USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
FACULTADE DE MATEMÁTICAS

Traballo Fin de Grao

INTRODUCCIÓN A LA SIMULACIÓN

Martina Basdediós Tilve

Julio 2022

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRAO DE MATEMÁTICAS

Traballo Fin de Grao

INTRODUCCIÓN A LA SIMULACIÓN

Martina Basdediós Tilve

Julio 2022

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Trabajo propuesto

Área de Coñecemento: Estadística e Investigación Operativa
Título: Introducción a la simulación
Breve descrición do contido
La simulación se puede definir de manera general como una técnica para estudiar sistemas del mundo real imitando su comportamiento y utilizando un modelo matemático del sistema implementado en una computadora. La simulación es una herramienta para evaluar el desempeño de un sistema, existente o propuesto, con el fin de evaluar y obtener la mayor información posible del sistema en estudio. La simulación también implica probar y comparar diseños alternativos, y validar y explicar los resultados de la simulación. El objetivo de este trabajo es conocer las técnicas básicas de simulación: generación de números pseudoaleatorios, generación de variables aleatorias con su implementación en la computadora para su utilización en las aplicaciones en los campos de Estadística y de la Investigación Operativa.
Recomendacións
Outras observacións

Índice

Resumen

La simulación estadística nos permite estudiar sistemas del mundo real a partir de modelos que imiten sus características y que puedan ser implementados en un ordenador. En este trabajo, haremos una introducción a diferentes técnicas que se emplean en simulación. Empezaremos haciendo un repaso de los conceptos básicos de probabilidad y estadística que vamos a necesitar conocer. A continuación, hablaremos sobre los números pseudoaleatorios y sobre cómo se puede contrastar su validez para ser empleados en simulación. Veremos también algunos métodos que podemos utilizar para generar valores de distintas variables aleatorias, que serán fundamentales para construir los modelos de los sistemas reales. Finalmente, presentaremos un par de ejemplos en los que vamos a simular sistemas de la vida real, poniendo en práctica las ideas que veremos a lo largo del trabajo.

Abstract

Statistical simulation allows us to study real world systems using models that imitate their characteristics and that can be implemented on a computer. In this work, we will give an introduction to different techniques used in simulation. We will start by reviewing the basic concepts of probability and statistics that we will need to know. Then, we will talk about pseudorandom numbers and how to test their validity in order to be used in simulation. We will also see some methods we can use to generate values of different random variables, which will be fundamental to build the models of real systems. Finally, we will present a couple of examples in which we will simulate real life systems, putting into practice the ideas that we will see throughout the work.

Introducción

¿Qué podemos entender por simulación estadística? La *simulación estadística* (en adelante, *simulación*) es una forma de "modelizar" acontecimientos que percibimos como aleatorios o impredecibles, de manera que los resultados simulados se aproximen a los del mundo real. De esta forma, al observar estos resultados, podemos obtener información sobre lo que está ocurriendo en la realidad. En este contexto, hablaremos de un *sistema real* como un conjunto de elementos que, relacionados entre sí, contribuyen a un determinado objeto. Podemos considerar como sistema real un banco, compuesto por clientes y cajeros, en el que podríamos estar interesados en conocer, por ejemplo, el número de clientes que entran en el banco cada hora, o el tiempo medio que tarda en ser atendido cada cliente. Las técnicas de simulación son esenciales para el estudio de los sistemas reales, pues en ocasiones puede no ser viable su observación directa.

El hecho de que no siempre sea viable estudiar un sistema de manera directa puede deberse a diversos factores, entre ellos el elevado coste que esto pueda suponer o la lentitud del proceso. Pensemos por ejemplo que queremos estudiar cómo afectaría la instalación de un nuevo semáforo en una determinada calle de una ciudad. Si queremos estudiarlo de manera directa tendríamos que esperar primero a que el semáforo sea instalado, con el coste económico que esto supone, para luego analizar el funcionamiento del mismo durante horas o incluso días.

El primer paso en la simulación es modelizar el sistema real que queremos estudiar mediante un modelo matemático, es decir, un modelo abstracto en el que es posible describir el comportamiento del sistema utilizando lenguaje matemático. El modelo suele recoger las características básicas del sistema de forma que se pueda trabajar con él de manera más práctica y económica que la experimentación directa. Uno de los principales objetivos del estudio de un modelo es dar respuesta a determinadas preguntas o cuestiones que nos podamos hacer para mejorar nuestra comprensión sobre lo que está ocurriendo en el sistema real.

Nos podríamos plantear ahora cómo elegir de manera adecuada un modelo matemático. ¿Es mejor optar por un modelo complejo que represente el sistema lo más fielmente posible o, por el contrario, se debe formular un modelo que recoja únicamente las características más importantes del sistema? Está claro que el principal inconveniente en el primer caso es la dificultad que conlleva

analizar matemáticamente un sistema demasiado complejo. Por esta razón, son preferibles los modelos matemáticos más simplificados que permitan obtener resultados exactos en un tiempo razonable. Aún así, a veces no siempre es posible la resolución analítica de un modelo, siendo necesario recurrir a valores aproximados a través de otras técnicas, como el cálculo numérico o la simulación. Estas técnicas han experimentado una gran evolución gracias al enorme avance que ha habido en el campo de la computación. Esto ha permitido abordar un nuevo enfoque en el estudio de los sistemas reales, pues ahora es posible analizar modelos cada vez más complicados.

En este trabajo haremos una introducción a la simulación estadística. Comenzaremos con el estudio de los números pseudoaleatorios para presentar a continuación algunos de los métodos utilizados para generar valores de distintas variables aleatorias. Finalmente, veremos un par de ejemplos en los que podemos utilizar la simulación para obtener información sobre sistemas de la vida real.

Capítulo 1

Introducción a la probabilidad y estadística

Como ya hemos mencionado, uno de los objetivos de la simulación estadística es el estudio de fenómenos aleatorios. Trabajaremos entonces con modelos estocásticos, para los que es imprescindible conocer y manejar adecuadamente los conceptos básicos de la teoría de la probabilidad, las variables aleatorias con sus distribuciones de probabilidad y los contrastes de hipótesis. Por esta razón, presentaremos estos conceptos a lo largo de este capítulo.

1.1. Espacio muestral y sucesos

Entendemos por *experimento aleatorio* un proceso para el que no podemos predecir cuál será su resultado, pero para el que sí conocemos todos los posibles resultados de antemano. Además, el experimento debe poder repetirse en condiciones idénticas, y llamamos *realización* a cada una de las repeticiones que hacemos del experimento. Podemos pensar en el lanzamiento de un dado o una moneda como ejemplos de experimentos aleatorios. El *espacio muestral* de un experimento aleatorio es el conjunto de todos sus posibles resultados, y lo denotamos por Ω. Llamamos *suceso* a cualquier subconjunto de Ω, y se dice que un suceso A *ocurre* si el resultado del experimento es un elemento de A. Por ejemplo, cuando lanzamos un dado, el espacio muestral de este experimento aleatorio es $\Omega = \{1, 2, 3, 4, 5, 6\}$, mientras que un posible suceso sería "que salga un múltiplo de 3", es decir, $A = \{3, 6\}$. En este caso, A *ocurre* si tras lanzar el dado obtenemos un 3 o un 6.

A partir de dos sucesos A y B podemos definir otros nuevos, como por ejemplo la *unión*, $A \cup B$, o la *intersección*, $A \cap B$. Mientras que el suceso unión ocurre siempre que alguno de los dos sucesos A o B ocurran, la intersección sólo ocurre en el caso de que ambos sucesos ocurran a la vez.

Estas definiciones se pueden generalizar para un número arbitrario de sucesos. Tenemos también otras definiciones, como el suceso complementario de un suceso A (A^c: "que no ocurra A"), y los sucesos *seguro* (Ω: "siempre ocurre") e *imposible* ($\emptyset$: "nunca ocurre").

1.2. Probabilidad

En el contexto de un experimento aleatorio podremos asignar a cada suceso A del espacio muestral una medida que refleje las posibilidades de que este ocurra, a la que llamaremos *probabilidad del suceso A*. Veremos a continuación la definición axiomática de probabilidad introducida por Kolmogorov. Para ello, necesitamos definir primero el concepto de σ-álgebra. Si Ω es un conjunto no vacío, decimos que $\mathcal{F} \subset \mathcal{P}(\Omega)$ es una *σ-álgebra* en Ω si $\emptyset,\ \Omega \in \mathcal{F}$, y además $\mathcal{F}$ cumple:

1. Si $A \in \mathcal{F}$, entonces $A^c \in \mathcal{F}$.
2. Si $\{A_n\}_{n\in\mathbb{N}} \subset \mathcal{F}$, entonces $\bigcup_{n\in\mathbb{N}} A_n \in \mathcal{F}$.
3. Si $\{A_n\}_{n\in\mathbb{N}} \subset \mathcal{F}$, entonces $\bigcap_{n\in\mathbb{N}} A_n \in \mathcal{F}$.

Tenemos así el par $(\Omega, \mathcal{F})$, donde Ω es el espacio muestral de un experimento aleatorio y $\mathcal{F}$ una σ-álgebra de sucesos asociada, $(\Omega, \mathcal{F})$. Una *probabilidad* en $(\Omega, \mathcal{F})$ es una función $P : \mathcal{F} \longrightarrow [0, 1]$ de modo que:

1. $P(A) \geq 0,\ \forall A \in \mathcal{F}$.
2. $P(\Omega) = 1$.
3. Si $\{A_i\}_{i=1}^{\infty}$ es una familia de sucesos disjuntos dos a dos, entonces:

$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$$

Esta última propiedad recibe el nombre de *aditividad numerable*. La terna $(\Omega, \mathcal{F}, P)$ conforma lo que conocemos como un *espacio de probabilidad*. Diremos entonces que $P(A) \in [0, 1]$ es la *probabilidad asociada al suceso A*. A partir de la anterior definición se obtienen una serie de propiedades de la probabilidad, que no presentaremos aquí. Sirva como ejemplo que $P(\emptyset) = 0$ y $P(A^c) = 1 - P(A)$.

La probabilidad de un suceso A puede verse afectada si sabemos que otro suceso B ha ocurrido previamente. Se define así la *probabilidad de A condicionada a B* como $P(A/B) = \frac{P(A\cap B)}{P(B)}$, siempre que $P(B) \neq 0$. Diremos que dos sucesos A y B son *independientes* si se cumple:

$$P(A \cap B) = P(A)P(B)$$

En este caso, se tiene que $P(A/B) = P(A)$.

1.3. Variables aleatorias y sus distribuciones de probabilidad

Llamaremos *variables aleatorias* a todas aquellas cantidades o medidas de interés determinadas a partir de los resultados de un experimento aleatorio. Por ejemplo, si nos encontramos ante un modelo que representa el funcionamiento de un supermercado, podríamos considerar como variables aleatorias el número de clientes que entran en el supermercado cada hora, o cuánto tiempo tarda cada cliente en ser atendido en la caja. Más formalmente, una *variable aleatoria* es una función real definida en el espacio muestral, es decir, $X : \Omega \longrightarrow \mathbb{R}$. Atendiendo a los distintos valores que puede tomar una variable aleatoria, podemos establecer dos grandes grupos: el de las variables aleatorias discretas y el de las variables aleatorias continuas.

Una *variable aleatoria discreta* es una función $X : \Omega \longrightarrow \mathbb{R}$ que toma un número finito o infinito numerable de valores. Una variable aleatoria discreta queda determinada una vez se establecen los valores que esta puede tomar y sus respectivas probabilidades, es decir, la distribución de probabilidad de la variable aleatoria. Se define entonces la *función de masa* o *función de probabilidad*, $p : \mathbb{R} \longrightarrow [0, 1]$, de una variable aleatoria discreta X como:

$$p(x) = P\{X = x\},\ x \in \mathbb{R}.$$

Esto es, para cada $x \in \mathbb{R}$ estamos considerando la probabilidad de que la variable aleatoria discreta tome precisamente ese valor. Si X toma los valores $\{x_1, x_2, \dots\}$ entonces se cumple:

$$p(x_i) > 0, \quad \forall i = 1, 2, \dots, \quad \sum_{i=1}^{\infty} p(x_i) = 1 \quad y \quad p(x) = 0, \quad \forall x \in \mathbb{R} \setminus \{x_1, x_2, \dots\}.$$

Si tenemos una variable aleatoria X y existe una función $f : \mathbb{R} \longrightarrow \mathbb{R}$ de forma que:

$$f(x) \geq 0,\ \forall x \in \mathbb{R}, \quad \int_{-\infty}^{\infty} f(x)\ \mathrm{d}x = 1 \quad y \quad P\{a \leq X \leq b\} = \int_a^b f(x)\ \mathrm{d}x,\ \forall a, b \in \mathbb{R},$$

entonces diremos que X es una *variable aleatoria continua*. La función f recibe el nombre de *función de densidad* de la variable X. De esta manera, la distribución de probabilidad de una variable aleatoria continua queda determinada a partir de su función de densidad.

Definimos a continuación la *función de distribución* de una variable aleatoria X como una función $F : \mathbb{R} \longrightarrow [0, 1]$, de forma que:

$$F(x) = P\{X \leq x\},\ \forall x \in \mathbb{R}.$$

Si X es una variable aleatoria discreta con función de masa $p(x_i),\ i = 1, 2, \dots$, entonces:

$$F(x) = \sum_{x_i \leq x} p(x_i)$$

Por otra parte, si X es una variable aleatoria continua con función de densidad $f(x),\ x \in \mathbb{R}$, se tiene:

$$F(x) = \int_{-\infty}^{x} f(t)\ \mathrm{d}t, \quad \forall x \in \mathbb{R},$$

por lo que la función de densidad de una variable aleatoria continua es la derivada de su función de distribución, o lo que es lo mismo, $F'(x) = f(x),\ \forall x \in \mathbb{R}$.

También nos interesará conocer alguna medida característica de las variables aleatorias y que vamos a necesitar más adelante. Como medida de centralización tenemos la *esperanza* o *media* de una variable aleatoria X, que se define como $E[X] = \sum_{i=1}^{\infty} x_i \cdot P\{X = x_i\}$ en el caso de que X sea una variable aleatoria discreta, mientras que si se trata de una variable aleatoria continua se tiene $E[X] = \int_{-\infty}^{\infty} x \cdot f(x)\ \mathrm{d}x$. Como medida de dispersión de una variable aleatoria X de media μ vamos a considerar su *desviación típica*, que se puede obtener sin más que calcular la raíz cuadrada positiva de la *varianza* de X, definida como $Var(X) = E[(X-\mu)^2]$.

La esperanza de una variable aleatoria posee una interesante propiedad que vamos a utilizar en este trabajo, y que exponemos a continuación.

Teorema 1.1. *Si X es una variable aleatoria y $g : \mathbb{R} \to \mathbb{R}$ una función real, entonces $Y = g(X)$*[1] *es una variable aleatoria y se verifica:*

- *Si X es una variable aleatoria discreta que toma valores $\{x_1,\ x_2,\ \dots\}$ y tiene función de masa $p(x_i)$, $i = 1, 2, \dots$, la variable aleatoria Y tiene media $E[Y] = \sum_{i=1}^{\infty} g(x_i) \cdot p(x_i)$.*
- *Si X es una variable aleatoria continua con función de densidad f, la media de la variable aleatoria Y es $E[Y] = \int_{-\infty}^{\infty} g(x) \cdot f(x)\ \mathrm{d}x$.*

Otro resultado que también utilizaremos es el siguiente.

Teorema 1.2 (**Ley fuerte de los grandes números**). *Sean $X_1, X_2, \dots$ variables aleatorias independientes e idénticamente distribuidas con media μ. Entonces se tiene:*

$$P\left\{\lim_{n\to\infty} \frac{X_1 + X_2 + \cdots + X_n}{n} = \mu\right\} = 1$$

Esto quiere decir que, para un n suficientemente grande, el promedio de n variables aleatorias independientes e idénticamente distribuidas es *casi seguro* su media.

Es bastante habitual trabajar con más de una variable aleatoria, por lo que nos podría interesar conocer las distribuciones de las variables aleatorias multidimensionales o vectores aleatorios.

[1] Notación: denotamos por $Y = g(X)$ a la variable aleatoria que resulta de aplicar la transformación g a la variable aleatoria X, esto es, $Y : \Omega \xrightarrow{X} \mathbb{R} \xrightarrow{g} \mathbb{R}$.

Entendemos por *vector aleatorio* un vector de la forma $(X_1, \ldots, X_n)$, donde cada componente es una variable aleatoria. Por ejemplo, podemos considerar el vector aleatorio bidimensional $(X, Y) : \Omega \longrightarrow \mathbb{R}^2$. Si X e Y son variables aleatorias discretas, se define la función de masa conjunta: $p(x, y) = P\{X = x,\ Y = y\}$, mientras que en el caso continuo tendríamos su función de densidad conjunta, $f : \mathbb{R}^2 \longrightarrow \mathbb{R}, (x, y) \to f(x, y)$, cumpliendo las propiedades establecidas anteriormente para una función de densidad.

Se puede enunciar un resultado análogo al Teorema 1.1 para el caso en que $X = (X_1, \ldots, X_n)$ es un vector aleatorio y $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ una función real de n variables, considerando la función de masa conjunta de X (si $X_1, \ldots, X_n$ son variables aleatorias discretas) o su función de densidad conjunta (si las mencionadas variables aleatorias son continuas).

En el caso de un vector aleatorio bidimensional, las variables X e Y se dicen *independientes* si, para cualesquiera conjuntos C y D, los sucesos $A = \{X \in C\}$ y $B = \{Y \in D\}$ son independientes. Esto se traduce en el caso discreto en: $P\{X = x,\ Y = y\} = P\{X = x\} \cdot P\{Y = y\},\ \forall x, y \in \mathbb{R}$, y en el caso continuo se tiene: $f(x, y) = f_X(x) \cdot f_Y(y)$, siendo $f_X(x)$ y $f_Y(y)$ las funciones de densidad *marginales* de X e Y, respectivamente.

Otro concepto importante es la *covarianza* de dos variables aleatorias X e Y, dada por:

$$Cov(X, Y) = E[(X - \mu_X)(Y - \mu_Y)]$$

siendo μ_X y μ_Y las medias de X e Y, respectivamente. Notemos que si dos variables aleatorias X e Y son independientes entonces se tiene $Cov(X, Y) = 0$. Finalmente, definimos la *correlación* de X e Y como sigue:

$$Corr(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}}$$

1.4. Variables aleatorias discretas

En esta sección introducimos las distribuciones de variables aleatorias discretas con las que trabajaremos en los siguientes capítulos.

1.4.1. Distribución de Bernoulli

Consideremos un experimento aleatorio en el que solo puede haber dos posibles resultados, a los que nos referiremos como *éxito* y *fracaso*. Por ejemplo, podemos estar interesados en observar si cierta sustancia está presente o no en un material. Sea X la variable aleatoria definida como:

$$X = \begin{cases} 1 & \text{si ocurre A (\textit{éxito}: la sustancia está presente)} \\ 0 & \text{si no ocurre A (\textit{fracaso}: la sustancia no está presente)} \end{cases} \tag{1.1}$$

Denotemos por $p = P(A)$ la probabilidad de obtener un éxito. Decimos que la variable aleatoria discreta X definida en (1.1) tiene *distribución de Bernoulli* de parámetro p, y la denotamos por $Ber(p)$. Se tiene que $E[X] = p$, y la función de masa de X es:

$$P\{X = x\} = p^x(1-p)^{1-x},\ x \in \{0, 1\}$$

1.4.2. Distribución binomial

Llevamos ahora a cabo n realizaciones independientes y en idénticas condiciones del experimento aleatorio definido en el apartado anterior. A cada una de estas realizaciones la llamamos *intento*. Nos interesará entonces conocer el número de veces que ocurre A, o lo que es lo mismo, el *número de éxitos* que se producen en los n intentos.

Se denomina variable *binomial* a la variable aleatoria discreta X que representa el número de éxitos en n intentos independientes de un experimento aleatorio, siendo p la probabilidad de éxito en cada intento. Los parámetros de la variable son por tanto n y p, y la denotamos por $Bin(n, p)$. La función de masa es en este caso:

$$P\{X = i\} = \binom{n}{i} p^i(1-p)^{n-i},\ i = 0, 1, \ldots, n.$$

Por ejemplo, si consideramos el lanzamiento de un dado, podríamos definir la variable aleatoria binomial $X =$*"número de veces que sale un 6 en 15 lanzamientos"*. Tenemos así $n = 15$ intentos, siendo un "éxito" la obtención de un 6. Además, para cada intento conocemos la probabilidad de que esto ocurra, pues $P(A =$ *"que salga un 6"* $) = 1/6$.

Como podemos deducir a partir de la definición, una variable binomial no es más que una suma de variables de Bernoulli independientes. En efecto, si $X_1, X_2, \ldots, X_n$ son n variables de Bernoulli independientes de un mismo experimento aleatorio definidas como en (1.1) se obtiene la variable aleatoria binomial $X = \sum_{i=1}^{n} X_i$.

1.4.3. Distribución de Poisson

Una variable aleatoria X tiene distribución *de Poisson* de parámetro $\lambda > 0$ si toma valores en $0, 1, 2, \ldots$ y presenta la siguiente función de masa:

$$P\{X = i\} = e^{-\lambda}\frac{\lambda^i}{i!},\ i = 0, 1, 2, \ldots$$

Denotaremos esta distribución como *Poisson(λ)*. Estas variables surgen, por ejemplo, en situaciones en las que contamos el número de *éxitos* que ocurren en un intervalo de tiempo determinado, como puede ser el número de llamadas que recibe una central telefónica cada minuto.

1.4.4. Distribución geométrica

Una variable aleatoria X se denomina *geométrica* con parámetro p si presenta la siguiente función de masa:

$$P\{X = n\} = p(1-p)^{n-1},\ n = 1, 2, 3 \dots$$

donde n indica el número de realizaciones independientes del experimento y p es la probabilidad de obtener un *éxito* en cada realización. La función de distribución de X se puede expresar como:

$$F(x) = 1 - (1-p)^{x+1},\ x = 0, 1, 2, \dots$$

Denotamos esta distribución por *Geom(p)*. De esta manera, una variable aleatoria geométrica va a representar el número de realizaciones del experimento hasta obtener el primer éxito, como, por ejemplo, el número de lanzamientos de una moneda hasta obtener una cara o el número de tornillos que se producen en una fábrica hasta que uno sale defectuoso (si consideramos este hecho como un "éxito"). La esperanza de estas variables viene dada por $E[X] = \frac{1}{p}$.

1.5. Variables aleatorias continuas

Al igual que en la sección anterior, veremos ahora las distribuciones de variables aleatorias continuas con las que trabajaremos más adelante.

1.5.1. Distribución uniforme

Se dice que una variable aleatoria X tiene *distribución uniforme* en el intervalo (a, b), $a < b$, si su función de densidad viene dada por:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{si } a \leq x < b \\ 0 & \text{en otro caso.} \end{cases}$$

Denotaremos esta distribución como *U(a,b)*. Su función de distribución es:

$$F(x) = P\{X \leq x\} = \begin{cases} 0 & \text{si } x < a \\ \dfrac{x-a}{b-a} & \text{si } a \leq x < b \\ 1 & \text{si } x \geq b \end{cases}$$

1.5.2. Distribución normal

Se dice que una variable aleatoria X sigue una *distribución normal* de media μ y varianza σ^2 si su función de densidad viene dada por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)},\ x \in \mathbb{R}.$$

La denotaremos por *N(μ,σ^2)*. Su función de distribución es la siguiente:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{X} e^{-(t-\mu)^2/(2\sigma^2)} \mathrm{d}t, \ x \in \mathbb{R}.$$

Los parámetros μ y σ^2 de esta distribución son la esperanza y la varianza de la variable, es decir, $E[X] = \mu$ y $Var(X) = \sigma^2$. La distribución normal es una de las más importantes y de mayor uso de todas las distribuciones de probabilidad, al considerarse la más idónea para representar una gran variedad de mediciones relacionadas con la Física, la Química o la Biología.

1.5.3. Distribución exponencial

Se dice que una variable aleatoria X sigue una *distribución exponencial* de parámetro $\lambda > 0$ si su función de densidad viene dada por:

$$f(x) = \lambda e^{-\lambda x}, \quad 0 \leq x < \infty.$$

Denotaremos esta distribución por *Exp(λ)*. Su función de distribución viene dada por:

$$F(x) = \int_0^x \lambda e^{-\lambda x} \, \mathrm{d}x = 1 - e^{-\lambda x}, \quad 0 \leq x < \infty.$$

La esperanza de una variable aleatoria exponencial es $E[X] = \frac{1}{\lambda}$. Estas variables son empleadas, por ejemplo, para modelizar el tiempo de espera hasta la ocurrencia de un determinado suceso, como el tiempo que pasa un cliente en la cola hasta ser atendido en un banco. También se suelen emplear variables exponenciales para modelizar tiempos de vida, como el de un aparato electrónico (la variable sería el tiempo que transcurre hasta que este aparato deja de funcionar).

1.5.4. Distribución doble exponencial

Se dice que una variable aleatoria X sigue una *distribución doble exponencial* de parámetro $\lambda > 0$ si su función de densidad es la siguiente:

$$f(x) = \frac{\lambda}{2} e^{-\lambda|x|}, \ x \in \mathbb{R}.$$

Denotaremos esta distribución por *Dexp(λ)*. Su función de distribución vendrá dada por:

$$F(x) = \int_{-\infty}^{x} f(t)\mathrm{d}t = \begin{cases} \frac{1}{2}e^{\lambda x} & \text{si } x < 0 \\ 1 - \frac{1}{2}e^{-\lambda x} & \text{si } x \geq 0. \end{cases}$$

Se denomina *distribución doble exponencial estándar* a la distribución $Dexp(1)$.

1.5.5. Distribución gamma

Una variable aleatoria X se dice que sigue una *distribución gamma* con parámetro de escala $a > 0$ y parámetro de forma $p > 0$ si su función de densidad viene dada por:

$$f(x) = \begin{cases} \frac{a^p}{\Gamma(p)} \cdot x^{p-1} e^{-ax} & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$$

donde $\Gamma(p) = \int_0^\infty x^{p-1} e^{-x} \, \mathrm{d}x$. Denotaremos esta distribución por $\Gamma(a, p)$. En el caso particular de que el parámetro $p \in \mathbb{Z}^+$, esta distribución recibe el nombre de *distribución de Erlang*.

1.6. Inferencia estadística

Antes de terminar el capítulo haremos una pequeña introducción a la inferencia estadística y, en particular, a los contrastes de hipótesis, pues los necesitaremos en el capítulo siguiente.

1.6.1. Población y muestra

La *población* no es más que el conjunto de elementos o individuos sobre los que queremos realizar un determinado estudio. Por ejemplo, los habitantes de un determinado país para conocer la prevalencia del COVID-19. En muchas ocasiones, no es posible o viable estudiar toda la población, por lo que es habitual trabajar con una selección "representativa" de la misma. A esta selección la denominamos *muestra*, y n, el *tamaño muestral*, hará referencia al número de elementos que la conforman. Supondremos que todos los individuos tienen las mismas posibilidades de ser elegidos en la muestra.

Si X es una variable aleatoria que mide una cierta característica de la población de estudio, nos referiremos a su función de distribución como *función de distribución teórica*, $F(x)$, $\forall x \in \mathbb{R}$. Entendemos como *muestra aleatoria simple* de X a n variables aleatorias $(X_1, \ldots, X_n)$ independientes e idénticamente distribuidas como X, donde cada X_i representará a una posible realización de la muestra. De esta forma, al medir la característica que nos interesa en cada uno de los elementos de la muestra obtendremos un conjunto de valores $(x_1, \ldots, x_n)$, que son el resultado de una realización de $(X_1, \ldots, X_n)$. En nuestro ejemplo, tendríamos, para cada X_i, $i = 1, \ldots, n$,

$$X_i = \begin{cases} 1 & \text{si la persona } i\text{-ésima tiene el COVID-19} \\ 0 & \text{en caso contrario} \end{cases}$$

A continuación definimos la *función de distribución empírica* de X como sigue:

$$F_n(x) = \frac{\#\{X_i \leq x\}}{n}, \quad i = 1, 2, \ldots, n, \quad x \in \mathbb{R},$$

donde # representa el cardinal del conjunto $\{X_i \leq x\}$ y n es el tamaño muestral. Esta distribución es siempre discreta, pues representa la proporción de los valores observados $(x_1, \ldots, x_n)$ (obtenidos a partir de una muestra aleatoria simple de X) que son menores o iguales que un cierto $x \in \mathbb{R}$. La función de distribución empírica de X es una buena aproximación de su función de distribución teórica pues, por la ley fuerte de los grandes números, $F_n(x)$ converge a $F(x)$ con probabilidad 1 cuando $n \to \infty$. De esta forma, cuanto mayor sea el tamaño muestral, más se va a aproximar la distribución empírica a la distribución teórica de X.

Llamamos *estadístico* a la variable aleatoria $Y = T(X_1, \ldots, X_n)$, siendo $T : \mathbb{R}^n \longrightarrow \mathbb{R}$. De esta forma, la *distribución en el muestreo* del estadístico Y será la función de distribución de la variable $T(X_1, \ldots, X_n)$. Como ejemplo de estadístico podemos considerar la media muestral $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

1.6.2. Contrastes de hipótesis

El objetivo de los contrastes de hipótesis es decidir si una determinada hipótesis sobre la población es aceptada o rechazada en favor de otra, basándose en las observaciones obtenidas de una muestra. Tendremos entonces dos hipótesis: llamaremos *hipótesis nula* o H_0 a la hipótesis que queremos contrastar y que se da como cierta, mientras que la hipótesis "contraria" frente a la cual es contrastada será la *hipótesis alternativa* o H_a. Para llevar a cabo un contraste de hipótesis es fundamental disponer de una muestra de la población, mediante la cual vamos a decidir si aceptamos o rechazamos H_0 empleando un estadístico, que proporciona la discrepancia entre lo que se espera que ocurra bajo la hipótesis nula y lo que se observa en la muestra dada. Nos referiremos a este estadístico como *estadístico del contraste* y su función de distribución será conocida bajo H_0.

Llamaremos *nivel de significación* α a la probabilidad de rechazar H_0 siendo esta cierta. A partir del nivel de significación fijado para el contraste y la distribución del estadístico considerado bajo H_0, establecemos un *criterio de decisión* que divide el espacio muestral en dos regiones disjuntas: la *región crítica* (o R_c) y la *región de aceptación* (o R_a). Si el valor del estadístico para la muestra considerada cae dentro de R_c, entonces se rechaza H_0 en favor de H_a, mientras que si se encuentra en R_a no habría evidencias para rechazar H_0. También es importante conocer el *nivel crítico* o *p-valor* del contraste, que es el nivel de significación más pequeño que lleva a rechazar H_0.

Una vez hemos presentado los conceptos de probabilidad y estadística que vamos a necesitar en este trabajo, veremos a continuación diferentes métodos para obtener números pseudoaleatorios, así como algunas aplicaciones dentro del campo de la simulación.

Capítulo 2

Generación de números pseudoaleatorios

Para llevar a cabo la simulación es imprescindible disponer de *números aleatorios*, es decir, números "impredecibles" y "equiprobables" que se encuentran dentro de un determinado rango de valores. Serán de especial importancia los números aleatorios en el intervalo $(0, 1)$, pues, por sus propias características, son valores que siguen una distribución $U(0, 1)$, y a partir de ellos se pueden obtener números aleatorios en cualquier otro intervalo o rango de valores que necesitemos. Antes de la aparición de los ordenadores, lo habitual era obtener dichos números haciendo uso de procedimientos físicos, como, por ejemplo, considerar los resultados obtenidos al lanzar uno o más dados, los resultados de una lotería o una ruleta, o a partir del ruido blanco, que no es más que una señal aleatoria generada por algunos procesos físicos (naturales o artificiales) cuyos valores son impredecibles. En general, en este tipo de procedimientos parece razonable aceptar que todos sus resultados sean impredecibles y equiprobables, lo que conoceremos como *hipótesis de aleatoriedad e independencia*. Estos valores se almacenaban en tablas y podían consultarse cada vez que fuese necesario utilizar números aleatorios. Uno de los primeros ejemplos de tablas de este estilo fueron las elaboradas por la organización RAND Corporation en 1955, que contenían millones de números aleatorios generados mediante procesos físicos.

En la actualidad, estos números son generados por ordenador, y se llaman *pseudoaleatorios* debido a que, fijado el valor inicial (que recibe el nombre de *semilla*) los restantes están determinados. Aún así, tenemos que poder garantizar que estas secuencias se comporten como "números aleatorios". Si los números aleatorios generados están entre 0 y 1, al menos deberían comportarse como una muestra de una variable aleatoria con distribución $U(0, 1)$. En R podemos encontrar las funciones *set.seed* y *runif*[1], que permiten fijar una semilla, y generar valores de una distribución

[1]En la sección 2.2 de este capítulo comentaremos un poco más estas funciones de R.

uniforme en un determinado intervalo (el que considera por defecto es el $(0, 1)$), respectivamente. Tenemos a continuación un pequeño ejemplo:

```
set.seed(7435)  # Fijamos la semilla
runif(12)       # Generamos 12 valores de una U(0,1)
```

```
##  [1] 0.15491309 0.10480164 0.68592499 0.30810908 0.51440981 0.68033474
##  [7] 0.93703164 0.03061975 0.30864556 0.29804359 0.65666886 0.52063688
```

Un *generador de números pseudoaleatorios* es un algoritmo que, a partir de la semilla, obtiene de manera recursiva los demás valores de la secuencia (los cuáles están determinados una vez fijados el algoritmo y la semilla). Este tipo de generadores deben satisfacer las hipótesis de aleatoriedad e independencia y ajuste de los valores a una $U(0, 1)$ (una vez transformados en números entre 0 y 1), y estas se pueden comprobar mediante diversos contrastes de hipótesis.

Se define el *período* o la *longitud de ciclo* de un generador como el menor número entero positivo p tal que existe un n_0 natural de forma que $x_{i+p} = x_i$, para todo $i = n_0, n_0 + 1 \ldots$. Es decir, es la longitud de la secuencia generada hasta que esta vuelve a repetirse. Por lo tanto, es lógico pensar que un buen generador debe tener una longitud de ciclo grande para evitar que los valores empiecen a repetirse demasiado pronto. Otras de las propiedades recomendables para un buen generador son de carácter computacional: el algoritmo tiene que ser eficiente, tanto para calcular valores a una gran velocidad como para ocupar poca memoria en el ordenador.

A lo largo de los años se han desarrollado una gran variedad de algoritmos para generar números pseudoaleatorios. Algunos de los primeros fueron el método de los cuadrados medios de von Neumann o el método de Lehmer. Ambos tienen una estructura similar: fijado un número entero (la semilla) se realizan una serie de operaciones que se repiten tantas veces como sea necesario hasta obtener la cantidad de valores deseada. Posteriormente, estos valores se trasladan al intervalo $(0, 1)$ mediante una transformación. Uno de los principales inconvenientes que pueden surgir son las longitudes de ciclo demasiado cortas.

Ejemplo 2.1 (**Generación de números pseudoaleatorios mediante el método de los cuadrados medios**)**.** En primer lugar, fijamos la semilla, x_0, que debe ser un número entero de $2n$ cifras ($n \geq 1$), y la elevamos al cuadrado, obteniendo un número de $4n$ cifras (en algunos casos es necesario añadir ceros a la izquierda del resultado para cumplir esto último). Las $2n$ cifras centrales formarán el número x_1. Para continuar el proceso solo tenemos que elevar x_1 al cuadrado y repetir los pasos tantas veces como necesitemos. De forma general, podemos escribir:

$$x_{i+1} = \left\lfloor \left(x_i^2 - \left\lfloor \frac{x_i^2}{10^{3n}} \right\rfloor \cdot 10^{3n} \right) / 10^n \right\rfloor$$

Por ejemplo, si consideramos el número 54, de $2 \cdot 1 = 2$ cifras, y lo elevamos al cuadrado, obtenemos 2916, un número de $4 \cdot 1 = 4$ cifras. Si queremos obtener las $2 \cdot 1 = 2$ cifras centrales, es decir, 91, podemos seguir la expresión indicada arriba, teniendo en cuenta que $n = 1$. La parte entera de $\frac{2916}{10^3}$ es 2, y al multiplicar por 10^3 obtenemos 2000. Al restar 2000 de 2916 nos queda 916, por lo que ya solo tenemos que "eliminar" la última cifra. Si hacemos la división $\frac{916}{10}$ y nos quedamos con la parte entera obtenemos 91, como queríamos conseguir.

Los números definidos como $u_i = \frac{x_i}{10^{2n}} \in (0, 1)$ son los números pseudoaleatorios generados por el método. Veamos cómo podemos implementar este método en *R*:

```
cuadrados_medios <- function(i, n, x0){
  u0=x0/(10^(2*n))
  x=numeric()
  for(j in 1:(i-1)){
    x1=floor((x0^2 - floor((x0^2)/(10^(3*n))) * 10^(3*n))/10^n)
    x=c(x,x1); x0=x1
  }
  u=c(u0, x/(10^(2*n))); u
}
```

Usando la función que hemos programado, vamos a generar una secuencia de 20 números pseudoaleatorios por el método de los cuadrados medios, con valores $n = 2$ y $x_0 = 2022$:

```
cuadrados_medios(20, 2, 2022)
```

```
##  [1] 0.2022 0.0884 0.7814 0.0585 0.3422 0.7100 0.4100 0.8100 0.6100 0.2100
## [11] 0.4100 0.8100 0.6100 0.2100 0.4100 0.8100 0.6100 0.2100 0.4100 0.8100
```

Podemos observar una secuencia de cuatro números que empiezan a repetirse: 0'41, 0'81, 0'61 y 0'21, por lo que la longitud de ciclo en este caso es 4. En efecto, es una longitud de ciclo demasiado corta, ejemplificando así el inconveniente mencionado para este método. Otro problema de este generador lo encontramos cuando aparece un 0 en la secuencia de valores generados, pues a partir de ahí todos los valores obtenidos serán también cero. Veamos un ejemplo con $n = 2$ y $x_0 = 1010$:

```
cuadrados_medios(20, 2, 1010)
```

```
##  [1] 0.1010 0.0201 0.0404 0.1632 0.6634 0.0099 0.0098 0.0096 0.0092 0.0084
## [11] 0.0070 0.0049 0.0024 0.0005 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

En la práctica, necesitamos trabajar con métodos que se ajusten lo mejor posible a los requisitos mencionados anteriormente para un buen generador.

2.1. Generadores congruenciales

Este tipo de generadores presentan una formulación sencilla, y a menudo se emplean como base para otros generadores más sofisticados. Si consideramos un método congruencial lineal, una vez fijada la semilla x_0 de forma que $0 \leq x_0 < m$ para un cierto entero m, la idea es obtener los demás valores de forma recursiva empleando la siguiente expresión:

$$x_n \equiv a \cdot x_{n-1} + c \ (mod\ m), \tag{2.1}$$

donde a (*multiplicador*) y c (*incremento*) son enteros que cumplen $0 < a < m$ y $0 \leq c < m$ (en el caso de que $c = 0$, el método se llama *congruencial multiplicativo*).

Notemos que, como estamos realizando divisiones módulo m, los valores de la secuencia serán los posibles restos, es decir, $0, \ldots, m-1$. Si dividimos cada uno de los valores por m, esto es, x_n/m, obtenemos los valores comprendidos entre 0 y 1. Al igual que hicimos con el método de los cuadrados medios, veamos cómo podemos implementar un generador congruencial de forma sencilla en *R*.

```
gen_congruencial <- function(n,x0,m,a,c){
  x=numeric(); x=c(x,x0)
  for(i in 1:(n-1)){
    x1=(a*x0+c)%%m
    x=c(x,x1); x0=x1
  }
  u=x/m; u
}
```

Vamos a generar 10 números pseudoaleatorios con esta función, empleando como semilla $x_0 = 7$ y como parámetros los valores $m = 9$, $a = 4$ y $c = 3$:

```
gen_congruencial(10,7,9,4,3)
```

```
##  [1] 0.7777778 0.4444444 0.1111111 0.7777778 0.4444444 0.1111111 0.7777778
##  [8] 0.4444444 0.1111111 0.7777778
```

Observamos que la longitud de ciclo en este ejemplo es 3, por lo que es demasiado pequeña. Como ya hemos visto, necesitamos que la longitud de ciclo de un generador sea lo más grande posible. En el caso de los generadores congruenciales lineales, la longitud de ciclo será como mucho m, pues hay m posibles valores distintos, y en cuanto uno de ellos se repita, todos se repetirán cíclicamente a partir de él (pues cada valor depende unívocamente del anterior al estar la secuencia definida de forma recursiva). Por esta razón, es conveniente escoger un valor de m muy grande y estudiar qué propiedades deben tener los parámeros m, a y c para que el período del generador congruencial sea máximo. Se tiene el siguiente resultado.

Teorema 2.2 (de Hull y Dobell, 1962). *La secuencia definida por la relación de congruencia* $x_n \equiv a \cdot x_{n-1} + c \ (mod\ m)$ *tiene período máximo* m *si se cumple:*

1. $mcd(c, m) = 1$ *(es decir,* c *y* m *son primos entre sí o coprimos).*
2. $a \equiv 1 \ (mod\ p)$ *si* p *es un factor primo de* m*.*
3. $a \equiv 1 \ (mod\ 4)$ *si 4 es un factor de* m*.*

Demostración. En primer lugar, observemos que si c y m son primos entre sí y $a = 1$, se tiene que el período es m, es decir, el máximo posible. Nos centraremos entonces en estudiar los casos en los que $a \neq 1$, teniendo en cuenta que c y m son coprimos.

Utilizando la expresión dada por (2.1) y calculando los valores de $x_i,\ i = 1, \ldots, n-1$ de forma recursiva, podemos obtener la expresión general de x_n en función de x_0:

$$x_1 \equiv a \cdot x_0 + c \ (mod\ m)$$

$$x_2 \equiv a \cdot x_1 + c \ (mod\ m) = a^2 x_0 + (a+1) \cdot c \ (mod\ m) = a^2 x_0 + \frac{a^2 - 1}{a - 1} \cdot c \ (mod\ m)$$

$$\vdots$$

$$x_n \equiv a^n x_0 + \frac{a^n - 1}{a - 1} \cdot c \ (mod\ m)$$

Como estamos interesados en conocer el período del generador, vamos a buscar el menor valor de n de forma que $x_n = x_0$, y para ello sustituimos x_n por x_0 en la expresión general de x_n:

$$x_0 \equiv a^n x_0 + \frac{a^n - 1}{a - 1} \cdot c \ (mod\ m) \quad \Leftrightarrow \quad \frac{(a^n - 1) \cdot (x_0(a-1) + c)}{a - 1} \equiv 0 \ (mod\ m)$$

Por las condiciones del teorema, los números m y $x_0(a-1) + c$ son primos entre sí (recordemos que $a \neq 1$). En efecto, si p divide a m, por la condición *2.* se tiene que p divide a $x_0(a-1)$. Por reducción al absurdo, si m y $x_0(a-1) + c$ no fuesen coprimos, p dividiría a $x_0(a-1) + c$, y como teníamos que p divide a $x_0(a-1)$, esto implicaría que p también divide a c. Llegamos así

a una contradicción con la condición *1.* de que c y m son coprimos. De esta manera, se reduce el problema a encontrar el menor n tal que:

$$\frac{a^n - 1}{a - 1} \equiv 0 \ (mod\ m) \tag{2.2}$$

Queremos probar que este n es igual a m, pues de esta forma habríamos probado que un generador siguiendo las condiciones del teorema (y con $a \neq 1$) tiene período máximo.

En primer lugar, vamos a probarlo para $m = p^\alpha$, con α un número entero positivo y p un primo impar. Si $\alpha = 1$, por la segunda condición del teorema se deduce que $a = 1$, valor para el cual ya habíamos visto que el período es máximo. Consideraremos entonces los casos en los que $\alpha \geq 2$. Como $a \neq 1$ y, por la segunda condición del teorema, $a - 1$ tiene que ser múltiplo de todos los factores primos de m (siendo p en este caso su único factor primo), podemos escribir $a = 1 + kp^\beta$, donde $k \neq 0$ es *coprimo* con p (es decir, $mcd(k, p) = 1$), y β es un número entero positivo. Veamos que $n = m = p^\alpha$ verifica la condición (2.2):

$$\frac{a^n - 1}{a - 1} = \frac{(1 + kp^\beta)^{p^\alpha} - 1}{1 + kp^\beta - 1} \overset{(a)}{=} \frac{(\sum_{j=0}^{p^\alpha} \binom{p^\alpha}{j} 1^{(p^\alpha - j)} \cdot (kp^\beta)^j) - 1}{kp^\beta} \overset{(b)}{=}$$

$$= \frac{1 + p^\alpha \cdot kp^\beta + \frac{p^\alpha \cdot (p^\alpha - 1) \cdot (kp^\beta)^2}{2!} + \ \ldots \ + (kp^\beta)^{p^\alpha} - 1}{kp^\beta} = p^\alpha + \frac{p^\alpha \cdot (p^\alpha - 1)}{2} kp^\beta + \cdots + (kp^\beta)^{p^\alpha - 1}$$

donde en (a) hemos empleado el binomio de Newton y en (b) la expresión del coeficiente binomial $\binom{p^\alpha}{j} = \frac{p^\alpha (p^\alpha - 1) \ldots (p^\alpha - j + 1)}{j!}$.

Tenemos que probar que la última suma obtenida en los cálculos anteriores es divisible por $m = p^\alpha$. Para ello vamos a escribir la expresión general del término j-ésimo de la mencionada suma:

$$\frac{p^\alpha}{j} \left[\frac{(p^\alpha - 1) \ldots (p^\alpha - j + 1)}{(j - 1) \ldots 2 \cdot 1} \right] (kp^\beta)^{j-1}, \quad j \geq 2, \ j \in \mathbb{Z} \tag{2.3}$$

El término $\frac{p^\alpha}{j} \left[\frac{(p^\alpha - 1) \ldots (p^\alpha - j + 1)}{(j-1) \ldots 2 \cdot 1} \right]$ se corresponde con el coeficiente binomial $\binom{p^\alpha}{j}$, por lo que es un número entero, así como $(kp^\beta)^{j-1} = k^{j-1} \cdot p^{(j-1)\beta}$.

De aquí se deduce que j es el único término del denominador que puede "necesitar" alguno de los factores de p^α para dividir a todo el numerador de la expresión (2.3). El número de veces que puede aparecer el factor p en j es menor o igual que $j - 1$. En efecto, para $j \geq 2$:

$$\frac{j}{p} + \frac{j}{p^2} + \cdots = j \cdot \left(\left(\sum_{i=0}^{\infty} (1/p)^i \right) - 1 \right) = j \cdot \left(\frac{1}{1 - 1/p} - 1 \right) = \frac{j}{p - 1} \overset{(a)}{\leq} \frac{j}{2} \leq j - 1$$

donde en (a) hemos usado que p es un primo impar. El factor p aparece al menos $j - 1$ veces en $p^{(j-1)\beta}$, $\beta \geq 1$, por lo que no es necesario utilizar el factor p^α para que j divida al numerador. Hemos probado así que la suma de términos de la que habíamos partido es divisible por p^α, y de esta forma $n = p^\alpha$ cumple (2.2) bajo las condiciones del teorema, como queríamos ver.

Vamos a comprobar que un valor n cumple (2.2) si y solo si es un múltiplo del menor de todos los valores que lo cumplen.

Como sabemos que para $n = p^\alpha$ se verifica (2.2), solo consideraremos valores de n que sean potencias de p, y será suficiente probar que $n = p^{\alpha-1}$ no satisface la condición (2.2) para asegurar que no hay ningún valor más pequeño que $n = p^\alpha$ que la cumpla. Si sustituimos $n = p^{\alpha-1}$ y $a = 1 + kp^\beta$ en (2.2) y seguimos un procedimiento similar al realizado para $n = p^\alpha$, obtenemos la siguiente suma:

$$\frac{a^n - 1}{a - 1} = p^{\alpha-1} + \frac{p^{\alpha-1} \cdot (p^{\alpha-1} - 1)}{2} kp^\beta + \cdots + (kp^\beta)^{p^{\alpha-1}-1}$$

El primer término, $p^{\alpha-1}$, no es divisible por p^α. Por tanto, si comprobamos que todos los demás términos de la suma sí son múltiplos de p^α, habremos probado que $n = p^{\alpha-1}$ no verifica la condición (2.2), como queríamos ver. La idea que vamos a seguir es la misma que antes, empleando la expresión explícita del j-ésimo término de la suma. En este caso vamos a tener $p^{\alpha-1}$ en lugar de p^α en la expresión (2.3). Para tener de nuevo p^α necesitamos conseguir otro factor p, y lo obtenemos del término $p^{(j-1)\beta}$. Podemos hacer esto último debido al hecho de que p es un primo impar y, de esta forma, el número de veces que puede aparecer el factor p en j es menor o igual que $j - 2$, por lo que j no necesita ningún factor de p^α para dividir al numerador de la expresión, como queríamos ver.

Hasta aquí hemos demostrado el teorema para $m = p^\alpha$, con p un número primo impar. Veamos el caso en el que $m = 2^\alpha$. Si $\alpha = 1$, se tiene que el período es máximo. Para $\alpha \geq 2$, y siguiendo las condiciones del teorema, llegamos a que $a = 1 + k \cdot 2^\beta$, con $k \neq 0$ coprimo con p y $\beta \geq 2$ (para cumplir, en particular, la tercera condición). Esta última restricción la vamos a necesitar para seguir un razonamiento análogo al caso anterior y afirmar que el factor p que necesitamos para completar p^α lo podemos obtener de $p^{(j-1)\beta}$.

Ya tenemos probado el teorema siendo m la potencia de un primo. Consideraremos finalmente el caso general, es decir, $m = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_s^{\alpha_s}$. De nuevo por las condiciones del teorema, se deduce que $a = 1 + kp_1^{\beta_1} p_2^{\beta_2} \dots p_s^{\beta_s}$ con p_i un primo, α_i un entero positivo, $k \neq 0$ coprimo con m, $\beta_i \geq 1$ o, si $p_i = 2$ y $\alpha \geq 2$, $\beta_i \geq 2$. El teorema queda entonces probado siguiendo un razonamiento análogo al empleado en los casos anteriores. □

Ejemplo 2.3 (**Generador congruencial con período máximo**)**.** Haciendo uso de la función de R que definimos antes y teniendo en cuenta las condiciones enunciadas en el teorema, vamos a poner un ejemplo de secuencia obtenida a partir de un generador congruencial con semilla $x_0 = 7$ y parámetros $m = 24$, $a = 13$ y $c = 17$. Como cabía esperar, se observa que tiene período máximo igual a 24.

```
gen_congruencial(30,7,24,13,17)
```

```
##  [1] 0.29166667 0.50000000 0.20833333 0.41666667 0.12500000 0.33333333
##  [7] 0.04166667 0.25000000 0.95833333 0.16666667 0.87500000 0.08333333
## [13] 0.79166667 0.00000000 0.70833333 0.91666667 0.62500000 0.83333333
## [19] 0.54166667 0.75000000 0.45833333 0.66666667 0.37500000 0.58333333
## [25] 0.29166667 0.50000000 0.20833333 0.41666667 0.12500000 0.33333333
```

Cuando se trabaja con ordenadores *binarios* (como los que usamos en nuestro día a día), es habitual elegir m de manera que este sea "el mayor entero representable en el ordenador". Se suele considerar m de la forma $m = 2^\beta$ o $m = 2^\beta - 1$, donde β depende del tamaño de la palabra del ordenador (es decir, del número máximo de bits que el ordenador puede gestionar "simultáneamente"). En los ordenadores de 32 bits es común la elección de los parámetros $m = 2^{31}$, $a = 314159269$ y $c = 453805245$, pues da lugar a un generador de período máximo. En el lenguaje de programación C podemos encontrar un generador congruencial con período máximo $m = 2^{64}$ y $a = 6364136223846793005$ y $c = 1$, que puede ser utilizado en ordenadores de 64 bits.

Una de las ventajas de considerar $m = 2^\beta$ es que las condiciones del teorema de Hull y Dobell se simplifican mucho. Como m es una potencia de 2, este va a ser su único factor primo, así que c debe ser impar (para cumplir la primera condición). Para valores de $\beta \geq 2$ tenemos que m va a ser múltiplo de 4, por lo que $a - 1$ también debe de serlo. Siendo 2 el único factor primo de m, debe cumplirse que $a - 1$ sea par, pero ya lo tenemos garantizado al exigir que sea múltiplo de 4. A modo de resumen, para que un generador congruencial con $m = 2^\beta$, $\beta \geq 2$, tenga período máximo, son condiciones suficientes y necesarias que c sea impar y $a - 1$ sea múltiplo de 4. Veamos otro ejemplo de secuencia obtenida por un generador congruencial de período máximo, considerando esta vez $m = 2^\beta$, $\beta \geq 2$. Sean entonces $x_0 = 9$, $m = 2^4 = 16$, $a = 9$ y $c = 5$, por lo que el período máximo es 16:

```
gen_congruencial(20,9,16,9,5)
```

```
##  [1] 0.5625 0.3750 0.6875 0.5000 0.8125 0.6250 0.9375 0.7500 0.0625 0.8750
## [11] 0.1875 0.0000 0.3125 0.1250 0.4375 0.2500 0.5625 0.3750 0.6875 0.5000
```

2.2. Números pseudoaleatorios en R

El software estadístico R ofrece una gran variedad de generadores de números pseudoaleatorios, así como diversas herramientas que podemos utilizar para trabajar con ellos. El comando

set.seed() nos permite fijar la semilla (que debe ser un número entero) con la que se inicializa el generador y podemos establecer este último a través de la función *RNGkind*(). Por defecto, *R* utiliza el generador de *Mersenne-Twister*, desarrollado por M. Matsumoto y T. Nishimura en 1997. Este generador tiene una longitud de ciclo igual a $2^{19937} - 1$, que se corresponde con un número primo de Mersenne, de ahí su nombre.

Al inicio del capítulo hablamos de cómo se pueden obtener números aleatorios a partir de procesos físicos. En *R* podemos encontrar el paquete *random*, que nos permite acceder a algunos de estos números aleatorios almacenados en el sitio web *random.org*.

Un tipo de funciones que resultan de mucha utilidad son las funciones que *R* usa para generar valores de algunas variables aleatorias con distribuciones "notables" (como pueden ser $runif$, $rnorm, rbinom, \dots$). De esta forma, si queremos obtener valores, por ejemplo, de una distribución geométrica, usaríamos la función $rgeom$ y tendríamos que indicar n el número de valores a generar y la probabilidad de éxito p de la distribución, es decir, *rgeom(n,p)*. Otro ejemplo podemos verlo al inicio de este capítulo, donde empleamos la función *runif(12)* para generar 12 valores de una distribución $U(0, 1)$.

2.3. Contrastes de hipótesis

Como ya hemos comentado, a la hora de utilizar un generador de números pseudoaleatorios debemos asegurarnos de la calidad del mismo, pues de esto depende a menudo la validez de los resultados obtenidos mediante simulación. Podemos evaluar la calidad de estos generadores de dos formas. La primera consiste en estudiar las propiedades teóricas del generador, como por ejemplo su longitud de ciclo. Las comprobaciones de este estilo se conocen como test *teóricos*. Solo es posible aplicar estos test en el caso de generadores con una estructura sencilla, como pueden ser los generadores congruenciales lineales que hemos estudiado en este capítulo.

Una segunda forma de evaluar la calidad de un generador de números pseudoaleatorios consiste en aplicar una serie de contrastes de hipótesis a la secuencia obtenida a partir de él, con el objetivo de comprobar si dicha secuencia respeta las hipótesis de aleatoriedad e independencia y ajuste a una $U(0, 1)$. Este tipo de comprobaciones o test se denominan *empíricos*, y podemos encontrar una gran variedad de contrastes de hipótesis diferentes, sirvan como ejemplo los siguientes: el *test de Kolmogorov-Smirnov*, que nos permite comprobar el ajuste a una $U(0, 1)$, los *contrastes de salto*, con el objetivo de contrastar la aleatoriedad, y finalmente el *test de Ljung-Box* para contrastar la independencia.

Podemos encontrar ejemplos de generadores de números pseudoaleatorios, como los generadores congruenciales, que, a pesar de cumplir diferentes test (tanto teóricos como empíricos) no pasan

algunos de los nuevos contrastes diseñados específicamente para los generadores de números pseudoaleatorios. Por esta razón, en ocasiones se emplean un gran número de pruebas de distinto tipo y específicas para los números pseudoaleatorios (*baterías de contrastes*), con el objetivo de obtener una mayor seguridad sobre las buenas propiedades de un determinado generador[2]. Como ejemplo de batería de contrastes podemos mencionar los *Dieharder tests*, que fueron desarrollados por Marsaglia en 1995. Las baterías de contrastes se utilizan con frecuencia en campos como la criptografía. En general, en simulación no es necesario que los generadores utilizados pasen todos los test de una batería de contrastes.

A continuación presentaremos los tres métodos empíricos que hemos nombrado en esta sección. Vamos a considerar una secuencia de números pseudoaleatorios obtenidos a partir del generador que queremos estudiar como una muestra aleatoria simple de una variable aleatoria que sigue una distribución F.

2.3.1. Test de Kolmogorov-Smirnov

El contraste de Kolmogorov-Smirnov se basa en comparar la función de distribución empírica de la muestra, F_n, con la función de distribución teórica, F_0, que se quiere contrastar, en este caso la de una distribución $U(0,1)$. De esta manera, tendremos como hipótesis nula:

$$H_0 : \ F = F_0$$

y como hipótesis alternativa:

$$H_a : \ F \neq F_0.$$

La comparación entre estas dos funciones vendrá dada por la siguiente expresión, que se corresponde con el estadístico del contraste:

$$D_n = \sup_{x \in \mathbb{R}} |F_n(x) - F_0(x)|$$

y cuya distribución es conocida bajo la hipótesis nula. Para valores grandes de este estadístico podemos decir que hay pruebas de que los valores observados en la muestra difieren en gran medida de los valores esperados si estos procediesen de una distribución $U(0,1)$, por lo que esta no sería la verdadera distribución de los valores que se obtienen a partir del generador que estamos estudiando.

Para llevar a cabo el contraste necesitamos calcular el nivel crítico o p-valor, es decir, la probabilidad de que el estadístico que estamos empleando (que refleja la discrepancia entre lo observado

[2]Se puede encontrar más información sobre este tema en la siguiente referencia: Marsaglia, G. y Tsang, W. W. (2002). Some Difficult-to-pass Tests of Randomness. *Journal of Statistical Software*, 7(3), 1-9. https://doi.org/10.18637/jss.v007.i03

y lo esperado) tome valores "tan alejados o más" que el obtenido para la muestra dada. Si esta discrepancia es grande rechazaremos H_0, y esto se corresponde con valores "muy pequeños" del nivel crítico. Para determinar qué valores del nivel crítico consideramos como pequeños, tomaremos como referencia los niveles de significación utilizados habitualmente, como pueden ser $\alpha = 0'05$ o $\alpha = 0'01$. Rechazaremos H_0 en favor de H_a cuando el nivel crítico obtenido sea menor que el α considerado (recordemos que α representa la probabilidad de rechazar la hipótesis nula siendo cierta).

Ejemplo 2.4 (**Test de Kolmogorov-Smirnov**)**.** Vamos a considerar una secuencia de 30 números pseudoaleatorios obtenidos a partir de la función *runif* de *R*. Haremos uso también de la función *ks.test* de *R*, que tiene implementado el test de Kolmogorov-Smirnov. Queremos contrastar si los datos provienen de una distribución $U(0,1)$, y para ello vamos a utilizar el comando *punif*, que nos devuelve el valor de la función de distribución uniforme en cada uno de los datos de la muestra.

```
set.seed(9876)
x=round(runif(30),4); x  # Redondeamos los valores a 4 cifras decimales

##  [1] 0.8467 0.3679 0.1240 0.5688 0.4245 0.2776 0.4630 0.6032 0.5075 0.1822
## [11] 0.1253 0.2417 0.7243 0.2741 0.0347 0.4605 0.7386 0.4680 0.4902 0.5222
## [21] 0.9824 0.8780 0.0695 0.6741 0.4571 0.1024 0.1161 0.8906 0.3616 0.4356

ks.test(x, punif)

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  x
## D = 0.1778, p-value = 0.266
## alternative hypothesis: two-sided
```

En la salida de *R* podemos observar tanto el valor del estadístico $D = 0'1778$ como el nivel crítico obtenido, en este caso 0'266. Este valor es mayor que cualquiera de los niveles de significación habituales, por lo que podemos aceptar la hipótesis nula, es decir, la secuencia de números pseudoaleatorios considerada se ajusta a una distribución $U(0,1)$.

2.3.2. Contrastes de salto

Emplearemos este tipo de contrastes para determinar si podemos aceptar la hipótesis de aleatoriedad de los números pseudoaleatorios obtenidos a partir de un generador, por lo que la hipótesis nula considerada será:

$$H_0 : \textit{ los valores de la muestra son aleatorios}$$

y la hipótesis alternativa:

$$H_a : \textit{ los valores de la muestra no son aleatorios}$$

Fijados dos números α y β de forma que $0 \leq \alpha < \beta \leq 1$, comprobamos si cada valor generado x_i cumple $\alpha \leq x_i \leq \beta$. Cuando esto ocurre, se anota un 1 y, en caso contrario, un 0, obteniendo una lista de ceros y unos. Por ejemplo, si elegimos $\alpha = 0$ y $\beta = 1/2$, para cada número de la muestra tenemos que ver si se encuentra en $[0, 1/2]$ (y anotaremos un 1), o en $(1/2, 1]$ (anotando entonces un 0).

Denotamos por T_1, T_2, ... las posiciones, en nuestra secuencia de números pseudoaleatorios, de los valores generados que caen dentro del intervalo (α, β) (es decir, las posiciones en la que anotamos un 1). Las variables Z_i definidas como $Z_i = T_i - T_{i-1} - 1$, $i = 1, 2, \ldots$ con $T_0 = 0$, representan la longitud de ceros que aparecen entre dos unos consecutivos.

La probabilidad de obtener un 1 (un éxito) en cada comprobación es $p = \beta - \alpha$, mientras que la probabilidad de que aparezcan k ceros hasta la aparición del siguiente 1 es $p_z = P\{Z = k\} = p(1-p)^k$, $k = 0, 1, 2, \ldots$ Se corresponde, por tanto, con la función de masa de una distribución geométrica. De esta forma, bajo la hipótesis nula las variables Z_i son independientes e idénticamente distribuidas con distribución $Geom(p)$. Atendiendo a los posibles valores de k, se obtienen las clases $Z = 0$, $Z = 1$, ..., $Z = r - 1$, $Z \geq r$, con probabilidades $p(1-p)^k$, $k = 0, \ldots, r-1$ para las r primeras clases y $(1-p)^r$ para la última (con $r < n$, donde n hace referencia al tamaño muestral).

Si consideramos la muestra de números pseudoaleatorios de tamaño n para la que queremos contrastar la aleatoriedad y definimos las clases correspondientes, podemos obtener las frecuencias observadas, es decir, el número de apariciones de k ceros consecutivos en la muestra hasta obtener un 1. Nuestro objetivo será contrastar las frecuencias observadas en cada clase (o_k, $k = 0, 1, \ldots, r$) con las frecuencias esperadas en cada una de ellas (e_k, $k = 0, 1, \ldots, r$). Las frecuencias esperadas se pueden calcular de la siguiente forma: $e_k = h \cdot p_k$, siendo h el número de "huecos" (entendiendo por hueco la longitud entre dos unos consecutivos) en la secuencia de n números pseudoaleatorios considerada. Vamos a emplear el siguiente estadístico:

$$D = \sum_{k=0}^{r} \frac{(o_k - e_k)^2}{e_k}$$

que, bajo la hipótesis nula, sigue aproximadamente una distribución χ^2 con tantos grados de libertad como el número de clases menos uno, es decir, con r grados de libertad (pues hemos definido $r+1$ clases)[3]. La aproximación será mejor cuanto mayor sea el valor de n, aunque en general es suficiente elegir los valores r y n de forma que se cumpla $e_k \geq 5$, $\forall k = 0, \ldots, r$.

Los valores grandes del estadístico D nos indican que las frecuencias observadas en cada clase difieren en gran medida de las frecuencias esperadas, y en ese caso proporcionarían pruebas de que los valores de la muestra no son aleatorios. Al igual que hicimos en el test de Kolmogorov-Smirnov, para realizar el contraste calculamos el nivel crítico, y rechazaremos H_0 cuando el valor obtenido sea menor que el α fijado para el contraste.

Ejemplo 2.5 (**Contraste de rachas bajo la mediana**)**.** Veamos un ejemplo de aplicación de los contrastes de salto, utilizando una secuencia de 60 valores generada en *R* con *runif*. Vamos a emplear, en particular, $\alpha = 0$, $\beta = 1/2$, que se conoce como *contraste de rachas bajo la mediana*, y así $p = 1/2 - 0 = 1/2$. El primer paso es obtener la secuencia de ceros y unos asociada a la muestra. Lo haremos en *R*. El comando *length* permite calcular la longitud de un vector dado.

```
set.seed(345); sec=numeric()
x=round(runif(60),4); x

##  [1] 0.2163 0.2748 0.3899 0.6557 0.4359 0.8035 0.3857 0.8333 0.4731 0.0934
## [11] 0.2632 0.9469 0.1768 0.6244 0.9564 0.0882 0.9510 0.2466 0.9649 0.8682
## [21] 0.9690 0.9675 0.0808 0.2099 0.1978 0.7908 0.6249 0.2358 0.8169 0.5739
## [31] 0.0803 0.3912 0.7151 0.9630 0.8522 0.5776 0.1567 0.7728 0.6186 0.3844
## [41] 0.6700 0.9777 0.7963 0.1446 0.0049 0.5746 0.1205 0.4297 0.5393 0.5552
## [51] 0.8097 0.4349 0.2593 0.5704 0.1495 0.9681 0.2812 0.1305 0.2796 0.0226

for(i in 1:length(x)){
  if(x[i]<=1/2) sec=c(sec,1) else sec=c(sec,0)
}
sec

##  [1] 1 1 1 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 1 0
## [39] 0 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 1
```

[3] La distribución χ^2 es un caso particular de distribución gamma. En efecto, la distribución $\Gamma(n/2, 1/2)$, con $n \geq 1$, recibe el nombre de distribución χ^2 con n grados de libertad.

En vista de la secuencia obtenida establecemos las clases $Z = 0$, $Z = 1$ y $Z \geq 2$, por lo que $r = 2$ y $n = 60$. El siguiente paso es calcular el número de huecos que hay en esta secuencia. Vamos a implementar en R un algoritmo que nos permite hallar tanto el número de huecos como la longitud de cada uno de ellos. Esto último nos será de utilidad a la hora de calcular las frecuencias observadas de cada clase. El comando *which* devuelve las posiciones de un vector dado en las que se cumple la condición establecida.

```
r=2; n=60
T=c(0, which(sec==1)); t=length(T)
Z=numeric()
for(i in 1:(t-1)){
  Z[i]=T[i+1]-T[i]-1
}
h=length(Z); h; Z

## [1] 30
##  [1] 0 0 0 1 1 1 0 0 1 2 1 4 0 0 2 2 0 4 2 3 0 1 0 3 0 1 1 0 0 0
```

En la salida de R podemos ver tanto el número de huecos $h = 30$ como la longitud de cada uno de ellos. Calculamos entonces las frecuencias esperadas y las observadas para cada una de las clases definidas previamente.

```
p=1/2; h=30; r=2
# Frecuencias esperadas
e=numeric(r+1)
for(i in 1:r){
  e[i]=h*p*(1-p)**(i-1)
}
e[r+1]=h*(1-p)**r
e

## [1] 15.0  7.5  7.5

# Frecuencias observadas
o=numeric(r+1)
for(i in 1:r){
  o[i]=length(which(Z==i-1))
}
```

```
o[r+1]=length(which(Z>=r))
o


## [1] 14  8  8
```

Podemos observar en la fila superior las frecuencias esperadas o_k para cada clase y en la fila inferior las frecuencias observadas. Finalmente, calculamos el estadístico y el p-valor del contraste. Recordemos que el p-valor refleja la probabilidad de que el estadístico de contraste tome valores que difieran en gran medida del obtenido para la muestra considerada. Por tanto, la probabilidad que debemos calcular es $P\{\chi^2_r > D_0\}$, donde D_0 denota el valor del estadístico para la muestra de números pseudoaleatorios que hemos considerado. Emplearemos el comando *pchisq* de *R* para calcular dicha probabilidad.

```
D=0; r=2
for(k in 1:length(o)){
  D=D+((o[k]-e[k])**2)/e[k]
}
D


## [1] 0.1333333


pvalor=1-pchisq(D,r); pvalor


## [1] 0.935507
```

En la salida de *R* podemos observar tanto el valor del estadístico $D_0 \approx 0'1333$ como el p-valor obtenido, 0'935507. Este es un valor mucho mayor que los niveles de significación habituales, por lo que podemos aceptar la hipótesis de aleatoriedad de la secuencia de números pseudoaleatorios considerada.

2.3.3. Test de Ljung-Box

El método que emplearemos para contrastar la hipótesis de independencia de la secuencia de números pseudoaleatorios es el test de Ljung-Box.

Diremos que una serie ordenada en el espacio o el tiempo (como es, en nuestro caso, la secuencia de números pseudoaleatorios) presenta *dependencia secuencial* cuando cada uno de los valores

puede ser previsible a partir del valor anterior (o de los anteriores). Esto quiere decir que existen correlaciones altas entre los valores de la secuencia. Para el test de Ljung-Box vamos a necesitar las autocorrelaciones muestrales de orden k, que se definen como sigue:

$$r(k) = \frac{\sum_{i=k+1}^{n}(x_i - \bar{x})(x_{i-k} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

De esta forma, si denotamos por ρ_k las autocorrelaciones teóricas y por m el orden máximo de las autocorrelaciones consideradas, establecemos la hipótesis nula:

$$H_0 : \ \rho_1 = \cdots = \rho_m = 0$$

o lo que es lo mismo, los valores de la secuencia son independientes, y la hipótesis alternativa:

$$H_a : \ \rho_k \neq 0 \ \text{ para algún } \ k = 1, \ldots, m$$

es decir, los valores de la secuencia no son independientes. Existen diversos criterios mediante los que se puede elegir un valor m adecuado para el contraste. Vamos a considerar, por ejemplo, $m \approx 10 \cdot \log_{10} n$, siendo n el tamaño muestral, pues de esta forma conseguimos que la estimación $r(m)$ de ρ_m sea buena.

El estadístico del contraste viene dado por:

$$Q = n(n+2)\sum_{k=1}^{m}\frac{r(k)^2}{n-k},$$

El estadístico Q tendrá una distribución aproximada de una χ^2 con $m-1$ grados de libertad bajo la hipótesis nula. Para obtener una buena aproximación debemos considerar al menos $n \geq 25$, y esta será mejor cuanto mayor sea el valor de n. Rechazaremos H_0 cuando el nivel crítico obtenido en el contraste sea menor que el nivel de significación α fijado.

Ejemplo 2.6 (**Test de Ljung-Box**)**.** Vamos a considerar una secuencia de 50 números pseudoaleatorios generada en *R* con *runif* y autocorrelaciones muestrales de hasta orden $m = 10 \cdot \log_{10} 50 \approx 17$. Vamos a usar la función *Box.test* de *R*, pues tiene implementado el test de Ljung-Box. Necesitamos calcular previamente los *residuos* de cada valor de la muestra, es decir, las diferencias $x_i - \bar{x}$ (donde $\bar{x}$ denota la media muestral), necesarias para el cálculo del estadístico. También debemos establecer el orden máximo de las autocorrelaciones que vamos a considerar, y podemos indicarlo con el argumento *lag*= 17 en la función *Box.test*. Veámoslo:

```
set.seed(987); x=round(runif(50),4); x
```

```
##  [1] 0.4773 0.9906 0.6064 0.5766 0.8091 0.2323 0.4245 0.3490 0.8645 0.7101
## [11] 0.1857 0.3811 0.1698 0.3115 0.1960 0.6389 0.1328 0.9666 0.0987 0.7095
```

```
## [21] 0.7757 0.5037 0.1905 0.7513 0.9607 0.0104 0.4961 0.4999 0.3269 0.1781
## [31] 0.4527 0.7118 0.3244 0.6377 0.1184 0.2221 0.3245 0.5335 0.8449 0.8879
## [41] 0.9492 0.6225 0.1916 0.2409 0.0603 0.2126 0.7571 0.0626 0.5284 0.4742
```

```
residuos=numeric()
for(i in 1:length(x)){
  e=x[i]-mean(x); residuos=c(residuos,e)
}
Box.test(residuos,lag=17,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuos
## X-squared = 15.2, df = 17, p-value = 0.5811
```

Como podemos ver en la salida de R, el valor del estadístico del contraste es $\chi^2 = 15'2$ y el nivel crítico obtenido es 0'5811 (mayor que los niveles de significación habituales). Por lo tanto, no hay evidencias para rechazar la independencia de la muestra considerada.

Observación 2.7. En los contrastes anteriores hemos considerado una única muestra de números pseudoaleatorios obtenida a partir de un generador, por lo que estaríamos aceptando o rechazando las hipótesis basándonos en una única prueba. Podríamos realizar un gran número de pruebas, de forma que en cada una de ellas comprobemos si el valor del estadístico se encuentra en la región crítica o no para un α dado, y finalmente observar si la proporción de rechazos de los contrastes de hipótesis considerados se aproxima o no al nivel de significación α fijado (habitualmente, $\alpha = 0'01$ o $\alpha = 0'05$).

2.4. Aplicaciones

En esta sección vamos a exponer tres ejemplos sencillos de simulación donde usaremos los números pseudoaleatorios.

2.4.1. Cálculo de integrales

Si queremos calcular la integral de una cierta función $g(x)$ en el intervalo $[0, 1]$, es decir, $\int_0^1 g(x)\,\mathrm{d}x$, y no es posible obtener una primitiva elemental de g, podemos calcularla mediante simulación.

Sea U una variable con distribución $U(0,1)$, por lo que su función de densidad será $f(x) = 1$, $\forall x \in [0,1)$, y denotemos $\theta = \int_0^1 g(x)\ \mathrm{d}x$. Esta integral se va a poder expresar como $\theta = E[g(U)]$. En efecto,

$$\theta = E[g(U)] = \int_{-\infty}^{\infty} g(x) \cdot f(x)\mathrm{d}x = \int_0^1 g(x)\mathrm{d}x$$

Consideremos ahora n variables aleatorias independientes e idénticamente distribuidas, U_1, U_2, $\ldots$, U_n, con distribución $U(0,1)$. De esta forma, las variables aleatorias transformadas $g(U_1), \ldots, g(U_n)$ son también independientes e idénticamente distribuidas con media θ. Se deduce ahora a partir de la ley fuerte de los grandes números lo siguiente:

$$\lim_{n\to\infty} \frac{g(U_1) + \cdots + g(U_n)}{n} = \theta = \int_0^1 g(x)\ \mathrm{d}x$$

por lo que para aproximar el valor de θ nos llegará con considerar el valor $\frac{1}{n} \cdot \sum_{i=1}^{n} g(U_i)$ para un n grande.

Ejemplo 2.8 (**Cálculo de una integral mediante simulación**). La función $g(x) = e^{-x^2}$ no tiene primitiva elemental, pero sabemos que es integrable por ser continua en todo $\mathbb{R}$. Calculemos entonces la integral $\theta = \int_0^1 e^{-x^2}\ \mathrm{d}x$ empleando el método que hemos visto. Lo haremos en *R*, considerando una muestra de $n = 100\ 000$ números pseudoaleatorios obtenidos con *runif*:

```
n=100000
set.seed(321); u=runif(n)
g<-function(x) exp(-x**2)
theta=sum(g(u))/n; theta


## [1] 0.7473219
```

Como podemos observar en la salida de *R*, obtenemos que $\theta \approx 0'7473219$.

2.4.2. Cálculo del área bajo una curva

Vamos a considerar para este ejemplo una curva con una función primitiva sencilla, como puede ser $y = -4x^2 + 4x$. Si calculamos la integral de esta curva en el intervalo $[0,1]$, es decir, $\int_0^1 y\ \mathrm{d}x$, obtenemos el siguiente valor, que se corresponde con el área comprendida entre esta curva y el eje de abscisas:

$$\frac{-4 \cdot 1^3}{3} + \frac{4 \cdot 1^2}{2} = \frac{2}{3} \approx 0'667$$

Nuestro objetivo será aproximar dicha área mediante simulación. Notemos que la porción de la curva que nos interesa está inscrita en el cuadrado con vértices $(0,0)$, $(0,1)$, $(1,1)$ y $(1,0)$. Si

consideramos dos variables aleatorias independientes X e Y con distribución $U(0,1)$, entonces su función de densidad conjunta será:

$$f(x,y) = f(x)f(y) = \frac{1}{1-0} \cdot \frac{1}{1-0} = 1$$

por lo que el vector aleatorio bidimensional (X,Y) tendrá distribución uniforme en el cuadrado $[0,1] \times [0,1]$.

Veamos cómo podemos aprovechar esto último para aproximar la integral $A = \int_0^1 y \, \mathrm{d}x$. Si generamos una gran cantidad de valores del vector (X,Y), podemos comprobar para cada uno de ellos si se encuentra por debajo de la curva o por encima. Para ello, será conveniente definir una variable aleatoria de Bernoulli, a la que llamaremos I, de la siguiente manera:

$$I = \begin{cases} 1 & \text{si } Y + 4X^2 - 4X \leq 0 \\ 0 & \text{en otro caso} \end{cases}$$

De esta forma,

$$E[I] = P\{Y + 4X^2 - 4X \leq 0\} = \frac{\text{Área bajo la curva}}{\text{Área del cuadrado}} = \frac{A}{1} = A$$

La ley fuerte de los grandes números nos garantiza que la fracción obtenida al considerar los valores de (X,Y) que caen debajo de la curva, frente a los valores totales generados, se va a aproximar a la media de la variable, es decir, a la integral y consecuentemente al área que queremos calcular. Hacemos los cálculos en *R*:

```
n=80000
set.seed(123); x=runif(n)
set.seed(456); y=runif(n)
c=0
for(i in 1:n){
  if(y[i]+4*x[i]**2-4*x[i] <= 0) c=c+1
}
A=c/n; A


## [1] 0.6672375
```

Habíamos calculado que el valor del área bajo la curva considerada es exactamente $\frac{2}{3}$. A partir de una muestra de $n = 80\,000$ valores aleatorios de (X,Y), hemos obtenido un valor para esta área de 0'6672375 $\approx$ 0'667, por lo que podemos decir que es una buena aproximación del valor real. En la Figura 2.1 podemos ver una representación gráfica del proceso que hemos seguido.

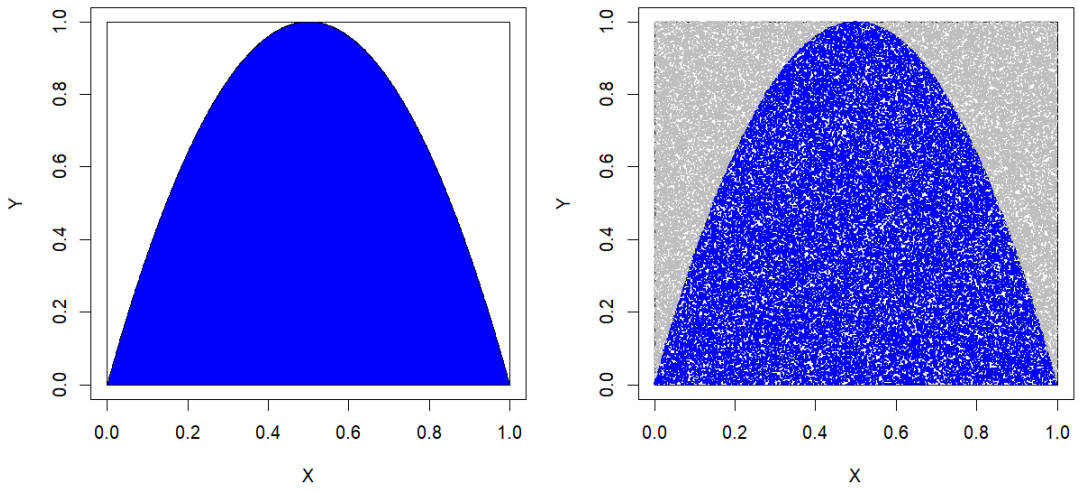

Figura 2.1: *A la izquierda, el área que queremos aproximar mediante simulación. A la derecha, la representación gráfica del proceso seguido con una muestra de 80 000 valores aleatorios de* (X, Y)*, donde están en azul los valores que caen sobre la curva o por debajo de ella y en gris los que caen por encima.*

2.4.3. Cálculo del área encerrada por una curva: aproximación de π.

Si consideramos un vector aleatorio bidimensional (X, Y) con distribución uniforme en el cuadrado de lado $l = 2$ y centro el origen de un sistema de coordenadas, sus posibles valores serán los puntos (x, y) con $x,\ y \in [-1, 1]$. Notemos que se puede inscribir un círculo de radio 1 en el cuadrado. Para calcular la probabilidad de que un valor cualquiera que tome (X, Y) se encuentre en el círculo, debemos obtener el valor de $P\{X^2 + Y^2 \leq 1\}$. Este se obtiene de la siguiente manera:

$$P\{X^2 + Y^2 \leq 1\} = \frac{\text{Área del círculo}}{\text{Área del cuadrado}} = \frac{\pi \cdot 1^2}{2 \cdot 2} = \frac{\pi}{4}$$

Vamos a considerar entonces dos variables aleatorias independientes X e Y, ambas con distribución $U(-1, 1)$. Su función de densidad conjunta sería:

$$f(x, y) = f(x)f(y) = \frac{1}{1 - (-1)} \cdot \frac{1}{1 - (-1)} = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

Así, el vector aleatorio bidimensional (X, Y) tiene distribución uniforme en el cuadrado $[-1, 1] \times [-1, 1]$. Notemos que si una variable aleatoria U sigue una distribución $U(0, 1)$, entonces la variable definida como $2U - 1$ seguirá una distribución $U(-1, 1)$. Sean U_1 y U_2 variables aleatorias con distribución $U(0, 1)$. Definimos entonces $X = 2U_1 - 1$ e $Y = 2U_2 - 1$, ambas con distribución $U(-1, 1)$. Si consideramos la variable aleatoria I definida como:

$$I = \begin{cases} 1 & \text{si } X^2 + Y^2 \leq 1 \\ 0 & \text{en otro caso} \end{cases}$$

seguirá una distribución de Bernoulli, con $E[I] = P\{X^2 + Y^2 \leq 1\} = \frac{\pi}{4}$. Al igual que en el ejemplo anterior, por la ley fuerte de los grandes números garantizamos que al generar una gran

cantidad de valores del vector (X, Y), podremos aproximar el valor de $\pi \approx 3'141592$ al considerar el número de puntos que caen dentro del círculo frente a los puntos totales generados. Veamos cómo podemos hacerlo con R.

```
n=80000
set.seed(246); u1=runif(n); x=2*u1-1
set.seed(468); u2=runif(n); y=2*u2-1
c=0
for(i in 1:n){
  if(x[i]**2 + y[i]**2 <= 1) c=c+1
}
pi_aprox=4*(c/n); pi_aprox

## [1] 3.1418
```

Para una muestra de $n = 80\,000$ valores aleatorios del vector (X, Y), hemos obtenido $\pi \approx 3'1418$. En la Figura 2.2 podemos ver una representación gráfica del proceso que hemos seguido.

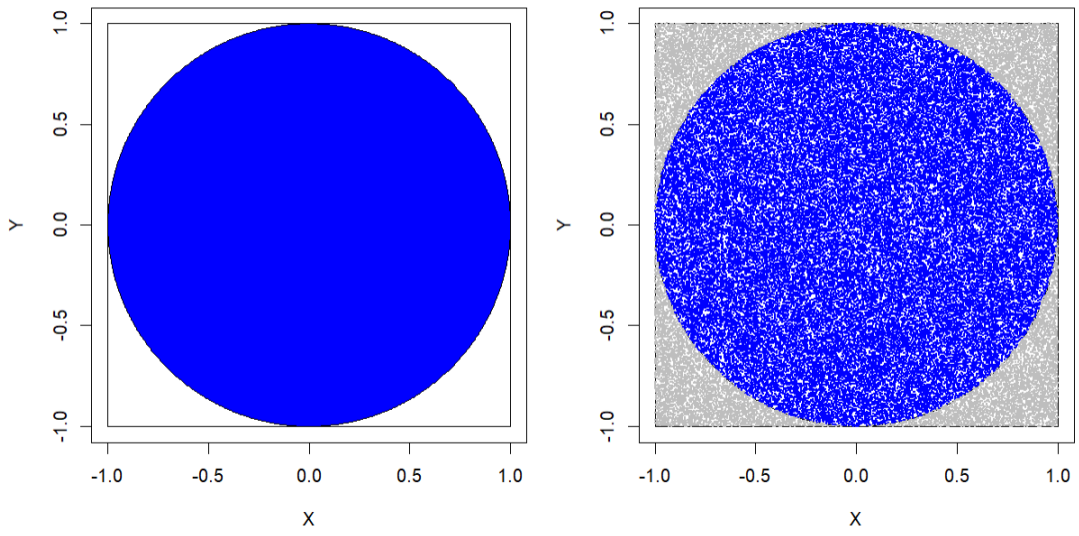

Figura 2.2: *A la izquierda, el área que queremos aproximar mediante simulación. A la derecha, la representación gráfica del proceso que hemos seguido en la aproximación de π, utilizando una muestra de 80 000 valores aleatorios de (X, Y), donde están en azul los puntos que caen sobre la curva o dentro de ella y en gris los que caen fuera.*

Si aumentamos el valor de n podríamos llegar a una aproximación mejor del valor real de π. Por ejemplo, vamos a considerar $n = 300\,000$ y repetimos el proceso.

```
n=300000
set.seed(20); u1=runif(n); x=2*u1-1
set.seed(22); u2=runif(n); y=2*u2-1
c=0
for(i in 1:n){
  if(x[i]**2 + y[i]**2 <= 1) c=c+1
}
pi_aprox=4*(c/n); pi_aprox

## [1] 3.14156
```

En este caso, hemos obtenido $\pi \approx 3'14156$. En efecto, esta aproximación es mejor que la obtenida en el ejemplo anterior.

En el siguiente capítulo presentaremos diferentes métodos que podemos utilizar para generar valores de variables aleatorias que siguen una determinada distribución conocida. Para llevar a cabo estos métodos, vamos a necesitar disponer de un buen generador de números pseudoaleatorios.

Capítulo 3

Simulación de variables aleatorias

En simulación, muchas veces será necesario trabajar con variables que siguen una distribución conocida. En el estudio de un modelo será imprescindible además que los valores que usemos correspondan a la distribución que queremos emplear, con el fin de garantizar la validez de las conclusiones obtenidas. En este capítulo vamos a ver distintos métodos mediante los que podemos generar valores de una determinada distribución (tanto discreta como continua).

3.1. Simulación de variables aleatorias continuas

En esta sección vamos a presentar dos de los métodos generales más empleados a la hora de simular variables aleatorias continuas: el *método de inversión* y el *método de aceptación-rechazo*.

3.1.1. Método de inversión

Este método se basa en el siguiente resultado:

Teorema 3.1 (**de inversión**). *Si tenemos una variable aleatoria X con función de distribución F, continua e invertible, entonces la variable aleatoria U, definida como una transformación de la variable original por su propia función de distribución, es decir, $U = F(X)$, sigue una distribución $U(0,1)$. De aquí se deduce entonces que la variable $F^{-1}(U)$ tiene la misma función de distribución que X, esto es, F.*

Demostración. Denotemos por G la función de distribución de U y consideremos $u \in (0,1)$. Se tiene en primer lugar:

$$G(u) \overset{(a)}{=} P\{U \leq u\} = P\{F(X) \leq u\} \overset{(b)}{=} P\{X \leq F^{-1}(u)\} \overset{(c)}{=} F(F^{-1}(u)) = u$$

En (a) y (c) aplicamos la definición de función de distribución, mientras que en (b) usamos que F es invertible. Se tiene además que $G(u) = 0$ si $u \leq 0$ y que $G(u) = 1$ si $u \geq 1$. De esta forma, G es la función de distribución de una distribución $U(0,1)$. Para finalizar veremos que, como F es invertible,

$$X = F^{-1}(F(X)) = F^{-1}(U),$$

con U siguiendo una distribución $U(0,1)$. □

A partir del anterior resultado, podemos establecer un algoritmo que nos permite simular cualquier variable aleatoria continua con función de distribución F, siempre que esta sea continua e invertible. Ilustramos a continuación el algoritmo del método de inversión:

`1. Generar` $U \backsim U(0,1)$
`2. Devolver` $X = F^{-1}(U)$

Podemos repetir estos pasos tantas veces como sea necesario para generar tantos valores de X como queramos. Notemos que para llevar a cabo el primer paso del algoritmo es necesario simular valores de una $U(0,1)$, es decir, números pseudoaleatorios como los que hemos visto en el capítulo anterior.

Ejemplo 3.2 (**Distribución exponencial mediante el método de inversión**)**.** Para dar un ejemplo de cómo se aplica este método, vamos a simular una cierta variable aleatoria X que sigue una distribución exponencial de parámetro $\lambda > 0$. En el capítulo 1 hemos visto la función de densidad y la función de distribución de este tipo de variables. Se puede comprobar que la función de distribución de X, sea $F(x)$, es continua e invertible en el intervalo $[0, \infty)$. Con el objetivo de encontrar una expresión explícita para $F^{-1}(u)$ y poder realizar el segundo paso del algoritmo, vamos a realizar los siguientes cálculos:

$$x = F^{-1}(u) \quad \Leftrightarrow \quad F(x) = u \quad \Leftrightarrow \quad 1 - e^{-\lambda x} = u \quad \Leftrightarrow \quad 1 - u = e^{-\lambda x} \quad \Leftrightarrow \quad x = -\frac{\ln(1-u)}{\lambda}$$

Una vez tenemos esta forma de expresar $F^{-1}(u)$, podemos llevar a cabo el algoritmo, pero antes vamos a ver cómo se puede simplificar dicha expresión para ahorrar cálculos. En efecto, si U es una variable aleatoria que sigue una distribución $U(0,1)$, entonces la variable $1 - U$ también seguirá dicha distribución. Por esta razón, podemos sustituir el valor $1-u$ por u en la expresión obtenida en los cálculos anteriores, sin que esto afecte a la validez del método. Si tenemos intención de ejecutar el algoritmo un gran número de veces, también puede ser útil definir $L = \frac{-1}{\lambda}$ al comienzo, que será un valor constante al depender únicamente del parámetro de la distribución. De esta manera, podemos escribir ahora $x = L \cdot \ln(u)$.

Veamos un pequeño ejemplo de aplicación de este algoritmo en R, donde vamos a simular 30 valores de una variable aleatoria X que sigue una distribución $Exp(2)$:

```
sim_exp<-function(lambda,n){
  L=-1/lambda
  u=runif(n)
  x=L*log(u); round(x,4)    # Redondeamos los valores a 4 cifras decimales
}
sim_exp(2,30)

##  [1] 0.1637 0.1070 0.1809 0.8219 0.0496 0.7447 0.9133 0.4982 0.1815 0.2504
## [11] 0.1650 0.4415 0.2215 0.0032 0.2030 0.1139 0.6656 0.3397 0.1131 1.1218
## [21] 0.2207 0.5552 0.5604 0.3412 0.1845 0.6511 0.1392 0.1113 0.5729 1.5831
```

Ejemplo 3.3 (**Distribución doble exponencial estándar mediante el método de inversión**). Recordemos que para una variable aleatoria X que sigue una distribución exponencial estándar se tiene $\lambda = 1$, por lo que su función de distribución vendrá dada por:

$$F(x) = \begin{cases} \frac{1}{2}e^{x} & \text{si } x < 0 \\ 1 - \frac{1}{2}e^{-x} & \text{si } x \geq 0. \end{cases}$$

Al igual que en el ejemplo anterior, realizamos ahora los cálculos necesarios para hallar $F^{-1}(u)$:

$$\frac{1}{2}e^{x} = u \quad \Leftrightarrow \quad e^{x} = 2u \quad \Leftrightarrow \quad x = \ln(2u), \quad x < 0 \quad \Rightarrow \quad u < 0{,}5$$

$$1 - \frac{1}{2}e^{-x} = u \quad \Leftrightarrow \quad e^{-x} = 2(1-u) \quad \Leftrightarrow \quad x = -\ln(2(1-u)), \quad x \geq 0 \quad \Rightarrow \quad u \geq 0{,}5$$

De esta manera, a la hora de aplicar este método tendremos en cuenta que si el número pseudo-aleatorio generado es $u < 0{,}5$, entonces $x = \ln(2u)$, y si $u \geq 0{,}5$ haremos $x = -\ln(2(1-u))$. A modo de ejemplo, implementamos ahora el algoritmo en R para generar 30 valores de X:

```
sim_doble_exp<-function(n){
  u<-runif(n)
  x=numeric()
  for (i in u){
    if (i<0.5){
      x=c(x, log(2*i))
    } else {
      x=c(x, -log(2*(1-i)))
    }
  }
  x
}
round(sim_doble_exp(30),4)    # Redondeamos los valores a 4 cifras decimales
```

```
##  [1] -0.1632 -1.6711  0.8861  1.4400  0.4835 -0.1490  1.1579 -1.4821  0.5238
## [10]  0.0532 -1.1082  0.5203  1.5291  1.1564  0.0351 -2.1826 -0.2290  0.3850
## [19]  0.3198  0.0008  0.4530 -3.3374 -0.2781 -0.6362  1.3153  0.2619 -0.5769
## [28] -0.0774  1.6935  1.3532
```

Aunque una de las ventajas más importantes del método de inversión reside en que se puede aplicar a una gran cantidad de variables aleatorias continuas, también presenta algunos inconvenientes. Uno de los más frecuentes surge cuando no se tiene una expresión explícita para la función de distribución (como en el caso de la distribución normal). Además, aún teniendo una expresión explícita, en ocasiones no es posible despejar el valor de x en la ecuación $F(x) = u$, o la expresión obtenida es demasiado complicada y ralentiza los cálculos. En este caso, una posible solución sería emplear métodos numéricos, pero hay que tener en cuenta la desventaja añadida de necesitar resolver una ecuación nueva para cada valor de la distribución que queremos simular.

3.1.2. Método de aceptación-rechazo

Con este método vamos a poder solucionar alguno de los inconvenientes del método de inversión. El método de aceptación-rechazo está pensado para simular variables aleatorias para las que sí se disponga de una expresión explícita de su función de densidad, pero se desconozca una expresión explícita de su función de distribución o resulte complicado resolver $F(x) = u$. Además, esta función de densidad tiene que estar acotada por otra función de densidad que sea fácilmente simulable.

Denotaremos por f a la función de densidad de la variable aleatoria continua que queremos simular, mientras que g será la *densidad auxiliar* del método, la cual tendrá que cumplir algunas condiciones. La idea consiste en generar valores de la variable con densidad g y comprobar para cada uno de ellos si se cumple una determinada *condición de aceptación*. En caso afirmativo, se acepta el valor generado como un posible valor de la variable con densidad f, mientras que se descarta el valor generado cuando no se cumple la condición de aceptación.

El método de aceptación-rechazo se basa en el siguiente resultado:

Teorema 3.4 (**de aceptación-rechazo**). *Consideremos f y g dos funciones de densidad de forma que los conjuntos en los que están definidas son compatibles (es decir, $g(x) > 0$ cuando $f(x) > 0$), y existe una constante $c > 0$ tal que $\frac{f(x)}{g(x)} \leq c$, $\forall\, x \in \mathbb{R}$. Sea X una variable aleatoria con función de densidad f. Sean Y y U dos variables aleatorias continuas e independientes, de forma que Y tiene función de densidad g y U sigue una distribución $U(0,1)$. Si $U \leq \frac{f(Y)}{c \cdot g(Y)}$, la variable aleatoria que resulta de considerar los valores de Y para los que se cumple dicha condición tiene la misma distribución que X.*

Demostración. Veamos que la función de distribución $P\{Y \leq x \ / \ U \leq \frac{f(Y)}{cg(Y)}\}$ coincide con la función de distribución de la variable aleatoria X, $P\{X \leq x\}$.

$$P\left\{Y \leq x \ / \ U \leq \frac{f(Y)}{cg(Y)}\right\} = \frac{P\{Y \leq x,\ U \leq \frac{f(Y)}{cg(Y)}\}}{P\{U \leq \frac{f(Y)}{cg(Y)}\}} =$$

$$= \frac{\int_{-\infty}^{x} \int_{0}^{f(y)/(cg(y))} \mathrm{d}u \ g(y) \ \mathrm{d}y}{\int_{-\infty}^{\infty} \int_{0}^{f(y)/(cg(y))} \mathrm{d}u \ g(y) \ \mathrm{d}y} = \frac{\int_{-\infty}^{x} [f(y)/(cg(y))] \ g(y) \ \mathrm{d}y}{\int_{-\infty}^{\infty} [f(y)/(cg(y))] \ g(y) \ \mathrm{d}y} = \frac{\int_{-\infty}^{x} f(y) \ \mathrm{d}y}{\int_{-\infty}^{\infty} f(y) \ \mathrm{d}y} = P\{X \leq x\}$$

□

A partir del teorema anterior se construye el algoritmo de aceptación-rechazo:

1. `Generar` Y `con función de densidad` g `y` $U \backsim U(0,1)$ `independientes.`
2. `Si se cumple` $U \leq \frac{f(Y)}{c \cdot g(Y)}$`, devolver` $X = Y$`.`

Eficiencia del algoritmo

Notemos que la constante c que empleamos en el algoritmo está sujeta a ciertas restricciones. Como $\frac{f(x)}{g(x)} \leq c \ \Leftrightarrow \ f(x) \leq c \cdot g(x)$, por la definición de función de densidad, se tiene:

$$1 = \int f(x) \ \mathrm{d}x \leq c \cdot \int g(x) \ \mathrm{d}x = c$$

De esta manera sabemos que $c \geq 1$. Si $c = 1$, entonces tendríamos $f = g$, y no necesitaríamos hacer nada más. Por lo tanto, nos centraremos en el caso $c > 1$.

En el paso 2 del algoritmo se hace la siguiente comprobación: $U \leq \frac{f(Y)}{c \cdot g(Y)}$. Vamos a definir la *eficiencia del método* como la probabilidad de aceptación de esta condición, es decir:

$$p = P\left\{u \leq \frac{f(y)}{c \cdot g(y)}\right\} = \int g(y) \ \int_0^1 I_{\{u \leq \frac{f(y)}{c \cdot g(y)}\}} \ \mathrm{d}u \ \mathrm{d}y = \int \frac{f(y)}{c} \ \mathrm{d}y = \frac{1}{c}$$

De esta manera, el número de repeticiones de los pasos 1 y 2 del algoritmo hasta que se cumple la condición de aceptación es una variable aleatoria, a la que denotaremos por N, que sigue una distribución geométrica de parámetro $p = \frac{1}{c}$. Así, el número medio de repeticiones del algoritmo hasta obtener un valor simulado de la variable X viene dado por $E(N) = \frac{1}{p} = c$. Podemos decir que cuánto más próximo a 1 sea el valor de p, más eficiente será el algoritmo.

Elección de c

Para obtener el mejor valor de c tenemos que encontrar el número real más pequeño de forma que $\frac{f(x)}{g(x)} \leq c$. De esta forma, el menor valor de c que cumple la condición es:

$$c_{opt} = \max_{g(x) \neq 0} \frac{f(x)}{g(x)}$$

donde c_{opt} denota el valor óptimo de c.

Elección de la densidad auxiliar g

Como ya hemos visto, para llevar a cabo el método de aceptación-rechazo necesitamos elegir una densidad auxiliar g de la que sea sencillo simular valores mediante otros métodos de simulación. En general, cuanto más se parezca la forma de la función de densidad g a la de f, más pequeño será el valor óptimo de c. Sin embargo, todavía no se sabe cómo elegir g de forma que dé lugar a un valor de c óptimo. Una posible solución al problema pasa por considerar una cierta familia paramétrica de densidades entre las que podamos encontrar una que se parezca lo suficiente a f. Sea $\{g_\theta / \ \theta \in \Theta\}$ una familia de densidades de este tipo. Para encontrar el valor óptimo de c debemos considerar primero, para cada $\theta \in \Theta$, el máximo en x del cociente $\frac{f(x)}{g_\theta(x)}$ y, posteriormente, quedarnos con el parámetro θ_0 que da lugar al menor cociente de todos ellos. Se tiene entonces:

$$c_{\theta_0} = \min_{\theta\in\Theta} \max_{x} \frac{f(x)}{g_\theta(x)}$$

Ejemplo 3.5 (**Distribución normal estándar a partir de la doble exponencial**)**.** En el apartado anterior hemos visto cómo podemos simular una distribución $Dexp(1)$ empleando el método de inversión. La distribución doble exponencial de parámetro $\lambda > 0$ también es fácilmente simulable usando este método. Como además su función de densidad tiene una forma similar a la función de densidad de una $N(0,1)$ y ambas tienen el mismo soporte (toda la recta real) podemos escogerla como densidad auxiliar en el método de aceptación-rechazo para simular la normal estándar. Veamos cuál sería el mejor valor de λ en lo que a eficiencia del algoritmo se refiere.

Denotamos por $g_\lambda(x)$ la función de densidad de una distribución $Dexp(\lambda)$. Buscamos el parámetro λ que nos devuelve el valor óptimo de c:

$$c_\lambda = \min_{\lambda>0} \max_{x\in\mathbb{R}} \frac{f(x)}{g_\lambda(x)} = \min_{\lambda>0} \max_{x\in\mathbb{R}} \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}}{\frac{\lambda}{2} e^{-\lambda|x|}} = \min_{\lambda>0} \frac{1}{\lambda}\sqrt{\frac{2}{\pi}} \max_{x\in\mathbb{R}} e^{\varphi_\lambda(x)} = \min_{\lambda>0} \frac{1}{\lambda}\sqrt{\frac{2}{\pi}} \cdot e^{\max\limits_{x\in\mathbb{R}} \varphi_\lambda(x)}$$

donde $\varphi_\lambda(x) = -\frac{x^2}{2} + \lambda|x|$. Como esta función es simétrica, continua en toda la recta real y diferenciable todas las veces que queramos, salvo en $x = 0$, solo necesitaremos buscar un máximo absoluto en $[0, \infty]$.

$$x > 0 \ \Rightarrow \ \varphi'(x) = -x + \lambda \ \ ; \ \ \varphi'(x) = 0 \ \Leftrightarrow \ x = \lambda \ \ ; \ \ \varphi''(\lambda) < 0$$

De esta forma, $\varphi(x)$ tiene un máximo relativo en $x = \lambda$, y como es una función simétrica, $x = -\lambda$ también será un máximo relativo. Se puede ver a través de los intervalos de crecimiento y decrecimiento de la función que ambos son máximos absolutos, y el valor máximo que alcanza la función será por tanto $\varphi(\pm\lambda) = \frac{\lambda^2}{2}$. En consecuencia:

$$c_\lambda = \min_{\lambda>0} \frac{1}{\lambda}\sqrt{\frac{2}{\pi}} e^{\frac{\lambda^2}{2}}$$

Para hallar ese mínimo, primero derivamos la expresión respecto a λ y la igualamos a 0. Haciendo las cuentas, llegamos a que $\lambda = 1$ (recordemos que $\lambda > 0$). Además, se puede ver que la segunda derivada en ese punto es positiva, por lo que $\lambda = 1$ es un punto mínimo de la expresión de partida. Esto quiere decir que, de entre todas las distribuciones doble exponencial con $\lambda > 0$, la de parámetro $\lambda = 1$ es la que genera un algoritmo de aceptación-rechazo más eficiente para simular valores de una normal estándar. En este caso, el valor óptimo de c es:

$$c_{opt} = \sqrt{\frac{2}{\pi}e^{\varphi(1)}} = \sqrt{\frac{2}{\pi}e^{1/2}} = \sqrt{\frac{2e}{\pi}} \approx 1'3155$$

El algoritmo, donde hemos incluido también el método de inversión para obtener valores de la función de distribución de la doble exponencial estándar, quedaría entonces como sigue:

1. `Generar` $U, V \backsim U(0,1)$.
2. `Si` $V < 0,5$ `hacer` $Y = \ln(2V)$, `en otro caso` $Y = -\ln(2(1-V))$.
3. `Si se cumple` $U \cdot e^{\frac{Y^2}{2}-|Y|+\frac{1}{2}} \leq 1$, `devolver` $X = Y$.

La condición que hay que comprobar en este algoritmo de aceptación-rechazo es:

$$U \leq \frac{f(Y)}{c \cdot g(Y)} \Leftrightarrow c \cdot U \cdot \frac{g(Y)}{f(Y)} \leq 1 \Leftrightarrow \sqrt{\frac{2e}{\pi}}U\sqrt{\frac{\pi}{2}}e^{\frac{Y^2}{2}-|Y|} = U \cdot e^{\frac{Y^2}{2}-|Y|+\frac{1}{2}} \leq 1$$

Como ya hemos visto, la probabilidad de aceptación será $p = \frac{1}{c} = \frac{1}{\sqrt{2e/\pi}} = 0'76017$, por lo que podemos decir que el algoritmo es "bastante" eficiente. Veamos cómo podemos implementarlo en R, donde n va a ser el número de repeticiones del algoritmo:

```
sim_normal<-function(n){
  u=runif(n); v=runif(n)
  t=numeric()
  for(i in 1:n){
    if (v[i] < 0.5){
      t=c(t,log(2*v[i]))
    } else {
      t=c(t,-log(2*(1-v[i])))
    }
  }
  x=numeric()
  for(i in 1:n){
    if (u[i]*exp(t[i]^2/2-abs(t[i])+1/2) <= 1) x=c(x,t[i])
  }
  round(x,4)
}
```

Empleando esta función que hemos definido en R, veamos un ejemplo de simulación de valores de una distribución normal estándar a partir de la función de densidad de la doble exponencial estándar, repitiendo el algoritmo $n = 50$ veces:

```
set.seed(678); sim_normal(50)
```

```
##  [1] -0.1919  1.7448  1.0051 -0.0001 -0.8321 -0.3135  0.2581  1.7415  0.5092
## [10] -1.1385  0.5559 -0.9290  1.2875  0.7231  0.0702  0.0733 -1.3784  1.0761
## [19] -0.3266 -0.1602 -0.1336 -0.3161 -0.1632 -0.1635 -0.8350  0.3998  0.6787
## [28] -0.0965  1.1617 -0.9927  0.0271 -0.3275 -0.6626 -0.5175 -0.8393  1.2518
## [37]  0.3624
```

En este caso hemos obtenido un total de 37 valores simulados de la distribución normal estándar.

También podemos comparar la función de densidad teórica de una distribución normal estándar con la distribución de los valores obtenidos por el método de aceptación-rechazo haciendo uso de un histograma. Usaremos la función *hist* de R. En el argumento *breaks*, que establece los puntos de separación entre las barras del histograma, usaremos el método de *Freedman-Diaconis*, "FD", mediante el que se puede obtener la anchura óptima de las barras. Con la función *curve* de R y el argumento *dnorm*, podemos dibujar sobre el histograma anterior la gráfica de la función de densidad de una distribución normal estándar. En la figura 3.1 podemos observar el resultado obtenido al implementar el siguiente código en R:

```
x=sim_normal(100000)
hist(x, breaks = "FD", freq = FALSE, main=" ")
curve(dnorm, add = TRUE)
```

3.2. Simulación de variables aleatorias discretas

Aunque hay varios métodos para simular distribuciones discretas, en esta sección nos centraremos en estudiar el *método de la transformación cuantil* y el *método de la tabla guía*.

3.2.1. Método de la transformación cuantil

Este es un método general que nos permite obtener valores de una cierta variable aleatoria discreta X, que tiene como posibles valores $\{x_1,\ x_2,\ \ldots,\ x_n,\ \ldots\}$ con probabilidades $p_k = P\{X = x_k\},\ k = 1, 2, \ldots, n, \ldots$

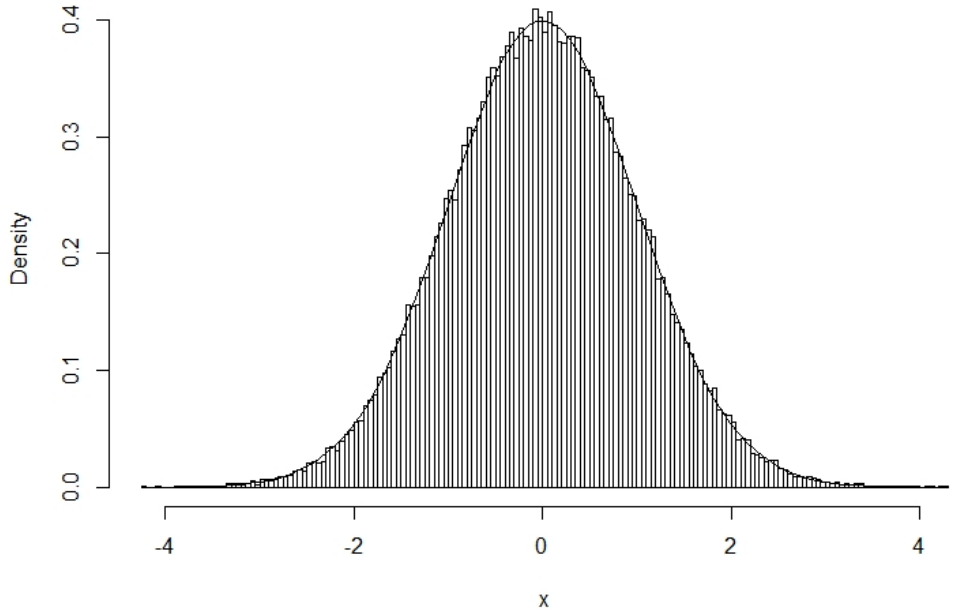

Figura 3.1: *Histograma de los valores simulados mediante $n = 100\ 000$ repeticiones del algoritmo de aceptación-rechazo frente a la función de densidad teórica de una distribución normal estándar.*

El teorema de inversión no se puede aplicar directamente a variables discretas, pues su función de distribución, sea esta F, no es continua ni invertible. Como F no es invertible (pues es una función constante a trozos con discontinuidades de salto) no podemos definir su función inversa. Sin embargo, podemos definir la denominada *función cuantil* (o *inversa generalizada*) para cualquier distribución F:

$$Q(u) = \text{ínf}\{x \in \mathbb{R}/F(x) \geq u\}, \quad \forall u \in (0,1)$$

En el caso de que F sea invertible, se cumple que $Q = F^{-1}$.

Notemos que, si consideramos una variable aleatoria con distribución $U(0,1)$, $U : \Omega \longrightarrow (0,1)$, y la función cuantil Q de una cierta función de distribución F, entonces podemos definir una nueva variable aleatoria, a la que denotaremos por $Y = Q(U)$, de la siguiente forma: $Q(U) : \Omega \xrightarrow{U} (0,1) \xrightarrow{Q} \mathbb{R}$. Se tiene el siguiente resultado:

Teorema 3.6 (de inversión generalizada). *Si X es una variable aleatoria con distribución F y función cuantil Q, y U es una variable aleatoria con distribución $U(0,1)$, entonces la variable aleatoria $Y = Q(U)$ tiene distribución F.*

Demostración. Llamemos G a la función de distribución de la variable $Y = Q(U)$. Queremos comprobar que $G = F$. Se tiene, $\forall x \in \mathbb{R}$:

$$G(x) = P\{Y \leq x\} \overset{(a)}{=} P\{\text{ínf}\{y \in \mathbb{R}/F(y) \geq U\} \leq x\} = P\{U \leq F(x)\} = \int_0^{F(x)} 1 \, \mathrm{d}u = F(x)$$

donde en (a) hemos usado la definición de función cuantil, y así $G = F$, como queríamos ver. □

A partir de este resultado, se deduce un algoritmo que nos permite simular valores de cualquier distribución discreta. Veámoslo.

`Algoritmo de transformación cuantil`
`1. Generar` $U \backsim U(0,1)$.
`2. Devolver` $X = Q(U)$.

Podemos aplicar este algoritmo de manera directa si es sencillo hallar de forma explícita el valor de la función Q que nos interesa para cualquier $u \in (0,1)$.

Ejemplo 3.7 (Distribución geométrica mediante el método de transformación cuantil). Sea X una variable aleatoria con distribución $Geom(p)$. A partir de su función de distribución podemos hallar el valor de la función cuantil para cualquier $u \in (0,1)$, siendo ese valor un número entero no negativo k cumpliendo la definición. Veámoslo:

$$F(k) \geq U > F(k-1) \Leftrightarrow 1-(1-p)^{k+1} \geq U > 1-(1-p)^k \Leftrightarrow (1-p)^k > 1-U \geq (1-p)^{k+1} \Leftrightarrow$$

$$\Leftrightarrow k \cdot \ln(1-p) > \ln(1-U) \geq (k+1) \cdot \ln(1-p) \Leftrightarrow k < \frac{\ln(1-U)}{\ln(1-p)} \leq k+1$$

Por lo tanto obtenemos:

$$k = \left\lfloor \frac{\ln(1-U)}{\ln(1-p)} \right\rfloor$$

Antes de establecer el algoritmo, podemos hacer un par de simplificaciones. En primer lugar, definimos $a = \frac{1}{\ln(1-p)}$. Ahora, al igual que hicimos en el ejemplo 3.2, como $U \backsim U(0,1)$, entonces $1-U \backsim U(0,1)$. De esta forma, se tiene el siguiente algoritmo para generar valores de X:

`1.Generar` $U \backsim U(0,1)$
`2.Devolver` $X = \lfloor a \cdot \ln(U) \rfloor$

Veámoslo en R, donde vamos a generar 20 valores de una variable geométrica con parámetro $p = 0{,}2$:

```
sim_geom<-function(p,n){
  a=1/(log(1-p))
  u=runif(n)
  x=floor(a*log(u)); x
}
sim_geom(0.2,20)
```

```
##  [1]  0  6  0  5  4  4  1  2  4  0  2 13  0  4  1  2  0  0  4  2
```

El mayor inconveniente de este algoritmo puede ser la dificultad de encontrar el valor de $Y = Q(U)$ de forma explícita, como hemos hecho en este último ejemplo. En general, se tiene:

$$Y = \text{ínf}\{x_k \ / \ \sum_{i=1}^{k} p_i \geq U\} = x_j$$

de forma que $\sum_{i=1}^{j} p_i \geq U > \sum_{i=1}^{j-1} p_i$. En lo sucesivo, nos será de utilidad considerar una función I a la que nos referiremos como *codificación* o *etiquetado*, que nos permite "guardar" el orden de los valores de X, por ejemplo, cada valor i de la variable I se correspondería con el subíndice i del valor x_i de la variable aleatoria X. El problema se reduce, entonces, a encontrar el valor j de la variable I que cumple la condición $\sum_{i=1}^{j} p_i \geq U > \sum_{i=1}^{j-1} p_i$, y luego tomar el valor x_j de X. Este procedimiento se puede ver en el siguiente algoritmo:

`Algoritmo de transformación cuantil con búsqueda secuencial`

`1. Generar` $U \backsim U(0,1)$.

`2. Establecer` $I = 1$ `y` $S = p_I$.

`3. Mientras` $U > S$, `hacer` $I = I + 1$ `y` $S = S + p_I$.

`4. Devolver` $X = x_I$.

Ejemplo 3.8 (**Distribución de Poisson mediante el método de la transformación cuantil**)**.** Sea X una variable aleatoria con distribución de Poisson de parámetro λ, con posibles valores $x_0 = 0,\ x_1 = 1,\ \ldots$ Aunque hayamos considerado que los etiquetados de los valores comenzaban en el valor 1, podemos modificarlo de forma que el etiquetado de cada valor coincida con el valor numérico del mismo, pues de este modo será más sencillo aplicar el algoritmo que hemos visto. Calculamos sus probabilidades haciendo uso de la función de masa de una distribución de Poisson:

$$p_j = P\{X = x_j\} = P\{X = j\} = \frac{e^{-\lambda}\lambda^j}{j!}, \quad j = 0, 1, 2, \ldots$$

Notemos además que estas probabilidades pueden calcularse recursivamente:

$$P\{X = j\} = \frac{e^{-\lambda}\lambda^j}{j!} = \frac{\lambda e^{-\lambda}\lambda^{j-1}}{j(j-1)!} = \frac{\lambda}{j} P\{X = j-1\}$$

Teniendo en cuenta todas estas simplificaciones, podemos emplear el siguiente algoritmo para generar valores de la variable X:

`1. Generar` $U \backsim U(0,1)$.

`2. Establecer` $I = 0,\ p = e^{-\lambda}$ `y` $S = p$.

`3. Mientras` $U > S$ `hacer` $I = I + 1,\ p = \frac{\lambda}{I} p$ `y` $S = S + p$.

`4. Devolver` $X = I$.

Veamos cómo podemos implementarlo en *R*:

```
sim_poisson<-function(n,lambda){
  x=numeric()
  for(i in 1:n){
    u=runif(1)
    I=0; p=exp(-lambda); S=p;
    while(u>S){
      I=I+1; p=(lambda/I)*p; S=S+p
    }
    x=c(x,I)
  }
  x
}
```

Usaremos esta función para generar una muestra de 30 valores aleatorios de una distribución de Poisson con parámetro $\lambda = 2$:

```
sim_poisson(30,2)
```

```
##  [1] 1 2 1 2 1 2 1 0 3 6 1 4 2 4 3 1 2 4 2 3 1 1 0 2 0 4 3 1 1 2
```

Hasta aquí siempre hemos considerado una ordenación de menor a mayor de los posibles valores que toma una variable aleatoria. Aunque no vamos a tratar este tema en este trabajo, cabe destacar que existen muchas otras formas de etiquetar estos valores, y se puede estudiar cómo conseguir el etiquetado que proporciona el algoritmo más eficiente en cada caso.

3.2.2. Método de la tabla guía

En el apartado anterior hemos visto que una complicación del método de la transformación cuantil puede surgir a la hora de encontrar el índice j tal que cumpla la condición $\sum_{i=1}^{j} p_i \geq U > \sum_{i=1}^{j-1} p_i$ (aunque para algunas distribuciones este índice se pueda calcular de forma directa, como en el ejemplo de la distribución geométrica). El método de la tabla guía permite reducir el número de comparaciones que se necesitan para comprobar la mencionada condición.

En primer lugar, consideramos una variable aleatoria discreta X que toma valores $\{x_1, \ldots, x_n\}$ y tiene función de masa $p_j,\ j = 1, 2, \ldots, n$. Calculamos recursivamente las sumas acumuladas de estas probabilidades, a las que denotaremos por $q_j = \sum_{i=1}^{j} p_i$, y entonces se tiene: $q_0 = 0$, $q_j = q_{j-1} + p_j,\ j = 1, 2 \ldots, n$. La variable I va a representar el etiquetado de los valores de X,

para los que suponemos que se encuentran ordenados de menor a mayor. Consideramos ahora n subintervalos de igual tamaño en $[0,1]$, es decir, $J_i = [\frac{i-1}{n}, \frac{i}{n}),\ i = 1, 2, \ldots, n$ para luego definir los valores:

$$g_i = \text{máx}\left\{ j \ / \ q_j < \frac{i}{n} \right\},\ i = 1, 2 \ldots, n$$

donde $g_i = 0$ si el conjunto correspondiente es vacío. Los valores g_i representan, para cada intervalo, el valor más alto del índice $j = 1, 2, \ldots, n$ de forma que la correspondiente suma acumulada de las probabilidades es menor que el extremo derecho del intervalo, y van a almacenarse en lo que conoceremos como *tabla guía*. Debemos tener en cuenta que q_n es siempre igual a 1 por ser la suma acumulada de las probabilidades de todos los posibles valores, y así, por definición, podemos deducir que $g_n = n - 1$.

Veamos entonces el algoritmo para calcular la tabla guía de una cierta variable aleatoria discreta con función de masa p_j. Podríamos hacerlo por definición, para lo que necesitaríamos calcular las sumas acumuladas q_j, o mediante este otro método que exponemos a continuación, que resulta más eficiente por necesitar un menor número de cálculos y comparaciones.

1. `Establecer` $g_i = 0$ `para` $i = 1, \ldots, n-1$, $g_n = n - 1$ `y` $S = 0$.
2. `Para` $i = 1, \ldots, n-1$ `hacer` $S = S + p_i$, $j = \lfloor n \cdot S \rfloor + 1$, $g_j = i$.
3. `Para` $i = 2, \ldots, n-1$ `hacer` $g_i = \text{máx}\{g_i,\ g_{i-1}\}$.

Una vez sabemos calcular la tabla guía correspondiente, vamos a ver cómo la podemos aprovechar para establecer un algoritmo que nos permita reducir el número de cálculos necesarios a la hora de hallar el índice j que cumple $\sum_{i=1}^{j} p_i \geq U > \sum_{i=1}^{j-1} p_i$ en el método de la transformación cuantil.

Sea U una variable aleatoria siguiendo una distribución $U(0,1)$. Si generamos un valor de U podemos saber en cuál de los intervalos J_i definidos antes ha caído, calculando el índice i del intervalo como sigue:

$$\frac{i-1}{n} \leq U < \frac{i}{n} \ \Leftrightarrow\ i - 1 \leq n \cdot U < i \ \Leftrightarrow\ i = \lfloor n \cdot U \rfloor + 1$$

donde $\lfloor n \cdot U \rfloor$ denota la parte entera de $n \cdot U$.

A continuación, para hallar el valor de la variable I que vamos a simular debemos tener en cuenta que este valor será $g_i + 1$ si se cumple $U > q_{g_i}$. En otro caso, tenemos que encontrar el primer índice $j = g_i - 1,\ g_i - 2,\ \ldots,\ 0$ para el que se tiene $U > q_j$ y establecer $I = j + 1$ (pues recordemos que j debe cumplir $\sum_{i=1}^{j} p_i \geq U > \sum_{i=1}^{j-1} p_i$). Tras simular los valores de la variable I no tenemos más que considerar sus correspondientes valores de X, como hemos visto en el método de la transformación cuantil.

A continuación, presentamos el algoritmo de simulación mediante una tabla guía:

1. Generar $U \backsim U(0,1)$.
2. Establecer $i = \lfloor n \cdot U \rfloor + 1$ y $j = g_i$.
3. Mientras $U \leq q_j$, hacer $j = j - 1$.
4. Establecer $I = j + 1$ y devolver $X = x_I$.

Ejemplo 3.9 (**Distribución binomial mediante el método de la tabla guía**). Se puede establecer un algoritmo para simular valores de una variable aleatoria binomial a partir de la definición de su función de masa, pero el método resultante es muy ineficiente para valores grandes del parámetro n. Por este motivo, vamos a ver cómo podemos simular una distribución binomial empleando el método de la tabla guía.

En este ejemplo vamos a considerar la situación expuesta en la sección 1.4.2, por lo que simularemos una variable aleatoria discreta X que sigue una distribución binomial de parámetros $n = 15$ y $p = 1/6$, es decir, $Bin(15, 1/6)$. Los posibles valores son: $x_1 = 0,\ x_2 = 1,\ \ldots,\ x_{16} = 15$, y esto implica que la variable I que guarda el etiquetado puede tomar los valores $\{1, \ldots, 16\}$.

En primer lugar, vamos a definir tres funciones en R que necesitaremos para llevar a cabo el método de la tabla guía. Cada una de ellas nos servirá para calcular la función de masa, las sumas acumuladas y la tabla guía de una distribución binomial dada, respectivamente. A continuación podemos ver cómo se definen estas tres funciones.

```
fmasa_binom<-function(n,p){
  m=n+1; P=numeric(m)
  for(i in 1:m){
    P[i]=choose(n,i-1)*(p**(i-1))*((1-p)**(n-(i-1)))
  }
  P
}

sum_acumuladas_binom<-function(n,p){
  m=n+1; q=numeric(m); q0=0
  P=fmasa_binom(n,p)
  for(i in 1:m){
    q[i]= q0+P[i]
    q0=q[i]
  }
  q
}
```

```
tab_guia_binom<-function(n,p){
  m=n+1
  g=numeric(m); g[m]=n; S=0
  P=fmasa_binom(n,p)
  for(i in 1:n){
    S=S+P[i]; j=floor(m*S)+1; g[j]=i
  }
  for(i in 2:n){
    g[i]=max(g[i],g[i-1])
  }
  g
}
```

Vamos a implementar ahora el algoritmo para simular valores de una variable binomial mediante el método de la tabla guía utilizando estas tres funciones auxiliares. Definimos entonces una función en la que debemos indicar t el número de valores que queremos generar y los parámetros n y p de la distribución binomial. Teniendo en cuenta cómo hemos establecido el etiquetado, si restamos 1 a cada valor generado de la variable I, obtendremos el valor de la variable binomial correspondiente. Veamos el algoritmo en R:

```
sim_binom<-function(t,n,p){
  m=n+1
  q=sum_acumuladas_binom(n,p)
  g=tab_guia_binom(n,p)
  X=numeric()
  for(k in 1:t){
    u=runif(1)
    i=floor(m*u)+1; j=g[i]
    if(j==0){
      I=j+1; X=c(X,I-1)
    } else {
      while(u <= q[j]){
      j=j-1
      if(j==0) break
      }
      I=j+1; X=c(X,I-1)
    }
  }
```

```
  print(g)    # Tabla guia
  print(X)    # Valores simulados de X
}
```

Haciendo uso de esta función, generamos 30 valores de la variable X que sigue una distribución $Bin(15, 1/6)$.

```
sim_binom(30,15,1/6)


##  [1]  0  1  1  1  2  2  2  2  3  3  3  3  4  4  5 15
##  [1] 0 2 4 1 2 3 5 4 2 4 3 5 1 5 0 4 2 2 2 3 5 4 1 2 1 3 1 2 1 4
```

En la salida de R podemos ver tanto la tabla guía que se ha utilizado (fila superior) como los valores de X generados mediante este método (fila inferior).

En el algoritmo de simulación mediante una tabla guía, si el valor generado de la variable U cae en el intervalo J_i, se puede comprobar que el número de comparaciones que realizamos en el paso 3 es menor o igual que 1 más el número de valores q_j que caen en el intervalo J_i. De esta forma, considerando la variable aleatoria $N =$*"número de comparaciones"* y teniendo en cuenta que U sigue una distribución $U(0,1)$, podemos acotar el número medio de comparaciones que necesitamos en el paso 3 del algoritmo como sigue:

$$E(N) \leq \frac{1}{n}\sum_{i=1}^{n}(1 + \#\{j \ / \ q_j \in J_i\}) = 1 + \frac{1}{n}\sum_{i=1}^{n}\#\{j \ / \ q_j \in J_i\} =$$

$$= 1 + \frac{1}{n}\#\{j \ / \ q_j \in [0,1)\} = 1 + \frac{n-1}{n} < 2$$

donde # hace referencia al cardinal de los conjuntos correspondientes.

Para una variable aleatoria discreta que toma n posibles valores, se puede aplicar el método de la tabla guía utilizando tablas guía de m elementos, donde m no tiene por qué coincidir con n. En este caso, el intervalo $[0,1)$ se dividiría en m subintervalos de igual longitud y el número medio de comparaciones estaría acotado por $E(N) \leq 1 + \frac{n}{m}$. De esta forma, cuando n es un número demasiado grande, podemos considerar un número m más reducido para construir la tabla guía con el objetivo de reducir el espacio que nos ocuparía en la memoria del ordenador y al mismo tiempo mantener una acotación razonable para el número medio de comparaciones. Por ejemplo, si queremos simular mediante este método una variable aleatoria discreta con 750 000 posibles valores, podemos considerar una tabla guía de 15 000 elementos, por lo que el número medio de comparaciones sería inferior o igual a $1 + \frac{750000}{15000} = 51$.

3.3. Métodos específicos

A lo largo de este capítulo, hemos presentado diferentes métodos generales para simular valores de variables aleatorias que siguen distribuciones tanto continuas como discretas. Para algunas de estas distribuciones, estos métodos pueden no ser los más eficientes, pues no aprovechan las propiedades particulares de cada distribución. En esta sección, vamos a ver dos casos de distribuciones notables para las que podemos desarrollar un método específico de simulación a partir de sus propiedades: la distribución de Poisson y la distribución de Erlang.

3.3.1. Método específico para la distribución de Poisson

Hemos visto que podemos generar valores de una variable aleatoria con distribución $Poisson(\lambda)$ utilizando el método de la transformación cuantil. Sin embargo, también podemos hacer uso de una relación existente entre esta distribución y la distribución $Exp(\lambda)$. Dadas $T_1, \ldots, T_n, \ldots$ variables aleatorias independientes y con distribución $Exp(\lambda)$, se tiene que la variable aleatoria X que cumple:

$$\sum_{i=1}^{X} T_i \leq 1 < \sum_{i=1}^{X+1} T_i \tag{3.1}$$

(definiendo $X = 0$ si $T_1 > 1$) sigue una distribución $Poisson(\lambda)$.

Podemos simular los valores de las variables aleatorias T_i utilizando el método de inversión, como hemos visto en el ejemplo 3.2. Siguiendo ese método, establecemos $T_i = L \cdot \ln(U_i)$, con $L = -\frac{1}{\lambda}$ y U_i un valor de la variable aleatoria U uniforme en $(0, 1)$. A partir de esta última expresión y de la relación (3.1), se obtiene lo siguiente:

$$\sum_{i=1}^{X} T_i \leq 1 < \sum_{i=1}^{X+1} T_i \;\Leftrightarrow\; -\sum_{i=1}^{X} \frac{\ln(U_i)}{\lambda} \leq 1 < -\sum_{i=1}^{X+1} \frac{\ln(U_i)}{\lambda} \;\Leftrightarrow\; -\frac{\ln\left(\prod_{i=1}^{X} U_i\right)}{\lambda} \leq$$

$$\leq 1 < -\frac{\ln\left(\prod_{i=1}^{X+1} U_i\right)}{\lambda} \;\Leftrightarrow\; \ln\left(\prod_{i=1}^{X} U_i\right) \geq -\lambda > \ln\left(\prod_{i=1}^{X+1} U_i\right) \;\Leftrightarrow\; \prod_{i=1}^{X} U_i \geq e^{-\lambda} > \prod_{i=1}^{X+1} U_i$$

Podemos establecer así a través de la distribución $Exp(\lambda)$ un algoritmo sencillo para simular valores de la variable aleatoria X con distribución de $Poisson(\lambda > 0)$:

1. Establecer $p = 1$ y $S = -1$.
2. Mientras $p \geq e^{-\lambda}$, generar $U \backsim U(0,1)$ y hacer $p = p \cdot U$, $S = S + 1$.
3. Devolver $X = S$.

Ejemplo 3.10 (Distribución de Poisson a través de la distribución exponencial). Vamos a utilizar el algoritmo anterior para generar 30 valores de una variable aleatoria X con distribución $Poisson(4)$. Lo implementamos en R:

```
sim_pois_con_exp<-function(lambda,n){
  x=numeric()
  for(i in 1:n){
    p=1; S=-1
    while(p >= exp(-lambda)){
      u=runif(1)
      p=p*u; S=S+1
    }
    x=c(x,S)
  }
  x
}
sim_pois_con_exp(4,30)

##  [1] 2 2 4 3 0 3 5 2 0 5 5 6 0 4 3 5 2 6 3 5 4 5 1 5 6 3 6 4 1 4
```

3.3.2. Método específico para la distribución de Erlang

Como hemos comentado en el primer capítulo, esta distribución es un caso particular de la distribución gamma con $p \in \mathbb{Z}^+$. Por otra parte, recordemos que $\Gamma(p) = \int_0^\infty x^{p-1}e^{-x}\,\mathrm{d}x$. En el cálculo de esta integral se obtiene la expresión recursiva $\Gamma(p) = (p-1)\,\Gamma(p-1)$. Además, $\Gamma(1) = \int_0^\infty e^{-x}\,\mathrm{d}x = 1$. Llegamos así a la expresión $\Gamma(p) = (p-1)!$, $p \in \mathbb{Z}^+$. Por tanto, la función de densidad de la distribución de Erlang tiene la siguiente forma:

$$f(x) = \begin{cases} \dfrac{a(ax)^{p-1}e^{-ax}}{(p-1)!} & \text{si } x > 0 \\ 0 & \text{si } x \leq 0 \end{cases}$$

Además, si $p = 1$, la función de densidad de la distribución $\Gamma(a, 1)$, $a > 0$, coincide con la función de densidad de una distribución $Exp(a)$, es decir:

$$f(x) = \begin{cases} ae^{-ax} & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$$

Otra propiedad que nos resultará útil es la reproductividad: si tenemos n variables aleatorias independientes $X_1, \ldots, X_n$ con distribuciones $\Gamma(a, p_1), \ldots, \Gamma(a, p_n)$, $a,\ p_1,\ \ldots,\ p_n > 0$, entonces la variable aleatoria $\sum_{i=1}^n X_i$ sigue una distribución $\Gamma(a, \sum_{i=1}^n p_i)$, $a > 0$.

Veamos entonces cómo podemos simular valores de una variable aleatoria con distribución de Erlang a partir de una suma de variables con distribución exponencial. En este caso, $p \in \mathbb{Z}^+$, por lo que si consideramos $X_1, \ldots, X_p$ variables aleatorias independientes con distribución $\Gamma(a,1)$, o lo que es lo mismo, $Exp(a)$, su suma, $\sum_{i=1}^{p} X_i$, es una variable aleatoria que sigue una distribución $\Gamma(a,p)$. Los valores de cada una de las variables X_i pueden simularse utilizando el método de inversión, es decir, $X_i = L \cdot U_i$, con $L = -1/a$ y U_i un valor de la variable aleatoria U uniforme en $(0,1)$ (como hemos visto en el ejemplo 3.2). Se tendría así lo siguiente:

$$\sum_{i=1}^{p} X_i = -\sum_{i=1}^{p} \frac{\ln(U_i)}{a} = L \cdot \ln\left(\prod_{i=1}^{p} U_i\right)$$

Resumiendo, el algoritmo obtenido para simular valores de una distribución de Erlang $\Gamma(a,p)$, con $p \in \mathbb{Z}^+$, puede expresarse como sigue:

1. Establecer $L = -1/a$ y $S = 1$.
2. Para $i = 1, \ldots, p$ generar $U \backsim U(0,1)$ y hacer $S = S \cdot U$.
3. Devolver $X = L \cdot \ln(S)$.

Ejemplo 3.11 (**Distribución de Erlang a partir de la suma de distribuciones exponenciales**)**.** Para ejemplificar cómo funciona el algoritmo anterior, vamos a implementarlo en R simulando 30 valores de una distribución de Erlang de parámetros $a = 3$ y $p = 4$:

```
sim_erlang<-function(a,p,n){
  L= -1/a
  x=numeric(n); set.seed(123)
  for(k in 1:n){
    S=1
    for(i in 1:p){
      u=runif(1); S=S*u
    }
    x[k]=L*log(S)
  }
  x
}
round(sim_erlang(3,4,30),4)  # Redondeamos los valores a 4 cifras decimales
```

```
##  [1] 0.8342 1.3008 0.7381 1.1087 1.9108 0.3118 0.6320 1.0993 1.6806 1.4716
## [11] 1.5689 2.0253 1.7922 1.4948 1.1453 1.6713 0.4751 0.5989 3.3259 1.5665
## [21] 0.9756 1.0784 1.4414 1.4334 1.3489 0.7913 0.4766 1.3578 1.7109 0.6955
```

Ahora que ya conocemos varios métodos para generar variables aleatorias continuas y discretas, podemos utilizarlos en simulación. Veremos un par de ejemplos de simulación de sistemas reales en el siguiente capítulo.

Capítulo 4

Aplicaciones de la simulación

En la introducción hemos visto de manera intuitiva en qué consiste la simulación estadística. Antes de continuar, vamos a introducir algunos conceptos relacionados con los distintos tipos de simulación que se utilizan dentro de este campo, y de los que veremos un par de ejemplos.

En primer lugar, podemos distinguir entre simulación *estática*, si el paso del tiempo no influye en el modelo del sistema real que estamos estudiando, y simulación *dinámica*, si el tiempo es una de las variables importantes del modelo. Algunas de las ventajas que presenta la simulación estática son la posibilidad de comparar distintas situaciones bajo las mismas condiciones y, en general, un menor coste computacional respecto a la simulación dinámica. Precisamente, un elevado coste computacional puede ser uno de los principales incovenientes de la simulación dinámica, pues necesitamos analizar los distintos estados por los que va pasando un sistema a lo largo del tiempo.

Dentro de la simulación dinámica podemos encontrar dos grandes categorías: la simulación *continua*, en la que suponemos que se producen cambios en el sistema constantemente, y la simulación *discreta*, si esos cambios se producen en instantes de tiempo muy concretos.

De nuevo, en la simulación discreta aparecen dos grandes tipos. Por una parte, tenemos la simulación *por eventos*, en la que podemos controlar la variable que representa el tiempo y "moverla" hasta el instante en el que ocurre el siguiente cambio en el sistema. A cada uno de estos cambios lo llamaremos *evento*. Como cabe esperar, es necesario saber en cada instante cuándo se va a producir el siguiente evento. Por otra parte, encontramos la simulación *por cuantos*, cuando estudiamos el sistema dejando pasar pequeños intervalos de tiempo de longitud fija δ (a los que llamaremos *cuantos*). Suponemos que en cada uno de estos intervalos se va a producir, como mucho, un cambio o suceso en el sistema. Aunque la simulación por eventos tiene, en general, una ejecución más rápida que la simulación por cuantos, en muchas ocasiones esta última es la única opción factible para estudiar un sistema (dentro de la simulación dinámica discreta), pues no siempre conoceremos de forma exacta los instantes de tiempo en los que se produce un suceso.

4.1. Simulación estática

Veremos en esta sección un juego para el que nos interesará conocer si resulta rentable jugar.

Ejemplo 4.1. Supongamos que un jugador lanza una moneda, pagando un euro en cada lanzamiento, hasta que el número de caras obtenidas sea una unidad mayor que el número de cruces. En ese momento, finaliza el juego, y el jugador recibe 10 euros. Veamos cómo podemos ver si compensa jugar, es decir, que no perdamos dinero en el juego.

Si analizamos el planteamiento del juego, observamos que solo podremos ganar en un lanzamiento impar. Si el número de caras tiene que superar en una unidad al número de cruces, entonces uno de los números es par y el otro impar, por lo que su suma es siempre un número impar. Además, si la condición para finalizar el juego se produce a partir del décimo lanzamiento, estaríamos perdiendo dinero (pues pagamos un euro por cada lanzamiento y solo recibiríamos 10 euros). Se deduce entonces que para obtener un beneficio en este juego tenemos que ganar en los lanzamientos 1, 3, 5, 7 o 9. Denotamos por GL_i el suceso "*ganar en el lanzamiento i* ". De esta forma, la probabilidad de "*ganar*" es $P(G) = P(GL_1 \cup GL_3 \cup GL_5 \cup GL_7 \cup GL_9)$.

Si en el primer lanzamiento obtenemos una cara, habremos ganado. La probabilidad de que esto ocurra es $1/2$, pues los posibles resultados son cara o cruz. De esta forma, $P(GL_1) = 1/2$.

Para llegar al tercer lanzamiento no pudimos haber ganado en el primero, por lo que se obtuvo una cruz en ese lanzamiento. La probabilidad de que esto ocurra es la complementaria a obtener una cara, es decir, $1-1/2 = 1/2$. Una vez llegamos al tercer lanzamiento, podemos encontrarnos con 4 situaciones distintas. Si denotamos la cara por "c" y la cruz por "+", estas situaciones serían: +cc, +c+, ++c, +++. Solo en la primera se cumple que hay una cara más que cruces, por lo que la probabilidad de ganar en el tercer lanzamiento (condicionado a que no hemos ganado en el primero) es $1/4$. Denotamos el suceso "*no haber ganado en el lanzamiento i* " por NL_i. De esta forma, la probabilidad de ganar en el tercer lanzamiento sería:

$$P(GL_3) = P(GL_3/NL_1) \cdot P(NL_1) = \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{8}$$

Se puede realizar un razonamiento análogo para los lanzamientos 5, 7 y 9, siempre teniendo en cuenta que la probabilidad de ganar en cada uno de ellos se obtiene al multiplicar la probabilidad de ganar en ese lanzamiento condicionada a no haber ganado en los lanzamientos impares anteriores por la probabilidad de llegar a ese lanzamiento (es decir, la probabilidad de no haber ganado en los lanzamientos impares anteriores). Se puede comprobar que:

$$P(GL_1) = \frac{1}{2} \qquad P(GL_3) = \frac{1}{8} \qquad P(GL_5) = \frac{1}{16} \qquad P(GL_7) = \frac{5}{128} \qquad P(GL_9) = \frac{7}{256}$$

Por tanto, en virtud de la aditividad numerable de la probabilidad, obtendremos la probabilidad

de G al sumar todas estas probabilidades:

$$P(G) = \frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{5}{128} + \frac{7}{256} = \frac{193}{256} \approx 0'75391$$

Vamos ahora a obtener esta probabilidad mediante la simulación del juego. La idea consiste en simular un gran número de partidas del juego, y hallar la proporción del número de partidas en las que hemos ganado frente al número total de partidas jugadas. Veamos cómo podemos hacer esta simulación en *R*. El comando *sample* con el argumento *size*= 1 permite obtener un elemento aleatorio de una lista de elementos dada. Simularemos $n = 1000$ partidas de este juego, y posteriormente calcularemos la proporción de partidas en las que hemos perdido y la proporción de partidas en las que hemos ganado, haciendo uso del comando *table* asociado a la condición "euros<10", o lo que es lo mismo, la proporción de partidas en las que hemos necesitado menos de 10 lanzamientos para ganar.

```
n=1000; set.seed(543); euros=numeric()
for(i in 1:n){
  caras=0; cruces=0
  while((caras-cruces)<1){
    x=sample(c("cara","cruz"), size=1)
    if(x=="cara"){
      caras=caras+1
    } else {
      cruces=cruces+1
    }
  }
  euros[i]=caras+cruces
}
table(euros<10)/n


##
## FALSE  TRUE
## 0.246 0.754
```

Como podemos observar, se obtiene una probabilidad de ganar de 0'754, muy próxima a la probabilidad teórica, 0'75391. Una de las ventajas de resolver este juego mediante simulación es la sencillez del proceso, pues no necesitamos analizar el juego con detalle (como hemos hecho antes) y es fácil de implementar en el ordenador, obteniendo además una buena aproximación de la probabilidad teórica.

4.2. Simulación dinámica

En esta sección vamos a exponer un ejemplo de simulación dinámica por eventos. Este ejemplo se enmarca dentro de la *teoría de colas*, una disciplina de la Investigación Operativa, que se encarga de estudiar y analizar situaciones en las que un cierto servicio es demandado, pero no puede satisfacerse de forma inmediata, por lo que se provocan colas y esperas. En este contexto, llamaremos *cliente* a cualquier individuo que solicita el servicio, y la *cola* será el conjunto de clientes que esperan ser atendidos. Podemos pensar, por ejemplo, en las colas formadas en las cajas de un supermercado o en los cajeros automáticos.

En general, en los sistemas de colas podemos distinguir tres componentes principales: el *proceso de llegada*, en el cual se describe la distribución de llegada de los clientes; el *mecanismo de servicio*, para el que necesitamos conocer tanto el número de servidores disponibles como la distribución de los tiempos de servicio; y la *disciplina de la cola*, que no es más que la regla que se utiliza para decidir qué cliente de la cola va a ser el siguiente en ser atendido.

Uno de los modelos más sencillos que podemos encontrar en los sistemas de colas es el *modelo M/M/1*, en el que existe un único servidor y las distribuciones de los tiempos entre llegadas y los tiempos de servicio son exponenciales de parámetros λ y μ, respectivamente. De esta forma, vamos a distinguir dos posibles eventos: "*llegadas*" y "*finalizaciones del servicio*". En este modelo, se considera que el número de clientes potenciales es infinito y hay una única cola, para la que no se establece un límite de longitud. Además, se supone que los clientes no abandonan el sistema una vez entran en la cola. Se utiliza la disciplina de cola FIFO, que establece que el primer cliente de la cola va a ser el primero en ser atendido (*first in, first out*).

Llamaremos TE a la variable aleatoria que indica el tiempo hasta el siguiente evento, es decir, el tiempo que transcurre desde que se produce un evento hasta que ocurre el siguiente. Esta variable sigue una distribución exponencial de parámetro $\lambda + \mu$. En efecto, si T_1 es la variable aleatoria que representa el tiempo hasta la siguiente llegada y T_2 la variable aleatoria que representa el tiempo hasta la siguiente finalización de un servicio, entonces la variable aleatoria $TE = min(T_1, T_2)$ representa el tiempo hasta el siguiente evento (llegada o finalización del servicio), y su distribución, siendo $T_1 \backsim Exp(\lambda)$ y $T_2 \backsim Exp(\mu)$, es $Exp(\lambda + \mu)$.

Ejemplo 4.2 (**Simulación de un sistema de colas**)**.** Vamos a simular un sistema de colas que representa un banco donde solo hay un cajero. Suponemos que los tiempos entre llegadas de clientes (de uno en uno) son variables aleatorias independientes con distribución exponencial de parámetro λ, y los tiempos de servicio (tiempo que permanecen en el cajero) son también variables aleatorias con distribución exponencial de parámetro μ.

Una vez se produce un evento, el siguiente evento puede ser la llegada de un cliente o la finalización de un servicio (y el cliente sale del sistema de colas). Hemos visto que $TE = min(T_1, T_2)$ sigue una

distribución $Exp(\lambda+\mu)$, mientras que $T_1 \backsim Exp(\lambda)$ y $T_2 \backsim Exp(\mu)$. Entonces, la probabilidad de que el siguiente evento sea una llegada es: $P(T_1 = TE) = \frac{\lambda}{\lambda+\mu}$, y la probabilidad de que el siguiente evento sea una finalización de servicio es: $P(T_2 = TE) = \frac{\mu}{\lambda+\mu}$. De esta forma, para determinar en la simulación el tipo de evento que se va a producir, generamos una variable aleatoria $p \backsim U(0,1)$. Si $p < \frac{\lambda}{\lambda+\mu}$, entonces el siguiente evento es una llegada, y en otro caso, una finalización de servicio, con lo cual cumplimos las condiciones anteriormente indicadas.

Vamos a detallar ahora el código de *R* que nos permite simular este sistema de colas, así como calcular la longitud media de la cola y visualizar cómo evoluciona la longitud de la cola respecto al tiempo transcurrido. Para este ejemplo, vamos a elegir los parámetros λ y μ de forma que sean próximos y se cumpla $\lambda < \mu$, para evitar que se forme una cola muy larga o que apenas se forme cola. Establecemos $\lambda = 1$ (llega, en media, un cliente por minuto) y $\mu = 1'25$ (finalizan el servicio, en media, 1'25 clientes por minuto). También estableceremos la duración de la simulación, es decir, cuánto tiempo vamos a simular el sistema de colas, en este caso, durante 60 minutos.

Asumiremos que en el primer instante de tiempo ($t = 0$) la cola está vacía ($cola$=0). Además, el primer evento tiene que ser la llegada de un cliente. En este caso, la distribución de TE será la correspondiente a los tiempos de llegada, $Exp(1)$. Utilizaremos el código hecho en el capítulo 3 para generar los valores de las distribuciones exponenciales que vamos a usar.

```
lambda=1; mu=1.25; duracion=60; set.seed(330)
t=0; cola=0            # Instante inicial y longitud de la cola en ese instante
suma=0                 # Para calcular la longitud media de la cola

# Primer evento: llegada de un cliente
TE=sim_exp(lambda,1)   # Tiempo hasta que ocurre el evento
tiempo_evento=TE
t=TE; cola_actual=0    # Instante actual
num_eventos=1          # Numero de eventos que han ocurrido

# Sigue adelante la simulacion
while(t<duracion){
  if(cola_actual>0){                # Cola no vacia
    TE=sim_exp(lambda+mu,1)         # Tiempo hasta que ocurre el evento
    cola[num_eventos]=cola_actual   # Longitud de la cola antes de este evento
    p=runif(1)                      # Para decidir el tipo de evento
    cola_actual=ifelse(p<lambda/(lambda+mu), cola_actual+1, cola_actual-1)
  } else {                          # Cola vacia
    TE=sim_exp(lambda,1)            # Siguiente evento: llegada
```

```
    cola[num_eventos]=cola_actual    # Longitud de la cola antes de este evento
    cola_actual=1                    # Longitud actual de la cola
  }
t=t+TE                               # Instante actual
tiempo_evento[num_eventos]=TE        # Tiempo entre eventos
suma=suma+TE*cola[num_eventos]
num_eventos=num_eventos+1            # Numero de eventos que han ocurrido
}
suma/t                               # Longitud media de la cola

## [1] 1.364067
```

Observamos en la salida de R que la longitud media de la cola en el cajero es, aproximadamente, de 1'3641 clientes. En la Figura 4.1 se puede ver una representación de la evolución de la cola respecto al tiempo, obtenida con el siguiente código en R:

```
plot(cumsum(tiempo_evento), cola, type="s", xlab="Tiempo",
     ylab="Longitud de la cola")
```

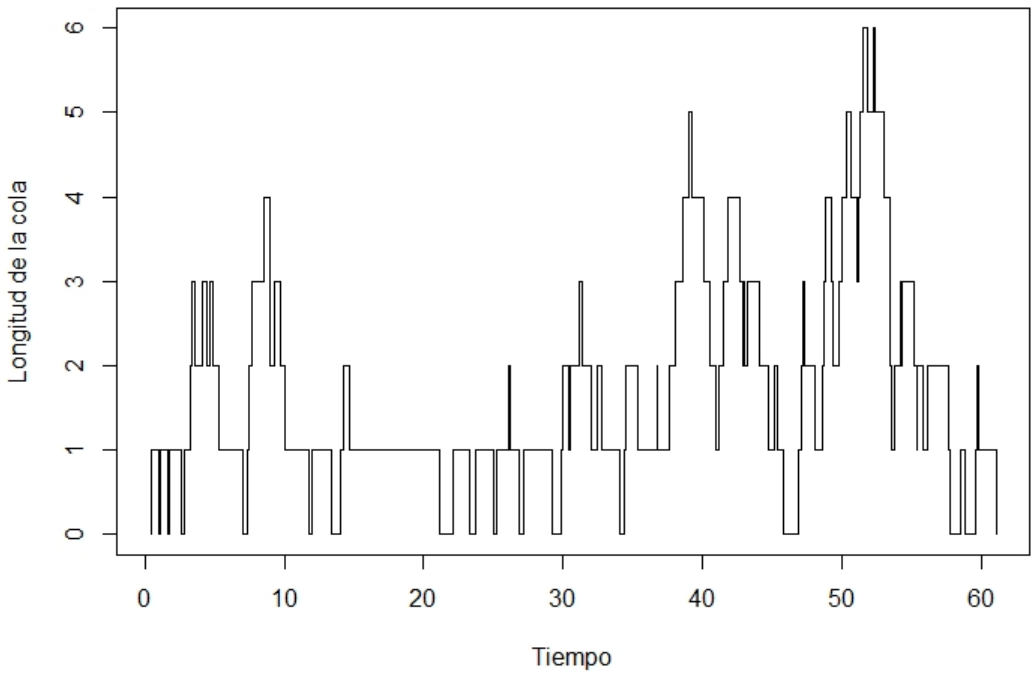

Figura 4.1: *Representación gráfica de la longitud de la cola formada en el cajero respecto al tiempo.*

Con los ejemplos que hemos visto en este capítulo, finalizamos el presente trabajo.

Anexo I

Códigos de *R*

I.1. Código correspondiente a la Figura 2.1

```
par(mar=c(4.2,4.2,1.5,1))
par(mfrow=c(1,2))
x=seq(0,1,length=1000)
y=-4*x^2 + 4*x

# Figura izquierda

plot(x, y, type = "l", xlab="X", ylab="Y", lwd=1, col=1)
polygon(x, y, col = "blue")
lines(x, y, col = 1, lwd = 1)
segments(0, 0, 0, 1, lwd=1, col=1)
segments(0, 1, 1, 1, lwd=1, col=1)
segments(1, 1, 1, 0, lwd=1, col=1)
segments(0, 0, 1, 0, lwd=1, col=1)

# Figura derecha

plot(x, y, type = "l", xlab="X", ylab="Y", lwd=1, col=1)
segments(0, 0, 0, 1, lwd=1, col=1)
segments(0, 1, 1, 1, lwd=1, col=1)
segments(1, 1, 1, 0, lwd=1, col=1)
segments(0, 0, 1, 0, lwd=1, col=1)
```

```
n=80000
set.seed(123); X=runif(n)
set.seed(456); Y=runif(n)
for(i in 1:n){
  if(Y[i]+4*X[i]**2-4*X[i]<0){
    points(X[i],Y[i],pch=20,cex=0.1,col="blue")
  } else {
    points(X[i],Y[i],pch=20,cex=0.1,col="grey")
  }
}
```

I.2. Código correspondiente a la Figura 2.2

```
par(mar=c(4.2,4.2,1.5,1))
par(mfrow=c(1,2))

# Figura izquierda

plot(c(-1,1),c(-1,1), xlab="X", ylab="Y", type="n")
theta <- seq(0, 2 * pi, length = 1000)
polygon(x=cos(theta), y=sin(theta), lwd=1, col="blue", border=1)
segments(-1, -1, -1, 1, lwd=1, col=1)
segments(-1, -1, 1, -1, lwd=1, col=1)
segments(-1, 1, 1, 1, lwd=1, col=1)
segments(1, -1, 1, 1, lwd=1, col=1)

# Figura derecha

plot(c(-1,1),c(-1,1), xlab="X", ylab="Y", type="n")
segments(-1, -1, -1, 1, lwd=1, col=1)
segments(-1, -1, 1, -1, lwd=1, col=1)
segments(-1, 1, 1, 1, lwd=1, col=1)
segments(1, -1, 1, 1, lwd=1, col=1)
polygon(x=cos(theta), y=sin(theta), lwd=1, col="white", border=1)
```

```
n=80000
set.seed(246); u1=runif(n); x=2*u1-1
set.seed(468); u2=runif(n); y=2*u2-1
for(i in 1:n){
  if(x[i]**2+y[i]**2<=1){
    points(x[i],y[i],pch=20,cex=0.1,col="blue")
  } else {
    points(x[i],y[i],pch=20,cex=0.1,col="grey")
  }
}
```

Bibliografía

[1] Cao, R. (2002). *Introducción a la Simulación y a la Teoría de Colas*. NetBiblo.

[2] Cao, R. y Fernández, R. (2022). *Simulación Estadística*. https://rubenfcasal.github.io/simbook/index.html

[3] Casella, G. y Robert, C.P. (2010). *Introducing Monte Carlo Methods with R*. Springer-Verlag.

[4] Dekking, F.M., Kraaikamp, C., Lopuhaä H.P. y Meester, L.E. (2005). *A Modern Introduction to Probability and Statistics: Understanding Why and How*. Springer-Verlag.

[5] García, A. y Vélez, R. (2012). *Principios de Inferencia Estadística*. Universidad Nacional de Educación a Distancia.

[6] Gentle, J.E. (2003). *Random number generation and Monte Carlo methods*. Springer-Verlag.

[7] Hull, T.E. y Dobell, A.R. (1962). Random Number Generators. *SIAM Review*, *4*(3), 230-254. http://www.jstor.org/stable/2027716

[8] Knuth, D.E. (1998). *The Art of Computer Programming: Vol.2. Seminumerical Algorithms*. Addison Wesley Longman.

[9] Kroese, D.P. (2011). *Monte Carlo Methods*. John Wiley & Sons.

[10] Ross, S.M. (2006). *Simulation*. Elsevier.