

La generación aleatoria de números por ordenador

AVANCE

Vicente Manzano Arrondo

Índice

Un previo conceptual	2
El método Monte Carlo	5
Procedimiento general	7
Estrategias para la construcción de generadores básicos	11
La calidad de los generadores: estrategias para su estudio a posteriori	15
Referencias	18
Apéndice I	
Simulación de la tirada de un dado para la aproximación a la normal	21
Apéndice II:	
Programa para la generación de números aleatorios	22
Apéndice III	
Listado en código C, de los algoritmos de tres métodos de generación	24

Un previo conceptual

La generación de números pseudo-aleatorios por parte de un programa de ordenador, lleva consigo algunas aparentes incoherencias, paradojas o puntos de difícil comprensión.

Sobre la previsión. El problema principal reside en que el azar es, en términos elementales o individuales, imprevisible. Sólo se elaboran leyes que, de forma aproximada, permiten acercarse a los comportamientos aleatorios en términos generales o grupales. Así, utilizando el ejemplo más clásico, no sabemos si el lanzamiento de una moneda dará como resultado una u otra cara. Pero esperamos que, si la moneda tiene una masa uniformemente distribuida por todo el volumen y la forma es exactamente cilíndrica, el porcentaje de cualquiera de los dos resultados posibles se acercará al 50% conforme el número de lanzamientos se hace cada vez más grande.

Pero *¿Cómo reproducir un proceso imprevisible?*

Sobre el conocimiento y el control. Por otro lado, un experimento *realmente* aleatorio ¿Qué es?. Supongamos que alguien extrae bolas de un bombo de lotería y lee los números resultantes. Es un ejemplo también clásico de fenómeno aleatorio. Se dice que el azar es sinónimo de ignorancia. Si se pudieran conocer y controlar las fuerzas que actúan en el movimiento del conjunto de bolas dentro del bombo, se podrían predecir los resultados. Pero la red de relaciones es tan complicada que nadie, en su sano juicio, pretende abordar el problema con intenciones de éxito.

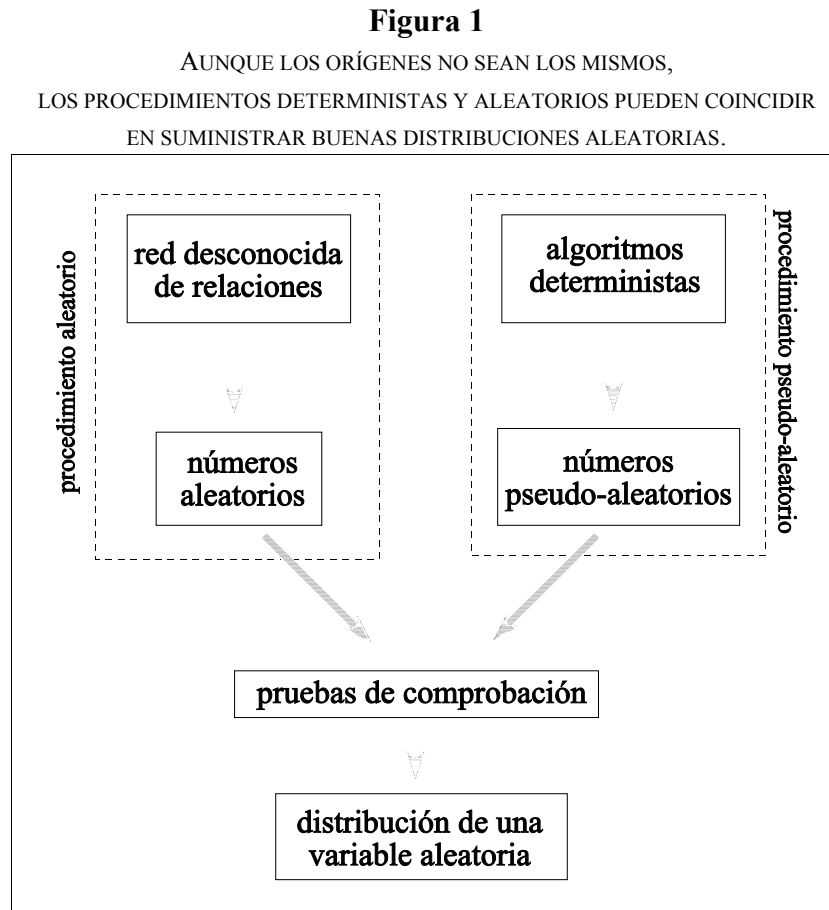
Pero *¿Cómo poner en marcha un procedimiento controlado por ordenador para producir un resultado incontrolado?*

Sobre los argumentos apriorísticos. Por último, los fenómenos se juzgan como aleatorios por argumentos establecidos “a priori”. Antes de que el dado pare, no sabemos cuál será la cara que muestre. Después, el resultado es evidente y no admite discusión ni reflexión.

Pero *¿Cómo creer en la aleatoriedad de un proceso que es determinístico “a priori” ?*

Es curioso, pero después de los problemas conceptuales apuntados, ocurre que la generación de números aleatorios por un ordenador es un hecho aceptado y útil, que permite poner en marcha técnicas y desarrollos imposibles desde otras perspectivas. La solución final que se adopta es: “un procedimiento generador de números aleatorios será

bueno si, analizado el conjunto de números resultante, no es posible indicar a posteriori, si tales cantidades fueron generadas por el ordenador o por un procedimiento genuinamente aleatorio”. La figura 1 muestra esta relación.



Esta equivalencia entre los resultados puede llevar a que el producto de un procedimiento determinista pueda llegar a ser juzgado como más satisfactoriamente aleatorio que el producto de un procedimiento aleatorio puro. La explicación es sencilla: los analizadores de la bondad de los procedimientos evalúan en qué medida los resultados cuentan con las características “ideales” de la aleatoriedad. No se intenta imitar el procedimiento real, aleatorio de referencia, sino simular un comportamiento aleatorio en el ordenador de tal forma que la generación de resultados respete las probabilidades que se establecen a priori. En un apartado posterior trataremos qué pruebas de comprobación son utilizadas para tal fin.

Hay una última cuestión para abordar antes de finalizar este apartado previo: ¿Son imprescindibles las «máquinas» para generar secuencias aleatorias de números? ¿No servimos las personas?

Ya Reichenbach (1949) señaló que las personas somos generadores aleatorios imperfectos. Otros estudios posteriores han recogido suficientes evidencias (por ejemplo, Wagenaar, 1972, Bar-Hille y Wagenaar, 1993 o Chapanis, 1995), puesto que existen tendencias claras en los humanos para la elaboración de secuencias sesgadas (Ginsburg y Wiegersma, 1991, Chapanis, op.cit.) y éstas suelen estar relacionadas con características personales como la salud mental (Rosenberg, Weber, Crocq y Duval, 1990; Brugger, Monsch, Salmon y Butters, 1996), los conocimientos o informaciones previas (Chapanis, op.cit.), como el concepto subjetivo de aleatoriedad (Spatt y Goldenberg, 1993; Brugger, Pietzsch, Weidmann, Biro y Alon, 1995; Ginsburg y Karpiuk, 1995) o la edad (Rabinowitz, Dunlap, Grant y Campione, 1989).

Así pues, es evidente que debe recurrirse a un procedimiento “objetivo” de generación de números aleatorios. De ello nos encargaremos más adelante. Antes, ¿Qué sentido tiene la utilización de secuencias de dígitos aleatorios?

El método Monte Carlo

El método de Monte Carlo se utiliza para generar variables aleatorias y, con éstas, realizar cálculos, simular comportamientos o resolver problemas (por ejemplo, Sobol, 1983).

Una variable aleatoria es aquella de la que se conoce la distribución de probabilidad (la probabilidad asociada a cada uno de los valores que puede adoptar la variable) pero no el resultado en cada proceso de medición de la variable. Un ejemplo clásico es el del lanzamiento de un dado. Se sabe que todas las caras son igualmente probables, pero ante un lanzamiento concreto, se desconoce el resultado a priori. La variable “resultado al tirar el dado” es aleatoria.

El método se basa en la utilización de un procedimiento determinista que permite la generación de una variable aleatoria. Su nombre, que se refiere a uno de los casinos (juegos de *azar*) más famosos del mundo, se hace así comprensible. No obstante, si bien era conocido desde hace tiempo, no comenzó su expansión hasta la difusión de los ordenadores.

Originalmente, el método de Monte Carlo se utilizó implementando en el ordenador tablas de números aleatorios generados mediante procedimientos aleatorios puros, como un bombo de lotería o una ruleta de casino (Sobol, op.cit.). Pero este procedimiento es lento, ocupa mucha memoria y arrastra posibles sesgos de forma indefinida.

El método que realmente se sigue en la actualidad, en la utilización de Monte Carlo, es la generación de números mediante algoritmos implementados en el ordenador. Ello permite ahorrar memoria, ganar en rapidez y reproducir las secuencias a voluntad, realizando todo tipo de pruebas de calidad con las secuencias generadas. Resulta imprescindible, pues, que los generadores de números sean de alta calidad, para que las simulaciones de Monte Carlo se realicen convenientemente (Barry, 1996).

Así pues, para hacer simulaciones aleatorias se requieren buenos generadores aleatorios.

Así mismo, las secuencias de cifras aleatorias son también utilizadas para otros objetivos que no encajan bien en el método Monte Carlo, como, por ejemplo, comparación de ejecuciones en personas normales y con deficiencias (Brugger, Monsch, Salmon y Butters, 1996), aprendizaje (Schlenker, 1996 o Speer, 1997) o someter a sujetos experimentales a tareas que permiten medir procesos básicos, diferencias individuales o procesos fisiológicos (Brugger, Milicevic, Regard y Cook, 1993; Vitulli, DePace y Holland, 1993;

Brugger, Pietzsch, Weidmann y Biro, 1995; Itagaki, Niwa, Itoh y Momose, 1995; Sakata, Shinohara, Hori y Sugimoto, 1995; Hori, Iwaki y Hayashi, 1996).

Procedimiento general

En el marco real, entendido éste como el conjunto de los acontecimientos espontáneos que se desencadenan a nuestro alrededor, existen las llamadas “distribuciones libres”. Son conjuntos de datos que no parecen seguir un patrón determinado, un modelo conocido. No obstante, cuando se generan números con el ordenador, se busca que éstos sigan un patrón concreto. Es más, si no lo siguen no es posible evaluar la calidad del generador utilizado, puesto que la única posibilidad es comparar el resultado obtenido (distribución generada) con el esperado (distribución teórica).

El procedimiento general que se sigue es:

1. Escoger un modelo de distribución (por ejemplo, la normal)
2. Escoger un generador de números aleatorios con distribución uniforme continua, acotada en el intervalo $(0,1)$ (abierto a la derecha.
3. Diseñar una estrategia que permita generar la distribución buscada a partir de la uniforme continua (en el ejemplo, de la uniforme a la normal).

El procedimiento parece algo enrevesado, pero tiene su sentido, como suele ocurrir. Existen, principalmente, dos justificaciones:

1. Tenemos así a investigadores que diseñan, analizan y comunican generadores para una distribución uniforme continua. Algunos procedimientos serán preferidos a otros y unos tendrán mayor difusión que otros. En definitiva, se contará con un conjunto más o menos limitado de *generadores aleatorios básicos*. Gracias al conocimiento en el comportamiento de éstos, otro grupo de investigadores podrá elaborar estrategias para traducir las distribuciones uniformes continuas a las que interese finalmente. Con ello, el trabajo global es mucho más eficiente que si se abordara directamente una distribución diferente cada vez.
2. Pensar en términos de distribución uniforme continua, permite utilizar a los decimales como suministradores de dígitos. Así, la mayoría de los procedimientos de generación de uniformes, trabajan con los decimales que dejan como residuo diferentes operaciones.

Un ejemplo muy sencillo de este procedimiento general es la obtención de dígitos para una distribución Bernoulli de ceros y unos. El esquema es el que sigue:

1. Se identifica o decide el valor de j o probabilidad de éxito. El éxito será codificado con un 1 y el fracaso con 0. Luego: $p(1) = j$ y $p(0) = 1 - j$.
2. Se genera un número u_i tal que $0 \leq u_i < 1$, utilizando un generador básico.
- 3.. Si $u_i \leq j$ el resultado es éxito (1). Si, por el contrario, $u_i > j$ el resultado será fracaso (0).

Si el experimento se repite n veces, cabe esperar que se observen nj unos y $n(1-j)$ ceros.

En el método, llamemos, *directo*, se debe contar con una máquina que genere valores que sigan una determinada distribución de probabilidad directamente. Así, por ejemplo, supongamos que interesa la relación:

valor	probabilidad
1	1/6
2	1/3
3	1/2

Se puede utilizar un dado y realizar la asignación:

valor	caras del dado
1	1
2	3 y 5
3	pares: 2, 4, 6

Como cada cara cuenta con una probabilidad de 1/6, entonces:

$$p(1) = 1/6$$

$$p(2) = 1/6 + 1/6 = 2/6 = 1/3$$

$$p(3) = 3 \cdot 1/6 = 3/6 = 1/2$$

El método directo parece funcionar, pero pensemos ahora en una situación más compleja, pero muy frecuente: se requiere una variable que siga una distribución normal. Ésta es continua, y la distribución de probabilidad es muy compleja. No es posible idear un artilugio mecánico, como el dado, que suministre valores en una distribución normal.

Pero puede utilizarse el dado, indirectamente. Veamos:

Se lanza el dado k veces y se anota el número de ellas en las que ha salido una cifra par. La proporción del suceso “resultado par” será una cantidad concreta c . Repitamos el experimento n veces y obtendremos n cantidades c_i . El conjunto de los valores c_i constituyen una variable aleatoria con una distribución que se aproxima a la normal conforme crece k y n . La figura 2 muestra las representaciones gráficas que han resultado para distintos valores de k y n . Obsérvese que conforme aumenta el número de experimentos, n , la aproximación a la normal mejora, si bien el número de valores posibles generado por la cantidad de tiradas, k , aumenta las exigencias para n . El procedimiento utilizado no es muy bueno: se ha recurrido al generador aleatorio que de un intérprete BASIC¹. Para acceder a un buen método de generación para la distribución normal, puede consultarse Kinderman y Ramage (1976)..

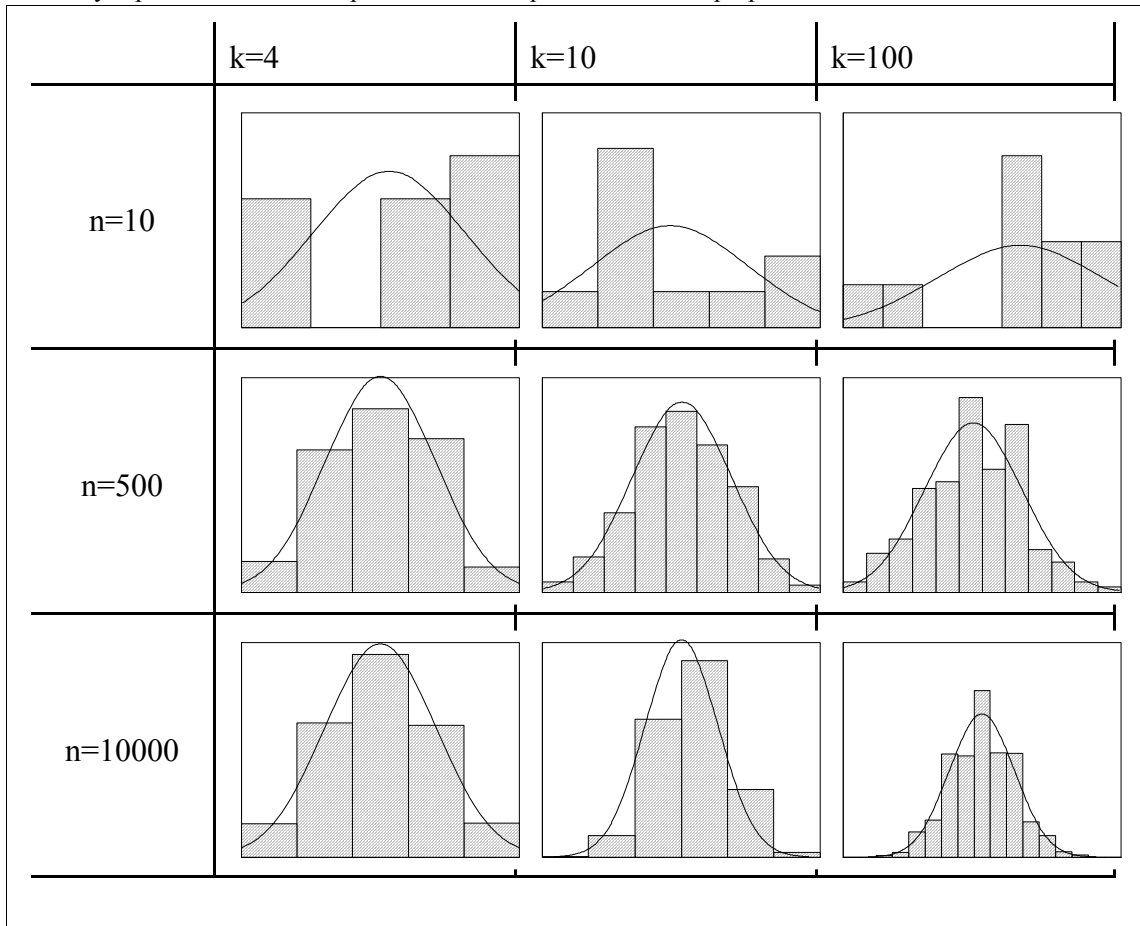
Así pues, los procedimientos para generar números aleatorios con el ordenador, se podrían clasificar en dos: los generadores básicos, ocupados por construir una distribución uniforme continua en el intervalo $(0,1)$; y los generadores específicos, encargados de obtener otras distribuciones a partir de la uniforme. Una buena revisión sobre generadores específicos puede encontrarse en Law y Kelton (1982). Una buena revisión histórica sobre los generadores básicos puede obtenerse en Teichroew (1965); a partir de entonces, casi todo ha sido congruencia lineal, tal y como veremos en un apartado posterior.

Por último, es importante señalar que la utilización de los métodos de generación de números aleatorios no goza de una aceptación plena, tal y como cabría sospechar. Su abuso en contextos poco idóneos o la utilización de generadores de baja calidad, dejan huella en las investigaciones. Algunos ejemplos se pueden encontrar en Ragland (1980) y Maddox (1994).

¹ En el Apéndice I, se encuentra la rutina utilizada para generar las 9 distribuciones.

Figura 2

Simulación del experimento “lanzamiento de un dado k veces”
y representación de n repeticiones del experimento con superposición de la curva normal.



Estrategias para la construcción de generadores básicos

Dado que interesa obtener una sucesión de valores comprendidos en el intervalo de 0 a 1, una aproximación inicialmente aceptable es trabajar directamente con decimales. Algunos ejemplos son:

1. Utilizar las cifras decimales del número π (citado en Teichroew, 1965)
2. El método de los cuadrados medios de Neyman (citado en Sobol, 1983), donde se utilizan cifras de 8 decimales, escogiéndose como aleatorios los cuatro centrales. En el siguiente paso, el decimal de cuatro dígitos resultante se eleva al cuadrado y vuelven a escogerse los cuatro dígitos centrales. Y así sucesivamente.
3. El método de Zielinski con irracionales (Zielinski, 1986)

No obstante, estos generadores cuentan con algunos inconvenientes acerca del sesgo en las generaciones o acerca de la comprobabilidad de sus propiedades.

Sin lugar a dudas, el procedimiento de generación más utilizado es un método recursivo, que utiliza cada valor para generar el siguiente, siguiendo el esquema:

$$u_i = f(u_{i-1})$$

donde u_i representa el valor uniforme generado en el lugar i -ésimo.

Y, dentro de esta categoría, los de mejor calidad y más ampliamente estudiados (Barry, 1996) son los llamados “generadores de congruencia lineal”. Su esquema general es:

$$u_i = a_i / m$$

$$a_i = \text{resto} ([a_{i-1} \cdot k + c] / m)$$

donde:

- u_i es el valor uniforme generado en el lugar i -ésimo
- a_i es el valor entero generado en el lugar i -ésimo, el primero, a_0 se denomina “semilla”
- k es una constante entera llamada “multiplicador”
- c es una constante entera llamada “sumador”
- m es una constante entera llamada “módulo”

Por lo general $c = 0$, por lo que el método se llama también “multiplicativo”. La tabla 1 muestra un ejemplo concreto en la generación de algunos números seguidos.

Tabla 1

Generación de un conjunto de números pseudoaleatorios con semilla 100, multiplicador 5 y módulo 64.

i	a_i	$a_i \cdot k$	resto ($a_i k/m$)	u_{i+1}
0	100	500	52	0,8125
1	52	260	4	0,0625
2	4	20	20	0,3125
3	20	100	36	0,5625
4	36	180	52	0,8125
5	52	260	4	0,0625

Obsérvese con atención el contenido de la tabla. Es evidente que los números se repiten. Hay un ciclo que permite únicamente 4 cantidades diferentes. Esta cantidad, 4, es lo que se denomina “periodo”. Veamos estos elementos independientemente.

Semilla: Dado que el procedimiento es determinístico, la misma semilla dará lugar siempre a la misma secuencia de números. Es una ventaja poder reproducir la secuencia en cualquier circunstancia. Pero también es un inconveniente utilizar cifras generadas de forma inamovible, como en una tabla impresa. Por esta razón, suele utilizarse como semilla, directa o indirectamente, el “reloj del sistema”, es decir el número de unidades de tiempo con que cuenta la máquina en donde se implementa el algoritmo.

Módulo: Conforme mayor es, mayor es también el periodo, pero más tiempo tarda la máquina en realizar los cálculos. En la práctica, se utilizan potencias de módulo 2, que son las más fáciles de manejar por el ordenador digital. A su vez, se recurren a tipos de datos concretos. Por ejemplo, el tipo *long* o “largo” requiere 4 bytes o, lo que es lo mismo, 32 bits. Ello permite un total de $2^{32} = 4.294.967.296$ estados diferentes. Si se utiliza el signo, éste consume un bit y los números generados son $2^{31} = 2147483648$. Por último, el número 0 está contemplado, por lo que el máximo valor para el módulo,

con n bits, será $2^n - 1$. Así, hay muchos estudios que recurren al módulo $2^{31} - 1$

Multiplicador:

Su decisión es altamente importante. El multiplicador confiere a la generación características especiales. Buena parte de la literatura específica se ha dedicado a estudiar multiplicadores. Así, por ejemplo, Fishman y Moore (1986) aconsejan los multiplicadores 742938285, 950706376, 1226874159, 62089911 y 1343714438 para el módulo $2^{31} - 1$. Sánchez-Bruno y San Luís-Costas (1995) aconsejan las cantidades 1343714438 y 742938285 como los mejores multiplicadores para el módulo $2^{31} - 1$, tras analizar los propuestos por Fishman y Moore (op.cit.) más otros de amplio uso. Atkinson (1980) estudia tres buenos multiplicadores (69069, 71365 y 100485) para los módulos 2^w con $w=30,32,33,35$, propuestos por Marsaglia.

Periodo:

Ya hemos mencionado su importancia. Existen generadores cuyo cometido principal es aumentar el periodo considerablemente. Así, el mencionado de Zielinski (1986) es un generador sin periodo. El método llamado “de Fibonacci” aumenta el periodo recurriendo a la suma de dos antecedentes en lugar de uno ($a_i = [a_{i-1} + a_{i-2}]/m$). Incluso, los métodos complejos derivados de la congruencia lineal, que abordamos seguidamente, buscan no sólo aumentar la aleatoriedad de la serie, sino romper el periodo. El estudio relativamente fácil del periodo en el método de la congruencia lineal, es otra de las características que lo hacen preferible a otros procedimientos de generación. Atkinson (1980) indica que el periodo llega a $m-1$ en el caso de módulos primos y $m/4$ si m es potencia de 2.

En el apéndice II, se encuentra un sencillo programa BASIC utilizado para generar números aleatorios siguiendo las indicaciones del usuario en cuanto a semilla, multiplicador, sumador y módulo.

Los métodos más rápidos utilizan únicamente un generador de congruencia lineal multiplicativo. Pero existen otros procedimientos que han dado buenos resultados. Así, una variación típica es utilizar dos generadores: el primero construye un vector de k números pseudo-aleatorios; el segundo selecciona un número concreto del vector. El procedimiento se hace más largo, pero el periodo aumenta y parece ser que también la aleatoriedad de la serie.

Un método con tres generadores es utilizado por Wichmann y Hill (1982). Y uno de los generadores más ambiciosos, anunciado como “universal” y con un complejo mecanismo interno, es dado a conocer por Marsaglia, Zaman y Tsang en 1990. En el apéndice III se encuentran los listados, en C, de ambos generadores, además del sencillo de recurrencia lineal que parte de los trabajos previos de Marsaglia, analizado por Atkinson (1980).

La calidad de los generadores: estrategias para su estudio a posteriori

Supongamos que alguien lanza un dado en 636 ocasiones, anotando los resultados, y obtiene las frecuencias que figuran en la siguiente tabla:

cara	frecuencias
1	61
2	64
3	315
4	71
5	60
6	65

parece ser que la cara numerada con un 3 ha aparecido un número de veces muy superior al resto. La conclusión más fácil es que el dado no se encuentra equilibrado, sino sesgado hacia la cara 3, que aparece casi la mitad de las veces. Cabría esperar que todas las caras aparecieran la misma cantidad de ocasiones, más o menos. 636 lanzamientos equivalen a 106 veces cada una de las caras.

Así pues, una conclusión fácilmente asumible en el ejemplo, es que el conjunto de lanzamientos del dado no se corresponde a una sucesión aleatoria, puesto que “las frecuencias observadas se alejan sensiblemente de las que cabría esperar si el procedimiento fuera aleatorio”. Una estrategia para comparar una distribución de frecuencias esperadas con otra observada es utilizar la prueba de Chi cuadrado. En el ejemplo, el valor sería $\chi^2 = 495,208$ y su grado de significación asociado 0,000. Así pues, la distribución observada se aleja significativamente de la esperada y se rechaza la hipótesis de aleatoriedad.

Este tipo de pruebas, comparación de las frecuencias esperadas y observadas, se denomina “test frecuencial” en las pruebas utilizadas para comprobar la aleatoriedad de una generación de dígitos. Si se cuenta con una variable continua, se puede utilizar la prueba de una sola muestra de Kolmogorov-Smirnoff con la distribución acumulada de la variable.

Otras estrategias² son:

- Test serial: se comparan los pares consecutivos de dígitos. Si éstos van de 0 a 9, cabe esperar 100 parejas con una frecuencia esperada de $N/100$.
- Otra estrategia fué utilizada por Mahalanobis (citado en Teichroew, 1965): comparar los momentos esperados con los observados. Mahalanobis comparaba los cinco primeros momentos.

Las pruebas mencionadas tienen un denominador común: se utiliza “la distribución” en su conjunto, no “la serie” o “secuencia”. Pero cabe esperar que sea, precisamente, la secuencia lo más aleatorio. De hecho, una distribución observada que coincida plenamente con la esperada es *demasiado* sospechosa. Observemos, por ejemplo, la siguiente secuencia de dígitos comprendidos en el intervalo (1,6):

1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6,...

La secuencia es claramente no aleatoria, es totalmente sistemática. No obstante, pasaría elegantemente una prueba de aleatoriedad basada en la comparación de distribuciones. Por esta razón, se requieren pruebas que aborden precisamente la sucesión y no los resúmenes de ésta. Estrategias concretas son:

- Análisis de las rachas. Es tan peculiar para la comprobación de la aleatoriedad que puede aparecer con la denominación “prueba de la aleatoriedad”. En la sucesión de dígitos aleatorios, se observarán repeticiones de 2, 3, 4, etc... dígitos de igual valor. Estas repeticiones se denominan “rachas”. Para que la sucesión pueda ser considerada aleatoria, las rachas observadas para cada dígito deben ser similares a las esperadas. Kermack y McKendrick, citados en Teichroew (1965) utilizan rachas de longitud 2 a 7 dígitos y realizan las comprobaciones “hacia delante” y “hacia atrás”, al igual que Fishman y Moore (1982).
- Análisis lagunar: entre dos dígitos de igual valor se encontrará un cantidad k de dígitos, es decir, una *laguna* de valor k . La frecuencia esperada para una laguna de valor i es:

$$\frac{N}{10} \left(\frac{9}{10} \right)^i$$

² Las denominaciones de los test frecuencial, serial, póker y lagunar han sido tomadas de Azorín (1969).

Así, por ejemplo, Kermack y McKendrick, citados en Teichroew (1965) utilizan lagunas de 3 a 9 dígitos.

- Test del póker: se compara la cantidad observada y esperada de k dígitos de igual valor consecutivos. Con dígitos en el intervalo (0,9) la frecuencia esperada es

$$\frac{1}{10^{k+1}} \frac{N}{k}$$

Usualmente, $k = 4$, lo que justifica el nombre de la prueba.

- Autocorrelaciones con retardo. Una autocorrelación es la correlación de una variable consigo misma, generando algún retardo en la serie. Un retardo de orden i implica que las parejas se establecen de cada dato con el que se encuentra distante en i posiciones. Así, en una autocorrelación con retardo 1, se correlaciona una variable consigo misma, una vez que se han desplazado todos los datos una posición. Cabe esperar que, en una sucesión aleatoria de dígitos, las autocorrelaciones con cualquier retardo tengan el valor aproximado 0.

Otros análisis, sensiblemente más complejos, parten de considerar n-uplas de cifras aleatorias. Marsaglia inició el análisis que más tarde se llamaría del enrejado (lattice test), sugiriendo los principios en 1968 y desarrollando una técnica concreta en 1972 (citado en Atkinson, 1980). Tanto la prueba del enrejado como la espectral (citado en Atkinson, op.cit.) tienen el mismo fundamento:

1. Se construye una tabla de doble entrada donde en filas y columnas se encuentran todas las cifras aleatorias diferentes generadas. En el caso de las unidades, (de 0 a 9), la tabla contendrá $10 \times 10 = 100$ casillas. Éste es el caso de las parejas, para trios serán matrices de tres dimensiones, para n-uplas, n dimensiones. Esta matriz es denominada estructura de enrejado.
2. En la prueba del enrejado, se calculan las razones entre las casillas. La aleatoriedad exige que las razones rondan la unidad.
3. En la prueba espectral, se calculan distancias entre planos bidimensionales.

Referencias

Atkinson, A.C. (1980). Tests of Pseudo-random Numbers. *Applied Statistics*, 29 (2), 164-171.

-
- Azorin, F. (1969). *Curso de muestreo y aplicaciones*. Madrid: Aguilar.
- Bar-Hillel, M. y Wagenaar, W.A. (1993). The perception of randomness. En G. Keren y C. Lewis (Eds.). *A handbook for data analysis in the behavioral sciences: methodological issues*. Hillsdale, New Jersey: Erlbaum. 369-393.
- Barry, T. M. (1996). Recommendations on the testing and use of pseudo-random number generators used in Monte Carlo analysis for risk assessment. *Risk Analysis*, 16 (1), 93-105.
- Brugger, P.; Milicevic, A.; Regard, M. y Cook, N.D. (1993). Random-number generation and the menstrual cycle: Preliminary evidence for a premenstrual alteration of frontal lobe functioning. *Perceptual and Motor Skills*, 77 (3), 915-921.
- Brugger, P.; Monsch, A. U.; Salmon, D.P. y Butters, N. (1996). Random number generation in dementia of the Alzheimer type: A test of frontal executive functions. *Neuropsychologia*, 34 (2), 97-103.
- Brugger, P.; Pietzsch, S.; Weidmann, G.; Biro, P. y Alon, E. (1995). Stroop-type interference in random-number generation. *Psychological Reports*, 77, 387-390.
- Chapanis, A. (1995). Human production of "random" numbers. *Perceptual and Motor Skills*, 81 (2), 1347-1363.
- Fishman, G.S y Moore, L.R. (1982). A Statistical Evaluation of Multiplicative Congruential Random Number Generators with Modulus $2^{31}-1$. *Journal of the American Statistical Association*, 77 (377), 129-136.
- Ginsburg, N. y Karpiuk, P. (1995). Simulation of human performance on a random generation task. *Perceptual and Motor Skills*, 81 (3), 1183-1186.
- Ginsburg, N. y Wieggersma, S. (1991). Response bias and the generation of random sequences. *Perceptual and Motor Skills*, 72 (2), 1332-1334.
- Hori, T.; Iwaki, T. y Hayashi, M. (1996). Effects of smoking on EEG alpha asymmetry during production of random sequences of numbers. *Perceptual and Motor Skills*, 82 (3), 827-834.
-

-
- Itagaki, F.; Niwa, S. I.; Itoh, K. y Momose, T. (1995). Random number generation and the frontal cortex. *International Journal of Psychophysiology*, 19 (1) 79-80.
- Kinderman, A.J. y Ramage, J.G. (1976). Computer generation of normal random variables. *Journal of American Statistical Association*, 71 (356), 893-896.
- Law, A.M. y Kelton, W.D. (1982). *Simulation Modeling and Analysis*. New York: McGraw-Hill.
- Maddox, J. (1994). The poor quality of random numbers. *Nature*. 372 (6505), 403.
- Marsaglia, G. (1968). Random numbers fall mainly in the planes. *Proceedings of the National Academy of Science*, 61, 25-28.
- Marsaglia, B.; Zaman, A. y Tsang, W.W. (1990). Toward a universal random number generator. *Statistics & Probability Letters*. 8, 35-39.
- Rabinowitz, F. M.; Dunlap, W. P.; Grant, M. J. y Campione, J. C. (1989). The rules used by children and adults in attempting to generate random numbers. *Journal of Mathematical Psychology*, 33 (3), 227-287.
- Ragland, R. (1980). On Playing Dice with the Universe: Problems in the Use of Random Number Tables in Social Science Research. *Journal of Thought*, 15 (1), 93-97.
- Reichenbach, H. (1949). *The theory of probability*. Berkeley, C.A.: University of California Press.
- Rosenberg, S; Weber, N.; Crocq, M.A. y Duval, F. (1990). Random number generation by normal, alcoholic and schizophrenic subjects. *Psychological Medicine*, 20 (4), 953-960.
- Sakata, S.; Shinohara, J.; Hori, T. y Sugimoto, S. (1995). Enhancement of randomness by flotation rest (restricted environmental stimulation technique). *Perceptual and Motor Skills*, 80 (3), 999-1010.
- Sánchez-Bruno, A. y San Luís-Costas, C. (1995). A statistical analysis of seven multipliers for linear congruential random numbers generators with modulus $2^{31}-1$. *Quality & Quantity*, 29, 331-337.
-

- Sobol, I. M. (1983). *Método de Montecarlo*. Colección “Lecciones populares de Matemáticas”. 2ª Edición. Moscú: MIR.
- Spatt, J. y Goldenberg, G. (1993). Components of random generation by normal subjects and patients with dysexecutive syndrome. *Brain and Cognition*, 23, 231-242.
- Teichroew, D. (1965). A history of distribution sampling prior to the era of the computer and its relevance to simulation. *Journal of American Statistical Association*, Marzo, 27-49.
- Vitulli, W.F.; DePace, A.N. y Holland, B.E. (1993). Gender effects on “predictions” of computer-random numbers as a function of feedback conditions. *Perceptual and Motor Skills*, 76 (2), 479-488.
- Wagenaar, W.A. (1972). Generation of random sequences by human subjects: a critical survey of literature. *Psychological Bulletin*, 77, 65-72.
- Zielinski, R. (1986). A quasi-random number generator with infinite period. *Statistics & Probability Letters*, 4, 259.
-

Apéndice I

Simulación de la tirada de un dado para la aproximación a la normal

```
10 RANDOMIZE TIMER
20 INPUT "n,k: ",N,K
30 INPUT "salida: ",SAL$
40 OPEN SAL$ FOR OUTPUT AS #1
50 FOR Z=1 TO N
51   locate 10,10 : print z
60   PAR=0
70   REM
80   REM Simula el lanzamiento de un dado k veces
90   REM
100  FOR Y=1 TO K
110    NUM=RND*6
120    IF NUM<1 THEN 180
130    IF NUM<2 THEN 170
140    IF NUM<3 THEN 180
150    IF NUM<4 THEN 170
160    IF NUM<5 THEN 180
170    PAR=PAR+1
180  NEXT Y
190  PRINT #1, USING "#####";PAR/K*10000
200 NEXT Z
210 CLOSE : SYSTEM
```

Apéndice II:**Programa para la generación de números aleatorios**

```

10 CLS : KEY OFF
20 LOCATE 1,25 : PRINT "Generación de números aleatorios"
30 LOCATE 2,25 : PRINT "=====
40 REM
50 REM   Presenta la información
60 REM
70 PRINT : PRINT "       Nombre para el archivo de salida:"
80 PRINT "       Número de dígitos a generar:"
90 PRINT : PRINT "       Características de la generación:" : PRINT
100 PRINT "           S:Semilla:"
110 PRINT "           C:Multiplicador:"
120 PRINT "           D:Sumador:"
130 PRINT "           M:Módulo:"
140 PRINT : PRINT "La fórmula seguida es" : PRINT
150 PRINT "           a                                     + , "
160 PRINT "           i                                     * a · C + D * "
170 PRINT "       u = )))) donde a = S   y   a = resto * i-1 * "
180 PRINT "           i M                               0   i * ))))))) * "
190 PRINT "                                           *           M * "
200 PRINT "                                           .           - "
210 PRINT
220 REM
230 REM   Pide información
240 REM
250 LOCATE 4,40 : INPUT " ",SAL$ : REM Archivo de salida
260   IF SAL$="" THEN SAL$="salida.gnr"
270   LOCATE 4,40 : PRINT "«";SAL$;"»"
280 LOCATE 5,40 : INPUT " ",NDIG# : REM Números a generar
290   IF NDIG#<1 THEN 570
300 LOCATE 9,25 : INPUT " ",S# : REM Semilla aleatoria
310   IF S#<1 THEN S#=TIMER*100
320   LOCATE 9,25 : PRINT S#;" "
330 LOCATE 10,25 : INPUT " ",C# : REM Multiplicador
340   IF C#<1 THEN C#=1
350   LOCATE 10,25 : PRINT C#;" "
360 LOCATE 11,25 : INPUT " ",D# : REM Sumador
370   IF D#<0 THEN D#=0
380   LOCATE 11,25 : PRINT D#;" "
390 LOCATE 12,25 : INPUT " ",M# : REM Módulo
400   IF M#<1 THEN M#=1
410   LOCATE 12,25 : PRINT M#;" "
420 FOR A=1 TO 400 : PRINT " "; : NEXT A
430 REM
440 REM   Realiza la generación
450 REM
460 OPEN SAL$ FOR OUTPUT AS #1
470 LOCATE 16,15 : PRINT "Dígitos generados:"
480 CUENTA#=0
490 CUENTA#=CUENTA#+1
500   J#=S#*C#+D#
510   S#=J#-INT(J#/M#)*M#
520   PRINT #1, USING " #"; INT(S#*10/M#);
530   LOCATE 16,40 : PRINT CUENTA#
540 IF CUENTA#<NDIG# THEN 490
550 LOCATE 19,10 : PRINT "Fin del proceso. Pulsa una tecla"
560 A$=INKEY$ : IF A$="" THEN 560
570 CLOSE : CLS : SYSTEM

```

Apéndice III

Listado en código C, de los algoritmos de tres métodos de generación

```
# include <time.h>
# include <math.h>
# include <stdio.h>

double U[97], C, CD, CM;
int IP, JP;
unsigned long I, J, K, L, IX, IY, IZ, AT;
long tiempo_antes;

unsigned long genera_semillas (unsigned long inicio)
{
    if (!inicio) inicio=time(0);
    // Marsaglia ...
    K=inicio;
    do {
        I=(K*53+1)%178+1; J=(I*53+1)%178+1; K=(J*53+1)%178+1;
        } while ((I+J+K)==3);
    L=(K*53+1)%169;
    // Wichmann ...
    IX=(170*inicio)%30000; IY=(170*IX)%30000; IZ=(170*IY)%30000;
    // Atkinson
    AT=inicio;
    return inicio;}

void inicializa_MZT (void)
{register char ii,jj;
 double s,t;
 int i=I, j=J, k=K, l=L, m;
 for (ii=0;ii<97;ii++)
     {s=0; t=.5;
      for (jj=0;jj<24;jj++)
          {m=(k*((i*j)%179)%179);
           i=j; j=k; k=m;
           l=(53*l+1)%169;
           if (((l*m)%64)>=32) s+=t;
           t*=.5;}
      U[ii]=s;}
 C= (double)362436/16777216;
 CD=(double)7654321/16777216;
 CM=(double)16777213/16777216;
 IP=96; JP=32;}

unsigned long inicializa (int modelo, unsigned long semilla)
{semilla=genera_semillas(semilla);
 if (modelo==1) inicializa_MZT();
 return semilla;}

double duniforme_MZT (void)
{double uni;
 uni=U[IP]-U[JP]; if (uni<0) uni+=1;
 IP--; if (IP<0) IP=96;
 JP--; if (JP<0) JP=96;
```

```
C-=CD;          if (C <0) C+=CM;
uni-=C;         if (uni<0) uni+=1;
return uni;};

double duniforme_WH (void)
{double suma;
  IX = ((171*IX)%30269);
  IY = ((172*IY)%30307);
  IZ = ((170*IZ)%30323);
  suma =(double) IX/30269 +(double) IY/30307 + (double) IZ/30323;
  return fmod(suma,1);}

double duniforme_A (void)
{AT = (AT * 100485)%4294967295;
return (double) AT/4294967295;}
```
