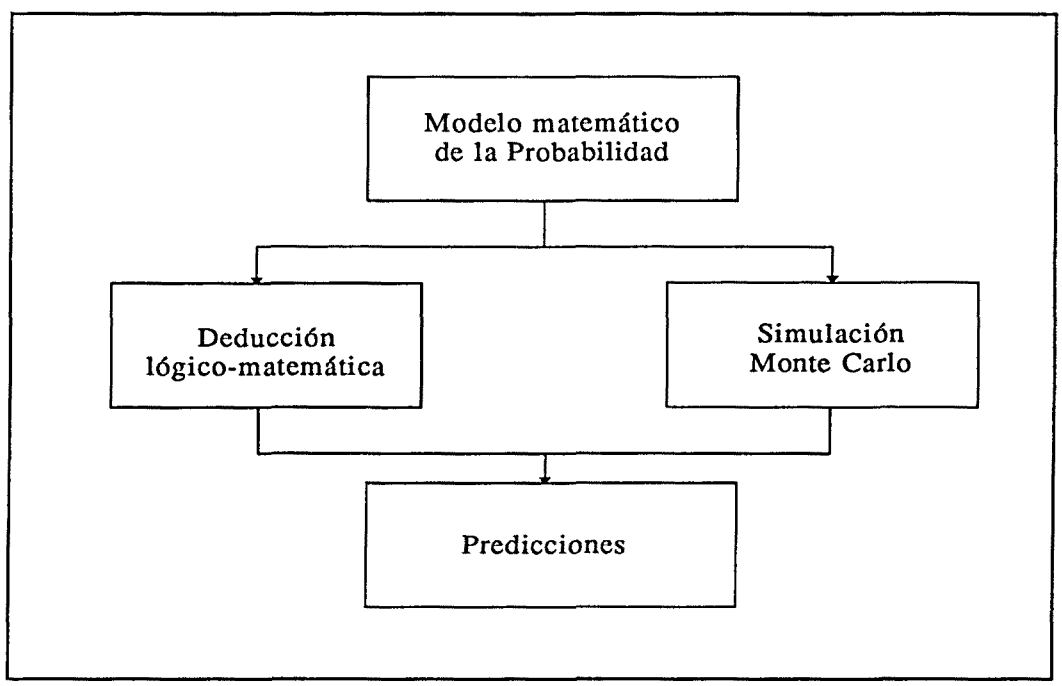


FIGURA 32. *Procedimientos para derivar predicciones a partir del modelo matemático de la Probabilidad.*

De este modo, *«el alumno alcanza la comprensión mediante la interacción y la experimentación. El hecho de que muchos estudiantes sean incapaces de conseguir estos objetivos mediante métodos tradicionales sugiere que el ordenador, con su habilidad interactiva natural, puede ser muy útil en estos casos.»* (Gordon y Hunt, 1986). De lo que se trata es de sustituir el principio de "debes calcularlo para comprenderlo" por el de ***"debes experimentarlo para comprenderlo"***.

Por otra parte, Gordon y Hunt señalan que es necesario involucrar a los alumnos en investigaciones que requieran de ellos lo siguiente:

- Aplicar técnicas conocidas a problemas o situaciones nuevas.
- Realizar preguntas pertinentes acera de problemas con los que no están familiarizados.
- Relacionar los modelos estadísticos con las situaciones reales, y tener la intuición suficiente para apreciar cuándo un modelo es más apropiado.
- Extraer conclusiones correctas a partir de los resultados observados.

Para Walton (1990), estas investigaciones deben motivar a los estudiantes a explorar diferentes métodos de solución de un problema, combinando datos empíricos (mediante ensayos reales y registro de resultados), simulaciones (mediante la ejecución reiterada y sistemática de los procesos), y soluciones analítico-teóricas (mediante cálculos matemáticos).

Los experimentos de simulación con monedas, cartas, dados, bolas, etc., que se realizaban en los inicios de la teoría de la probabilidad y la estadística eran excesivamente tediosos y requerían demasiado tiempo. El ordenador es la herramienta que puede substituye estos objetos ofreciendo una solución rápida y sencilla. A nivel docente, la aplicación más simple del muestreo Monte Carlo sólo requiere conocer la interpretación de un histograma y ser capaz de generar muestras de datos mediante un proceso de muestreo aleatorio.

Según Gordon y Hunt (1986), hay dos tipos de procesos que se deben implementar en un ordenador personal:

- Experimentos *gráficos* que permitan investigar los aspectos más teóricos. El profesor debe ofrecer las herramientas (y algunas ideas en una Guía de Usuario) al alumno para que éste pueda llevar a cabo las investigaciones.
- Experimentos de *simulación* que estimulen la participación del alumno. Esto significa, en un primer nivel, que exista la posibilidad de alterar parámetros de la simulación para observar los efectos que estos cambios producen en los resultados. En un segundo nivel, el alumno podría alterar el

procedimiento de simulación en sí mismo. Estos experimentos deben ser rápidos y deben incorporar animación, efectos sonoros, etc. que permitan mantener la atención del alumno, y aumenten la impronta visual y mnemotécnica después de la simulación.

Siguiendo esta línea, hemos incorporado en el *toolbox MonteCarlo* el mandato **DMUESTRAL**, que encierra en un cuadro de diálogo una familia de experimentos cuyo objetivo es el estudio gráfico e interactivo de las distribuciones muestrales de los estadísticos. Este cuadro de diálogo permite seleccionar o manipular los siguientes aspectos:

- El estadístico a investigar.
- La función de distribución de probabilidad de la población.
- El número de muestras aleatorias que se generan.
- El tamaño de las muestras generadas.

Los cambios que se realizan en estos parámetros se ven reflejados al instante en el histograma que muestra la forma de la distribución muestral.

Las *Figuras 33 a 35* muestran los distintos controles del cuadro de diálogo y el contenido del menú 'Distribución' del mandato DMUESTRAL.

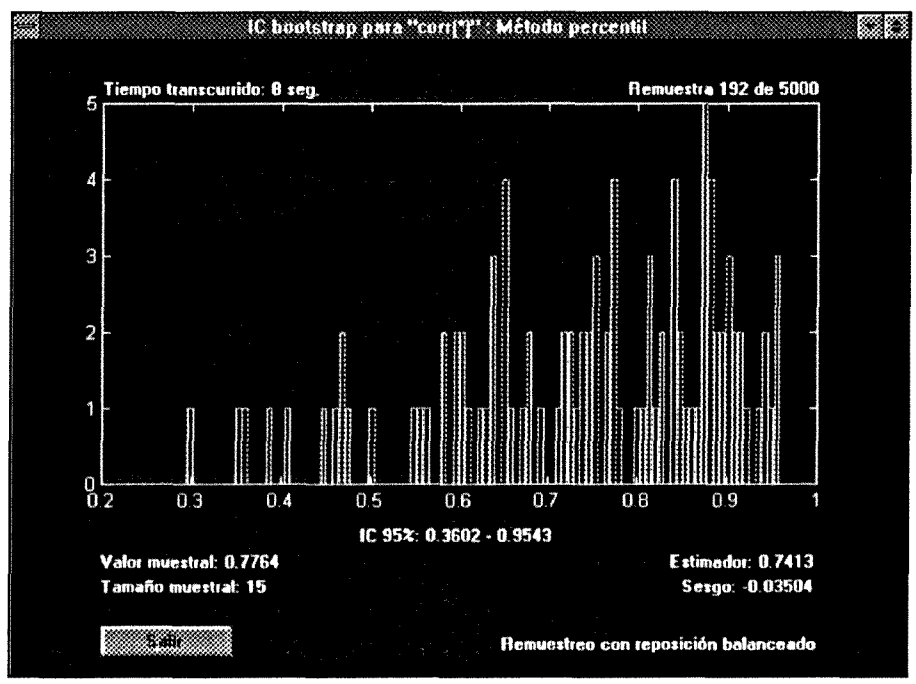


FIGURA 33. *Mandato DMUESTRAL: controles del cuadro de diálogo.*

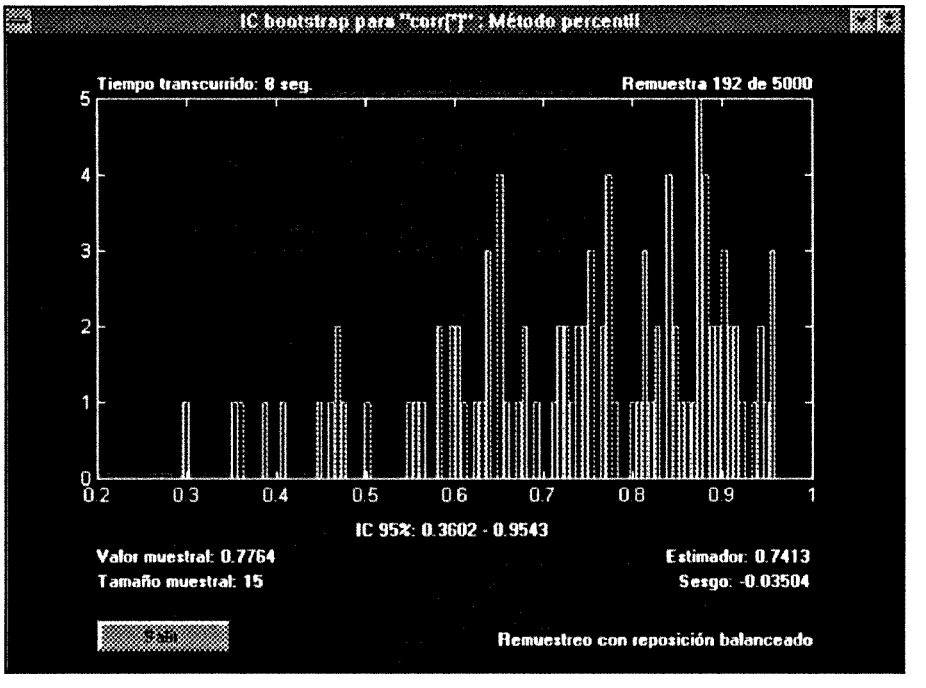


FIGURA 34. *Mandato DMUESTRAL: menú 'Distribución'.*

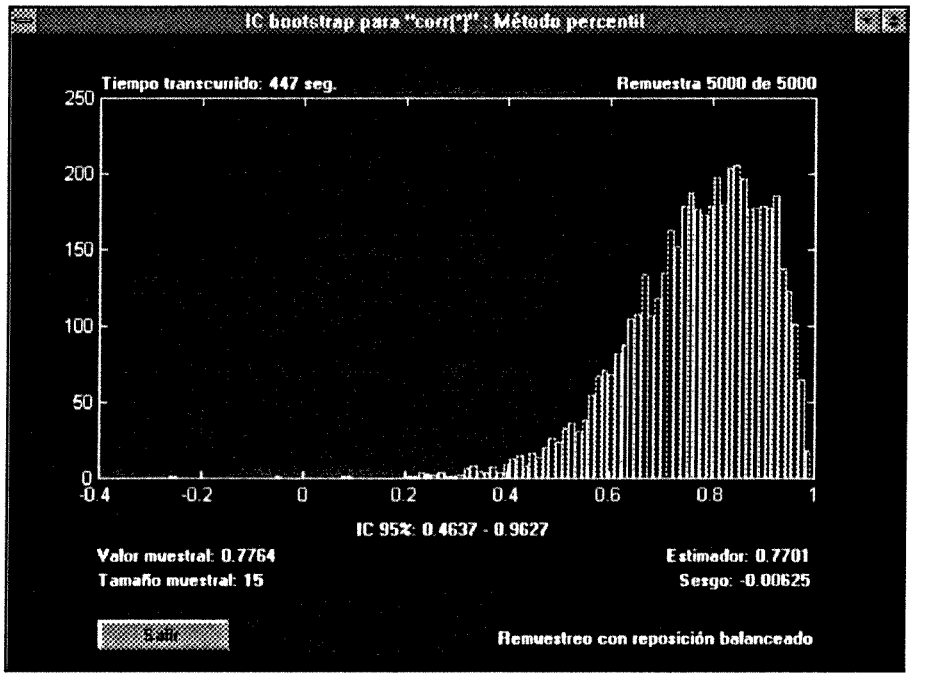


FIGURA 35. *Mandato DMUESTRAL: histograma de la distribución muestral.*

3.4. SÍNTESIS DE LOS MANDATOS DEL *TOOLBOX MONTECARLO*

La figura siguiente muestra la salida del mandato:

HELP *MonteCarlo*

que visualiza la lista de mandatos que conforman el *toolbox MonteCarlo*.

```
» help MonteCarlo

Toolbox para experimentos de simulación Monte Carlo
Versión 1.0, 20-Septiembre-1994

MUESTREO ALEATORIO:

 DISTRND     - Muestreo aleatorio a partir de distribución de probabilidad
 RANDOMIZE   - Remuestreo aleatorio sin reposición (por permutación)
 RESAMPLE    - Remuestreo aleatorio con reposición
 SEED        - Inicializar semilla para generadores de números aleatorios

MANDATOS PARA CONSTRUIR ALGORITMOS MONTE CARLO

 Distribuciones de probabilidad:
  DISTCDF    - Función de distribución acumulada
  DISTPDF    - Función de densidad de probabilidad
  DISTINV    - Inversa de la función de distribución acumulada

 Recuentos:
  COUNT      - Número de filas con valores entre [mín,máx]
  SIGRES     - Grado de significación en técnicas de muestreo Monte Carlo

 Recodificación de valores:
  RECODE     - Cambia valores entre [mín,máx] por valor especificado

 Obtención de rangos de ordenación:
  RANKS      - Devuelve las posiciones que ocuparán los valores ordenados
  TAGSORT    - Devuelve un vector de posiciones de los valores ordenados
```

Continúa ...

```
 Evaluación de series y repeticiones:
  DUPLICATES- Número de repeticiones de longitud [mín,máx]
  RUNS      - Número de repeticiones consecutivas de longitud [mín,máx]

 Eliminación de valores:
  DEDUP     - Elimina las filas con valores duplicados
  WEED      - Elimina las filas que contengan un valor entre [mín,máx]

INTERVALOS DE CONFIANZA:

 Intervalos clásicos:
  ICMEAN    - IC de una media (con base en la distribución t)
  ICMEDIAN  - IC de una mediana (con base en la distribución t)
  ICPROP    - IC de una proporción (con base en la distribución F)
  ICVAR     - IC de una variancia (con base en la distribución χ²)

 Intervalos de confianza bootstrap:
  ICBT      - IC bootstrap: método percentil-t
  ICBP      - IC bootstrap: método percentil
  ICBBC     - IC bootstrap: método BC
  ICBBCa    - IC bootstrap: método BCa

CONTRASTE DE HIPÓTESIS:

 Pruebas clásicas:
  CHI2TEST  - Prueba ji cuadrado de conformidad
  CHI2TEST2 - Prueba ji cuadrado de independencia (PD, PR y OR si procede)
  CHI2MN    - Prueba ji cuadrado de McNemar
  TTEST     - Prueba t de conformidad de una media observada a una teórica
  TTESTG    - Prueba t de comparación de 2 medias: grupos independientes
  TTESTP    - Prueba t de comparación de 2 medias: datos apareados
  ZCORR     - Prueba de significación del coef. de correlación de Pearson
  ZTEST     - Prueba z de conformidad de una media observada a una teórica

 Muestreo Monte Carlo:
  MTEST     - Prueba de conformidad por muestreo Monte Carlo

 Pruebas de aleatorización:
  RFISHER   - Prueba de aleatorización de Fisher
  RTESTG    - Prueba de independencia: grupos independientes
  RTESTP    - Prueba de independencia: datos apareados (o medias repetidas)
  RCORR     - Asociación entre dos variables: medidas repetidas
```

Continúa ...

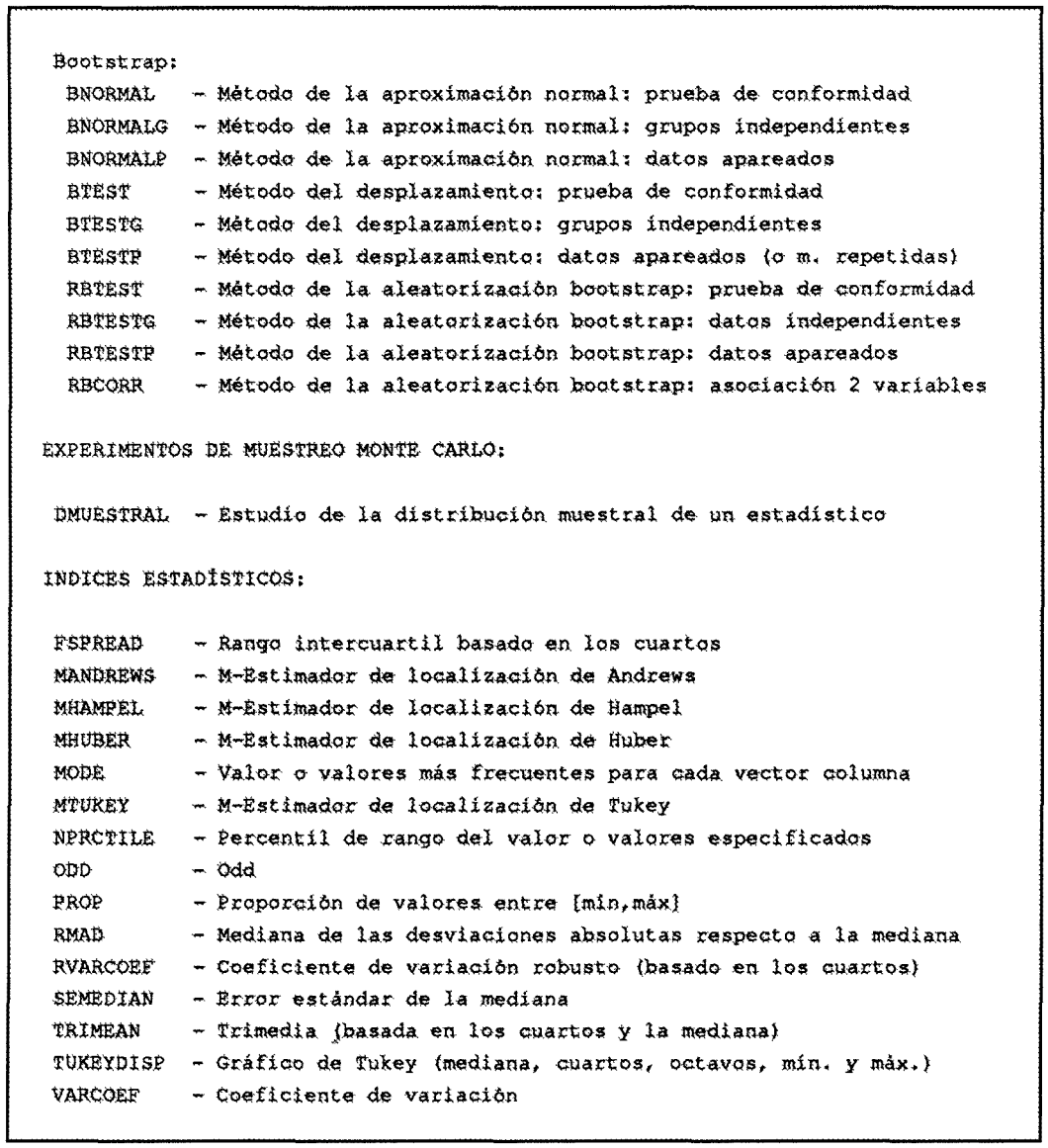


FIGURA 36. Resumen de los mandatos del toolbox MonteCarlo.

Puede obtenerse la descripción del propósito y sintáxis de cualquiera de los mandatos mediante la instrucción:

HELP *nombre_mandato*

La siguiente figura muestra un ejemplo de este tipo de ayuda:

```
» help ICBP

ICBP  - Intervalo de confianza bootstrap. Método percentil.

      Sintaxis: icbp(X, B, cEstad, alpha/2, xEE, nBalance, nDib, nHist, nTpo)

                    X: nombre del vector columna que contiene los datos
                    B: número de remuestreos a efectuar
               cEstad: debe escribir la llamada a la función que calcula el
                       estadístico entre comillas simples, utilizando el
                       carácter # para representar la matriz de datos sobre
                       la que desea calcular el estadístico.
                       Ejemplos: 'mean(#)', 'mhampel(#, 1.7, 3.4, 8.5)', etc.
            alpha/2: precisión unilateral del intervalo
                xEE: tipo de estimación del error estándar, pudiendo indicar:
                       * 'cálculo': fórmula de cálculo
                       *  0: sin cálculo del error estándar
                       * >0: número de remuestreos para estimación bootstrap
                       * -1: estimación jackknife
           nBalance: *  0: se realizará un remuestreo simple
                       *  1: se aplicará el algoritmo de remuestreo balanceado
               nDib: *  0: se visualizará el histograma al final
                       * >0: se visualizará el histograma cada b remuestras
                       * -1: no se visualizará el histograma
              nHist: número de intervalos de clase para el eje de abcisas
                       del histograma
               nTpo: *  0: no se mostrará el tiempo empleado en la estimación
                       *  1: se mostrará el tiempo empleado en la estimación
```

FIGURA 37. Ejemplo de ayuda obtenida mediante el mandato HELP ICBP.

4

EXPERIMENTOS DE SIMULACIÓN MONTE CARLO PARA VALIDAR EL FUNCIONAMIENTO DEL *TOOLBOX MONTECARLO*

«For Neyman, the idea of probability is fairly straightforward: It represents an idealization of long-run frequency in a long sequence of repetitions under constant conditions.»

E.L. Lehmann, 1993

En este capítulo se presentan diversos experimentos de simulación Monte Carlo diseñados para comprobar el correcto funcionamiento y análisis de los mandatos que conforman el *toolbox MonteCarlo*, así como la adecuación de MATLAB como plataforma de simulación estadística.

Para ello se han seleccionado experimentos llevados a cabo por investigadores relevantes en el campo de las técnicas de remuestreo que incorporan el mayor número posible de estas técnicas.

4.1. PRECISIÓN DE LOS INTERVALOS DE CONFIANZA *BOOTSTRAP*

Para comprobar el correcto funcionamiento de los mandatos ICBT, ICBP, ICBBC e ICBBCA, se realizan dos tipos de pruebas.

En primer lugar, y a partir de los famosos datos de Diaconis y Efron (1983) y Efron (1987), se compararan los resultados obtenidos por estos autores y los obtenidos con el *toolbox MonteCarlo* respecto a la estimación por intervalo del coeficiente de correlación de Pearson.

Diaconis y Efron (1983), relacionan los datos de las variables PAU (prueba de acceso a la universidad) y TFD (test de admisión a facultades de derecho), correspondientes a una muestra de 15 facultades de derecho, y obtienen un coeficiente de correlación de Pearson de .7764. La estimación por intervalo *bootstrap* del 68% por el método percentil, con 1000 remuestreos, da como resultado [.654, .909]. Por su parte, Efron (1987) presenta los resultados para la misma muestra de datos, aplicando esta vez el método percentil con corrección acelerada del sesgo, con una cobertura del 90%, y utilizando el valor –.0817 para la constante de aceleración *a*; los límites del intervalo de confianza obtenido, efectuando 100,000 remuestreos, es [.43, .92].

La *Tabla 24* contiene los resultados de aplicar los mandatos ICBT (con 50 remuestreos para la estimación *bootstrap* del error estándar), ICBP, ICBBC e ICBBCA de *MonteCarlo*. En todos los casos se han realizado 1000 remuestreos (la semilla utilizada ha sido 65536).

Los resultados obtenidos concuerdan con los presentados por Diaconis y Efron. Asimismo puede constatarse que la estimación por el método percentil-t es la menos exacta, debido al fuerte supuesto de normalidad de la distribución que se impone en este procedimiento.

Por otra parte, el segunto tipo de pruebas consiste en la realización de un experimento de muestreo Monte Carlo para estimar la distribución muestral de un estadístico y, una vez estimada, obtener los límites del intervalo de confianza. Seguidamente, los límites del intervalo construido se comparan con los que se obtienen al aplicar los diferentes métodos *bootstrap*. Por tanto, se trata de un estudio dirigido a comprobar la precisión de estas técnicas de estimación por intervalo.

En este experimento se generan 1000 muestras aleatorias según una distribución normal con parámetros $\mu = 100$ y $\sigma^2 = 225$. Para cada una de las muestras se realizan las estimaciones por intervalo *bootstrap* (con extracción de 1000 remuestras) de la media aritmética y de la variancia. El experimento se realiza para dos tamaños muestrales distintos: 16 y 32.

TABLA 24. *Resultados de la aplicación de los mandatos IC bootstrap con 1000 remuestreos (50 remuestreos para estimar error estándar en ICBT).*

Mandato	I.C. 68%	I.C. 90%
ICBT (método percentil-t):		
estim. bootstrap del EE	[.5114 , .8729]	[.1270 , .9280]
estim. jackknife del EE	[.4634 , .8586]	[.0569 , .9247]
ICBP (método percentil):		
Toolbox MonteCarlo	[.6557 , .9078]	[.5456 , .9500]
Diaconis y Efron (1983)	**[.6540 , .9090]**	----------
ICBBC (método BC)	[.6006 , .8807]	[.4850 , .9374]
ICBBCA (método BCa):		
Toolbox MonteCarlo	[.5743 , .8725]	[.4255 , .9273]
Efron (1987). a = -.0817	----------	**[.4300 , .9200]**

La *Tabla 25* resume, para cada método *bootstrap* aplicado y para los intervalos *t* y χ^2 clásicos, el porcentaje de intervalos estimados que contienen los verdaderos valores de los parámetros μ y σ^2 poblacionales.

Puede comprobarse que los intervalos de confianza *bootstrap* para la variancia contienen el verdadero valor poblacional en un porcentaje inferior al nivel $1-\alpha$ fijado "a priori". Este hecho puede ser debido a que, al tratarse de muestras pequeñas (especialmente para n = 16), no se produce la normalización de la distribución muestral *bootstrap* que predice el teorema central del límite para muestras grandes, resultando una distribución demasiado asimétrica.

Las *Figuras 38 a 41* muestran las distribuciones muestrales obtenidas para los estadísticos media y variancia y para los tamaños muestrales 16 y 32.

TABLA 25. *Porcentajes de inclusión del parámetro en 1000 IC 95% (B = 1000)*

MÉTODO	N = 16		N = 32	
	MEDIA	VARIANCIA	MEDIA	VARIANCIA
Clásico	94.9%	94.8%	95.0%	94.9%
Percentil-t	94.8%	81.5%	94.7%	88.4%
Percentil	93.9%	81.5%	94.5%	88.4%
BC	93.2%	85.1%	94.1%	90.3%

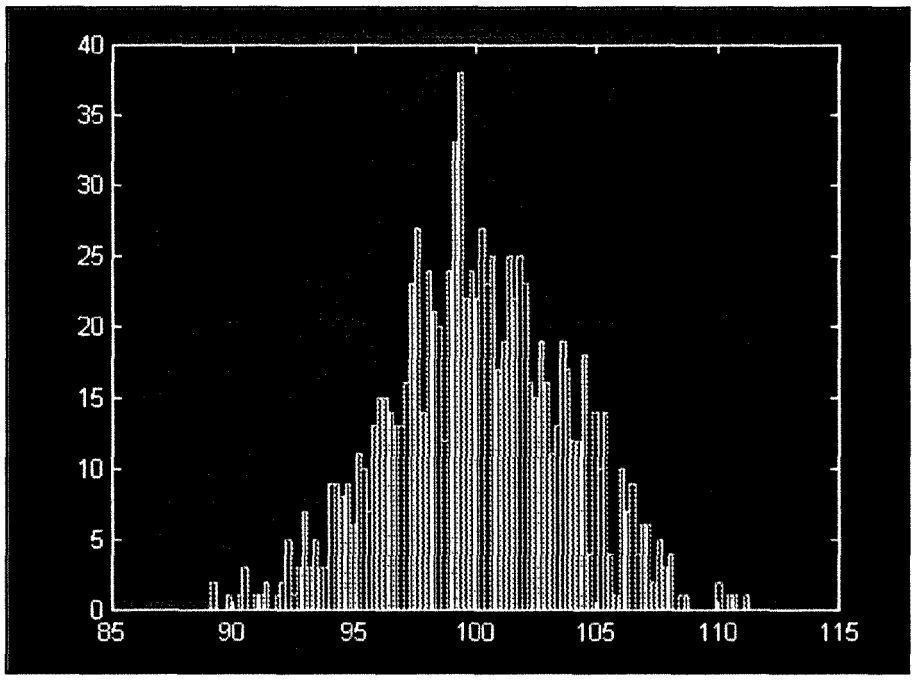


FIGURA 38. *Distribución muestral de 1000 medias (n = 16).*

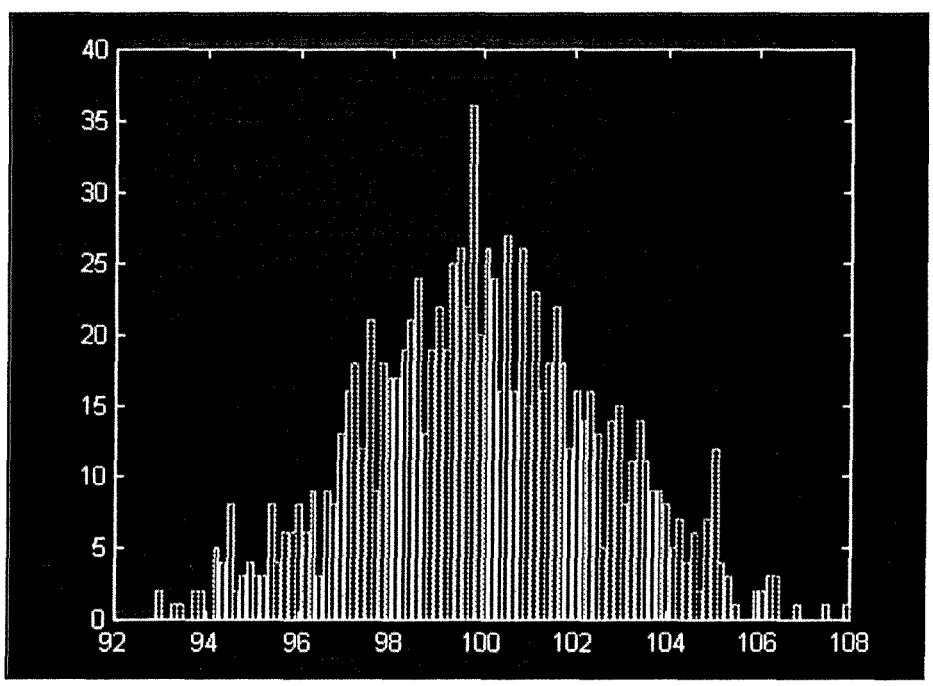


FIGURA 39. *Distribución muestral de 1000 medias (n = 32).*

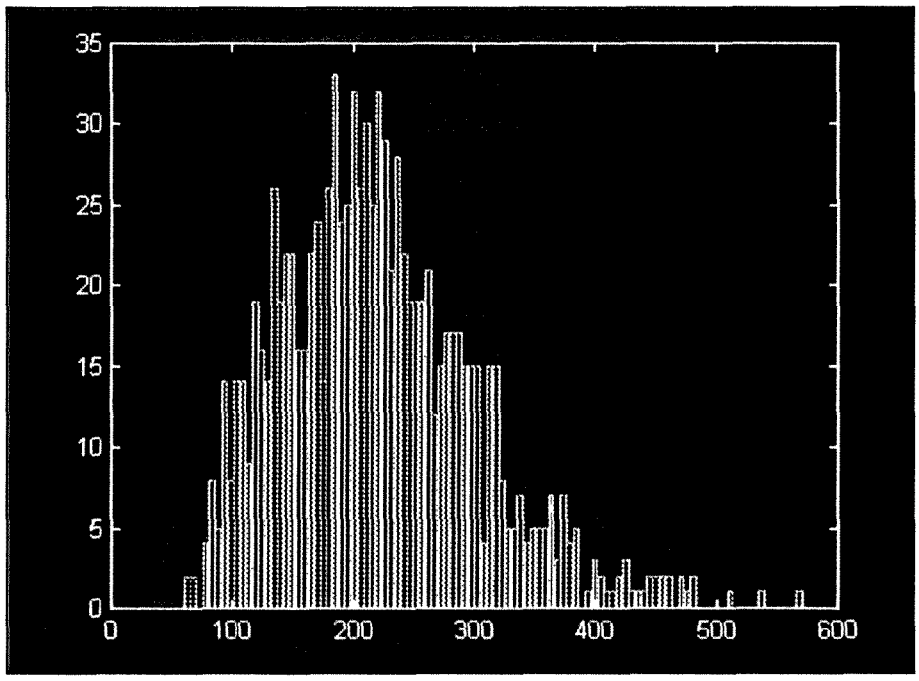


FIGURA 40. *Distribución muestral de 1000 variancias (n = 16).*

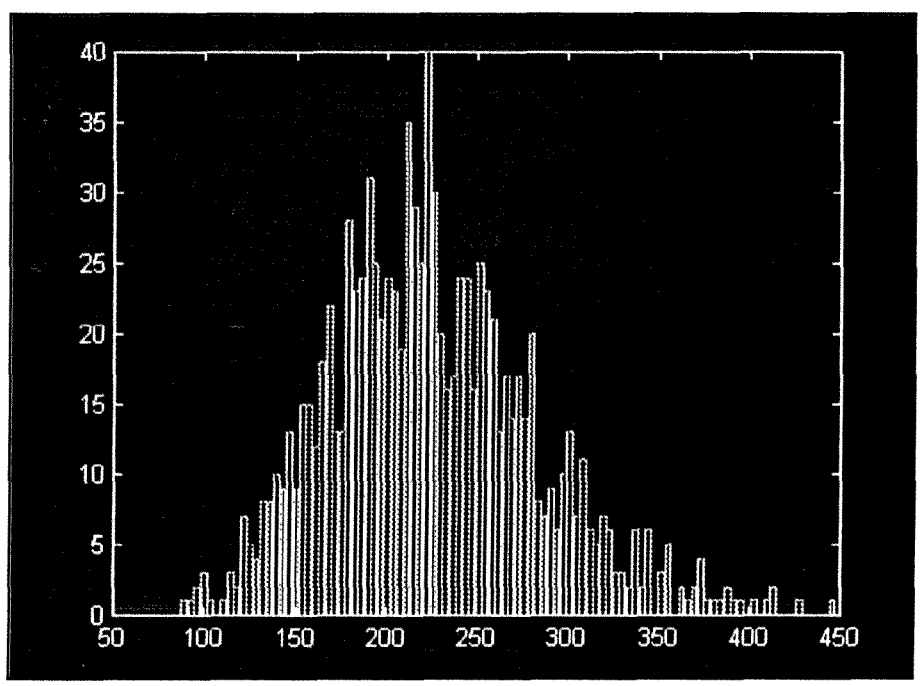


FIGURA 41. *Distribución muestral de 1000 variancias (n = 32).*

La siguiente figura muestra el algoritmo utilizado para este experimento:

```
function sim = simic( SIM, B, n, alpha );
%SIMIC - Simulación Monte Carlo para comprobar la precisión de
%        los intervalos de confianza bootstrap.
%

% Inicialización de variable para registro del tiempo de ejecución
tpo = clock;

% Pre-dimensionamiento de los vectores
sim   = zeros(SIM,2);
res   = zeros(8,1);
boot  = zeros(B,2);
kb    = zeros(n,1);
t     = distinv('T',1-alpha,n-1);
ji1   = distinv('Chi2',1-alpha,n-1);
ji2   = distinv('Chi2',alpha,n-1);

% Muestreo Monte Carlo
for m = 1:SIM,

   x = distrnd('Normal',0,1,n) .* 15 + 100;
   sim(m,:) = [mean(x) ; var(x)];
   zo = distinv('Normal',(1-alpha),0,1);

   % Remuestreo bootstrap
   for b = 1:B,
      kb = resample(x,1);
      boot(b,1) = mean(kb);
      boot(b,2) = var(kb);
      boot(b,3) = (boot(b,1) - sim(m,1))/sqrt(boot(b,2));
      boot(b,4) = ((n-1)*boot(b,2))/sim(m,2);
   end
```

Continúa ...

```
% Método clásico
% Media
ic = [sim(m,1)-(t*sqrt(sim(m,2)/n)) ;  sim(m,1)+(t*sqrt(sim(m,2)/n))];
if (ic(1) <= 100) & (ic(2) >= 100)
   res(1) = res(1) + 1;
end
% Variancia
ic = [(n-1)*sim(m,2)/ji1 ;  (n-1)*sim(m,2)/ji2];
if (ic(1) <= 225) & (ic(2) >= 225)
   res(2) = res(2) + 1;
end

% Método percentil-t
% Media
prct = prctile(boot(:,3), [alpha*100 (1-alpha)*100]);
ic = [sim(m,1) - (prct(2)*sqrt(sim(m,2))) ;...
      sim(m,1) - (prct(1)*sqrt(sim(m,2)))];
if (ic(1) <= 100) & (ic(2) >= 100)
   res(3) = res(3) + 1;
end
% Variancia
prct = prctile(boot(:,4), [alpha*100 (1-alpha)*100]);
ic = [prct(1)*sim(m,2)/(n-1) ;  prct(2)*sim(m,2)/(n-1)];
if (ic(1) <= 225) & (ic(2) >= 225)
   res(4) = res(4) + 1;
end

% Método percentil
% Media
ic  = prctile(boot(:,1), [alpha*100 (100-(alpha*100))]);
if (ic(1) <= 100) & (ic(2) >= 100)
   res(5) = res(5) + 1;
end
```

Continúa ...

```
    % Variancia
    ic  = prctile(boot(:,2), [alpha*100 (1-alpha)*100]);
    if (ic(1) <= 225) & (ic(2) >= 225)
       res(6) = res(6) + 1;
    end

    % Método BC
    % Media
    zb = distinv('Normal',length(find(boot(:,1)<=sim(m,1)))/B,0,1);
    bcinf = distcdf('Normal',(2*zb)-zo,0,1);
    bcsup = distcdf('Normal',(2*zb)+zo,0,1);
    ic = prctile(boot(:,1), [bcinf*100, bcsup*100]);
    if (ic(1) <= 100) & (ic(2) >= 100)
       res(7) = res(7) + 1;
    end
    % Variancia
    zb = distinv('Normal',length(find(boot(:,2)<=sim(m,2)))/B,0,1);
    bcinf = distcdf('Normal',(2*zb)-zo,0,1);
    bcsup = distcdf('Normal',(2*zb)+zo,0,1);
    ic = prctile(boot(:,2), [bcinf*100, bcsup*100]);
    if (ic(1) <= 225) & (ic(2) >= 225)
       res(8) = res(8) + 1;
    end

end

fprintf(['Tiempo transcurrido: ' int2str(etime(clock,tpo)) ' seg.']);
fprintf(['    Método clásico.  Media: ' num2str(res(1)*100/SIM) '%'...
                        '  Var: ' num2str(res(2)*100/SIM) '%']);
fprintf(['Método percentil-t.  Media: ' num2str(res(3)*100/SIM) '%'...
                        '  Var: ' num2str(res(4)*100/SIM) '%']);
fprintf(['  Método percentil.  Media: ' num2str(res(5)*100/SIM) '%'...
                        '  Var: ' num2str(res(6)*100/SIM) '%']);
fprintf(['         Método BC.  Media: ' num2str(res(7)*100/SIM) '%'...
                        '  Var: ' num2str(res(8)*100/SIM) '%']);
```

FIGURA 42. *Simulación Monte Carlo para evaluar la precisión de IC bootstrap.*

4.2. PROBABILIDAD DE ERROR TIPO I EN EL CONTRASTE DE HIPÓTESIS *BOOTSTRAP*

Los experimentos realizados para comprobar el correcto funcionamiento de los mandatos *bootstrap* para el contraste de hipótesis son una réplica de los realizados por Noreen (1989) y Westfall y Young (1993).

En el primer experimento, realizado por Noreen (1989), el procedimiento experimental consiste en extraer 1000 muestras aleatorias e independientes de datos, con tamaños muestrales 20, 40, 80 y 160, de una población con distribución Normal estandarizada. Para cada muestra, se aplican las pruebas de conformidad *bootstrap*, con 99 remuestreos, de la media observada respecto a la media teórica *0*, así como la prueba *t* clásica para comparar los resultados. Puesto que la verdadera media poblacional es *0*, si la prueba es válida la hipótesis nula debería de rechazarse aproximadamente un 5% de las veces cuando el nivel α fijado "a priori" es igual a 0.05, y un 10% de las veces cuando este nivel α es igual a 0.10.

Las pruebas de conformidad *bootstrap* aplicadas son las siguientes:

- Método del desplazamiento *bootstrap.*
- Método de la aproximación normal *bootstrap.*
- *Bootstrap* conjunto (resultado significativo al nivel α fijado "a priori" tanto en el método del desplazamiento como en el método de la aproximación normal).
- Método del desplazamiento *bootstrap* con transformación pivotal. Este método sólo se aplica en nuestro estudio.

La *Tabla 26* muestra los resultados de nuestro experimento (columna *MonteCarlo*) y del experimento de Noreen (1989). Se puede constatar que las estimaciones de la probabilida de error Tipo I son muy cercanas en ambos casos. Las pequeñas diferencias entre los resultados de nuestro experimento y los de Noreen pueden explicarse debido al relativamente pequeño número de muestras extraídas (1000) y de remuestreos realizados (99), así como al hecho de que este autor no indica ni los algoritmos concretos que utiliza para la generación de valores aleatorios uniformes y según ley Normal, ni la semilla utilizada para dichos generadores. En nuestro experimento, la semilla utilizada para los generadores de números aleatorios ha sido 65536.

TABLA 26. *Probabilidad de error Tipo I: 1000 muestras de tamaño N de una población Normal estandarizada (B = 99).*

MÉTODO	N = 20		N = 40		N = 80		N = 160	
	Noreen	*MonteCarlo*	Noreen	*MonteCarlo*	Noreen	*MonteCarlo*	Noreen	*MonteCarlo*
α = 0.05								
Unilateral superior Prueba *t*	0.059	0.050	0.047	0.054	0.053	0.047	0.057	0.053
Desplazamiento *bootstrap*	0.076	0.068	0.064	0.063	0.068	0.050	0.064	0.050
Aproximación normal *bootstrap*	0.072	0.068	0.056	0.059	0.059	0.048	0.057	0.050
Bootstrap conjunto	0.068	0.068	0.056	0.059	0.058	0.048	0.052	0.050
Transformación pivotal	-----	0.051	-----	0.052	-----	0.050	-----	0.051
Unilateral inferior Prueba *t*	0.050	0.056	0.037	0.038	0.050	0.039	0.048	0.049
Desplazamiento *bootstrap*	0.064	0.063	0.047	0.043	0.048	0.042	0.053	0.047
Aproximación normal *bootstrap*	0.063	0.072	0.049	0.044	0.057	0.044	0.050	0.050
Bootstrap conjunto	0.059	0.059	0.040	0.037	0.046	0.038	0.043	0.044
Transformación pivotal	-----	0.047	-----	0.048	-----	0.049	-----	0.049
α = 0.10								
Unilateral superior Prueba *t*	0.106	0.103	0.092	0.117	0.115	0.099	0.112	0.102
Desplazamiento *bootstrap*	0.117	0.113	0.095	0.114	0.112	0.109	0.114	0.098
Aproximación normal *bootstrap*	0.125	0.117	0.101	0.124	0.114	0.107	0.116	0.108
Bootstrap conjunto	0.111	0.104	0.092	0.110	0.107	0.094	0.106	0.092
Transformación pivotal	-----	0.102	-----	0.105	-----	0.103	-----	0.104
Unilateral inferior Prueba *t*	0.101	0.104	0.106	0.088	0.091	0.083	0.098	0.083
Desplazamiento *bootstrap*	0.113	0.115	0.109	0.105	0.093	0.09	0.095	0.085
Aproximación normal *bootstrap*	0.125	0.119	0.104	0.098	0.095	0.092	0.096	0.083
Bootstrap conjunto	0.107	0.107	0.102	0.094	0.086	0.077	0.087	0.076
Transformación pivotal	-----	0.093	-----	0.094	-----	0.096	-----	0.095

En cuanto a la validez de los métodos *bootstrap*, puede comprobarse que cuando la muestra es pequeña (N = 20) la hipótesis nula es rechazada un 5% (5.9% según el resultado de Noreen) de las veces al nivel 0.05 al aplicar la prueba *t* clásica y la hipótesis alternativa es que la media observada es superior a la media poblacional. La prueba *bootstrap*, según el método del desplazamiento de la distribución, ofrece un 6.8% de rechazos (7.6% en el experimento de Noreen), el mismo resultado que al aplicar el método de la aproximación Normal. Si se utilizan los dos métodos *bootstrap* conjuntamente, rechazando la hipótesis nula sólo si ambos métodos rechazan esta hipótesis, el resultado obtenido es también del 6.8%. Cuando la hipótesis alternativa es que la media observada es inferior a la poblacional, o cuando el nivel α es igual a 0.10, el resultado del *bootstrap* conjunto es más conservador que cuando se evalúan los dos métodos por separado.

Aplicando el método *bootstrap* con transformación pivotal, y variando el número de muestras extraídas (4000) y de remuestreos (999), la hipótesis nula es rechazada el 5.2% de las veces. En general, esta prueba ofrece los mismos resultados que la prueba *t* clásica tanto para los niveles α 0.05 y 0.10, como en los dos tipos de hipótesis alternativa unilateral, y para los diferentes tamaños muestrales utilizados. Los resultados que se obtienen aplicando los métodos *bootstrap* del desplazamiento de la distribución sin transformación pivotal y el método de la aproximación Normal se alejan 1 ó 2 puntos porcentuales cuando los tamaños muestrales son 20 ó 40.

El segundo experimento realizado es similar al primero, pero en este caso la población se distribuye según una ley Lognormal estandarizada, incumpliéndose el supuesto de normalidad de la prueba *t* clásica. Dado que los métodos *bootstrap* no asumen ningún tipo de distribución en particular, deberían de ofrecer en esta situación mejores resultados que la prueba *t* clásica. Los valores de las muestras extraídas de la población Lognormal se definen por la relación $x = \exp(z)$, donde z es una variable Normal estandarizada. La media de esta población es $\sqrt{e}$, valor utilizado para las pruebas de conformidad unilaterales aplicadas. El número de muestras que se extraen es de 1000, con 99 remuestreos, al igual que en el estudio de Noreen (1989). Westfall y Young (1993) también replican el estudio de Noreen, pero utilizando el método *bootstrap* del desplazamiento de la distribución con transformación pivotal, y extrayendo 4000 muestras y realizando 999 remuestreos para cada una de ellas (nuestro experimento también utiliza este número de iteraciones para contrastar los resultados de Westfall y Young).

La *Tabla 27* muestra los resultados de nuestro experimento y del experimento de Noreen (N en la tabla) y de Westfall y Young (WY en la tabla).

TABLA 27. *Probabilidad de error Tipo I: muestras de tamaño N de una población Lognormal estandarizada.*

MÉTODO	N = 20		N = 40		N = 80		N = 160	
	N / WY	*MonteCarlo*	N / WY	*MonteCarlo*	N / WY	*MonteCarlo*	N / WY	*MonteCarlo*
α = 0.05								
Unilateral superior Prueba *t* (N)	0.008	0.010	0.010	0.012	0.015	0.017	0.022	0.024
Desplazamiento *bootstrap* (N)	0.014	0.013	0.014	0.012	0.018	0.020	0.027	0.016
Aproximación normal *bootstrap* (N)	0.011	0.015	0.013	0.016	0.017	0.021	0.024	0.025
Bootstrap conjunto (N)	0.009	0.015	0.009	0.014	0.013	0.021	0.021	0.024
Transformación pivotal (WY)	0.030	0.033	0.038	0.044	0.044	0.044	0.043	0.044
Unilateral inferior Prueba *t* (N)	0.153	0.168	0.149	0.156	0.124	0.113	0.109	0.098
Desplazamiento *bootstrap* (N)	0.180	0.204	0.167	0.177	0.136	0.123	0.119	0.104
Aproximación normal *bootstrap* (N)	0.167	0.193	0.161	0.171	0.124	0.118	0.113	0.101
Bootstrap conjunto (N)	0.165	0.186	0.158	0.162	0.119	0.110	0.107	0.092
Transformación pivotal (WY)	0.106	0.106	0.087	0.080	0.076	0.071	0.066	0.062
α = 0.10								
Unilateral superior Prueba *t* (N)	0.032	0.034	0.038	0.042	0.062	0.059	0.067	0.053
Desplazamiento *bootstrap* (N)	0.041	0.040	0.040	0.047	0.053	0.064	0.060	0.058
Aproximación normal *bootstrap* (N)	0.049	0.052	0.051	0.052	0.063	0.065	0.068	0.054
Bootstrap conjunto (N)	0.037	0.034	0.037	0.038	0.049	0.057	0.055	0.047
Transformación pivotal (WY)	0.082	0.086	0.085	0.093	0.092	0.093	0.094	0.087
Unilateral inferior Prueba *t* (N)	0.210	0.229	0.211	0.220	0.178	0.171	0.152	0.150
Desplazamiento *bootstrap* (N)	0.229	0.251	0.216	0.238	0.191	0.180	0.162	0.158
Aproximación normal *bootstrap* (N)	0.223	0.242	0.217	0.235	0.183	0.173	0.159	0.152
Bootstrap conjunto (N)	0.219	0.238	0.211	0.228	0.180	0.167	0.154	0.144
Transformación pivotal (WY)	0.163	0.165	0.143	0.139	0.134	0.129	0.119	0.125

Los resultados obtenidos en nuestro experimento son semejantes a los que presentan estos autores. La semilla utilizada para los generadores de números aleatorios ha sido 65536 (Noreen y Westfall y Young no indican cuál es el valor de la semilla utilizada en sus experimentos).

A partir de los resultados obtenidos se puede constatar que el método del desplazamiento con transformación pivotal aproxima los niveles nominales fijados "a priori" mejor que la prueba *t* y que el método del desplazamiento sin utilización de un estadístico pivote.

Westfall y Young (1993) realizan unos comentarios generales sobre la elección del número de muestras Monte Carlo y el número de remuestreos *bootstrap* que deberían aplicarse en este tipo de experimentos, concluyendo que es más importante el número de iteraciones en el bucle externo (muestreo Monte Carlo) que en el interno (remuestreo *bootstrap*). En la mayor parte de situaciones, el número de iteraciones del bucle externo debería oscilar entre 5000 y 10000, cifra adecuada también para el número de remuestreos *bootstrap*. Hope (1968) y Oden (1991) presentan complejos análisis que permiten evaluar la magnitud de error de la simulación debida al número finito de iteraciones.

La siguiente figura muestra el algoritmo utilizado para este experimento:

```
function simtest( SIM, B, n, alpha )
%SIMTEST  Simulación Monte Carlo para comprobar la validez de los métodos
%        bootstrap de contraste de hipótesis. Población Lognormal.
%

% Media poblacional
mu = sqrt( exp(1) );

% Inicialización de valores criterio
tcritl  = distinv('T',alpha,n-1);
tcritu  = distinv('T',1-alpha,n-1);
zcritl  = distinv('Normal',alpha,0,1);
zcritu  = distinv('Normal',1-alpha,0,1);
```

Continúa ...

```
% Inicialización de contadores
tsigl  = 0;
tsigu  = 0;
bssigl = 0;
bssigu = 0;
bnsigl = 0;
bnsigu = 0;
bcsigl = 0;
bcsigu = 0;
bpsigl = 0;
bpsigu = 0;

% Muestreo Monte Carlo
for sim = 1:SIM

   x = exp( distrnd('Normal',0,1,n) );
   m = mean(x);
   s = std(x);
   difmu = (m - mu);

   % t-test
   t = difmu/(s/sqrt(n));
   if t <= tcritl
      tsigl = tsigl + 1;
   end
   if t >= tcritu
      tsigu = tsigu + 1;
   end

   nbssigl = 0;
   nbssigu = 0;
   nbpsigl = 0;
   nbpsigu = 0;
   lbssigl = 0;
   lbssigu = 0;
   lbnsigl = 0;
   lbnsigu = 0;
```

Continúa ...

```
% Remuestreo bootstrap
for boot = 1:B

    xb  = resample(x,1);
    mb(boot)  = mean(xb);

    dif = mb(boot) - m;
    if dif <= difmu,  nbssigl = nbssigl + 1,  end;
    if dif >= difmu,  nbssigu = nbssigu + 1,  end;

    tb = (mb(boot) - m)/(std(xb)/sqrt(n));
    if tb <= t,  nbpsigl = nbpsigl + 1,  end;
    if tb >= t,  nbpsigu = nbpsigu + 1,  end;

end

% Shift-method
nbssigl = (nbssigl+1)/(B+1);
nbssigu = (nbssigu+1)/(B+1);
if nbsigl <= alpha
    lbssigl = 1;
    bssigl = bssigl + 1;
end
if nbsigu >= (1-aplha)
    lbssigu = 1;
    bssigu = bssigu + 1;
end

% Normal-approximation
zb = difmu/std(mb);
if zb <= zcritl
    lbnsigl = 1;
    bnsigl = bnsigl + 1;
end
if zb >= zcritu
    lbnsigu = 1;
    bnsigu = bnsigu + 1;
end
```

Continúa ...

```
    % Bootstrap Conjunto (shift & normal-approximation)
    if lbssigl & lbnsigl
       bcsigl = bcsigl + 1;
    end
    if lbssigu & lbnsigu
       bcsigu = bcsigu + 1;
    end

    % Shift & pivot method
    nbpsigl = (nbpsigl+1)/(B+1);
    nbpsigu = (nbpsigu+1)/(B+1);
    if nbpsigl <= alpha
       bpsigl = bpsigl + 1;
    end
    if bssigu >= (1-alpha)
       bpsigu = bpsigu + 1;
    end

end

fprintf(['n = ' int2str(n) '\n']);
fprintf(['\nAlpha = ' int2str(alpha) '\n');
fprintf('Upper-Tail\n');
fprintf(['          t-test: ' num2str(tsigu/SIM)  '\n']);
fprintf(['    shift-method: ' num2str(bssigu/SIM) '\n']);
fprintf(['   normal-method: ' num2str(bnsigu/SIM) '\n']);
fprintf(['   boot-conjunto: ' num2str(bcsigu/SIM) '\n']);
fprintf(['    pivot-method: ' num2str(bpsigu/SIM) '\n']);
fprintf('Lower-Tail\n');
fprintf(['          t-test: ' num2str(tsigl/SIM)  '\n']);
fprintf(['    shift-method: ' num2str(bssigl/SIM) '\n']);
fprintf(['   normal-method: ' num2str(bnsigl/SIM) '\n']);
fprintf(['   boot-conjunto: ' num2str(bcsigl/SIM) '\n']);
fprintf(['    pivot-method: ' num2str(bpsigl/SIM) '\n']);
```

FIGURA 43. *Simulación Monte Carlo para evaluar la probabilidad de error Tipo I de las pruebas de contraste de hipótesis bootstrap y de la prueba t clásica.*

A simple vista se constata que los algoritmos presentados son más simples que los que deberían realizarse con otros lenguajes de programación para llevar a cabo estos mismos experimentos. Hay que señalar también, que se podrían simplificar todavía más si se incluyeran en ellos las llamadas a los mandatos de estimación por intervalo o de contraste de hipótesis del *toolbox MonteCarlo*; no se ha hecho así para que los algoritmos mostraran la totalidad del proceso, así como para aumentar la velocidad de ejecución ya que se incluye un único bucle de remuestreo común para todos los métodos *bootstrap* que se comparan en cada experimento.