Simulación Estocástica

Fernando Baltazar-Larios
Sergio Iván López Ortega

16 de enero de 2023

Agradecimientos

Este trabajo se realizó con el apoyo de los proyectos PAPIME de la Dirección General de Asuntos del Personal Académico (DGAPA) UNAM, PE100216 (libro) y PE102618 (versión electrónica y paquetería).

Índice general

Capítulo 1

Introducción a la Simulación Estocástica

En este capítulo se presentan varias definiciones del concepto de simulación estocástica y se propone un algoritmo genérico para el uso adecuado de la simulación estocástica. Además, se discuten las ventajas y limitaciones de esta herramienta. Por último, a manera de motivación, se plantean un par de ejemplos que se podrán resolver con el uso adecuado de la simulación estocástica.

1.1. Introducción

La simulación es una herramienta poderosa ampliamente utilizada en las ciencias actuales para analizar problemas complejos. La idea básica de la simulación es reproducir artificialmente un fenómeno. En este capítulo abordaremos el marco conceptual y el espíritu metodológico de la simulación estocástica.

El inicio de la simulación estocástica se remonta a 1940, cuando John Von Neumann[1] y Stanisław Marcin Ulam [2] trabajando en problemas matemáticos relativos a la física nuclear, cuya solución analítica era intratable y la solución de manera experimental resultaba muy costosa, acuñaron el término "Análisis de Monte Carlo". Este concepto, llamado *Simulación Estocástica* en la actualidad, consiste en resolver un problema determinista simulando computacionalmente a algún proceso estocástico cuyas características probabilistas satisfacen las condiciones matemáticas del problema original. En palabras más simples: con la ayuda de una computadora, se recrean condiciones aleatorias para analizar a una cierta dinámica o fenómeno determinista, y se analizan los resultados computacionales obtenidos.

1.2. Modelación matemática

Debido a que el objetivo de la simulación es recrear computacionalmente a un modelo matemático, tenemos la necesidad de definir o al menos dar una noción de qué es un modelo matemático. Esta es una cuestión profunda en matemáticas y en ciencia en general. No abordamos tal discusión debido a que se aleja de los objetivos de este texto y nos limitamos a da la definición de modelo y modelo matemático, cada definición puntualiza aspectos diferentes de un modelo.

[1] Matemático húngaro (1903-1957) que realizó aportaciones muy diversas con aplicaciones a la física, a la economía y al cómputo, entre otras áreas.

[2] Matemático polaco (1909-1984), célebre por haber contribuido en la invención del *Método Monte Carlo* y participado en el proyecto armamentista *Manhatan*. Contribuyó también en álgebra, análisis y topología.

Definición 1.1. *Un modelo es una representación de una situación o sistema real, que plasma los efectos acción-reacción de los elementos del sistema que se desean investigar.*

Definición 1.2. *Un modelo matemático es una representación matemática de un sistema no matemático.*

En este trabajo nos concentramos en modelos matemáticos que representen de manera adecuada a la realidad utilizando conceptos matemáticos ya existentes (dentro de las matemáticas). Nuestra intención es diseñar un modelo abstracto en donde existan variables (o parámetros) controlables, y que el modelo represente a alguna situación real.

En general, en un modelo matemático se pueden destacar tres fases.

1. La construcción del modelo. Es la traducción y aproximación de acciones en el problema original en conceptos matemáticos adecuados.
2. El análisis del modelo, utilizando herramientas existentes dentro de la matemática.
3. La interpretación de los resultados matemáticos obtenidos por dicho análisis, en el sistema original.

1.3. Definición de simulación estocástica

Definir formalmente al concepto de *Simulación* no es una tarea sencilla. En 1975, R.E. Shannon (ver [26]) la definió como el proceso de diseñar un modelo de un sistema real y realizar experimentos con éste para entender su comportamiento. Una definición formal propuesta en 1963 por C. West Churchman (ver [28]) es la siguiente:

Definición 1.3. *X simula a Y si:*

1. *X e Y son sistemas formales.*
2. *Y se considera un sistema real.*
3. *X es una aproximación de tal sistema real.*
4. *La validez de X no está exenta de error.*

Existen otras definiciones, pero quizá la que goza de mayor aceptación es la de M. Shubik (ver [27]), que es la siguiente.

Definición 1.4. *La simulación de un sistema es la operación de un modelo (simulador), el cuál es una representación del sistema. Este modelo puede someterse a manipulaciones que serían imposibles de realizar, demasiado costosas o poco prácticas para el sistema.*

Es necesario garantizar la eficacia de la simulación estocástica, es decir, que los resultados obtenidos a través de tal procedimiento coinciden o aproximan razonablemente a la dinámica que estamos estudiando. Para ello, se utilizan resultados matemáticos pertenecientes a la teoría de la probabilidad y la estadística; disciplinas que nos sirven para estudiar fenómenos que ocurren bajo condiciones de incertidumbre. Por lo tanto la simulación estocástica consiste en dos etapas:

1. El diseño de un modelo que aproxime adecuadamente al fenómeno en estudio y, por otro lado,

2. la justificación rigurosa (prueba matemática) de que la aproximación realmente funciona.

Sin embargo, desde el punto de vista aplicado, es necesario enfocarse exclusivamente en la primera etapa. Existe ya una batería de resultados teóricos que sustentan con rigor la aplicación de la simulación estocástica a una gran variedad de modelos que describen situaciones en la física, la ingeniería, la actuaría y otras áreas del conocimiento. El objetivo de este libro es presentar resultados teóricos para sistemas en los cuales por lo menos una de sus características es incierta y algunas aplicaciones clásicas, esperando que el lector sea capaz de encontrar nuevas aplicaciones de tales resultados a modelos particulares de su interés.

Respecto al diseño del modelo, hace apenas unas décadas era común que un investigador se enfrentara al dilema entre proponer un modelo muy detallado del fenómeno cuyo análisis matemático fuese prácticamente imposible, o bien, proponer uno que fuera tratable matemáticamente, pero que omitiese aspectos relevantes del problema. Con la llegada de las computadoras, en los últimos años ha sido posible el planteamiento de modelos más complicados y su solución a través de la simulación estocástica. Queremos señalar que, sin embargo, este dilema sigue existiendo: el planteamiento de un modelo demasiado específico se traduce en costo computacional que podría volverse no costeable; el investigador debe ponderar este costo en la elección del modelo.

1.4. Metodología para simular

En esta sección se describen algunos aspectos metodológicos importantes en la simulación estocástica.

Para planificar y hacer uso de un modelo de simulación sugerimos tener en cuenta el siguiente procedimiento:

Algoritmo 1 :Simulación de un sistema

1: Planteamiento del problema.
2: Recolección y organización de la información.
3: Diseño del modelo matemático.
4: Implementación computacional.
5: Validación del programa computacional.
6: Análisis de resultados.
7: Validación de la simulación.

Es importante evitar errores en la implementación de la simulación para el análisis de un problema; a continuación se mencionan algunos de los errores más comunes que pueden cometerse:

- El abuso en la especificidad del modelo diseñado. Esto tiene como consecuencia un costo en el tiempo de ejecución que convierte a la simulación en una herramienta poco práctica. Una solución es plantear un modelo simple para el problema, e ir aumentando paulatinamente los detalles necesarios hasta llegar a un modelo que represente adecuadamente las características esenciales del mismo.

- Presentar una simulación sin calibrar, es decir, realizar una implementación sin tener la certeza de que la simulación realmente representa al problema que se está estudiando.

- Agregar supuestos que no representan al problema aún cuando la simulación produzca los resultados esperados bajo tales supuestos. Esta práctica está alejada de un procedimiento científico y puede traer malinterpretaciones conceptuales del problema.
- Realizar iteraciones computacionales que no son suficientes para garantizar la cercanía de la solución computacional a la solución real. Siempre que sea posible, se debe estimar el error en términos del número de iteraciones y realizar las iteraciones que sean necesarias para garantizar que nuestro error es menor que una constante dada.
- Ejecutar un programa que utiliza siempre un conjunto de sucesiones particulares de números aleatorios; provoca que la simulación no satisfaga las hipótesis estadísticas necesarias para garantizar un error pequeño cuando se tienen muchas iteraciones.
- Por último, que nuestro generador de números aleatorios no esté trabajando de manera adecuada.

En la siguiente sección se enuncian las principales virtudes y limitaciones de la simulación.

1.5. Virtudes y limitaciones de la simulación

Además de las virtudes intrínsecas de la simulación podemos enunciar las siguientes:

- Es posible estudiar el efecto de cambios internos y externos del sistema, haciendo alteraciones en el modelo del sistema y observar los efectos de esas alteraciones en el comportamiento del sistema.
- Puede conducir a un mejor entendimiento del sistema y por consiguiente a sugerir estrategias que mejoren la operación y eficiencia del sistema.
- La técnica de simulación puede ser utilizada para experimentar con nuevas situaciones, sobre las cuales se tiene poca o nula información.
- El modelo es siempre perfectible, lo que permite mejoras en el tiempo computacional requerido para su ejecución.
- Es posible realizar diseño de experimentos para identificar causalidad: puede elegirse un subconjunto de las variables originales de estudio y mantenerse constante, para estudiar la influencia del resto de las variables en el fenómeno de estudio.
- Es posible reproducir un experimento aleatorio en condiciones idénticas tantas veces como sea necesario; esto se logra con el uso de números aleatorios independientes.

En contraparte, es importante puntualizar algunas limitaciones que se tienen en el uso de la simulación:

- En la mayoría de las situaciones no se producen resultados exactos.
- A pesar de permitir exploración del fenómeno en muy diversas situaciones, su desempeño como herramienta de optimización suele ser muy pobre.
- Existen costos inherentes para su uso. Además del tiempo requerido para el diseño teórico del modelo, se requiere tiempo para su traducción (humana) en lenguaje computacional, y finalmente, el costo en tiempo-máquina para obtener los resultados deseados.

- Abuso de la simulación. Puede elegirse erróneamente simular en situaciones en donde obtener resultados teóricos exactos es posible y a menores costos que simular; la decisión de simular un sistema debe ser un último recurso tras evaluar las distintas maneras de resolver el problema de estudio o bien como un complemento exploratorio en el estudio del mismo. Es importante señalar que al reportar los resultados obtenidos por simulación, se debe ser muy específico en el modelo utilizado y las condiciones computacionales específicas. Con ello el lector tendrá una idea de que tan verosímil es que la realidad coincida con los resultados obtenidos y también la posibilidad de replicar la simulación.

1.6. Ejemplos

Finalmente, como motivación, concluimos este capítulo presentando algunos ejemplos específicos que se pueden solucionar adecuadamente con la simulación estocástica.

Ejemplo 1.6.1. ***El problema del viajero****. Se tienen k ciudades que un viajero debe visitar una y sólo una vez. El viaje de ciudad a ciudad tiene un costo (por ejemplo, que puede ser proporcional a la distancia entre ellas) y lo que se busca es una ruta que minimice estos costos.*

Este problema tiene una solución obvia: basta listar todos los posibles recorridos que el viajero puede realizar y elegir el que tenga costo mínimo. No obstante, es fácil verificar que el número total de posibles viajes es $(k-1)!$. La complejidad del problema se hace evidente al observar que, por ejemplo para 51 ciudades tenemos $50! = 3{,}04141 \times 10^{64}$ posibles viajes, y para 101 ocurre que $100! = 9{,}3326 \times 10^{157}$, así que listar todas las posibles trayectorias en estos casos es imposible, aún para las computadoras más poderosas.

En el capitulo 8 nos enfocaremos en resolver este tipo de problemas utilizando simulación estocástica.

Ejemplo 1.6.2. ***Un ejemplo basado en ocurrencia de eventos es el de una cola simple (M/G/1/k)****. Supongamos que queremos modelar la cantidad de personas que hacen fila en un banco. Consideramos llegadas aleatorias (en intervalos de tiempo muy pequeños tenemos una probabilidad p pequeña de que llegue un único cliente, y probabilidad $1-p$ de que no llegue nadie, donde hay independencia entre lo que ocurre en cada intervalo). Consideremos que cada cliente, tiene un tiempo de servicio asociado que consideramos aleatorio con distribución de probabilidades F, porque el tiempo depende de la necesidad de cada cliente. Además, nuestro banco analizado tiene un salón de espera con una capacidad máxima k; esto quiere decir que si un cliente llega al salón de espera y está lleno, decide abandonar el lugar. En el momento en el que un usuario llega, el tiempo promedio de espera que hará debe ser inversamente proporcional a la cantidad de cajeros que están en servicio, y directamente proporcional a la cantidad de personas esperando en el salón en ese instante.*

Una pregunta interesante es, dado que tenemos información estimada de la cantidad de personas que llegan a un banco por día, y los tiempos de servicio necesarios para atenderlos, cuántos cajeros necesitamos tener en operación para que ocurra simultáneamente que:

- *La cantidad de clientes atendidos es la máxima posible.*
- *La proporción del tiempo que cada cajero permanece sin atender a nadie es la mínima posible.*

Para dar una intuición acerca de este problema, notemos que si colocamos un número gigantesco de cajeros en servicio tendremos con mucho probabilidad los clientes esperarán muy poco antes de ser atendidos, pero habrá una gran proporción de tiempo humano desperdiciado (cajeros en espera de que llegue algún cliente por atender). Por el contrario, si mantenemos a un único cajero en servicio y tenemos una sucursal en donde llega una cantidad considerable de usuarios, el cajero se mantendrá ocupado casi todo el tiempo, pero eventualmente el salón de espera se llenará y habrá muchos clientes que no podrán ser atendidos.

Este problema no tiene una solución analítica satisfactoria general. Sin embargo, la simulación estocástica del problema se puede implementar intuitiva y eficazmente lo cual permite realizar análisis numéricos adecuados e incluir variaciones del sistema para adaptarlo a distintas situaciones de la vida real. La rama de las matemáticas que estudia este tipo de sistemas se denomina *Redes Estocásticas*; en esta área existen muchas otras preguntas y fenómenos que se pueden estudiar, además de la pregunta planteada en los párrafos anteriores.

Las técnicas de simulación que veremos en este libro, nos permitirán resolver estos problemas con un esfuerzo de cómputo razonable. La simulación puede aplicarse prácticamente en cualquier ámbito actuarial; a lo largo de este trabajo se presentan ejemplos de aplicaciones en distintas áreas de la Actuaría así como en otras disciplinas.

Capítulo 2

Simulación de números aleatorios

En este capítulo se presentan algunos algoritmos para generar sucesiones de números aleatorios con distribución uniforme en el intervalo $(0, 1)$ (tal distribución será denotada por $U(0, 1)$). Además, se discuten algunas pruebas estadísticas para verificar la aleatoriedad e independencia de las sucesiones generadas. Finalmente, se presenta la manera de utilizar esas muestras aleatorias de números uniformes para generar muestras de cualquier distribución de probabilidad.

2.1. Introducción

La simulación estocástica está basada en la generación de números aleatorios que provienen de diferentes leyes de probabilidad. Sin embargo, como veremos más adelante, basta generar números aleatorios uniformemente distribuidos en el intervalo (0,1) para incorporar la aleatoriedad en esquemas de simulación estocástica para distribuciones arbitrarias. En este sentido, las variables uniformes en (0,1) conforman la base para simular sistemas estocásticos generales, por lo que comenzaremos aprendiendo a generar números aleatorios provenientes de tal distribución.

En el inicio de la simulación, la aleatoriedad se generaba por medio de técnicas manuales, tales como lanzar volados, dados, cartas barajadas, ruletas y urnas con bolas. En aquel entonces se creía que únicamente a través de artefactos mecánicos y/o electrónicos se podía producir verdaderas sucesiones de números aleatorios. Estas sucesiones eran preservadas en *tablas de números aleatorios* que podían ser consultadas cada vez que se querían utilizar tales números.

Actualmente la gran mayoría de los generadores de números aleatorios no dependen de dispositivos físicos sino que están basados en simples algoritmos que pueden ser fácilmente implementados en un ordenador, lo cual ha disminuido considerablemente el costo en la generación de números aleatorios. La discusión, en parte filosófica, de si realmente existe o no el azar ha motivado el desarrollo de la generación de números aleatorios basados en sistemas físicos microscópicos reales. Por ejemplo, utilizando el ruido atmosférico [1] o bien la división de un haz de luz [2]. Tales números pueden ser utilizados también en la simulación estocástica, siempre que satisfagan las hipótesis estadísticas necesarias.

[1] Randomness and Integrity Services Ltd: www.random.org/.
[2] ANU. Quantum Optics: http://photonics.anu.edu.au/qoptics/Research/qrng.php

2.2. Números pseudoaleatorios

Un buen generador de números aleatorios debe tener todas las características estadísticas importantes de las verdaderas sucesiones de números aleatorios, aún si tales sucesiones son generadas mediante un algoritmo determinista. Los valores aleatorios generadas mediante algoritmos deterministas se conocen con el nombre de pseudoaleatorios.

2.2.1. Generadores lineales congruenciales

Los métodos más comunes para generar sucesiones de números aleatorios utilizan los llamados **generadores lineales congruenciales** que han sido propuestos como versiones del generador propuesto por Lehmer en 1951 (ver [17]) que genera una sucesión determinística de números por medio de la fórmula recursiva

$$X_{i+1} = (aX_i + c) \ (mod \ m), \tag{2.1}$$

donde el valor inicial es X_0, conocido como **semilla**, y a, c y m, todos enteros positivos, llamados el **multiplicador**, el **incremento** y el **módulo** respectivamente.

Para el caso especial cuando $c = 0$ la fórmula (2.1) se reduce a

$$X_{i+1} = aX_i \ (mod \ m);$$

y a tal generador se le conoce como generador congruencial multiplicativo.

Nota 2.1. *Cada X_i $(i \geq 1)$ puede tomar valores únicamente en el conjunto $\{0, 1, \ldots, m-1\}$.*

Definición 2.1. *Si definimos sucesiónes de valores $\{U_i\}_{i\geq 1}$ como*

$$U_i := \frac{X_i}{m},$$

*llamadas **números pseudoaleatorios** (debido a que son generadas por un método determinista), entonces (como veremos más adelante), $\{U_i\}_{i\geq 1}$ es una aproximación a una verdadera sucesión de variables aleatorias uniformes.*

Nota 2.2. *Observe que la sucesión de la Definición 2.1 se repetirá después de a lo más m pasos, y que por lo tanto, serán periódicos con período no mayor a m.*

La observación anterior motiva la siguiente definición.

Definición 2.2. *El período de un generador se define como*

$$k := mín\{j \geq 1 | X_i = X_{i+j}\}.$$

Ejemplo 2.2.1. *Sean $a = c = X_0 = 3$ y $m = 5$. Entonces la sucesión obtenida de la fórmula recursiva*

$$X_{i+1} = (3X_i + 3) \ (mod \ 5)$$

es 3, 2, 4, 0, 3, la cual tiene período 4.

Un requerimiento natural es pedir que el generador tenga período m pues en tal caso no caemos en ciclos repetitivos de valores antes de completar el período total de m valores. Es fácil verificar que una elección arbitraria para X_0, a, c y m difícilmente conducirá a una sucesión de números pseudoaleatorios con buenas propiedades estadísticas. De hecho, la teoría de números nos permite ver que sólo unas cuantas combinaciones de estos producen resultados satisfactorios:

Proposición 2.1. *Sea a un natural. Un generador tiene período m si y sólo si se cumple que:*

1. *El máximo común divisor entre c y m es 1.*
2. *$a \equiv 1 \ (mod\ p)$ para todo primo p factor de m.*
3. *$a \equiv 1 \ (mod\ 4)$ en el caso en que m es un múltiplo de 4.*

Este resultado es conocido como el teorema de Hull y Dobell, cuya prueba puede consultarse en [8].

Para las implementaciones en computadora, se elige m como un número primo suficientemente grande que pueda ser soportado por la longitud de una palabra en la computadora. Por ejemplo, en una computadora con longitud de palabra de 32-bits, se obtienen generadores estadísticamente aceptables eligiendo $m = 2^{31} - 1$ y $a = 7^5$, suponiendo que el primer bit es usado para el signo. Un ordenador con longitud de palabra de 64-bits ó 128-bits naturalmente producirá mejores resultados estadísticos. Una elección típica de valores para los parámetros multiplicativos también son $m = 2^{31} - 1$, $a = 7^5$ y $c = 0$. A tal elección se le conoce como el "generador estándar mínimo", que satisface propiedades deseables.

Siguiendo la misma idea, se ha propuesto alternativamente encontrar una sucesión de números pseudoaleatorios realizando operaciones similares al generador congruencial antes visto pero suponiendo que es posible tomar la semilla libremente. Este supuesto permite realizar iteración directa y el uso de números aleatorios en sistemas más sencillos como las hojas de cálculo (y planillas electrónicas en general).

El Generador Congruencial Semilla-Controlado Simple propuesto por Lewis, Goodman y Miller en 1969 (ver [18]) empieza con un valor Z llamado *Zin* que debe estar en el intervalo $(0, 1)$ y representa el número pseudoaleatorio inicial de la sucesión. De esta manera se produce iterativamente la sucesión:

$$U_{i+1} = red(m * a * U_i, 0)(mod\ m)/m, \tag{2.2}$$

donde m y a son enteros positivos y $red(x, y)$ es la función redondear x a y decimales. Por la naturaleza de la fórmula es evidente que esta recursión puede implementarse fácilmente en una hoja de cálculo o en cualquier lenguaje de programación. La función `Gen.Cong.SemCont` (ver B.1.2) implementa el método anterior.

2.2.2. Otros métodos

Distintas versiones se han propuesto en virtud de mejorar los resultados estadísticos obtenidos cuando se analiza el comportamiento de los números obtenidos, pensados como observaciones de variables aleatorias independientes y con distribución uniforme en el intervalo (0,1).

En 1982, Wichmann y Hill propusieron un modelo más particular (ver [30]) que presenta un sistema que utiliza más de una semilla para un número pseudoaleatorio.

En 1988, Park y Miller (ver [21]) propusieron una versión del Generador Congruencial Lineal en el que m es una potencia de un número primo, a debe ser un entero de orden multiplicativo módulo m mayor, $c = 0$ y la semilla un primo relativo de m. De ahí surgen los parámetros típicos antes vistos, que están pensados en máquinas de 32-bits de forma estándar.

En general hay una extensa discusión sobre estos métodos para presentar propiedades muestrales óptimas. L'Ecuyer, por ejemplo, desde 2001 ha desarrollado y probado extensamente, a través de la teoría de módulos, nuevos generadores pensando en los parámetros ideales en un mundo de b-bits de forma más generalizada (ver [15]).

Nota 2.3. *Todos los paquetes para cómputo estadístico y la gran mayoría de los lenguajes de programación tienen ya incluidas rutinas para la generación de números pseudoaleatorios.*

Usualmente, dependiendo del generador, lo único que se solicita al usuario es la semilla inicial, X_0, y al llamar a la función se produce una sucesión de variables aleatorias independientes, provenientes de la distribución $U(0, 1)$.

En el lenguaje de programación R la función que permite generar números de la distribución $U(a, b)$ es `runif(n, a, b)`, donde:

`n` cantidad de números aleatorios que se desea generar

`a` límite inferior de la distribución

`b` límite superior

En particular hacemos `u <- runif(1,0, 1)` para generar un solo valor de $U(0, 1)$.

Hasta ahora hemos propuesto métodos para generar números pseudoaleatorios en el intervalo (0,1), pero es necesario justificar las siguientes dos afirmaciones:

1. Es suficiente poder generar números aleatorios uniformemente distribuidos en el intervalo (0,1) para generar cualquier variable aleatoria con distribución arbitraria.
2. Los números que estamos generando provienen realmente de una sucesión de variables uniformes independientes.

Ambos puntos son abordados en las siguientes secciones.

2.3. Pruebas Estadísticas

Hemos visto que se pueden utilizar algoritmos deterministas que generan números pseudoaleatorios, imitando el comportamiento aleatorio de números aleatorios provenientes de una sucesión de variables alea-

torias con distribución $U(0,1)$. En esta sección, mostraremos como probar estadísticamente si un conjunto de observaciones cualquiera satisface dos supuestos:

1. Son independientes (*Aleatoriedad*).
2. Provienen o no de una distribución $U(0,1)$ (*Bondad de ajuste*).

Esto servirá, en particular, para probar estadísticamente que muestras generadas a través de los algoritmos deterministas de la sección anterior satisfacen tales supuestos. Para dicha tarea haremos uso de algunas pruebas conocidas en la estadistica no paramétrica.

2.3.1. Aleatoriedad

Existen varios procedimientos para probar independencia entre las variables que generaron una muestra dada de números aleatorios. Entre los más usados están algunas pruebas no paramétricas (ver [3]) o el uso de modelos autorregresivos de medias móviles (ver [7]). En el sitio de internet [3] se encuentra un conjunto de pruebas clásicas conformadas en 1995 por G. Marsaglia, titulado *Diehard*. Su uso consiste en someter a una muestra de datos aleatorias a esa batería de test, en donde deben ser todos aprobados en el caso en que los datos realmente provengan de variables aleatorias independientes. Revisaremos en esta subsección una de las pruebas más clásicas.

Prueba de rachas

Esta prueba, también conocida como prueba de Wald-Wolfowitz, es una prueba estadística no paramétrica útil para contrastar la hipótesis de aleatoriedad para una sucesión de datos dicotómicos: {águila, sol}, {0, 1}, {éxito, fracaso}, etcétera. En general, esta prueba es útil también para probar la hipótesis de que los elementos de una sucesión son mutuamente independientes. El juego de hipótesis de interés es:

H_0 : el proceso que genera la muestra **es aleatorio** contra

H_1 : el proceso que genera la muestra **no es aleatorio**.

Para implementar la prueba primero definimos el concepto de racha.

Definición 2.3. *Una racha es una subsucesión de elementos de una sucesión consistente de elementos adyacentes iguales.*

Ejemplo 2.3.1. *La sucesión:*

$$++++--+++--+++++++---+$$

consiste de 7 rachas, 4 de las cuales consisten de $+$ y las restantes de $-$.

[3] https://webhome.phy.duke.edu/ rgb/General/dieharder.php

Sea N^+ el número de ocurrencias de + y N^- la cantidad de −, de manera que $N = N^+ + N^-$. Si los + y los − se alternan aleatoriamente, el número de rachas en la sucesión de N elementos, para N grande, resulta ser una variable aleatoria cuya distribución es aproximadamente normal con media:

$$\mu = \frac{2N^+N^-}{N} + 1,$$

y varianza

$$\sigma^2 = \frac{2N^+N^-(2N^+N^- - N)}{N^2(N-1)} = \frac{(\mu-1)(\mu-2)}{N-1}.$$

Estos parámetros no dependen de la probabilidad de obtener + o la probabilidad (complementaria) de obtener −, bajo el supuesto que los elementos son independientes e idénticamente distribuidos. Si la cantidad de rachas (r) es significativamente mayor o menor de lo esperado la hipótesis de independencia estadística de los elementos puede ser rechazada.

Entonces, bajo H_0

$$z = \frac{r - \mu}{\sigma}$$

se rechaza la hipótesis nula al nivel α si $z < Z_{\alpha/2}$, o bien, $z > Z_{1-\alpha/2}$, donde $Z_{\alpha/2}$ son los cuantiles correspondientes una variable normal estándar.

La función **wawotest** (de **R**), realiza la prueba de Wald-Wolfowitz para la aleatoriedad de un vector de valores. Consideremos la asociación $0 \equiv -$, y $+ \equiv 1$ para tener valores numéricos sencillos de capturar en una computadora. Entonces, la estadística de la prueba es

$$A = \sum_{i=1}^{n} y_i y_{i+1},$$

donde $y_{n+1} = y_1$ y n es el tamaño del vector. Bajo la hipótesis de aleatoriedad de los valores en el vector, esta estadística se distribuye normal con la media y varianza teóricas mencionadas anteriormente.

Notemos que cuando estamos trabajando con datos en el intervalo (0,1) necesitamos una manera de convertir tales datos a números binarios en $\{0, 1\}$. Un procedimiento simple es convertir el dato u al valor 0 cuando es menor que 0.5 y al valor 1 en otro caso. El código B.1.3 convierte los datos uniformes en (0,1) a binarios $\{0, 1\}$.

La Tabla 2.1 presenta el resultado de realizar una prueba de rachas sobre una trayectoria generada de manera aleatoria de ceros o unos utilziando la función **wawotest** (ver el código B.1.3) y, como se espera, el valor obtenido para la variable "estadística normalizada" está entre los cuantiles de la distribución normal estándar. Otro dato importante obtenido a través del test es el `p-valor`, que es el máximo valor

Valor de la prueba	Esperanza	Varianza	Estadística normalizada	p-valor
15.1582474	-1.0000000	10497.9996794	0.1577034	0.4373453

Tabla 2.1: Resultado de una prueba de aleatoriedad Wald-Wolfowitz para $n = 1,000$.

de confianza con el cuál podemos afirmar que los datos cumplen la hipótesis de aleatoriedad.

2.3.2. Bondad de ajuste.

Es importante mencionar que las pruebas de bondad de ajuste suponen que las observaciones son independientes, por lo que primero deberá verificarse si éstas pueden ser consideradas independientes (problema que abordamos en la subsección anterior).

Un método para probar si una muestra proviene de la distribución $U(0,1)$ consiste en probar la hipótesis

$$H_0 : \{x_1, \ldots, x_n\} \text{ es una muestra aleatoria de } U(0,1),$$

contra la alternativa de que

$$H_1 : \{x_1, \ldots, x_n\} \text{ es una muestra con una distribución diferente a la } U(0,1).$$

Existen diferentes aproximaciones para tales hipótesis, pero una muy recomendable, en función de su potencia, es la estadística Anderson-Darling (ver [29]). Dicha estadística está dada por

$$A_n^2 = n \int_0^1 \frac{(G_n(t) - t)^2}{t(1-t)} dt,$$

donde

$$G_n(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{(-\infty, t]}(x_i)$$

es la función de distribución empírica de la muestra $\{x_1, \ldots, x_n\}$.

Sea $\{x_1, \ldots, x_n\}$ la muestra ordenada, es decir, ocurre que $x_i \leq x_j$ para cualquier par de números $i \leq j$. El estadístico A_n^2 determina si los datos $x_1 \leq \cdots \leq x_n$ provienen de una distribución con función acumulativa F:

$$A_n^2 = -n - \sum_{k=1}^{n} \frac{2k-1}{n} \left[\ln F(x_k) + \ln \left(1 - F(x_{n+1-k}) \right) \right].$$

El estadístico de la prueba se puede comparar entonces contra las distribuciones del estadístico de prueba, dependiendo de que F es utilizada para determinar el `p-valor`.

En el caso de la distribución $U(0,1)$ el estadístico de prueba se reduce a

$$A_n^2 = -n - \sum_{k=1}^{n} \frac{2k-1}{n} \left[\ln(x_k) + \ln(1 - x_{n+1-k}) \right] \text{ para } x_k \in (0,1).$$

		$1-\alpha$			
Caso		0.9	0.95	0.975	0.99
Todos los parámetros desconocidos	A_n^2 para $n \geq 5$	1.933	2.492	3.07	3.99

Tabla 2.2: Cuantiles para la prueba de Anderson-Darling.

Algunos cuantiles de interés para realizar la prueba se presentan en la Tabla 2.2.

A continuación se presentan ejemplos sobre cómo realizar la prueba de Anderson-Darling en **R** con la función **ad.test** (ver B.1.3). En nuestro primer ejemplo se realiza sobre una muestra simulada por la función `runif` de **R**. En el segundo ejemplo se realiza una prueba similar para el caso en que la muestra fue generada a través del generador congruencial multiplicativo. En ambos casos la hipótesis nula es aceptada. En la Tabla 2.3 se presentan los resultados obtenidos.

Generador	Valor de la prueba	p-valor
runif	0.2853	0.9487
Gen.Lin.Cong	0.66784	0.5859

Tabla 2.3: Resultado de una prueba de bondad de ajuste Anderson-Darling para $n = 100$.

Para terminar esta sección, presentamos la comparación del Generador Congruencial Lineal y el Generador Congruencial de Semilla-Controlado bajo el criterio del p-valor en la prueba de Anderson-Darling a muestras de diferente tamaño n. Se eligió $m = 2^{31} - 1$, $a = 7^5$ para ambos métodos y $c = 1$ para el Generador Congruencial Lineal. La diferencia entre los dos se refleja en que el período de la sucesión se ve modificado al sumar c distinto de cero.

En la Tabla 2.4 se reportan los p-valores de la prueba Anderson-Darling para diferentes valores de n. Concluimos que, por lo que en general, un generador con parámetros más completos da mejores resultados estadísticos.

	n	LinCong	CongSC
1	3	0.000150	0.094056
2	4	0.000120	0.118201
3	5	0.000100	0.157030
4	6	0.000086	0.029503
5	7	0.000075	0.553804
6	8	0.000067	0.821272
7	9	0.000060	0.112792
8	10	0.000055	0.016056

Tabla 2.4: p-valores de la prueba Anderson-Darling

La Figura 2.1 muestra la comparación cuantil-cuantil de ambos generadores contra la distribución teórica.

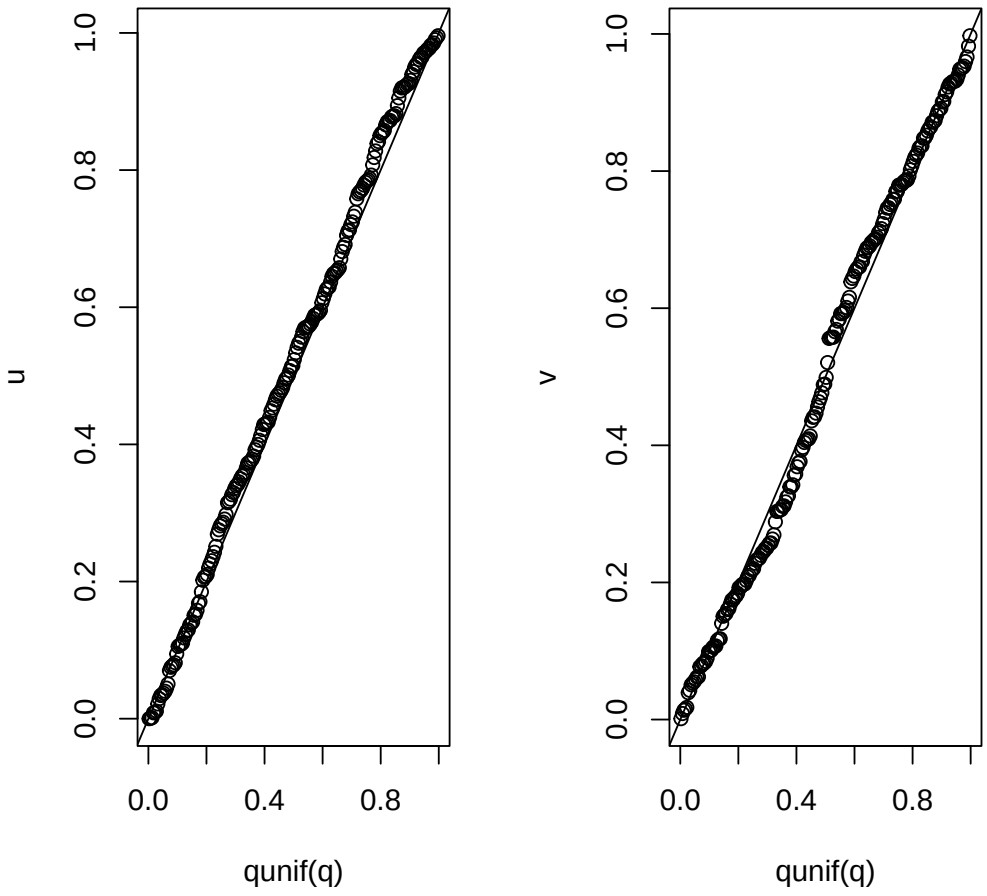


Figura 2.1: Comparación cuantil a cuantil del generador lineal congruencial (u) y el congruencial de semilla controlada (v) contra los valores teóricos q.

2.4. Simulación de otra distribución a partir de la distribución uniforme

En esta sección mostraremos teóricamente que, a partir de variables aleatorias uniformes en el intervalo $(0,1)$, podemos generar variables aleatorias con una distribución arbitraria dada. La implementación de tal algoritmo se puede dar en algunos casos prácticos, aunque no en todos. Lo presentamos aquí para mostrar la gran importancia de ser capaces de generar números pseudoaleatorios provenientes de $U(0,1)$. Será hasta el próximo capítulo que presentaremos una batería de técnicas para generar números aleatorios con una distribución dada.

Supongamos que queremos simular a una variable aleatoria X, con función de distribución F. Por las propiedades de las funciones de distribución, F es un mapeo de $\mathbb{R}$ al intervalo $(0,1)$. Notemos que en el caso en que X es una variable aleatoria continua (es decir su función de distribución F es continua) dicho mapeo es inyectivo y como consecuencia su función inversa mapea el intervalo $(0,1)$ a $\mathbb{R}$. Definamos a la variable

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

donde U es una variable aleatoria uniforme en $(0,1)$. Ocurre que

$$F_{\tilde{X}}(t) = \mathbb{P}(\tilde{X} \leq t) = \mathbb{P}(U \leq F(t)) = F(t),$$

pues la función de distribución de la uniforme U es igual a la identidad cuando toma valores entre 0 y 1. Eso prueba que $\tilde{X}$ tiene la distribución F, es decir que tiene la misma distribución X. Concluimos que para obtener un número aleatorio proveniente de una variable aleatoria continua X es suficiente obtener un número aleatorio proveniente de una variable uniforme U y aplicarle la función F^{-1} a tal valor.

La prueba se puede adaptar a variables aleatorias generales, es decir, sin utilizar el supuesto de que X es continua. Damos primero una definición necesaria.

Definición 2.4. *La inversa generalizada de una función de distribución F está dada por*

$$F^{-}(y) = \text{ínf}\{x \in \mathbb{R} : F(x) \geq y\},$$

para todo y en $(0,1)$.

Notemos que en el caso particular en que la función F es continua, F^{-} coincide con F^{-1}; es por ello que es llamada inversa generalizada.

Lema 2.1. *Sea F la función de distribución de una variable aleatoria X y U una variable aleatoria con distribución $U(0,1)$. Definamos*

$$\tilde{X} := F^{-}(U),$$

entonces $\tilde{X}$ tiene la misma distribución que X.

Demostración. Para un real fijo t probaremos la igualdad de eventos:

$$\{\tilde{X} \leq t\} = \{U \leq F(t)\}. \tag{2.3}$$

$\subseteq$) Supongamos que $\tilde{X} \leq t$, lo que es equivalente por la definición de $\tilde{X}$ a que $F^{-}(U) \leq t$. Debido a la definición de F^{-} y a que F es creciente ocurre que para toda $\varepsilon > 0$

$$U \leq F(t + \varepsilon),$$

y por la continuidad por la derecha de la función F se concluye que $U \leq F(t)$.

$\supseteq$) En el evento $\{U \leq F(t)\}$ tenemos que $t \in \{x \in \mathbb{R} : F(x) \geq U\}$ lo que implica que

$$t \geq \text{ínf}\{x \in \mathbb{R} : F(x) \geq U\} = F^{-}(U) = \tilde{X}.$$

Usando la igualdad de eventos de la Ecuación (2.3) tenemos que

$$\mathbb{P}(\tilde{X} \leq t) = \mathbb{P}(U \leq F(t)) = F(X),$$

lo que prueba que $\tilde{X}$ tiene la misma distribución que X.

□

El lema anterior puede aplicarse en particular a variables aleatorias discretas. En este caso, la función de distribución no es invertible, pero sabemos que el conjunto de discontinuidades de F, que es el soporte de la variable aleatoria, es a lo más numerable. Tales puntos definen a la inversa generalizada de la función y nos permiten aplicar el lema anterior. Veremos en el Capítulo 3 aplicaciones concretas del resultado anterior.

2.5. Ejercicios

1. Pobrar la Proposición 2.1.

2. Implementar en R un generador de números pseudoaleatorios utilizando el siguiente algoritmo, propuesto por Wichman & Hill (1982):

   a) Dar x, y, z enteros mayores a cero y menores a 30,000.

   b) Calcular

   $$\begin{aligned} x &= 171x \; mod[177] - 2x/177, \\ y &= 172y \; mod[176] - 35y/176, \\ z &= 170z \; mod[178] - 63x/178. \end{aligned}$$

   c) Evaluar:

   Si $x \leq 0$ entonces $x = x + 30269$,
   Si $y \leq 0$ entonces $y = y + 30307$,
   Si $z \leq 0$ entonces $z = z + 30323$.

   d) Definir $u = \left(\frac{x}{30269} + \frac{y}{30307} + \frac{z}{30323}\right) \; mod[1]$.

   Dependiendo del equipo de cómputo, u podría resultar 0 ó 1 en alguna iteración; en este caso, redefinimos a u como $u \pm eps$, en donde eps es la precisión de la computadora. Aplicar el test de rachas y el de bondad de ajuste vistos a una muestra de números aleatorios generados y concluir si el generador es confiable o no, bajo tales criterios.

3. Diseñar un test estadístico para saber si un conjunto de números $\{u_1, ..., u_n\}$ provienen de una distribución uniforme continua en $(0, 1)$, utilizando la distribución $\chi^2(k)$ para algún k, bajo el supuesto de independencia de tales números. Recomendamos no utilizar librerías para pruebas estadísticas, sino utilizar las funciones de distribución F y de cuantiles F^{-1} asociadas a una variable aleatoria $\chi^2(k)$.

a) Justificar porqué funciona, con teoría estadística.

b) Proponer un estadístico de prueba, pruebas de hipótesis y región de rechazo.

c) Programar el test en R (se pueden utilizar la función de distribución y su inversa de una variable aleatoria (v.a.) χ^2).

d) Hacer el test para datos generados por el generador de R, el generador lineal congruencial, y el generador de semilla continua (experimentar con distintas constantes aditivas c).

Capítulo 3

Simulación de Variables Aleatorias

En este capítulo se presentan los principales métodos para simular valores de variables aleatorias. Además se presentan algoritmos para generar valores de vectores aleatorios con componentes dependientes, en particular un algoritmo para generar cópulas.

3.1. Método de la Transformada Inversa

Tal como vimos en el capítulo anterior, para generar la aleatoriedad requerida será suficiente generar variables aleatorias uniformes en el intervalo $(0, 1)$. Supongamos que queremos simular a una variable aleatoria X, con función de distribución F. Utilizando la función de distribución inversa (Definición 2.4), tenemos el siguiente método para generar números provenientes de la distribución F.

Algoritmo 2 : Transformada Inversa

1: Generar un número aleatorio $U \sim U(0, 1)$.
2: Tomar $X = F^{-}(U)$.

La validez de este algoritmo está sustentada por el Lema 2.1.

En las próximas secciones haremos una especialización a los casos en que la variable aleatoria es continua, es decir, cuando su función de distribución es continua; o discreta, donde ocurre que la variable tiene soporte finito o numerable.

3.1.1. Variables Aleatorias Continuas.

Para el caso de variables aleatorias continuas, en el Algoritmo 2 la inversa generalizada se reduce a la inversa de la función de distribución F. Entonces, basta con encontrar la inversa de la función de distribución de la variable aleatoria que se quiere simular para aplicar este algoritmo.

Ilustraremos el algoritmo con algunos ejemplos donde encontrar la inversa es tarea fácil pero es importante mencionar que esto no siempre es así.

Ejemplo 3.1.1. ***Distribución Exponencial.*** *Supongamos que queremos generar números aleatorios de la distribución exponencial con parámetro $\lambda > 0$, es decir, la función de distribución está dada por:*

$$F(t) = \begin{cases} 0 & si\ t < 0 \\ 1 - e^{-\lambda t} & si\ t \geq 0. \end{cases}$$

Es trivial mostrar que $F^{-1}(t) = -\frac{1}{\lambda}\log(1-t)$. Entonces, si definimos a la variable aleatoria

$$X := F^{-}(U) = -\frac{1}{\lambda}\log(1-U),$$

ésta tendrá una distribución exponencial con parámetro λ si $U \sim U(0,1)$. Una pequeña observación es que en este caso $1-U \sim U(0,1)$, así que la variable aleatoria

$$\tilde{X} := -\frac{1}{\lambda}\log(U),$$

también tiene una distribución exponencial con parámetro λ, y un costo computacional ligeramente menor, ya que redujimos una operación en la definición de $\tilde{X}$. Entonces el Algoritmo 2, en este ejemplo, trabaja de la siguiente forma.

Algoritmo 3 : Transformada Inversa para distribución exponencial

1: Generar un número aleatorio $U \sim U(0,1)$.
2: Tomar $X = -\frac{1}{\lambda}\log(U)$.

Una implementación en R de este algoritmo es la función `G.exp.inv` (ver B.2.2), que genera observaciones de variables aleatorias exponenciales. En **R** podemos utilizar la función interna **rexp(n,**λ) para generar números de la distribución exponencial, donde n es el tamaño de la muestra.

Ejemplo 3.1.2. ***Estadísticos de Orden.*** *Sean $X_1, X_2, \ldots, X_n$ variables aleatorias independientes e identicamente distribuidas (v.a.i.i.d.) con función de distribución F. Queremos generar al primer estadístico de orden,*

$$X_{(1)} = \min_{1 \leq i \leq n} X_i,$$

y al último,

$$X_{(n)} = \max_{1 \leq i \leq n} X_i.$$

Es fácil probar que tales variables tienen a $F_1(x) = 1-[1-F(x)]^n$ y $F_n(x) = [F(x)]^n$ como sus respectivas funciones de distribución. Usando el Algoritmo 2 en estos casos obtenemos que las variables

$$\begin{aligned} Y_1 &:= F^{-}(1-U_1^{1/n}), \\ Y_n &:= F^{-}(U_2^{1/n}), \end{aligned}$$

tienen la misma distribución que $X_{(1)}$ y $X_{(n)}$ respectivamente, donde $U_1, U_2 \sim U(0,1)$.

Notemos que la eficiencia del método de la transformación inversa en los ejemplos anteriores es muy alta, por la relación explícita existente entre la variable uniforme y la variable que se quiere generar. Sin embargo, aún en el caso en que la función de distribución inversa exista, esta relación explícita puede no existir. Por ejemplo, en el caso en que X distribuye normal estándar, la función inversa de la distribución $\Phi(x)^{-1}$ existe, pero no existe una expresión analítica explícita para tal función, por lo que la implementación del Algoritmo 2 sería muy poco práctica.

3.1.2. Variables Aleatorias Discretas.

Nos enfocaremos ahora en variables aleatorias con soporte en un conjunto a lo más numerable.

Sea X una variable aleatoria discreta con soporte $S_X = \{x_1, x_2, \ldots\}$ y función de densidad $P(X = x_i) = p_i$ donde $x_1 < x_2 < \ldots$ para $i = 1, 2, \ldots$. Si definimos $\tilde{X} := F^-(U)$, donde F es la distribución de X y U una variable aleatoria uniforme en $(0, 1)$, ocurre que

$$P(\tilde{X} = x_i) = P\left(\sum_{k=1}^{i-1} p_k \leq U < \sum_{k=1}^{i} p_k\right) = p_i,$$

entonces el Algoritmo 2 se reduce en este caso a lo siguiente:

Algoritmo 4 : Transformada inversa caso discreto

1: Generar $U \sim U(0, 1)$.
2: Tomar

$$X = \begin{cases} x_1 & \text{si } U < p_1 \\ x_2 & \text{si } p_1 \leq U < p_1 + p_2 \\ \vdots & \\ x_i & \text{si } \sum_{k=1}^{i-1} p_k \leq U < \sum_{k=1}^{i} p_k \\ \vdots & \end{cases}$$

Una implementación del algoritmo anterior es la función `rTInv` (ver B.2.1).

Ejemplo 3.1.3. ***Distribución uniforme discreta.*** *Supongamos que queremos generar números aleatorios de una variable aleatoria $X \sim U\{1, 2, \ldots, n\}$, es decir, $P(X = i) = 1/n$ para $i = 1, ..., n$.*

El paso 2 del Algoritmo 4 toma la siguiente forma específica:

$$X = i \quad \text{si} \quad \frac{i-1}{n} \leq U < \frac{i}{n}, \quad i = 1, ..., n,$$

donde $U \sim U(0, 1)$.

Ejemplo 3.1.4. ***Distribución Bernoulli.*** *Para generar una v.a. $X \sim$ Bernoulli(p) con $p \in (0, 1)$, generamos un número aleatorio proveniente de $U \sim U(0, 1)$ y hacemos*

$$X = \begin{cases} 1 & si\ U \leq p, \\ 0 & si\ U > p. \end{cases}$$

Ejemplo 3.1.5. ***Distribución geométrica.*** *Queremos simular X cuando tiene función de densidad $P(X = i) = p(1 - p)^{i-1}$, $i = 1, 2, \ldots$, para $p \in (0, 1)$.*

En este caso es directo probar que

$$P(X \leq i - 1) = \sum_{k=1}^{i-1} P(X = k) = 1 - (1 - p)^{i-1}, \quad i = 1, 2, ...,$$

y el paso 2 del Algoritmo 4 adquiere la forma siguiente: Se elige $X = i$ cuando ocurre que

$$1 - (1-p)^{i-1} \leq U < 1 - (1-p)^{i}.$$

En **R**, podemos utilizar la función interna **rgeom(n,p)** para generar números provenientes de una variable aleatoria con distribución geométrica de parámetro p.

Nota 3.1. *Una manera alternativa de generar $X \sim Geo(p)$ es generar variables aleatorias independientes Bernoulli del mismo parámetro hasta obtener un éxito. La variable resultado será el número de lanzamientos que fue requerido para obtener tal éxito.*

3.1.3. Variables discretas con probabilidades iterativas

Un característica interesante de las siguientes distribuciones es la posibilidad de representar y calcular la probabilidad de tomar el i-ésimo valor, denotada por p_i, en términos de la probabilidad de tomar el valor anterior, es decir, en terminos de p_{i-1}.

Este resultado es utilizado, por ejemplo, para calcular las fórmulas de De Pril y Panger en Teoría del Riesgo. Puede demostrarse que las únicas distribuciones que cumplen tal propiedad son la Binomial, Poisson y Binomial Negativa.

Ejemplo 3.1.6. ***Distribución Binomial.*** *Para generar valores de una variable aleatoria $X \sim Binomial(n,p)$, utilizaremos la siguiente identidad recursiva:*

$$p_{i+1} = \left(\frac{n-i}{i+1}\right)\left(\frac{p}{1-p}\right)p_i.$$

Haciendo $F = F(i) = P(X \leq i)$ y $pr = p_i$ tenemos el siguiente algoritmo.

Algoritmo 5 : Distribución Binomial

1: Generar $U \sim U(0,1)$.
2: Definir $c = \frac{p}{1-p}$, $i = 0, pr = (1-p)^n, F = pr$.
3: Si $U < F$, hacemos $X = i$ y paramos.
4: $pr = c(n-i)/(i+1)pr$, $F = F + pr$, $i = i + 1$.
5: Volver al paso 3.

En **R** usamos **rbinom(muestra,n,p)**.

Nota 3.2. *Una manera alternativa de generar $X \sim Binomial(n,p)$ es generar primero n v.a.i.i.d. $Y_i \sim Bernoulli(p)$ y hacer $X = \sum_{i=1}^{n} Y_i$.*

Ejemplo 3.1.7. ***Distribución Poisson.*** *Si $X \sim Poisson(\lambda)$ con $\lambda > 0$, usando la identidad*

$$p_{i+1} = \left(\frac{\lambda}{i+1}\right)p_i,$$

y haciendo $p = p_i$ y $F = F(i)$, podemos simular valores de X con el siguiente algoritmo.

Algoritmo 6 : Distribución Poisson

1: Generar $U \sim U(0,1)$.
2: Definir $i = 0$, $p = e^{-\lambda}$, $F = p$.
3: Si $U < F$, hacemos $X = i$ y paramos.
4: $p = \lambda p/(i+1)$, $F = F + p$, $i = i + 1$.
5: Volver al paso 3.

Podemos ver una implementación de este algoritmo en nuestra función `G.poi.in` (ver B.2.3). Alternativamente, tenemos la función interna en **R** dada por **rpois(n,λ)**. En la Figura 3.1 se puede apreciar la comparación de muestras generadas por cada una de estas funciones.

3.1.4. Distribuciones discretas con esperanza grande.

Respecto a la eficiencia del Algoritmo 2 en el caso de variables aleatorias discretas es importante tener presente las siguientes observaciones:

1. Si $X = x_i$ fue necesario realizar i comparaciones. Entonces, el número esperado de iteraciones del Algoritmo2 es:

$$\sum_{i=1}^{\infty} i p_i,$$

   lo que puede ser muy ineficiente si la cardinalidad del soporte es muy grande o infinito.

2. El Algoritmo 2 comienza a hacer tales comparaciones empezando siempre por el menor valor que toma la variable aleatoria, sigue con el segundo valor más pequeño y así sucesivamente. Esto puede ser muy ineficiente para algunas distribuciones particulares.

En el caso en que conocemos algunas de las características de la distribución (esperanza, simetría, sesgo, etc.), es natural pensar que podemos utilizar esa información para modificar al Algoritmo 2 y obtener un algoritmo más eficiente. En esta sección nos focalizaremos en el caso de variables aleatorias discretas con valor esperado muy grande.

La idea general de la modificación es empezar a hacer comparaciones a partir del valor de la media $\mu := \mathbb{E}[X]$ y decidir si continuar hacia valores menores o bien continuar hacia valores mayores que μ, dependiendo del resultado de la primera comparación. De manera específica: nuestra primera comparación será entre la probabilidad acumulada hasta el valor μ y una variable aleatoria uniforme en $(0,1)$. Debido a que el valor esperado podría ser no entero, utilizamos el mayor de los enteros que están antes del valor μ, denotado por $\lfloor \mu \rfloor$. El siguiente algoritmo presenta la modificación mencionada.

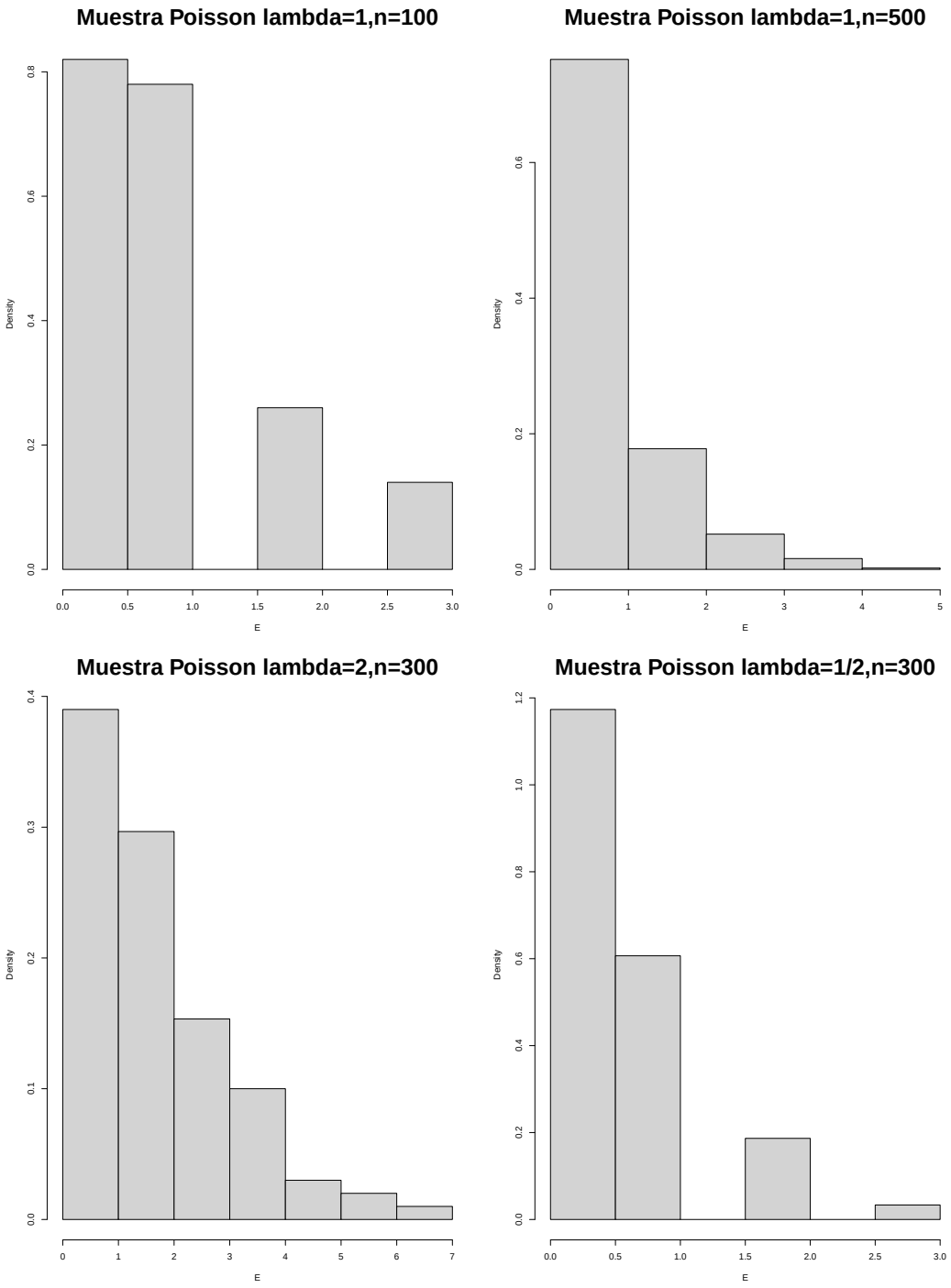


Figura 3.1: comparación cuantil a cuantil muestras de Poisson usando `G.poi.in` y **rpois**.

Algoritmo 7 : Transformada inversa caso discreto con esperanza grande

1: Generar $U \sim U(0,1)$.
2: Si $U < F(\lfloor \mu \rfloor)$, tomar

$$X = \begin{cases} x_1 & \text{si } U < p_1 \\ x_2 & \text{si } p_1 \leq U < p_1 + p_2 \\ \vdots & \\ x_{\lfloor \mu \rfloor} & \text{si } \sum_{k=1}^{\lfloor \mu \rfloor - 1} p_k \leq U < \sum_{k=1}^{\lfloor \mu \rfloor} p_k \end{cases}$$

en otro caso, tomar

$$X = \begin{cases} x_{\lfloor \mu \rfloor + 1} & \text{si } \sum_{k=1}^{\lfloor \mu \rfloor} p_k \leq U < \sum_{k=1}^{\lfloor \mu \rfloor + 1} p_k \\ x_{\lfloor \mu \rfloor + 2} & \text{si } \sum_{k=1}^{\lfloor \mu \rfloor + 1} p_k \leq U < \sum_{k=1}^{\lfloor \mu \rfloor + 2} p_k \\ \vdots & \end{cases}$$

Ejemplo 3.1.8. ***Distribución Poisson (λ grande)***. *Para la distribución de Poisson, el valor esperado coincide con el parámetro λ de la distribución, por lo que será necesario considerar a las probabilidades de que la variable tome valores alrededor de este valor. La función* `G.poi.mod` *(ver B.2.4) implementa el algoritmo modificado 7.*

Mostraremos ahora cuánto mejora el desempeño al hacer la modificación. En la Tabla 3.1, evaluamos la eficiencia (en número de iteraciones necesarias) variando el parámetro λ. Fueron generados 1000 números aleatorios con distribución Poisson, utilizando $\lambda = 1, 10$ y 100, con la función `G.poi.inv` (ver B.2.3).

	Lambda	Iteraciones	Promedio
1	1	1044	1.044000
2	10	10049	10.049000
3	100	100259	100.259000

Tabla 3.1: Eficiencia al simular 1000 valores de una variable aleatoria Poisson con diferentes medias.

Ahora, resumimos en la Tabla 3.2 la comparación de los algoritmos `G.poi.inv` y `G.poi.mod` cuando tenemos fijo el parámetro ($\lambda = 50$) y variamos el tamaño de muestra $n = 10, 100$ y 1000. Evaluamos el promedio de iteraciones realizadas en cada caso.

	N	Iteraciones_Inv	Promedio_Inv	Iteraciones_Mod	Promedio_Mod
1	10	487	48.700000	63	6.300000
2	100	4883	48.830000	473	4.730000
3	1000	50054	50.054000	5052	5.052000

Tabla 3.2: Compara la eficiencia del algoritmo de la transformada inversa y el modificado al simular n variables aleatorias Poisson(50)

Los resultados nos muestran que pueden tenerse algoritmos mucho más eficientes, cuando consideramos propiedades específicas de la función de densidad. El ejemplo de la distribución Poisson, nos mostró que entre más grande sea el parámetro λ, más frecuente será el número de iteraciones alrededor de este valor por coincidir con la media de la distribución. Esto lo ilustramos en la Figura 3.2 donde se presentan gráficas de la función de densidad de una variable aleatoria Poisson con diferentes valores de λ. La función

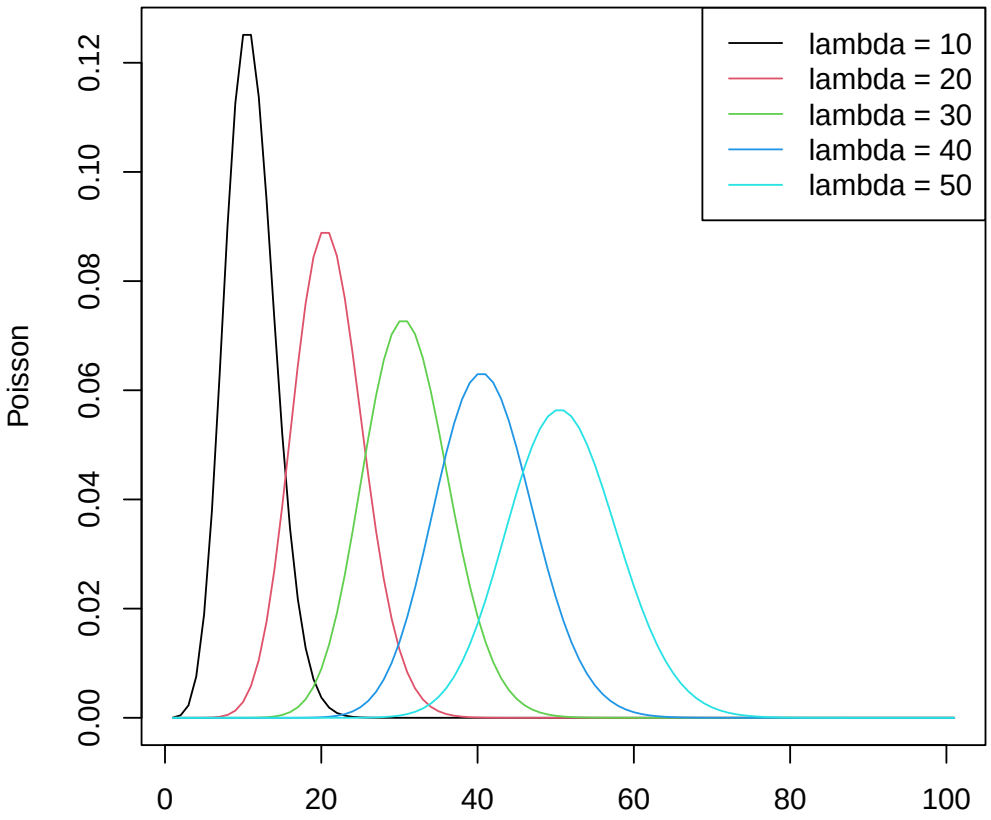


Figura 3.2: Función de densidad de variables aleatorias Poisson.

de densidad está extendida naturalmente a todos los valores positivos, utilizando la función gamma (esto es sólo para fines visuales; la densidad Poisson tiene soporte exclusivamente en los enteros no negativos).

3.2. Método de aceptación y rechazo.

El método de simulación descrito en esta sección sirve para obtener muestras de variables aleatorias de cualquier distribución. Fue propuesto por Stan Ulam y John von Neumann (ver [5]), aunque existe el antecedente de Buffon y su famoso problema de la aguja (ver [1]). La idea de dicho método consiste en muestrear variables aleatorias de una distribución apropiada y someterlas a una prueba para ser o no aceptadas como muestra proveniente de otra distribución.

Supongamos que tenemos una variable aleatoria X con función de densidad f con soporte en el intervalo

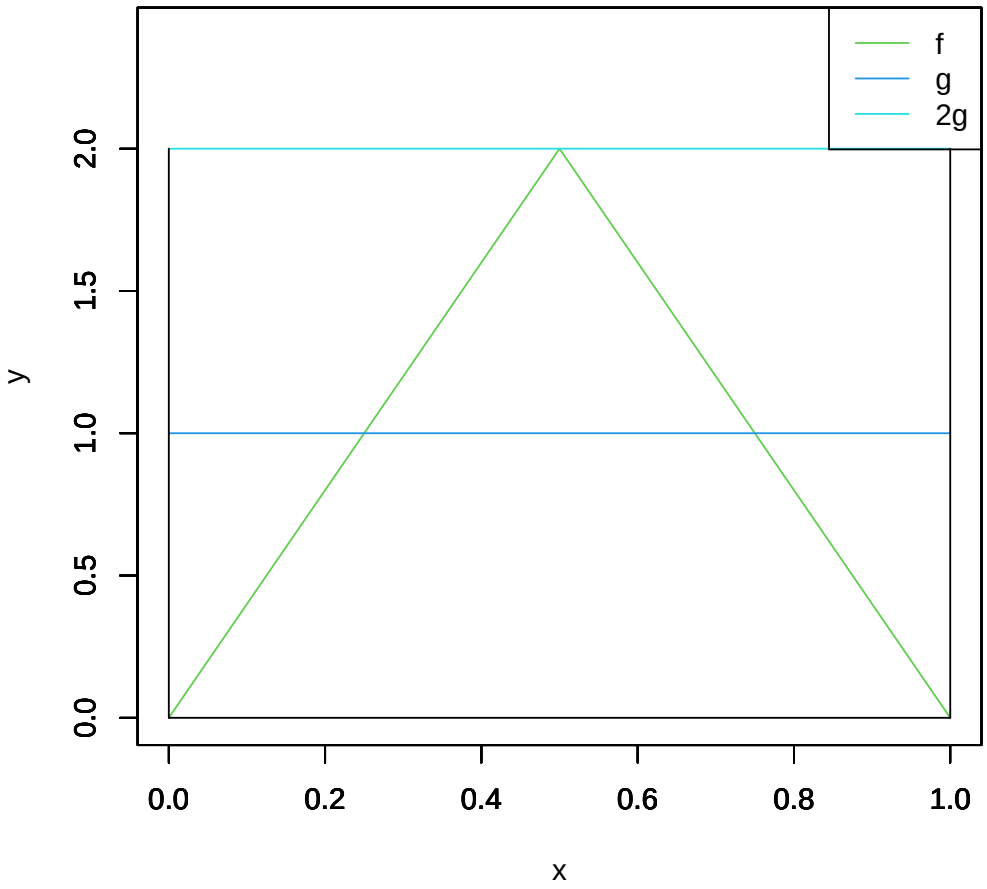


Figura 3.3: f es una distribución triangular, g una distribución uniforme en $[0, 1]$

$[a, b]$. Si definimos

$$c := sup\left\{f(x) : x \in [a, b]\right\},$$

entonces, para simular valores de X, podemos usar los siguientes pasos de aceptación y rechazo:

1. Generamos $Y \sim U(a, b)$.
2. Generamos $Z \sim U(0, c)$ independientemente de Y.
3. Si $Z \leq f(Y)$, hacemos $X = Y$. En otro caso, regresamos al paso 1.

Nota 3.3. *Dado que el vector (Y, Z) está uniformemente distribuido sobre $[a, b] \times [0, c]$, ocurre que una pareja aceptada (Y, Z), está uniformemente distribuida bajo la curva de f, teniendo la siguiente implicación: Si nos fijamos en tiras de tamaño ε en el eje x delimitadas por arriba por la curva f, entonces la probabilidad de que (Y, Z) caiga dentro de la tira que contiene a un punto x_0 en su base es aproximadamente $\varepsilon\, f(x_0)$.*

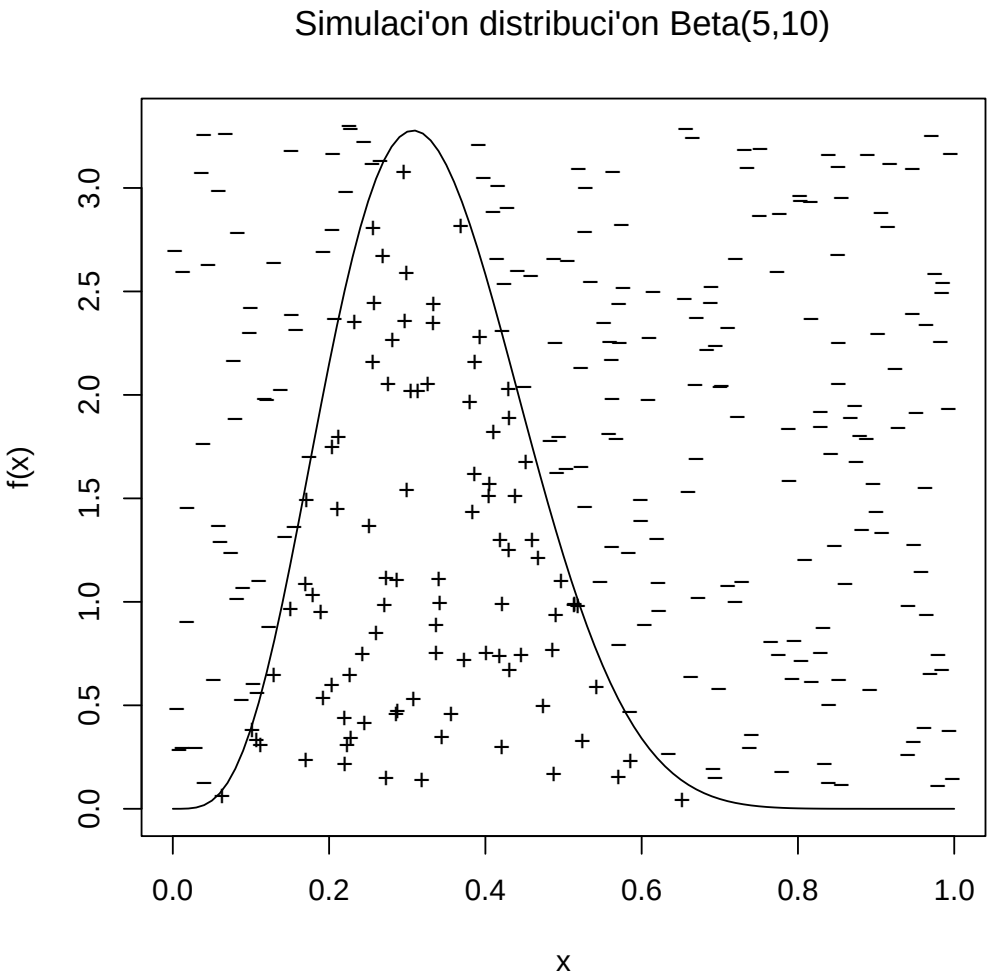


Figura 3.4: + representa realizaciones aceptadas y − realizaciones rechazadas

La idea anterior funciona en un contexto más general. Supongamos que f es una densidad cuyo soporte está contenido en el soporte de otra densidad g (que es fácil de muestrear) y sea $c > 0$ una constante tal que

$$f(x) \leq cg(x),$$

para cualquier x en el soporte de f. Entonces el siguiente algoritmo nos proporciona un número aleatorio proveniente de la densidad f.

Algoritmo 8 : Aceptación-Rechazo

1: Generar Y con función de densidad g.
2: Generar $Z \sim U(0, 1)$.
3: Aceptamos, $X = Y$, si $Z \leq \frac{f(Y)}{cg(Y)}$. En otro caso, regresar al paso 1.

Antes de justificar que el Algoritmo 8 realmente genera una variable aleatoria de la función de densidad deseada (f), mencionaremos algunas virtudes de este algoritmo:

1. No es necesario tener definida explícitamente a la función de densidad de la variable aleatoria que

se quiere simular.

2. Debido a que el método no utiliza directamente una expresión analítica de la densidad, se puede utilizar cuando el método de la función inversa no es posible o es computacionalmente ineficiente.

3. En estadística Bayesiana, es común conocer a las densidades de probabilidad sin conocer las constantes de normalización asociadas. En estos casos este método permite conocer tales constantes.

Presentamos ahora la prueba de la eficacia del Algoritmo 8.

Lema 3.1. *La variable aleatoria generada por el Algoritmo 8 tiene función de densidad f.*

Demostración. Siguiendo la estructura del algoritmo, definimos a la variable aleatoria X como la variable Z con función de densidad g cuando se satisface la condición de aceptación del algoritmo. Entonces, nos basta probar que

$$f_X(x) = g\Big(x\Big|Y \leq \frac{f(Z)}{cg(Z)}\Big) = f(x).$$

Utilizando la fórmula de Bayes tenemos que

$$f_X\Big(x\Big|Y \leq \frac{f(Z)}{cg(Z)}\Big) = \frac{P(Y \leq \frac{f(Z)}{cg(Z)}\Big|X = x)\, g(x)}{P\Big(Y \leq \frac{f(Z)}{cg(Z)}\Big)}. \tag{3.1}$$

Por un lado tenemos que

$$P\Big(Y \leq \frac{f(Z)}{cg(Z)}\Big|Z = x\Big) = P\Big(Y \leq \frac{f(x)}{cg(x)}\Big) = \frac{f(x)}{cg(x)}, \tag{3.2}$$

y por otro lado

$$\begin{aligned} P\Big(Y \leq \frac{f(Z)}{cg(Z)}\Big) &= \int_{-\infty}^{\infty} P\Big(Y \leq \frac{f(Z)}{cg(Z)}\Big|Z = x\Big)g(x)\,dx \\ &= \int_{-\infty}^{\infty} P\Big(Y \leq \frac{f(x)}{cg(x)}\Big|X = x\Big)g(x)\,dx = \int_{-\infty}^{\infty} \frac{f(x)}{c}\,dx = \frac{1}{c}. \end{aligned} \tag{3.3}$$

Utilizando las expresiones (3.2) y (3.3) en la expresión (3.1) obtenemos el resultado. □

Como parte de la eficacia del Algoritmo 8 es necesario que en alguna iteración se satisfaga la condición de acepatación. El siguiente corolario prueba que el Algoritmo 8 termina en tiempo finito, es decir, que es seguro que en alguna iteración se cumple la condición de acepatación.

Lema 3.2. *La probabilidad de generar una variable aleatoria de la densidad deseada, utilizando el Algoritmo 8 es uno, es decir, el algoritmo termina en tiempo finito*

Demostración. Sea I la variable aleatoria que denota el número de iteraciones totales realizadas por el Algoritmo 8. Notemos que la probabilidad de terminar en cada una de las iteraciones es la misma; denotemos a esa probabilidad por p. Entonces, la probabilidad de que el algoritmo termine en la i-ésima iteración está dada por

$$P(I = i) = p(1-p)^{i-1}, \qquad i = 1, 2, \ldots.$$

Así, la probabilidad de que el algoritmo termine en tiempo finito está dado por

$$P(I < \infty) = \sum_{i=0}^{\infty} (1-p)^i p.$$

Sabemos que esta serie converge solamente si $p \in (0, 1]$ y en dicho caso tenemos que $P(I < \infty) = 1$. Por lo tanto, basta demostrar que $p > 0$.

Sabemos que para terminar en cualquier iteración es necesario y suficiente que se cumpla la condición de aceptación establecida en el algoritmo. Del Teorema 3.1 tenemos que

$$p = P\Big(Y \leq \frac{f(X)}{cg(X)}\Big) = \frac{1}{c},$$

donde $c > 0$, entoces $p > 0$.

□

Otro punto importante es considerar la eficiencia del algoritmo, cuantificada por el número esperado de iteraciones necesarias para aceptar un valor. En este sentido, tenemos el siguiente resultado como consecuencia del Lema 3.2.

Corolario 3.1. *El número de iteraciones que el Algoritmo 8 necesita para aceptar un valor que tenga la distribución de la variable aleatoria X es una variable aleatoria geométrica con media c.*

Demostración. De la prueba del Lema [**?**] tenemos que $I \sim Geo(p)$ donde $p = \frac{1}{c}$, entonces,

$$\mathbb{E}[I] = \frac{1}{p} = c.$$

□

De acuerdo al Corolario 3.1 el Algoritmo 8 es más eficiente mientras c es menor. Es por ello que es importante elegir a una función g lo más similar a nuestra función objetivo f, no sólo en soporte, sino en forma y esperanza para elegir a la constante c lo más pequeña posible.

Veamos algunos ejemplos de aplicación del Algoritmo 8.

Ejemplo 3.2.1. *Consideremos a una variable aleatoria X con función de densidad*

$$f(x) = 20x(1-x)^3,$$

para $0 < x < 1$.

Usaremos $g(x) = \mathbb{I}_{(0,1)}(x)$. Para determinar alguna constante c de forma que la desigualdad en las hipótesis del Algoritmo 8 funcione, definimos

$$h(x) := \frac{f(x)}{g(x)}$$

y utilizando cálculo elemental podemos ver que h tiene un máximo en $x = 1/4$. Así, tras definir $c = h(1/4) = 135/64$, el Algoritmo 8, en este caso, tiene la forma.

Algoritmo 9 :Aceptación-Rechazo, caso continuo

1: Generar $U_1, U_2 \sim U(0,1)$, independientes.
2: Si $U_2 \leq \frac{256U_1(1-U_1)^3}{27}$, hacemos $X = U_1$. En otro caso, regresar al paso 1.

Podemos tener noción de la eficiencia del Algoritmo 8 generando muestras de distinto tamaño utilizando el Algoritmo 9 y comparando el promedio de iteraciones obtenido en cada caso con la constante c. En la Tabla 3.3 se muestra el promedio de iteraciones para aceptar un valor de la variable aleatoria de este ejemplo utilizando el Algoritmo 9 para muestras de diferentes tamaños. El Algoritmo 9 eatá implementado en la función `AR.cont` (ver B.2.5).

	n	c	Promedio
1	10	2.109375	1.800000
2	100	2.109375	1.990000
3	1000	2.109375	2.039000
4	10000	2.109375	2.128700

Tabla 3.3: Ilustra la eficiencia del Algoritmo 7

En la Figura 3.5 se ilustra el comportamiento de la distribición empírica de las muestras generadas en este ejemplo comparadas con la función de densidad teórica

Ejemplo 3.2.2. ***Distribución Gamma.*** *Supongamos que deseamos simular números aleatorios de* $X \sim Gamma(3/2, 1)$ *cuya función de densidad*

$$f(x) = \frac{x^{1/2}e^{-x}}{\Gamma(3/2)}\mathbb{I}_{(0,\infty)}(x).$$

Un candidato natural, como función g, es la densidad de variable aleatoria exponencial con media 3/2por dos razones: comparte el soporte que X y también su media. Consideremos a la función $h(x) := f(x)/g(x)$ y notemos que tiene un máximo en $x = 3/2$. Tras definir $c = h(3/2)$ obtenemos

$$\frac{f(x)}{cg(x)} = (2xe/3)^{1/2}e^{-x/3},$$

por lo que el Algoritmo 8 para este ejemplo es:

Algoritmo 10 :Aceptación-Rechazo de la distribución Gamma

1: Generar $U_1 \sim U(0,1)$.
2: Definir $Y = -\frac{3}{2}log(U_1)$.
3: Generar $U_2 \sim U(0,1)$.
4: Si $U_2 \leq (2eY/3)^{1/2}e^{-Y/3}$, definimos $X = Y$. En otro caso regresamos al paso 1.

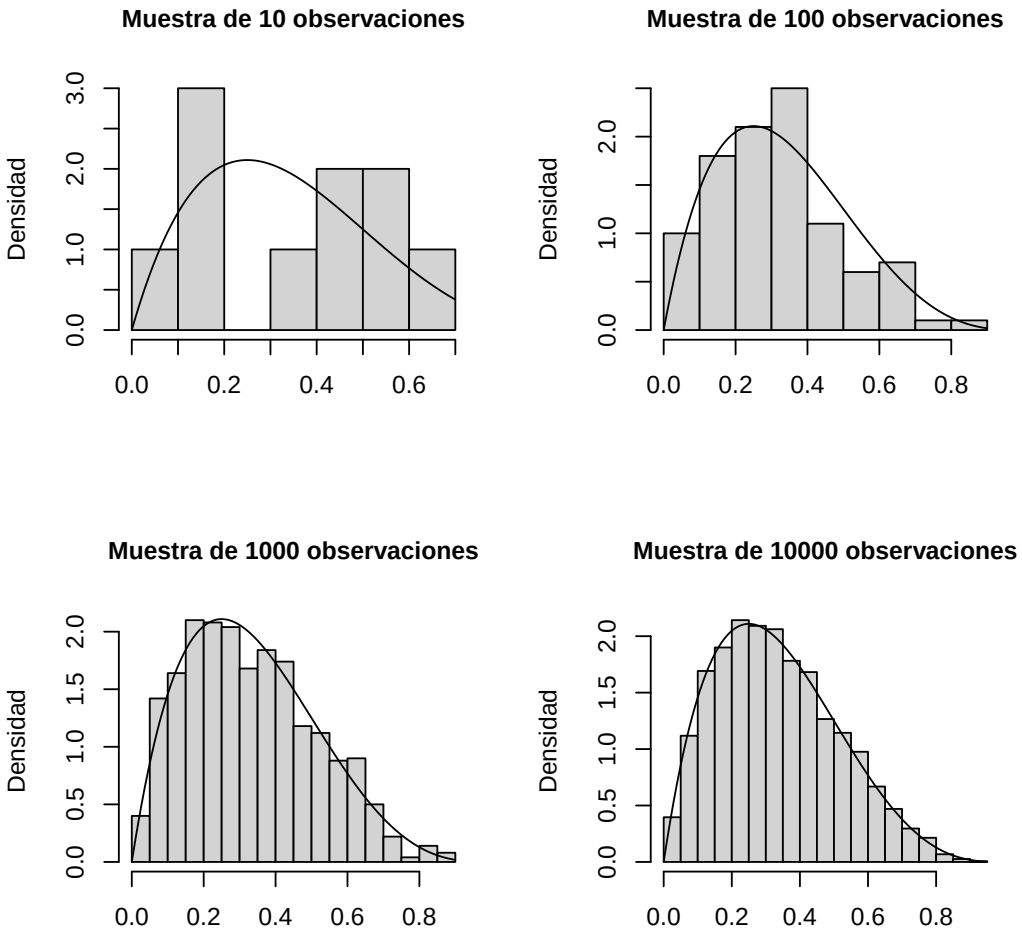


Figura 3.5: Compara histogramas con la densidad correspondiente .

Mostraremos a continuación cómo el Algoritmo 8 puede ser utilizado también en el caso discreto.

Ejemplo 3.2.3. *Supongamos que queremos simular una variable aleatoria X con soporte $S_X = \{1, 2, 3, 4, 5\}$ y $f(x) = p_x$ con $x \in S_X$, donde $(p_1 = 0{,}2, p_2 = 0{,}15, p_3 = 0{,}35, p_4 0{,}2, p_5 0{,}1)$ respectivamente.*

Usaremos como g a la función de densidad de una variable aleatoria uniforme discreta con soporte S_X.

En este caso tenemos que

$$c = máx_{x\in\{1,\ldots,5\}} \frac{f(x)}{g(x)} = \frac{f(3)}{g(3)} = 1.75,$$

y el Algoritmo 8 trabaja de la siguiente forma.

Algoritmo 11 : Aceptación-Rechazo, caso discreto

1: Generar $U_1 \sim U(0, 1)$ y hacer $Y = \lfloor 5U_1 \rfloor + 1$.
2: Generar $U_2 \sim U(0, 1)$.
3: Si $U_2 \leq 5f(Y) \,/\, 1.75$, definimos $X = Y$. En otro caso regresar al paso 1.

La función `AR.dis` (ver B.2.6) implementa el Algoritmo 11. La Tabla 3.4 compara la eficiencia de este algoritmo en término del promedio de iteraciones realizadas con muestras de distintos tamaños (n).
Además, en la Figura 3.6 se comparan los histogramas obtenidos con distintos tamaños de muestra con

	n	c	Promedio
1	10	1.750000	1.800000
2	100	1.750000	1.810000
3	1000	1.750000	1.762000
4	10000	1.750000	1.760400

Tabla 3.4: Eficiencia del Algoritmo 7 para el ejemplo del caso discreto

la densidad teórica.

3.3. Cociente de Uniformes

Este método para simular valores de variables aleatorias fue propuesto por Kinderman y Monahan en 1977 (ver [13]). Algunas consideraciones importantes en este método son las siguientes:

- La función de densidad de la variable aleatoria a simular debe ser continua.
- Es fácil de implementar.

El siguiente lema establece y justica el métod del cociente de uniformes.

Lema 3.3. *Supongamos que queremos simular valores de una variable aleatoria X con función de densidad $f(x) = cg(x)$ para toda x en el soporte de f , donde $g(x) \geq 0$ y ocurre que $\int_{\infty}^{\infty} g(x)dx < \infty$. Definamos al conjunto:*

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

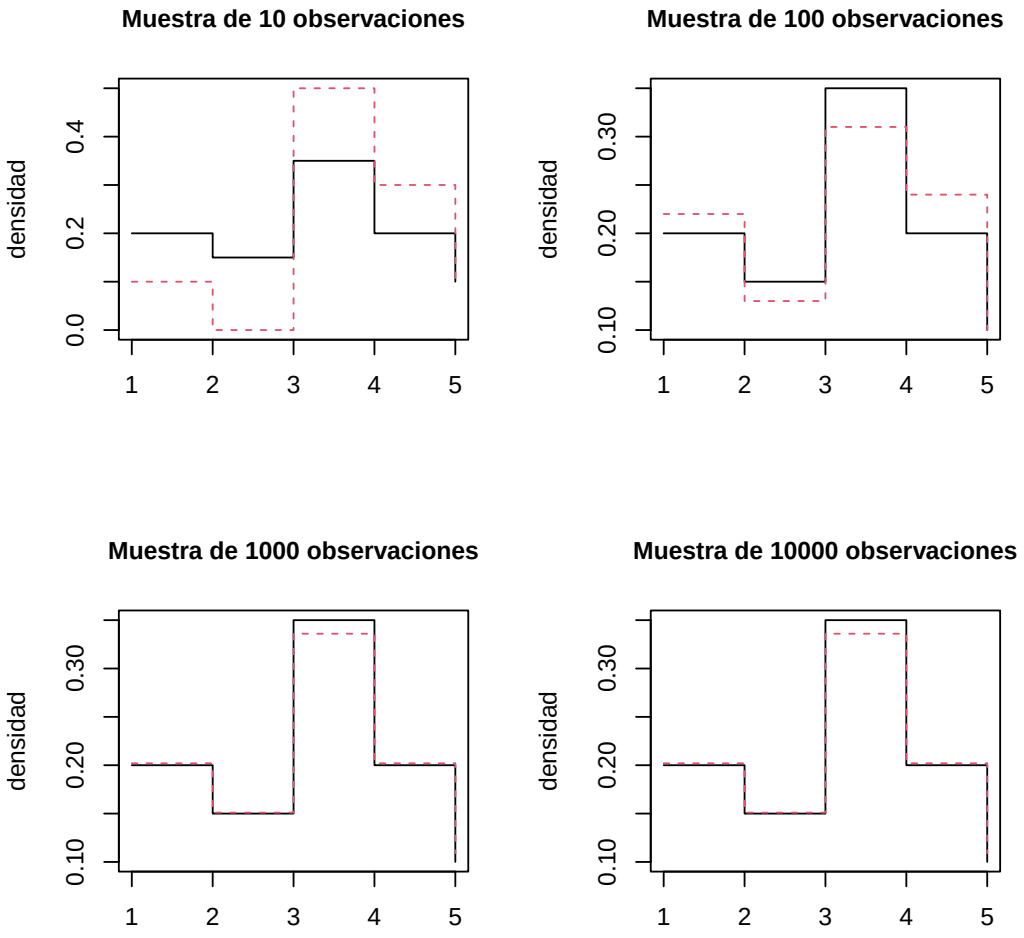


Figura 3.6: Compara histogramas con la densidad correspondiente.

Entonces se cumple que:

1. *C_g tiene área finita.*
2. *Si (U, V) tiene una distribución uniforme sobre C_g entonces la función de densidad de $\frac{V}{U}$ es f.*

Demostración. Primero demostraremos que C_g tiene área finita. Recordemos que su área está determinada por

$$|C_g| = \iint_{C_g} 1 \, dudv.$$

Para calcular el área usaremos el cambio de coordenadas $x = \frac{v}{u}$ y $y = u$. Con tal cambio nuestra región se expresa como

$$C_g = \{(x, y) \in \mathbb{R}^2 : 0 \leq y \leq \sqrt{g(x)}\}.$$

Notemos que $v \in (-\infty, \infty)$ y $u \geq 0$, lo cual implica que $x = \frac{v}{u} \in (-\infty, \infty)$. La transformación inversa está dada por $u = y$, $v = xy$, por lo que tenemos las siguientes derivadas:

$$\frac{du}{dx} = 0, \quad \frac{du}{dy} = 1, \quad \frac{dv}{dx} = y, \quad \frac{dv}{dy} = 1.$$

Lo anterior implica que el valor absoluto del determinante de la matriz jacobiana, $|J_g|$, está dado por el valor y. Así, aplicando el teorema de cambio de variable, tenemos que:

$$\begin{aligned} |C_g| &= \int \int_{C_g} 1 dudv \\ &= \int_{-\infty}^{\infty} \int_{0}^{\sqrt{g(x)}} y \, dydx \\ &= \int_{-\infty}^{\infty} \frac{g(x)}{2} dx < \infty, \end{aligned}$$

por lo que C_g tiene área finita.

Supongamos ahora que (u, v) se distribuye uniformemente en C_g. Es decir,

$$f_{U,V}(u, v) = \frac{\mathbb{I}_{C_g}(u, v)}{|C_g|}.$$

Hacemos la misma transformación $x = \frac{v}{u}$ y $y = u$. Así, por el teorema de cambio de variable, obtenemos que

$$\begin{aligned} f_{X,Y}(x, y) &= f_{U,V}(u(x, y), v(x, y))|J_g| \\ &= 2y \frac{\mathbb{I}_{\{0 \leq y \leq \sqrt{g(x)}\}}(x, y)}{\int_{-\infty}^{\infty} g(x)dx}. \end{aligned}$$

Ahora nos basta calcular la distribución marginal de X:

$$\begin{aligned} f_X(x) &= 2 \frac{\int_0^{\sqrt{g(x)}} y \, dy}{\int_{-\infty}^{\infty} g(x)dx} \\ &= \frac{g(x)}{\int_{-\infty}^{\infty} g(x)dx}, \end{aligned}$$

por lo que concluimos que $X = \frac{V}{U}$ tiene función de densidad f.

□

Utilizando el Lema 3.3 tenemos el siguiente algoritmo para simular a la variable aleatoria X.

Algoritmo 12 : Cociente de uniformes

1: Definir
$$C_g := \left\{(u, v) : 0 \leq u \leq g^{1/2}(v/u)\right\}.$$
2: Generar $(U, V) \sim U(C_g)$.
3: Tomar $X = V/U$.

Para ilustrar este método presentamos el siguiente ejemplo.

Ejemplo 3.3.1. *Distribución Exponencial. Sea* $X \sim Exp(\lambda)$*, tomamos* $g(x) = \lambda e^{-\lambda x}\mathbb{I}_{(0,\infty)}(x)$ *y* $c = 1$.

Entonces, para este ejemplo el Algoritmo 12 tiene la siguiente forma.

Algoritmo 13 : Cociente de uniformes para distribución exponencial

1: Definir
$$C_g = \left\{(u, v) : 0 \leq u^2 \leq \lambda e^{-v\lambda/u}\right\},$$
es decir,
$$0 \leq v \leq -\frac{u}{\lambda}(2log(u) - log(\lambda))$$
y
$$0 \leq u \leq \lambda^{1/2}.$$
2: Generar $U \sim U(0, \lambda^{1/2})$,
$$V \sim Unif\left(0, -\frac{u}{\lambda}(2log(u) - log(\lambda))\right)$$
3: Tomar$X = V/U$.

La función `G.exp.counif` (ver B.2.7) implementa el Algoritmo 13. Ahora podemos comparar los métodos disponibles para la generación de variables aleatorias exponenciales. En la Figura 3.7 se comparan los histogramas obtenidos con distintos tamaños de muestra con la densidad teórica usando los algoritmos 3 y 13.

Sabemos que la distribución exponencial es de cola ligera, en la Figura 3.7 podemos observar que las colas de los histogramas obtenidas con el Algoritmo 13 parecen ser ligeramente más pesadas que la teórica y esta situación no se presenta en los correspondientes obtenidos con el Algoritmo 3. Este análisis nos revela que el método de Transformada Inversa (para la distribuciń exponencial) es más preciso al del Cociente de Uniformes. Una causa de esta consecuencia es que el método de la Transformada Inversa utiliza únicamente un número aleatorio.

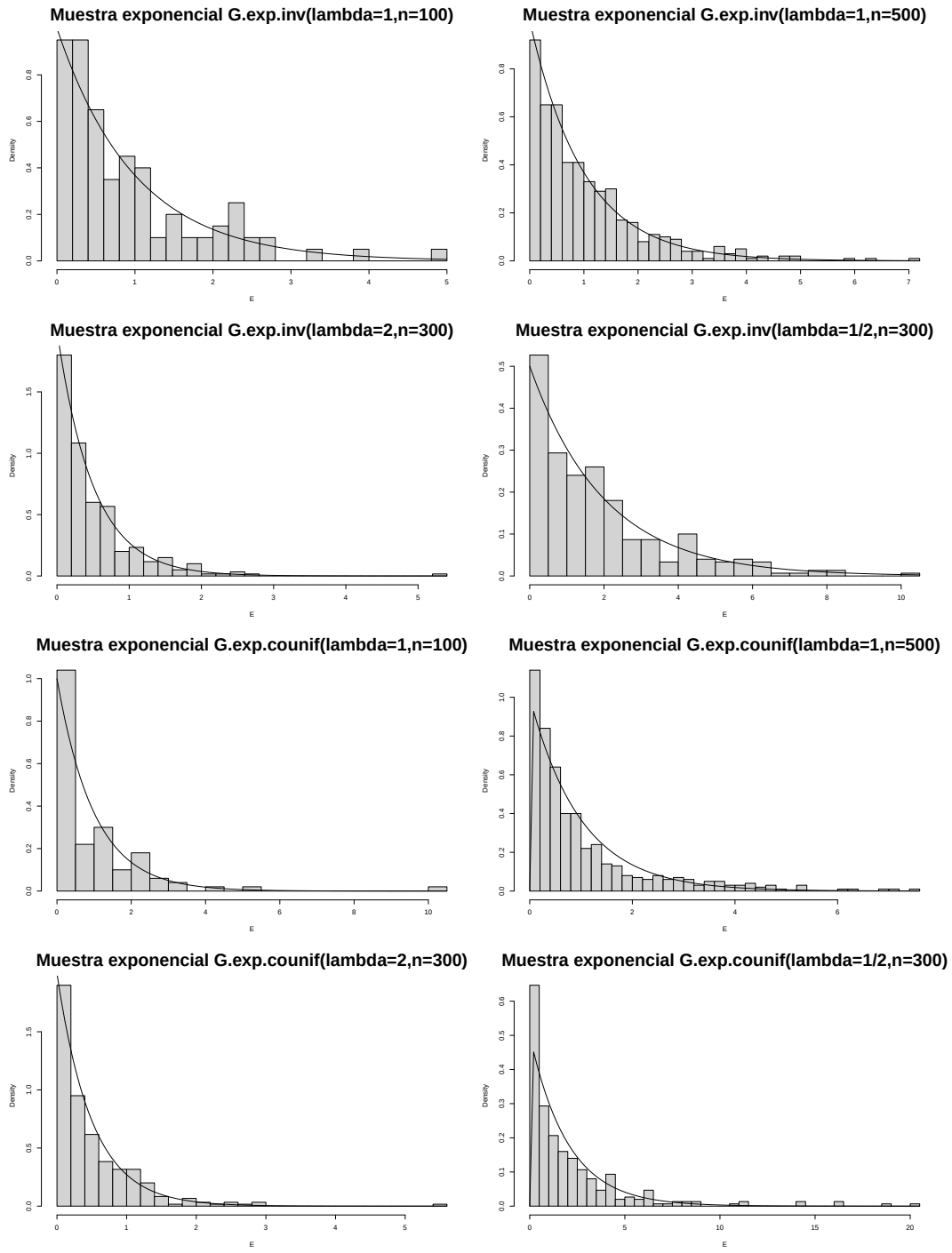


Figura 3.7: Compara histogramas con la densidad correspondiente.

3.4. Simulación de algunas distribuciones continuas

En esta sección se presentan algoritmos para tres distribuciones continuas muy importantes, Normal, Gamma y Beta.

3.4.1. Distribución Normal

Supongamos que deseamos simular una variable aleatoria $X \sim N(\mu, \sigma^2)$. Su función de densidad está dada por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}, \qquad -\infty < x < \infty,$$

donde μ es la media y σ^2 la varianza. Debido a que no existe una expresión analítica para la inversa de la función de distribución correspondiente, y aproximar tal inversa es numéricamente ineficiente, no se recomienda el uso del método de la Transformada Inversa. Será entonces necesario recurrir a otros procedimientos.

Por simplicidad, consideraremos únicamente la simulación de una variable aleatoria con distribución Normal estándar, es decir, $X \sim N(0,1)$ ya que para cualquier otra variable $Y \sim N(\mu, \sigma^2)$, ésta se puede representar como $Y = \mu + \sigma X$.

Uno de los primeros métodos para simular una variable aleatoria Normal estándar es el conocido como método **Box-Müller**. Describimos a continuación la justificación teórica del método.

Sean X y Y variables aleatorias independientes normales estándar, de modo que (X, Y) es un punto aleatorio del plano $\mathbb{R}^2$. Sean (R, Θ) sus correspondientes coordenadas polares, entonces $x = r\cos\theta$ y $y = r\,\mathrm{sen}\,\theta$, de manera que el Jacobiano (J) de la transformación es:

$$J = \det\begin{pmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{pmatrix} = \begin{vmatrix} \cos\theta & -r\,\mathrm{sen}\,\theta \\ \mathrm{sen}\,\theta & r\cos\theta \end{vmatrix} = r.$$

Entonces la densidad conjunta $f_{R,\Theta}(r, \theta)$ está dada por:

$$f_{R,\Theta}(r, \theta) = \frac{1}{2\pi} r e^{-r^2/2},$$

con $r \geq 0$ y $\theta \in [0, 2\pi]$. De la factorización de esta densidad en sus densidades marginales, uno puede observar que Θ y R son independientes, donde $\Theta \sim U[0, 2\pi]$ y R tiene la distribución de Rayleigh, es decir, su función de densidad es:

$$f_R(r) = r e^{-r^2/2}, \qquad r \geq 0.$$

Es fácil ver que R sigue la misma distribución que $\sqrt{V}$, con $V \sim Exp(1/2)$. A saber,

$$P(\sqrt{V} > v) = P(V > v^2) = e^{-v^2/2} \qquad v \geq 0.$$

Entonces, es fácil generar a las variables aleatorias Θ y R y posteriormente transformarlas en variables aleatorias independientes normales estándar. De esta manera tenemos el siguiente algoritmo para simular variables aleatorias normales.

Algoritmo 14 : Box-Müller

1: Generar dos variables aleatorias independientes, U_1 y U_2 de $U(0,1)$.
2: Devolver dos variables aleatorias independientes normales estándar, X y Y, como

$$\begin{aligned} X &= (-2\ln U_1)^{1/2}\cos(2\pi U_2), \\ Y &= (-2\ln U_1)^{1/2}\operatorname{sen}(2\pi U_2). \end{aligned}$$

Ahora describiremos un método alternativo para generar variables aleatorias normales estándar que se basa en el método de aceptación y rechazo.

Observemos que para generar una variable aleatoria $Y \sim N(0,1)$ podemos primero generar una variable aleatoria positiva X con función de densidad

$$f(x) = \sqrt{\frac{2}{\pi}} e^{-x^2/2}, \qquad x \geq 0,$$

y posteriormente asignar aleatoriamente un signo a X. La validez de este procedimiento se sigue de la simetría de la distribución normal estándar alrededor de cero.

Para simular una variable aleatoria con función de densidad $f(x) = \sqrt{\frac{2}{\pi}} e^{-x^2/2}$ utilizamos el método de Aceptación-Rechazo. Comenzamos acotando $f(x)$ por una $cg(x)$, donde $g(x) = e^{-x}$ es la función de densidad de una variable aleatoria $X \sim Exp(1)$. La mínima constante c tal que $f(x) \leq cg(x)$ es $\sqrt{2e/\pi}$. Por tanto, la eficiencia del método es $\sqrt{\pi/2e} \approx 0.76$. La condición de aceptación, $U \leq f(X)/(ce^{-X})$, se puede escribir como

$$U \leq \exp\left\{\frac{-(X-1)^2}{2}\right\},$$

la cual es equivalente a

$$-\ln U \geq \frac{(X-1)^2}{2}.$$

Dado que $-\ln U \sim Exp(1)$, la última desigualdad se puede escribir como

$$V_1 \geq \frac{(V_2-1)^2}{2}$$

donde $V_1 = -\ln U$ y $V_2 = X$ son independientes y ambas se distribuyen $Exp(1)$.
Por lo tanto, tenemos el siguiente algoritmo para simular una variable aleatoria (X) Normal estándar.

Algoritmo 15 : Normal estándar Aceptación-Rechazo

1: Generar $U \sim U(0,1)$.
2: Generar dos variables aleatorias independientes, $V_1, V_2 \sim Exp(1)$.
3: Hacer $Y = V2$ si $V_1 \geq \frac{(V_2-1)^2}{2}$, en otro caso, regresar 2.
4: Tomar

$$X = \begin{cases} Y & \text{si } U < 1/2, \\ -Y & \text{en otro caso.} \end{cases}$$

En la Figura 3.8, se presenta una comparación de las simulaciones de 500 variables aleatorias normal estándar, utilizando los algoritmos 14 y 15 mediante la gráfica del histograma y la gráfica cuantil-cuantil.

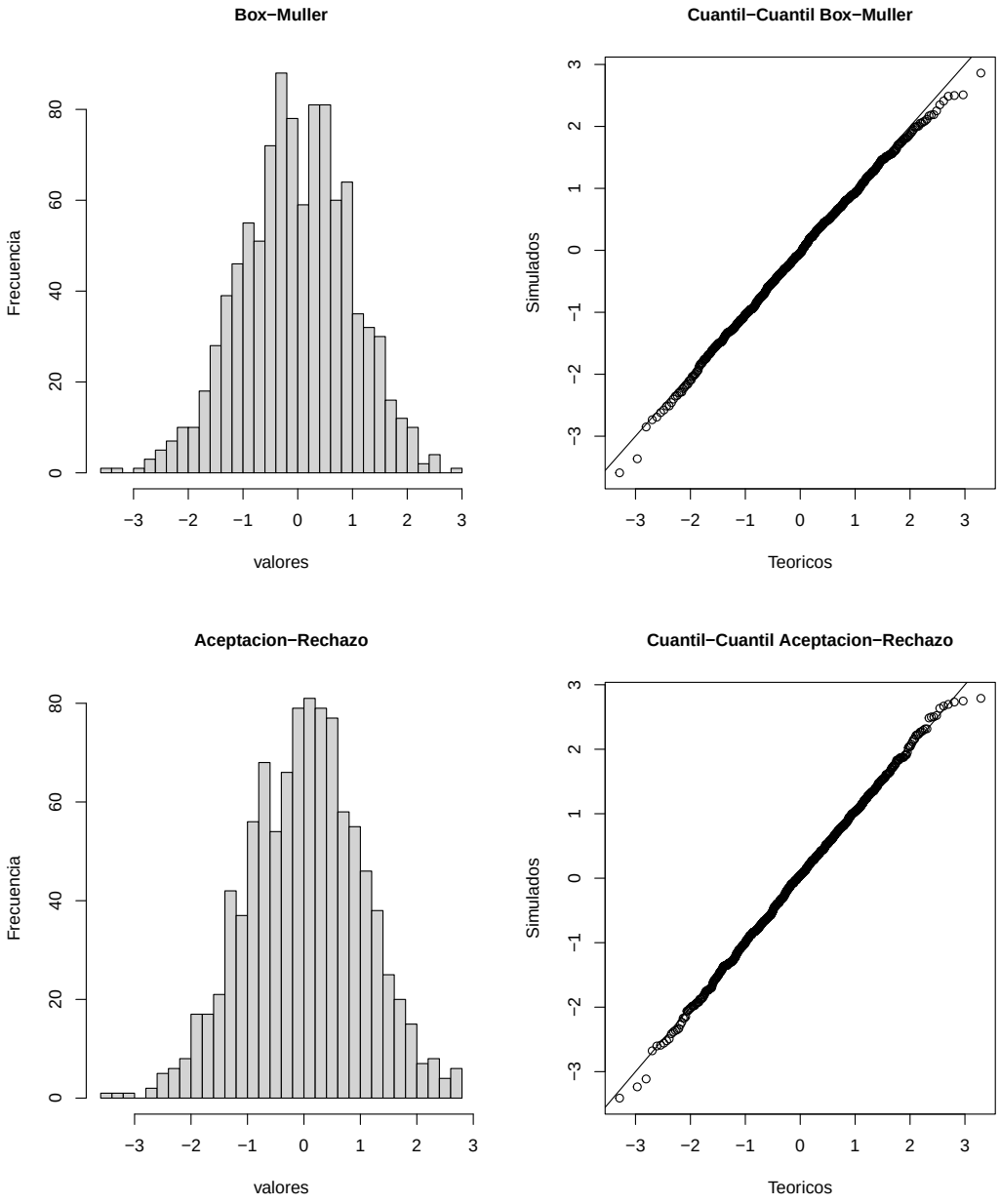


Figura 3.8: Comparación de normales estándar

3.4.2. Distribución Normal truncada

Se le llama distribución Normal truncada en un intervalo $I = (a, b)$ (donde podría ocurrir que $a = -\infty$ o $b = \infty$) a la distribución Normal estándar condicionada a tomar valores en tal intervalo. La densidad de una variable aleatoria X Normal truncada al intervalo $I = (a, b)$ ($X \sim NT_I(\mu, \sigma^2)$) tiene la forma

$$f(t) = C \exp^{-\frac{t^2}{2}} \mathbb{I}_{(a,b)}(t),$$

donde C es la constante de normalización (dependiente de a y b).

Para simular a variable aleatoria Normal truncada en I, tenemos que el método de Aceptación-Rechazo, para este caso, trabaja de la siguiente manera:

Algoritmo 16 : Normal truncada Aceptación-Rechazo

1: Simular X proveniente de una distribución Normal estándar.
2: Si X está en el intervalo en el I, aceptamos tal valor como resultante. En otro caso volvemos al paso anterior.

Notemos que este método puede ser altamente ineficiente, debido a que la probabilidad de rechazar puede ser muy alta. Por ejemplo, en el caso en que $I = (3, \infty)$, la probabilidad de aceptar un valor es igual a $1 - \Phi(3) \approx 0{,}0014$, por lo que para simular cada valor proveniente de la Normal truncada necesitaremos en promedio 714 intentos. Es necesario encontrar algoritmos más eficientes.

Marsaglia en 1964 (ver [19]) propuso el siguiente algoritmo para Normales truncadas en un intervalo de la forma $I = (a, \infty)$ donde a es un número positivo.

Algoritmo 17 : Normal truncada Marsaglia

1: Generar U_1, U_2 variables aleatorias independientes uniformes en $(0, 1)$.
2: Si ocurre que $U_2 < a(a^2 - 2\log U_1)^{-\frac{1}{2}}$ aceptamos y definimos a $X = (a^2 - 2\log U_1)^{\frac{1}{2}}$. En otro caso, volvemos al paso anterior.

Mostraremos a continuación que el Algoritmo 17 funciona.

Proposición 3.1. *Si X es obtenida mediante Algoritmo 17, entonces $X \sim NT_I(\mu, \sigma^2)$ con $I = (a, \infty)$.*

Demostración. Denotemos por A al evento $\{U_2 < a(a^2 - 2\log U_1)^{-\frac{1}{2}}\}$. Tenemos que

$$\mathbb{P}(U_1 \leq t|A) = \frac{1}{\mathbb{P}(A)} \int_0^t \int_0^{a(a^2-2\log u_1)^{-\frac{1}{2}}} du_2\, du_1,$$

por lo que la función de densidad de U_1 condicional al evento A está dada por

$$f_{U|A}(t) = Ca(a^2 - 2\log t)^{-\frac{1}{2}} 1_{\{0<t<1\}},$$

para alguna constante de integración C. Así, si definimos $X := (a^2 - 2\log U_1)^{\frac{1}{2}}$, podemos calcular

$$\mathbb{P}(X \leq t|A) = \mathbb{P}((a^2 - 2\log U_1)^{\frac{1}{2}} \leq t|A) = \mathbb{P}(U_1 \geq e^{\frac{a^2-t^2}{2}}|A) = 1 - \mathbb{P}(U_1 \leq e^{\frac{a^2-t^2}{2}}|A).$$

Derivando la expresión anterior obtenemos la función de densidad de X condicional al evento A:

$$\begin{aligned} f_{X|A}(t) &= -f_{U_1|A}(e^{\frac{a^2-t^2}{2}})e^{\frac{a^2-t^2}{2}}(-t) \\ &= Ca(a^2 - 2\log(e^{\frac{a^2-t^2}{2}}))^{-\frac{1}{2}}1_{\{0<e^{\frac{a^2-t^2}{2}}<1\}}e^{\frac{a^2-t^2}{2}}t \\ &= Cat^{-1}1_{\{a<t\}}e^{\frac{a^2-t^2}{2}}t \\ &= C_2e^{-\frac{t^2}{2}}1_{\{a<t\}}, \end{aligned}$$

donde C_2 es una constante. Concluimos que X condicional al evento A tiene distribución Normal truncada en el intervalo (a, ∞). □

La eficiencia del Algoritmo 17 es muy superior a la del Algoritmo 16; por ejemplo para $a = 3$ la probabilidad de aceptación (la probabilidad del evento A) es cercano a 0,88. En la Figura 3.9 se presenta la distribución empírica de una muestra simulada de 10000 valores proveniente de la distribución Normal truncada en el intervalo $(3, \infty)$. Se realizó utilizando la función Normal.Truc (ver B.2.8).

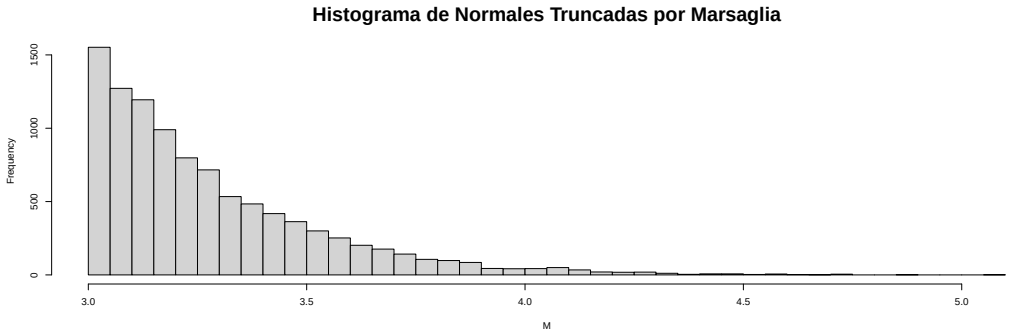


Figura 3.9: Histograma de Normal Truncadas

3.4.3. Distribución Gamma

Supongamos que queremos simular valores de una variable aleatoria $X \sim Gamma(\alpha, \lambda)$ cuya función de densidad es

$$f(x) = \frac{x^{\alpha-1}\lambda^{\alpha}e^{-\lambda x}}{\Gamma(\alpha)}, \qquad x \geq 0.$$

Los parámetros $\alpha > 0$ y $\lambda > 0$ se conocen como forma y escala, respectivamente.

Nota 3.4. *Si multiplicamos por un escalar a una variable aleatoria Gamma, simplemente cambia la escala, es decir, si $X \sim Gamma(\alpha, 1)$ entonces $X/\lambda \sim Gamma(\alpha, \lambda)$. Entonces, para simular cualquier variable aleatoria Gamma, basta con considerar únicamente la simulación de variables aleatorias $Gamma(\alpha, 1)$.*

Primero discutiremos un método para el caso en que $\alpha \geq 1$. Supongamos que queremos generar variables aleatorias independientes $Gamma(\alpha, \lambda)$, para $\alpha \in [1, \infty)$ no necesariamente entera. Sin pérdida de generalidad, supongamos que $\lambda = 1$.

Dado que podemos generar cualquier variable aleatoria Gamma con parámetro de forma en los naturales usaremos el método de aceptación y rechazo para generar la variable deseada considerando como variable

alterna a la distribución $Gamma(a, b)$ con $a \in \mathbb{N}$.

El cociente ambas funciones de densidad Gamma está dado por

$$h(x) := \frac{Gamma(x|\alpha, 1)}{Gamma(x|a, b)} \propto \frac{x^{\alpha-1}e^{-x}}{b^a x^{a-1}e^{-bx}} = x^{\alpha-a}b^{-a}e^{-x(1-b)}. \tag{3.4}$$

Si derivamos con respecto a x encontramos que la expresión anterior se maximiza en $x = (\alpha - a)/(1 - b)$ siempre que $b < 1$ y $a < \alpha$, obteniendo como cota a

$$c = b^{-a}\left(\frac{\alpha - a}{(1 - b)e}\right)^{\alpha-a}. \tag{3.5}$$

Como $a < \alpha$, podemos considerar $a = \lfloor\alpha\rfloor$ y elegimos a c que a su vez se minimiza en $b = a/\alpha$. Así, el mejor juego de parámetros (a, b) para generar variables $Gamma(\alpha, 1)$ es $(a, b) = (\lfloor\alpha\rfloor, \lfloor\alpha\rfloor/\alpha)$. Entonces el algoritmo de aceptación y rechazo, en este caso, trabaja de la siguiente forma.

Algoritmo 18 : Gamma $\alpha \geq 1$

1: Simular $U \sim U(0, 1)$.
2: Simular $Y \sim Gamma(a, b)$ donde $(a, b) = (\lfloor\alpha\rfloor, \lfloor\alpha\rfloor/\alpha)$.
3: Hacer $X = Y$ si $U < h(Y)/c$ para h definida en 3.4 y c de la expresión 3.5. En otro caso, regresar al paso 2.

Para el caso en que $\alpha < 1$, podemos usar el hecho de que si $X \sim Gamma(1 + \alpha, 1)$, $U \sim U(0, 1)$ y son independientes, entonces $XU^{1/\alpha} \sim Gamma(\alpha, 1)$. Entonces, para este caso, podemos utilizar el siguiente algoritmo.

Algoritmo 19 : Gamma $\alpha < 1$

1: Simular $U \sim U(0, 1)$.
2: Simular $Y \sim Gamma(1 + \alpha, 1)$ utilizand el Algoritmo 18.
3: Hacer $X = YU^{1/\alpha}$.

3.4.4. Distribución Beta

Consideremos a una variable aleatoria $X \sim Beta(\alpha, \beta)$, entonces su función de densidad es de la forma

$$f(x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1 - x)^{\beta-1}, \qquad 0 \leq x \leq 1$$

ambos parámetros, α y β, mayores que 0.

Nota 3.5. *$Beta(1, 1)$ corresponde a la densidad de una variable $U(0, 1)$.*

Para muestrear números provenientes de la distribución Beta, consideremos primero el caso donde alguno de los parámetros, ya sea α o β, es igual a 1. Para ese caso podemos usar el método de la transformada inversa.

Por ejemplo, para $\beta = 1$, la función de densidad de $Beta(\alpha, 1)$ es

$$f(x) = \alpha x^{\alpha-1}, \qquad 0 \leq x \leq 1,$$

y la correspondiente función de distribución es

$$F(x) = x^{\alpha}, \qquad 0 \leq x \leq 1,$$

entonces, una variable aleatoria $X \sim Beta(\alpha, 1)$ se puede generar a partir de una $U \sim U(0,1)$ y definiendo $X \sim U^{1/\alpha}$.

Un procedimiento general para generar una variable aleatoria $Beta(\alpha, \beta)$ se basa en el hecho de que si $Y_1 \sim Gamma(\alpha, 1)$, $Y_2 \sim Gamma(\beta, 1)$ y son independientes, entonces

$$X = \frac{Y_1}{Y_1 + Y_2}$$

se distribuye $Beta(\alpha, \beta)$. El algoritmo correspondiente es como sigue.

Algoritmo 20 : $Beta(\alpha, \beta)$

1: Generar independientemente $Y_1 \sim Gamma(\alpha, 1)$ y $Y_2 \sim Gamma(\beta, 1)$.
2: Definir $X = \dfrac{Y_1}{Y_1 + Y_2}$.

La Figura 3.10 compara muestras de variables aleatorias betas obtenidad con el Algoritmo 20 y la función `rbeta` de **R**.

Ejemplo 3.4.1.

En el caso particular en que los parámetros de la variable aleatoria $Beta(\alpha, \beta)$ son números naturales ($\alpha = m$ y $\beta = n$) tenemos un procedimiento alternativo para generar números provenients de tal distribución, basado en la teoría de las estadísticos de orden.

Sean $U_1, \ldots, U_{m+n-1}$ variables aleatorias $U(0,1)$. Desde que la m-ésima estadística de orden $U_{(m)}$ tiene distribución $Beta(m, n)$. Lo anterior da origen al siguiente algoritmo.

Algoritmo 21 : $Beta(\alpha, \beta)$ **con estadísticos de orden**

1: Generar $m + n - 1$ v.a.i.i.d. $U_1, \ldots, U_{m+n-1}$ de $U(0,1)$.
2: Regresar la m-ésima estadística de orden $U_{(m)}$ como una variable aleatoria de $Beta(m, n)$.

Es posible mostrar que el número total de comparaciones en el Algoritmo 21 necesario para encontrar $U_{(m)}$ es $(m/2)(m + 2n - 1)$, por lo que este procedimiento es poco eficiente cuando m y n son grandes.

3.5. Otros métodos

En esta sección presentamos algunos métodos alternativos para generar variables aleatorias discretas.

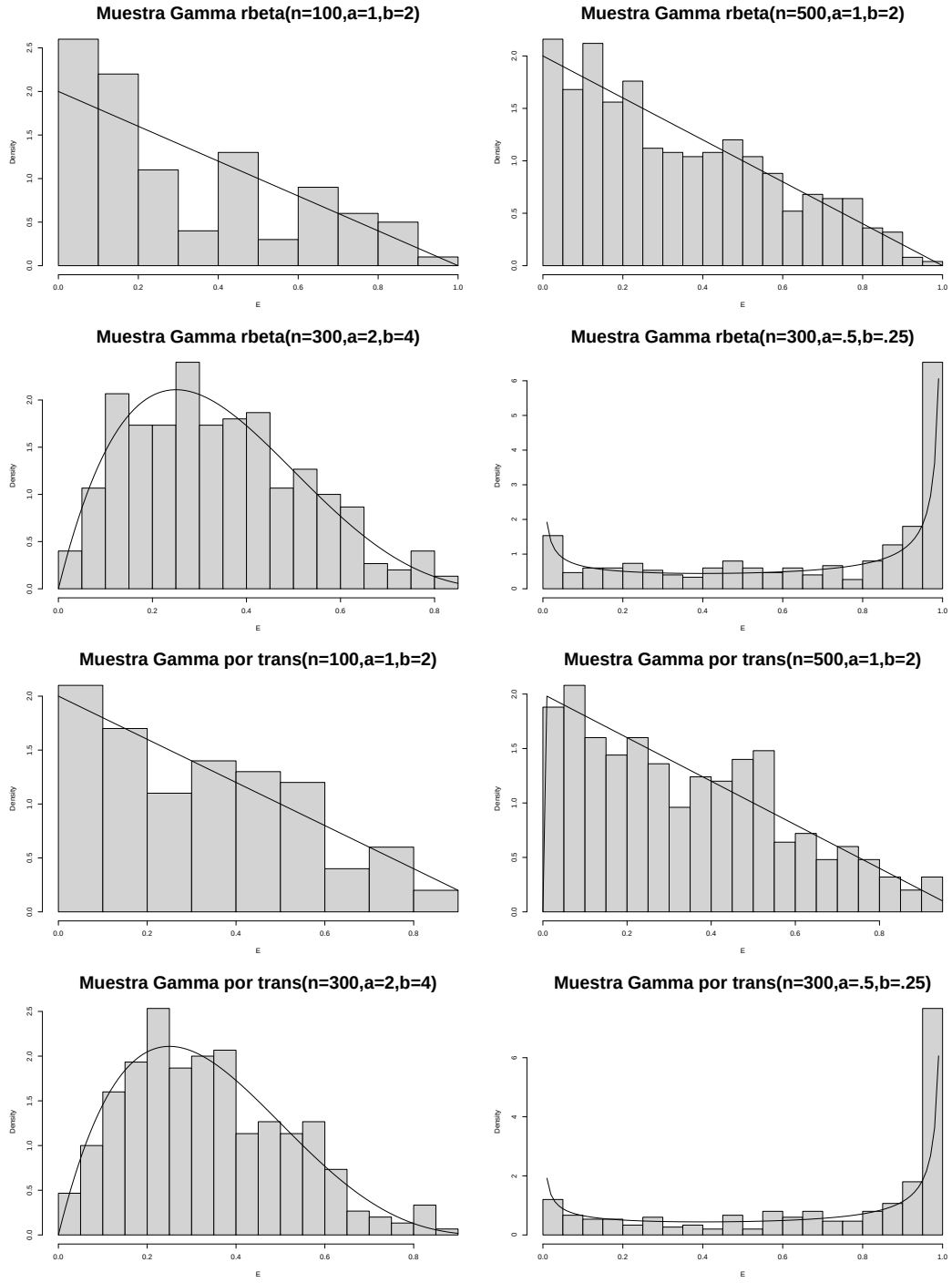


Figura 3.10: Comparación de muestras de Betas

3.5.1. Ensayos de Bernoulli

Distribución Binomial

Si $X \sim Bin(n, p)$ entonces su función de densidad es de la forma

$$f(x) = \left(\begin{array}{c} n \\ x \end{array} \right) p^x (1-p)^{n-x}, \qquad x = 0, 1, \ldots, n.$$

Recordemos que una variable aleatoria binomial X se puede intepretar como el número total de éxitos de n experimentos independientes Bernoulli, cada uno con probabilidad de éxito p. Si denotamos al i-ésimo resultado de un experimento $Ber(p)$ por $X_i = 1$ (éxito) y $X_i = 0$ (fracaso) podemos escribir $X = X_1 + \cdots + X_n$, con X_i v.a.i.i.d. $Ber(p)$. Entonces tenemos el siguiente algoritmo para simular $X \sim Bin(n, p)$.

Algoritmo 22 : Binomial como suma de Bernoullis

1: Generar v.a.i.i.d $X_1, \ldots, X_n$ de una $Ber(p)$.

2: Tomar $X = \sum_{i=1}^{n} X_i$ como una variable aleatoria de $Bin(n, p)$.

3.5.2. Relación entre distribuciones

Distribución Geométrica

Si $X \sim Geom(p)$ entonces su función de densidad es de la forma

$$f(x) = p(1-p)^{x-1}, \qquad x = 1, 2, \ldots$$

La variable aleatoria X puede interpretarse como el número de ensayos necesarios antes de que ocurra el primer éxito en una serie de ensayos independientes Bernoulli con probabilidad de éxito p. Observe también que $P(X > m) = (1-p)^m$.

3.5.3. Relación entre distribuciones

Geométrica y exponencial

Ahora veremos un algoritmo basado en la relación entre las distribuciones exponencial y geométrica.
Sea $Y \sim Exp(\lambda)$, con λ tal que $1 - p = e^{-\lambda}$. Entonces $X := \lfloor Y \rfloor + 1$ tiene distribución $Geom(p)$. Esto se debe a que

$$P(X > x) = P(\lfloor Y \rfloor > x - 1) = P(Y \geq x) = e^{-\lambda x} = (1-p)^x.$$

Por lo tanto, para generar una variable aleatoria $Geom(p)$ primero debemos generar una variable aleatoria de la distribución exponencial con $\lambda = -\ln(1-p)$, truncar el valor obtenido al entero más cercano y sumar uno. De esta forma tenemos el siguiente algoritmo para generar una variable aleatoria $Geom(p)$.

Algoritmo 23 : Geomérica via exponencial

1: Generar $Y \sim Exp(-\ln(1-p))$.
2: Regresar $X = 1 + \lfloor Y \rfloor$ como una variable aleatoria de $Geom(p)$.

Poisson y exponencial

Si $X \sim Poi(\lambda)$, entonces su función de densidad es de la forma

$$f(n) = \frac{e^{-\lambda}\lambda^n}{n!}, \qquad n = 0, 1, \cdots,$$

donde λ es el parámetro de tasa. Hay una íntima relación entre la distribución Poisson y las variables aleatorias exponenciales, resaltado por las propiedades del proceso Poisson. En particular, una variable aleatoria Poisson X se puede interpretar como el máximo número de variables exponenciales i.i.d. con parámetro λ cuya suma no supera a 1. Es decir,

$$X = \text{máx}\left\{ n : \quad \sum_{j=1}^{n} Y_j \leq 1 \right\},$$

donde las $\{Y_i\}$ son independientes y se distribuyen $Exp(\lambda)$. Puesto que $Y_j = -\frac{1}{\lambda}\ln U_j$, con $U_j \sim U(0,1)$ se puede escribir como:

$$\begin{aligned} X &= \text{máx}\left\{ n : \sum_{j=1}^{n} -\ln U_j \leq \lambda \right\} \\ &= \text{máx}\left\{ n : \quad \prod_{j=1}^{n} U_j \geq e^{-\lambda} \right\} \end{aligned}$$

Lo anterior da lugar al siguiente algoritmo para generar variables aleatorias de una distribución Poisson.

Algoritmo 24 : Poisson via exponencial

1: Defina $n = 1$ y $a = 1$.
2: Genere $U_n \sim U(0,1)$ y haga $a = aU_n$.
3: Si $a \geq e^{-\lambda}$, entonces defina $n = n + 1$ y regrese al paso 2.
4: En caso contrario, regrese $X = n - 1$ como una variable aleatoria $Poi(\lambda)$

Es fácil ver que para λ grande el Algoritmo 24 se vuelve lento pues $e^{-\lambda}$ es pequeño para λ grande, y se requieren más números aleatorios, U_j, para satisfacer la condición $\Pi_{j=1}^{n} U_j < e^{-\lambda}$.

3.6. Método de la composición

Este método es muy útil si la función de distribución F de la v.a. de la cual se desea muestrear puede ser expresada como una mezcla de n funciones de distribución $\{G_i\}_{i=1}^{n}$. Esto se expresa formalmente en la siguiente definición.

Definición 3.1. *Decimos que X tiene función de distribución compuesta si*

$$P(X_i \leq x) = G_i(x) \qquad i = 1, ..., n,$$

donde $\{X_i\}_{i=1}^n$ *es una colección de variables independientes y ocurre que*

$$X = \begin{cases} X_1 & \text{con probabilidad } p_1, \\ X_2 & \text{con probabilidad } p_2, \\ \vdots & \\ X_n & \text{con probabilidad } p_n, \end{cases}$$

donde $p_i \geq 0$ *para* $i = 1, \ldots, n$ *y* $\sum_{i=1}^n p_i = 1$.

Cuando X tiene una distribución compuesta tal como en la Definición 3.1, ocurre que su distribución F tiene la forma:

$$F(x) = \sum_{i=1}^{n} p_i G_i(x),$$

para toda $x \in \mathbb{R}$, $p_i > 0$ y $\sum_{i=1}^n p_i = 1$. En adelante, supondremos que somos capaces de generar variables aleatorias con distribución G_i para toda $i = 1, 2, \ldots, n$.

Utilizando la definición de la variable compuesta X, y de su función de distribución F en términos de las G_i's, tenemos una manera natural de simular a X: Simulamos una variable aleatoria discreta que tenga como soporte al conjunto $\{1, 2, \ldots, n\}$ con distribución $(p_1, \ldots, p_n)$, obteniendo un número natural $i^* \in \{1, 2, \ldots, n\}$, y posteriormente simulamos un número aleatorio con la distribución G_{i^*}.

Expresemos el método anterior por medio de variables aleatorias: Definamos a $\{X_i\}_{i=1}^n$ como una sucesión de variables aleatorias en donde X_i tiene distribución G_i. Definamos a Y como una variable aleatoria independiente, con densidad $P(Y = i) = p_i$ para $i = 1, 2, \ldots,$n. Entonces, la variable aleatoria X con función de distribución F puede ser expresada en términos de las variables anteriores:

$$X = \sum_{i=1}^{n} X_i 1_{\{Y=i\}}.$$

Ahora, presentamos el algoritmo anterior explícitamente.

Algoritmo 25 : Composición.

1: Generamos Y con densidad $P(Y = i) = p_i$ con $i = 1, 2, \ldots,$n.
2: Dado $Y = i$, generamos X como un número aleatorio proveniente de la función de distribución G_i.

Ejemplo 3.6.1. ***Composición de Normales.*** *Supongamos que* X *se distribuye como composición de tres variables aleatorias normales* $N(\mu_i, \sigma_i)$*, con* $\bar{\mu} = (-3, 0, 3)$*,* $\bar{\sigma} = (1, 1, 1)$ *y ponderación* $p = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$*. Entonces el Algoritmo 25 tiene la siguiente forma.*

Algoritmo 26 : Composición normales.

1: Generamos Y con densidad $P(Y = i) = p_i$ con $i = 1, 2, 3$.
2: Dado $Y = i$, generamos $Z \sim N(\mu_i, \sigma_i)$, y hacemos $X = Z$.

El Algoritmo 26 está implementado en la función B.2.9. La Fugura 3.11 presenta histogramas de muestras generadas con el Algoritmo 26.

Utilizando como motivación el algoritmo descrito en esta sección, presentamos en la siguiente sección un algoritmo general para generar vectores aleatorios.

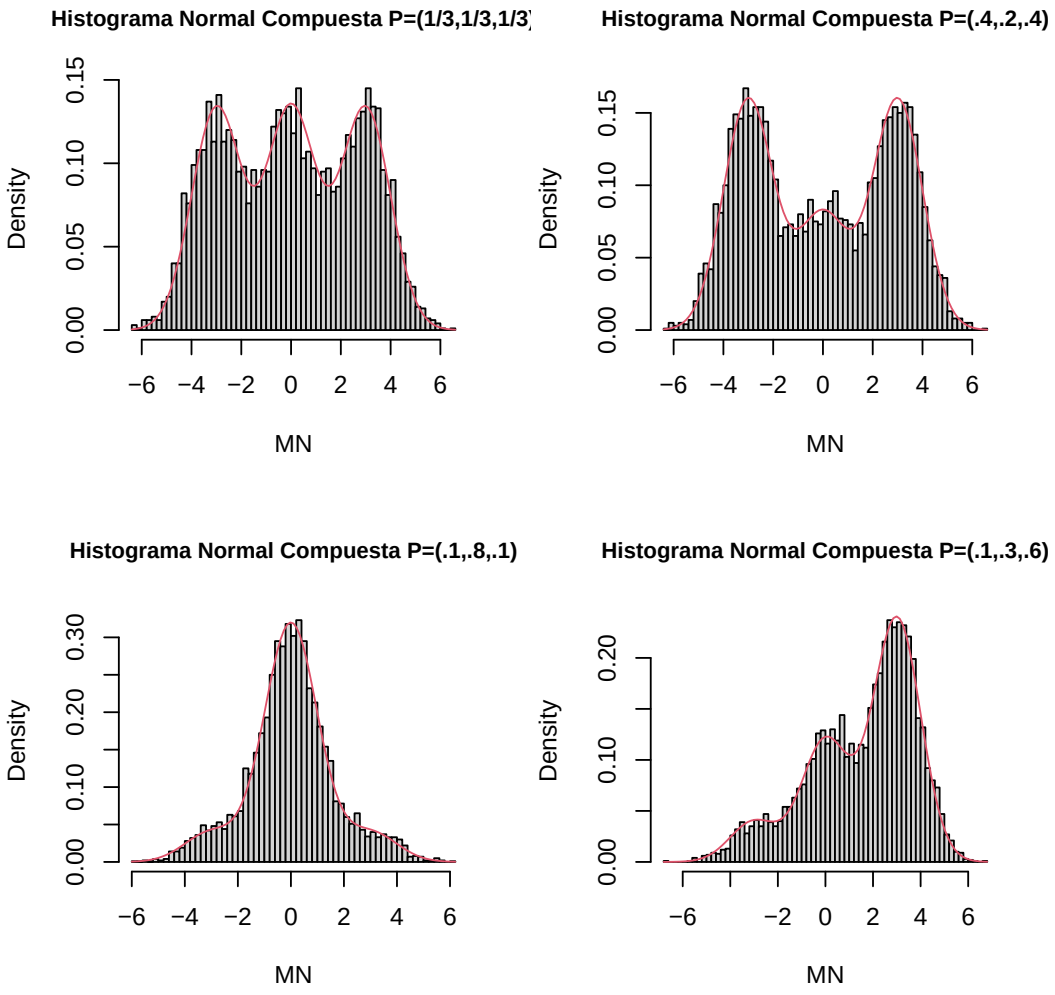


Figura 3.11: Histogramas de Mezclas de Normales

Ejemplo 3.6.2. ***Mezcla de Erlangs***

Definición 3.2. *Sea $\pi = (\pi_1, \pi_2, \ldots)$ con $\pi_k \geq 0$ y $\sum_{k=1}^{\infty} \pi_k = 1$. Se dice que una variable aleatoria X tiene distribición mezcla de Erlangs con parámetros π y $\beta > 0$ ($X \sim ErM(\pi, \beta)$) si su función de densidad es:*

$$f(x) = \sum_{k=1}^{\infty} \pi_k Erl(k, \beta)(x),$$

donde $Erl(k, \beta)$ es la función de densidad de una variable aleatoria Erlang de parámetros k y β.

La siguiente proposición muestra que las distribuciones **mezclas de Erlangs** son densas en el espacio de distribuciones continuas con soporte en $[0, \infty)$.

Proposición 3.2. *Sea F la función de distribución con soporte en $[0, \infty)$. Sea $\{F_n\}_{n=1}^{\infty}$ una sucesión de funciones de distribución definidas por*

$$F_n(x) = \sum_{k=1}^{\infty} b(k, n) Erl(k, n)(x),$$

donde $b(k, n) = F(k/n) - F((k-1)/n)$ y $Erl(k, n)$ es la función de distribución de una variable aleatoria Erlang de parámetros k y β. Entonces

$$lim_{n \to \infty} F_n(x) = F(x).$$

Además, si F tiene soporte acotado, la convergencia es uniforme.

Demostración. Ver [16] □

3.7. Generación de vectores aleatorios

Supongamos que quereremos obtener muestras aleatorias de un vector aleatorio n-dimensional $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ con función de distribución F y densidad f. Para el caso en que las variables aleatorias que componen a X son independientes, basta generar a cada una utilizando alguno de los métodos abordados en las secciones anterioses de este capítulo. En esta sección nos enfocaremos en vectores aleatorios compuestos por variables aleatorias que no son independientes.

Sabiendo que somos capaces de generar números aleatorios provenientes de una distribución marginal, podemos a la funcion de densidad conjunta del vector $\boldsymbol{X}$ como el siguiente producto:

$$f_X(x_1, x_2, \ldots, x_n) = f_{X_1}(x_1) f_{X_2|X_1}(x_2|X_1 = x_1) \ldots f_{X_n|X_1,\ldots,X_{n-1}}(x_n|X_1 = x_1, \ldots, X_{n-1} = x_{n-1}). \quad (3.6)$$

Esta expresión nos sugiere un método iterativo para simular al vector aleatorio $\boldsymbol{X}$. Se genera primero a la entrada 1, con su densidad marginal. Dado el valor de la entrada 1, se genera a la entrada 2, con la densidad marginal de la segunda entrada dada el valor de la primera entrada. Se prosigue este método hasta que la última entrada se genera de manera condicional, dadas todas la entradas. Siguiendo esta idea, el algoritmo toma la siguiente forma específica.

Es importante mencionar las siguientes observaciones respecto al Algoritmo 27.

Algoritmo 27 : Vectores Aleatorios

1: Simulamos X_1 de f_{X_1}.
2: Dado $X_1 = x_1$, generamos X_2 de $f_{X_2|X_1}(x_2|X_1 = x_1)$.
3: $\vdots$
4: [n] Generamos X_n de $f_{X_n|X_1,\ldots,X_{n-1}}(x_n|X_1 = x_1, \ldots, X_{n-1} = x_{n-1})$.

- Para implementar este algoritmo, es necesario ser capaces de simular a las variables marginal y condicionalmente, dadas las otras entradas del vector.

- El orden para simular el vector puede ser elegido arbitrariamente. Esto es una observación simple ya que la densidad conjunta se puede expresar como un producto de una densidad marginal y densidades condicionales, para cualquier orden en las entradas, de manera similar a la Ecuación 3.6. El orden a elegir puede influir fuertemente en la dificultad de la simulación (ver Ejercicio 16, 17).

Ejemplo 3.7.1. ***Distribución Multinomial****. Sea* $\boldsymbol{X}$ *un vector aleatorio con distribución multinomial de parámetros* $(p_1, \ldots, p_r, n)$ *y función de densidad*

$$f_X(x_1, x_2, \ldots, x_r) = \frac{n!}{x_1! \ldots x_r!} p_1^{x_1} \ldots p_r^{x_r},$$

donde $\sum_{i=1}^{r} x_i = n$*. En este caso podemos escribir al Algoritmo 27 de la siguiente manera:*

Algoritmo 28 : Multinoimial

1: Generamos variables aleatorias Y_j de $P(Y_j = i) = p_i$ con $i = 1, 2, \ldots, r$ y $j = 1, 2 \ldots n$.
2: Definimos $X_i = \sum_{j=1}^{n} I_{(Y_j=i)}$.

El Algoritmo 28 está implementado en la funcón `G.MNom` (ver B.3.1).

La Figura 3.7 ilustra la simularon de vectores (en la primera gráfica $n = 5$, en la segunda $n = 20$, en la tercera $n = 100$), en donde cada vector es la frecuencia resultante de asignar 20 objetos que se asignaron en 5 casillas, de forma independiente, donde las probabilidades para cada objeto de ser asignado a cada casilla están dadas por $P = (.1, .2, .1, .5, .1)$. Notemos que conforme n crece, nos aproximamos más y más a la densidad multinomial teórica para 20 objetos en 5 casillas. Esta es una manera de realizar inferencia, cuando tenemos n vectores simulados.

3.8. Cópulas

En esta sección presentaremos una herramienta con un enfoque que nos permite modelar dependencias en un vector aleatorio. Lo poderoso de esta herramienta es que nos permite parametrizar tal dependencia y además separar la información dada por la densidad conjunta de variables aleatorias en dos partes: la información dada por las marginales mismas y la información dada por la dependencia existente entre ellas. El tema es amplio, por lo que veremos una breve introducción.

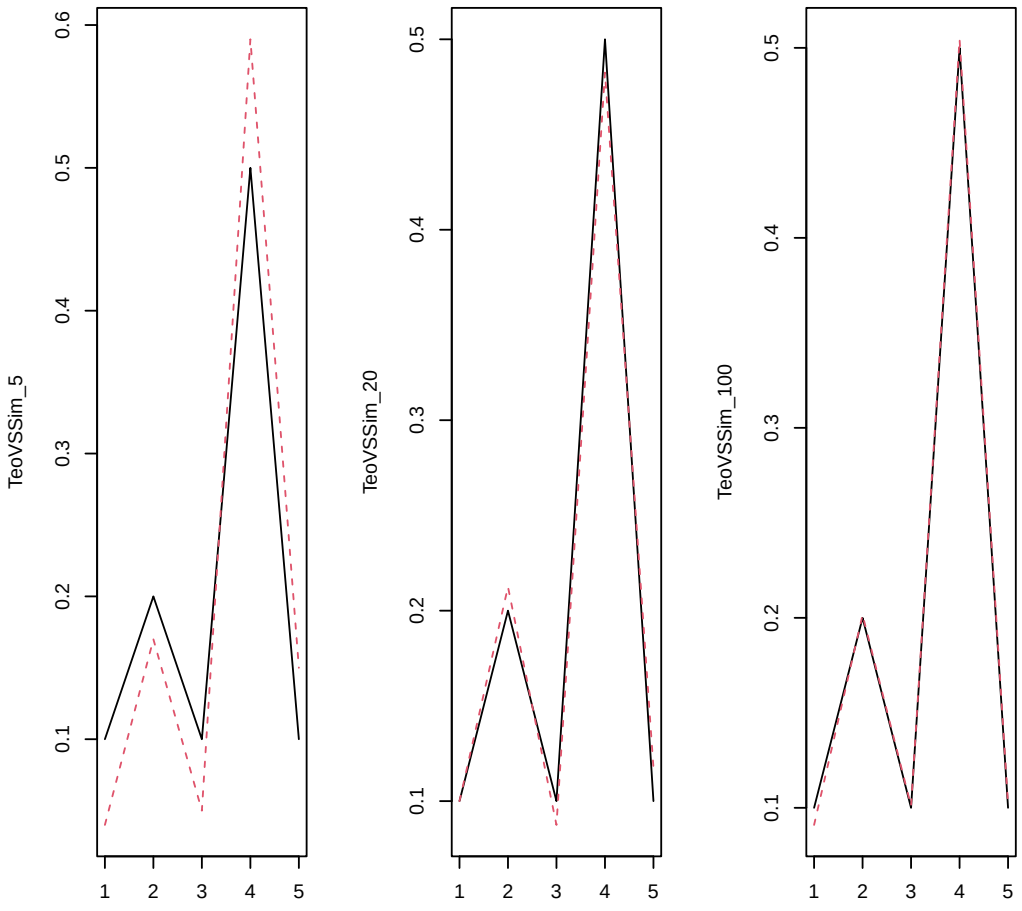


Figura 3.12: Frecuencia de una distribución multinomial.

3.8.1. Introducción

Consideremos un vector aleatorio $\boldsymbol{X} = (X_1, ..., X_n)$ en donde las distribuciones marginales $F_i(x) = \mathbb{P}(X_i \leq x)$ son continuas. Definamos nuevas variables aleatorias

$$U_i := F_i(X_i).$$

Note que por el teorema de la transfomación inversa, ocurre que U_i tiene distribución uniforme en $(0, 1)$. Definamos ahora a la función $C : [0, 1]^n \to [0, 1]$ dada por

$$C(u_1, ..., u_n) := \mathbb{P}(U_1 \leq u_1, ..., U_n \leq u_n),$$

Tenemos algunas observaciones:

- F_i tiene la información de cómo se comportan las variables aleatorias X_i marginalmente.
- Por otro lado, la función C tiene la información de dependencia entre las variables X_i, sin tomar en cuenta cómo se comportan marginalmente ya que utiliza en su definición siempre un vector $(U_1, ..., U_n)$ con marginales fijas (uniformes) pero con posible dependencia entre sus entradas.
- La función C es en sí misma una función de distribución conjunta en el cubo unitario con marginales uniformes.

La última observación puede escribirse explícitamente; lo hacemos a continuación.

Definición 3.3. *Una cópula es una función C que cumple*

1. $C : [0, 1]^n \to [0, 1]$.
2. $C(u_1, ..., u_{i-1}, 0, u_i, ..., u_n) = 0$ *para toda* i.
3. $C(1, ..., 1, u, 1, ..,1) = u$ *para toda* $i = 1, \ldots, n$.
4. $C(\square) \geq 0$, *donde tal notación quiere decir que el incremento de un* n*-volumen dentro del cubo unitario siempre es no negativo.*

El último elemento de la definición puede parecer abstracto, pero es el requisito usual en una función de distribución. Por ejemplo, para el caso $n = 2$, lo que expresa es que para todo $a < b$ y $c < d$, ocurre que

$$C(b, d) - C(b, c) - C(a, d) + C(a, c) \geq 0.$$

Esto quiere decir que si C fuese la distribución conjunta de un vector aleatorio (U, V), la probabilidad de que (U, V) esté en el cuadrado $(a, b] \times (c, d]$ debe ser mayor o igual a cero.

El siguiente teorema nos muestra la forma explícita que relaciona a cualquier función de distribución conjunta con alguna cópula y sus funciones de distribución marginales.

Teorema 3.1 (Sklar)**.** *Si H es una función de distribución conjunta en $\mathbb{R}^n$ con distribuciones marginales F_i, $1 \leq i \leq n$, existe una cópula C tal que*

$$H(x_1, ..., x_n) = C(F_1(x_1), ..., F_n(x_n)).$$

Algo que hay que destacar, es que este teorema de existencia no nos asegura que en todos los casos prácticos seamos capaces de encontrar a la función C de forma explícita. En esto es similar al teorema de la función inversa, en donde tal función inversa no es ni siquiera analítica en algunos casos.

Queremos ahora tener una forma paramétrica que nos permita tener una idea del tipo de dependencia entre las variables aleatorias que conforman al vector, y también el grado o intensidad en que son independientes. Es por ello que nos restringimos a una clase de cópulas que es importante en la teoría y en las aplicaciones.

Definición 3.4. *Decimos que una cópula es arquimediana si existe* $\phi : [0,1] \to [0,\infty]$ *continua, decreciente y convexa, con* $\phi(1) = 0$ *tal que*

$$C(u_1, ..., u_n) = \phi^{-1}(\phi(u_1) + ... + \phi(u_n)).$$

A ϕ *se le conoce como el generador de la cópula.*

Una observación directa de la definición de cópula arquimediana es que resulta ser intercambiable, es decir, si se permutan las coordenadas del vector $(u_1, ..., u_n)$ la cópula que se obtiene es la misma.

Mostraremos ahora algunos ejemplos.

Ejemplo 3.8.1. *Sea* $\phi(t) := -\log t$ *para* $t \in [0,1]$. *Notemos que en este caso* $\phi^{-1}(t) = \exp(-t)$. *Y entonces*

$$C(u_1, ..., u_n) = \phi^{-1}(-\log u_1 - ... - \log u_n) = \exp(\log u_1 + ... + \log u_n) = u_1...u_n.$$

Es decir, que a este generador corresponde la cópula que multiplica a sus entradas. Observando el teorema de Sklar, esto implica que cuando tenemos esta cópula, la función de distribución factoriza en sus marginales, por lo que llamaremos a esta la ***cópula independiente.***

Ejemplo 3.8.2 (Cópula de Gumbel)**.** *De manera similar al ejemplo anterior, podemos definir el generador* $\phi(t) := (-\log t)^\theta$ *donde* θ *es un parámetro en* $[0,1]$. *En este caso* $\phi^{-1}(t) = \exp(-t^{\frac{1}{\theta}})$ *y la cópula tiene la forma explícita*

$$C(u_1, ..., u_n) = \exp\Big(-\sum_{i=1}^{n}(\log u_i)^\theta\Big).$$

Notemos que cuando $\theta = 1$ *tenemos el caso de la cópula independiente. De hecho, el parámetro* θ *cumple que*

$$\theta = \frac{1}{1 - \tau_{X,Y}},$$

donde $\tau_{X,Y}$ *es la* ***Tau de Kendall****, una medida no paramétrica de correlación que está definida en* $[-1,1]$. *Esto nos indica que cuando* θ *crece hacia infinito,* $\tau_{X,Y}$ *crece hacia* 1*; es decir, la dependencia tiende a ser completa.*

Ocurrirá en general para las cópulas arquimedianas lo que vimos en el ejemplo anterior: la cópula elegida nos determinará el tipo de dependencia que tienen las variables aleatorias y habrá un parámetro θ que nos indicará el grado de dependencia que tienen las variables aleatorias.

3.8.2. Simulación de cópulas

Supongamos que queremos simular a un vector $\boldsymbol{X} = (X_1, ..., X_n)$ proveniente de una distribución F, en donde ocurre que

$$F(x_1, ..., x_n) = C(F_1(x_1), ..., F_n(x_n)),$$

en donde conocemos a las marginales F_i y a la cópula C. Debido a que la cópula C es en sí misma una función de distribución en $[0,1]^n$, el siguiente algoritmo funciona.

Algoritmo 29 : Simulación de vectores aleatorios

1: Simular al vector $(U_1, ..., U_n)$ proveniente de la cópula C.
2: Definir, para cada $1 \leq i \leq n$, $X_i := F_i^{-1}(U_i)$.
3: Devolver $\boldsymbol{X} = (X_1, ..., X_n)$.

Debido al teorema de la función inversa, cada X_i tiene la distribución marginal correcta. Por otro lado, las X_i's heredan la dependencia que tienen las v.a. uniformes U_i's. La prueba de este resultado es directa.

El Algoritmo 29 nos ha fragmentado el problema de generar números provenientes de la distribución F en dos partes: si sabemos cómo generar a la cópula C y el teorema de la función inversa se puede implementar para cada marginal F_i habremos terminado. Por ello, nos enfocaremos a continuación únicamente en métodos para generar a la cópula C.

Empezaremos mostrando un algoritmo general para generar cópulas que sin embargo es adecuado exclusivamente para dimensiones bajas. Mostraremos el caso particular de dimensión 2.

Notemos que dada una cópula C con densidad asociada c ocurre que

$$\frac{\partial}{\partial v} C(u,v) = \frac{\partial}{\partial v}\Big[\int_0^v \int_0^u c(s,t)dsdt\Big] = \int_0^u c(s,v)ds = \int_0^u c(s|v)F_{U|V}(u|v)ds,$$

donde en la penúltima igualdad hemos utilizamos que $c(v) = 1$, debido a que la marginal de V es uniforme en $(0,1)$, por lo que la densidad conjunta de U y V es igual a la densidad condicional de U dado V.

Por lo tanto, si tal derivación es posible podemos encontrar a $F_{U|V}$, y si además la función $F_{U|V}$ tiene una inversa explícita, podemos utilizar el algoritmo para vectores aleatorios que condiciona secuencialmente. Este algoritmo toma la siguiente forma explícita para generar una cópula.

Algoritmo 30 : Simulación de cópulas

1: Simular V y $W \sim [0,1]$ de forma independiente.
2: Dados $V = v$ y $W = w$, definir $U = F_{U|V}(w|v)$.
3: Devolver (U, V).

Ejemplo 3.8.3 (Cópula de Cuadras-Augé)**.** *En este caso*

$$C(u,v) = \text{mín}(u,v)\,\text{máx}(u,v)^{1-\theta},$$

para $\theta \in [0,1]$. Notemos que el caso $\theta = 0$ corresponde a la cópula independiente.
En este caso

$$\frac{\partial}{dv} C(u,v) = \begin{cases} u(1-\theta)v^{-\theta} & si\ u < v, \\ u^{1-\theta} & u > v. \end{cases}$$

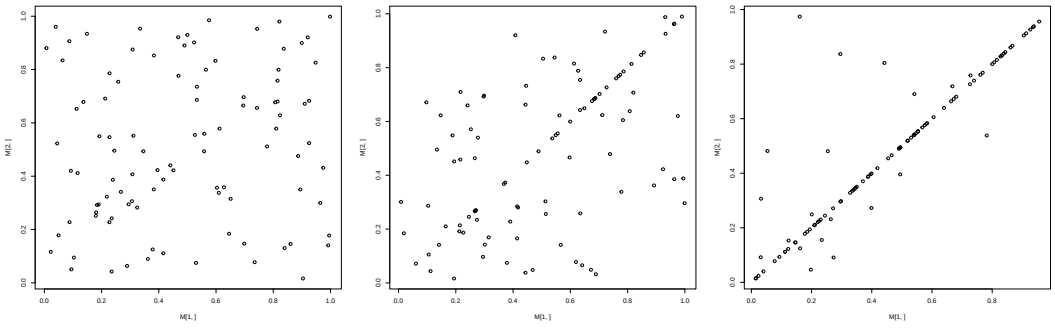


Figura 3.13: Simulación de la cópula de Cuadras-Auge

En el punto $u = v$, la derivada anterior no existe, sin embargo por ser función de distribución sabemos que la distribución $F_{U|V=v}(u)$ es continua por la derecha por lo que debe ocurrir que $F_{U|V=v}(v) = v^{1-\theta}$. Utilizando la fórmula obtenida, podemos encontrar la inversa $F^{-1}_{U|V=v}$ que es también una función definida a trozos:

$$F^{-1}_{U|V=v}(u) = \begin{cases} \frac{u}{1-\theta}v^{\theta} & si\ u < (1-\theta)v^{1-\theta} \\ v & cuando\ (1-\theta)v^{1-\theta} \leq u < v^{1-\theta} \\ u^{\frac{1}{1-\theta}} & si\ u \geq v^{1-\theta} \end{cases}$$

Una implementación del Algoritmo **??** *se encuentra en la función B.3.2 en el Apéndice.*

La Figura 3.13 muestra la simulación de 100 *valores con distintos valores para el parámetro de dependencia a; cada gráfica corresponde a los valores $a = 0.2, a = .5, a = .9$.*

Podemos ver cómo el valor de a nos da la fuerza de dependencia entre las variables aleatorias generadas y la dependencia específica de la Cópula Cuadras-Auge, que tiende a colocar valores sobre la diagonal del cubo $[0,1]^2$.

3.8.3. Cópulas arquimedianas

En el caso en que las cópulas son arquimedianas tenemos un algoritmo poderoso propuesto por Marshall y Olkin en 1988 (ver [20]) que funciona en una dimensión arbitraria n y que requiere exclusivamente la simulación de $n+1$ números aleatorios. Lo desafortunado de tal algoritmo, es que es necesario conocer a $\mathcal{L}^{-1}(\phi^{-1})$, la transformada de Laplace inversa de la función inversa del generador. A continuación presentamos el algoritmo.

Algoritmo 31 : Simulación de cópulas arquimedianas

1: Generar V con distribución dada por $\mathcal{L}^{-1}(\phi^{-1})$.
2: Generar de manera independiente $W_i \sim Unif(0,1)$.
3: Devolver $(U_1, ..., U_n)$ donde $U_i = \phi^{-1}\Big(-\frac{\log(W_i)}{V}\Big)$.

Presentamos ahora algunos ejemplos de cópulas arquimeadianas en donde es posible implementar el Algoritmo 31.

Notemos que en los casos de la cópula de Ali-Mikhail-Haq y Frank, la distribución de $\mathcal{L}^{-1}(\phi^{-1})$ resulta ser una variable aleatoria discreta.

Rango de θ	Cópula	$\phi(t)$	$\phi^{-1}(t)$	$\mathcal{L}^{-1}(\phi^{-1})$
[0, 1)	Ali-Mikhail-Haq	$\frac{1-\theta}{e^t-1}$	$\log\left(\frac{1-\theta(1-t)}{t}\right)$	$p_k = (1-\theta)\theta^{k-1}$, $k \in \mathbb{N}$
$(0, \infty)$	Clayton	$(1+t)^{-\frac{1}{\theta}}$	$t^{-\theta} - 1$	$\Gamma(\frac{1}{\theta}, 1)$
$(0, \infty)$	Frank	$-\frac{\log(e^{-t}(e^{-\theta}-1)+1)}{\theta}$	$-\log\left(\frac{\exp(-\theta t)-1}{\exp(-\theta)-1}\right)$	$p_k = \frac{(1-e^{-\theta})^k}{k\theta}$, $k \in \mathbb{N}$

Tabla 3.5: Tabla de Cópulas.

3.8.4. La paquetería `copula`

El lenguaje R tiene disponible a la paquetería `copula` para la simulación de vectores aleatorios con función de distribución dada por algunas de las cópulas conocidas. A continuación daremos un ejemplo de su uso.

```
> library(copula)
> library(scatterplot3d)
> myCop.gumbel=archmCopula(family="gumbel",dim =2,param=3)
```

En lo anterior, definimos cópulas utilizando tres parámetros: el tipo de cópula, la dimensión o tamaño del vector aleatorio asociado, y el parámetro de dependencia. En nuestro ejemplo, elegimos una cópula de Gumbel, de dos dimensiones y parámetro de dependencia igual a 3.

```
> myMvd=mvdc(copula=myCop.gumbel,margins=c("norm","norm"),
+ paramMargins=list(list(mean=0,sd=1),list(mean=0,sd=1)))
```

En el código anterior estamos definiendo funciones de distribución conjunta a través de cópulas dentro de la clase mvdc (multivariate distributions constructed from copulas, en inglés). Tiene tres componentes: la cópula (`copula`), el tipo de marginales (`margins`), y los parámetros de las correspondientes marginales (`paramMargins`). Para nuestro ejemplo utilizamos a la cópula Gumbel definida arriba, con marginales que son normales estándar.

Si escribimos en la consola el nombre de nuestra distribución recién definida (en nuestro ejemplo `myMvd`) obtenemos la información asociada a ella:

```
> myMvd
```

```
Multivariate Distribution Copula based ("mvdc")
 @ copula:
Gumbel copula, dim. d = 2
Dimension:  2
Parameters:
  alpha   = 3
 @ margins:
```

```
[1] "norm" "norm"
   with 2 (not identical)  margins; with parameters (@ paramMargins)
List of 2
 $ :List of 2
  ..$ mean: num 0
  ..$ sd  : num 1
 $ :List of 2
  ..$ mean: num 0
  ..$ sd  : num 1
```

Podemos calcular explícitamente la función de densidad de nuestra distribución conjunta recién construida (usando la función `dMvdc`) o bien calcular en la función de distribución acumulada en un punto; en este caso elegimos el punto $(x, y) = (1, 1)$ para calcular los valores de la densidad y la distribución.

```
> den_myc=pMvdc(c(1,1),myMvd)
> den_myc
```

```
[1] 0.804402
```

```
> dist_myc=dMvdc(c(1,1),myMvd)
> dist_myc
```

```
[1] 0.2690301
```

Tenemos también un simulador que nos permite muestrear puntos de nuestra distribución conjunta. A continuación simulamos 100 vectores provenientes de nuestra función de distribución ejemplo.

```
> ran_mycop=rMvdc(100,myMvd)
```

En la Figura 3.14 se presenta una gráfica la distribucón conjunta correspondiente.

3.9. Ejercicios

1. En el Ejemplo 3.1.2 mostrar que en el caso en que las variables originales tienen distribución uniforme en $(0, 1)$ ocurre que $Y_1 = 1 - U^{1/n}$ y $Y_n = U^{1/n}$, e implementar los algoritmos en este caso.

2. Considere una variable aleatoria X con función de densidad $f(x) = 2xI(x)_{(0,1)}$, escribir e implementar un algoritmo para generar valores de dicha variable.

3. Considere una variable aleatoria X, con soporte en el conjunto $\{1, 2, 3, 4, 5\}$ y con distribución $\{1/3, 1/5, 1/6, 1/6, 4/30\}$. Implementar un algoritmo para generar una muestra de 1, 000 elementos de dicha distribución . $\diamondsuit$

4. Implementar algoritmos para generar números provenientes de las siguientes variables aleatorias:

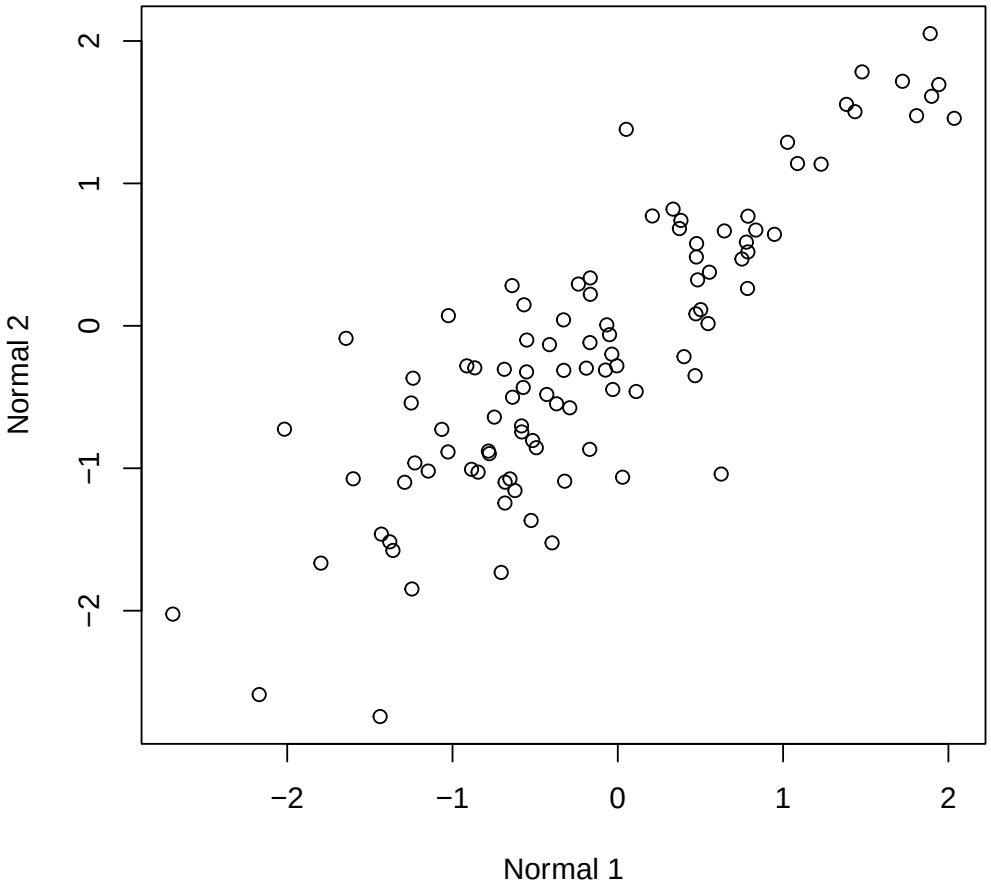


Figura 3.14: Distribución conjunta de una distribución Normal bidimensional

a) Uniforme discreta con soporte en los primeros 10 naturales. ♢

b) Binomial con $n = 10$ y $p = 1/2$.

c) Poisson(9).

d) Geométrica con un parámetro adecuado para tener un número de iteraciones menor a 3.

5. Describir un algoritmo para generar la distribución Binomial utilizando el algoritmo de la distribución Bernoulli.

6. Generar una muestra aleatoria de una distribución Bin(1000, 1/2) utilizando la definición de función inversa generalizada.

7. Sea $X \sim BinNeg(r, p)$

   a) Use la relación de las distribuciones Binomial Negativa con la Geométrica para dar un algoritmo para generar valores de X e implementarlo.

   b) Demostrar la relación
   $$p_{i+1} = \frac{i(1-p)}{i+1-r} p_i.$$

   c) Usar la relación anterior para dar un segundo algoritmo e implementarlo.

8. Generar muestras de tamaño 1,000 para $X \sim Exp(9)$ usando ♢:

   a) Aceptación-rechazo.

   b) Transformada inversa.

   c) Cociente de uniformes.

   ¿Qué método considera mejor y por qué?.

9. Justificar el Algoritmo 12.

10. Justificar el Algoritmo 25.

11. Implementar los dos algoritmos vistos en este capítulo para la distribución Normal ♢.

12. Probar lo siguiente: Si $X \sim Gamma(1+\alpha, 1)$, $U \sim U(0,1)$, y además son independientes, entonces $XU^{1/\alpha} \sim Gamma(\alpha, 1)$.

13. Implementar un algoritmo para la distribución Gamma con parámetros ♢:

    a) (1.5,3),

    b) (0.5,6).

14. Implementar el algoritmo para generar una muestra de variables aleatorias $Beta(1, 2.3)$. ♢

15. Implementar el algoritmo del Ejemplo 3.4.1 y compararlo con el método para generar variables aleatorias Beta en general.

16. Implementar los algoritmos de las secciones 3.5.3 y 3.5.3 ♢.

17. Generar muestras del vector aleatorio (X, Y) con la siguiente densidad:
    $$f(x, y) = \frac{e^{-\frac{x}{y}} e^{-y}}{y},$$
    cuando $x, y > 0$ y que vale 0 en otro caso.

18. Sea (X, Y) con función de densidad

$$f(x, y) = e^{-y},$$

cuando $0 < x < y$ y que vale 0 en otro caso:

a) Simular Y y $X|Y$

b) Simular X y $Y|X$

19. Considere a la distribución Pareto dada por $F(x) = [1 - \left(\frac{c}{x}\right)^{\alpha}]I(x)_{(c,\infty)}$ donde los parámetros de la distribución son la escala $c > 0$ y la forma $\alpha > 0$. Utilizar el método de la función inversa para implementar un algoritmo que genere números aleatorios con tal distribución.

20. Encontrar las formas explícitas de las cópulas generadas por $\phi(t) = \log(1 - \theta \log t)$ (Cópula de Gumbel-Hougaard) y $\phi(t) = -\log(1 - (1 - t)^{\theta})$ (Cópula de Joe).

21. Implementar el Algoritmo 31 para el caso de la cópula de Frank y Clayton. Para la cópula de Frank, note que la distribución discreta de la transformada inversa de la función inversa del generador puede calcularse fácilmente de forma iterativa.

Capítulo 4

Simulación de Procesos Estocásticos

La importancia del estudio de los procesos estocásticos en el campo actuarial se debe a que son una herramienta poderosa en la solución de problemas financieros, de seguros, economía, investigación de operaciones, entre otros así en toda aquella situación que se presente un cierto grado de incertidumbre en función del tiempo.

En este capítulo nos concentraremos en estudiar el problema de generar trayectorias de una variedad de procesos estocásticos.

4.1. Simulación de cadenas de Markov a tiempo discreto

Consideremos una cadena de Markov $X = \{X_n\}_{n\geq 0}$ a tiempo discreto, con espacio de estados $E = \{1,2,\ldots, N\}$ y homogénea en el tiempo. Supongamos además que su distribución inicial está dada por $\pi = \{\pi_1, \ldots \pi_N\}$ y con matriz de probabilidades de transición:

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix}.$$

Sabemos que, dada una cadena de Markov homogénea, podemos obtener su matriz de transición y su distribución inicia. Ahora nos preguntamos si dada una matriz estocástica y un vector de distribución de probabilidades, es posible asociar una cadena de Markov que tenga a tal matriz como matriz de transición y a tal vector como distribución inicial. El siguiente resultado asegura que sí se puede y su demostración constructiva proveé un algoritmo para la simulación de una cadena de Markov.

Teorema 4.1. *Dada una matriz estocástica de $P \in \mathcal{M}_{N\times N}$ y una distribución inicial π sobre el espacio de estados $E = \{1, 2 \ldots, N\}$, existe un espacio de probabilidad en el que están definidas una sucesión de variables aleatorias $\{X_n\}_{n\in\mathbb{N}}$ con soporte en E, que conforman una cadena Markov homogénea con espacio de estados E, matriz de transición P y distribución inicial π.*

Demostración. Consideremos un espacio de probabilidad $(\Omega, \mathscr{F}, \mathbb{P})$ en el cual está definida una sucesión $\{U_n\}_{n\in\mathbb{N}}$ de variables aleatorias independientes uniformes en el intervalo (0,1). Definimos $\phi_i(y)$ como la función de cuantiles sobre E asociada a la distribución $\{p_{i,j}\}_{1\leq j\leq N}$ y a $\phi_0(y)$ como la función de cuantiles

sobre E asociada a la distribución π, para y en $(0,1)$.

Construimos a continuación el proceso deseado de forma iterativa. Definamos

$$X_0 = \phi_0(U_0)$$

y, habiendo definido a X_n, definimos $X_{n+1} = \phi_{X_n}(U_{n+1})$ para $n \geq 0$.

Analicemos las probabilidades de transición del proceso $\{X_n\}_{n\geq 0}$:

$$\mathbb{P}[X_n = x_n | X_{n-1} = x_{n-1}] = \mathbb{P}[\phi_{X_{n-1}}(U_n) = x_n | X_{n-1} = x_{n-1}].$$

Notemos que X_{n-1} depende exclusivamente de $\{U_k\}_{k\leq n-1}$, además de que las variables aleatorias $\{U_k\}_{k\in\mathbb{N}}$ son independientes. Esto implica que

$$\mathbb{P}[\phi_{X_{n-1}}(U_n) = x_n | X_{n-1} = x_{n-1}] = \mathbb{P}[\phi_{x_{n-1}}(U_n) = x_n] = p_{x_{n-1}x_n},$$

por definición de la función de cuantiles.

Mostraremos ahora que el proceso $\{X_n\}_{n\geq 0}$ propuesto satisface la propiedad de Markov.

$$\begin{aligned}
\mathbb{P}[X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n] &= \mathbb{P}[\phi_0(U_0) = x_0, \phi_{X_0}(U_1) = x_1, \ldots, \phi_{X_{n-1}}(U_n) = x_n] \\
&= \mathbb{P}[\phi_0(U_0) = x_0, \phi_{x_0}(U_1) = x_1, \ldots, \phi_{x_{n-1}}(U_n) = x_n], \\
&= \mathbb{P}[\phi_0(U_0) = x_0]\mathbb{P}[\phi_{x_0}(U_1) = x_1] \ldots \mathbb{P}[\phi_{x_{n-1}}(U_n) = x_n] \\
&= \mathbb{P}[X_0 = x_0]\mathbb{P}[X_1 = x_1 | X_0 = x_0] \ldots \mathbb{P}[X_n = x_n | X_{n-1} = x_{n-1}] \\
&= \mathbb{P}[X_0 = x_0] \prod_{i=0}^{n-1} p_{x_i x_{i+1}}.
\end{aligned}$$

De lo cual se sigue directamente la propiedad de Markov. Por lo tanto $\{X_n\}_{n\in\mathbb{N}}$ es una cadena Markov, con distribución inicial π y matriz de transición P. □

El resultado anterior nos proporciona una justificación de la existencia de cadenas de Markov además de indicarnos un procedimiento para simular trayectorias de éstas. A continuación mostramos tal procedimiento explícitamente, en forma de algoritmo. El algoritmo siguiente genera trayectorias de una cadena de Markov homogénea con distribución inicial π y matriz de transición P hasta el horizonte de tiempo m.

Algoritmo 32 : Cadenas de Markov a tiempo discreto

1: Generar X_0 de la distribución inicial π. Inicializamos n=1.
2: Generamos X_n de la distribución correspondiente al renglón de la matriz de transición asociado al estado X_{n-1}.
3: Si n es igual al horizonte de tiempo m, entonces detenemos el proceso, si no es así actualizamos $n = n + 1$ y volvemos al paso 3.

Para la simulación de cadenas de Markov y el estudio de las características y resultados utilizando el software R está disponible la paquetería `markovchains`. Con esta paquetería podemos constatar cómo se cumplen los resultados teóricos de cadenas de Markov (ver la sección A.2 en el Apéndice A) y diversas herramientas interesantes como el ajuste de una Cadena de Markov a partir de una secuencia de estados.

Ejemplo 4.1.1. *Se simuló una trayectoria de una cadena de Markov usando el Algoritmo 32 con las siguientes características: Matriz de transición*

$$P = \begin{pmatrix} 0{,}2 & 0{,}1 & 0{,}5 & 0{,}2 \\ 0{,}5 & 0{,}2 & 0{,}2 & 0{,}1 \\ 0{,}2 & 0{,}5 & 0{,}1 & 0{,}2 \\ 0{,}3 & 0{,}5 & 0{,}1 & 0{,}1 \end{pmatrix},$$

vector de distribución inicial π

$$\begin{pmatrix} 0{,}2 & 0{,}4 & 0{,}2 & 0{,}2 \end{pmatrix},$$

vector de espacio de estados E

$$\begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix},$$

y horizonte de tiempo m=30. En la Figura 4.1 podemos observar la gráfica de la trayectoria simulada.

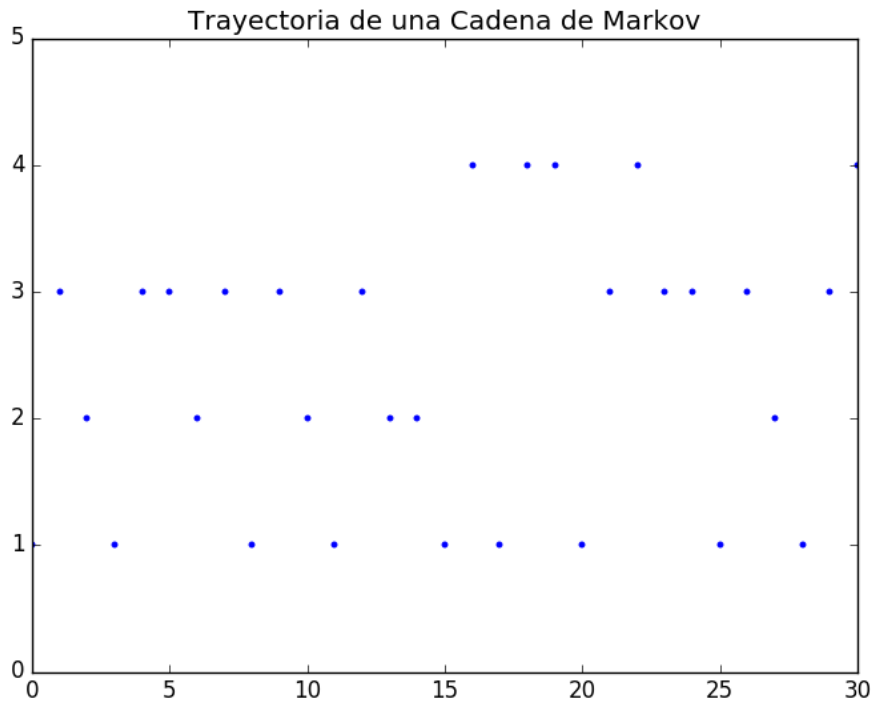


Figura 4.1: Gráfica de una trayectoria de la cadena hasta el tiempo 30.

Veamos a continuación un ejemplo (abordado con más profundidad en la sección A.2.)

Ejemplo 4.1.2. ***Laberinto***

Coloquemos una rata en la esquina inferior izquierda del laberinto de la Figura 4.2 y comida en la esquina superior derecha. Supongamos que estando en cualquiera de los cuartos del laberinto, la rata puede ir a cualquiera de los cuartos vecinos con la misma probabilidad, y que olvida en cada momento las decisiones que tomó con anterioridad. Entonces, su posición en el laberinto se puede modelar por una cadena de Markov con matriz de transición dada por:

$$P = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{pmatrix}$$

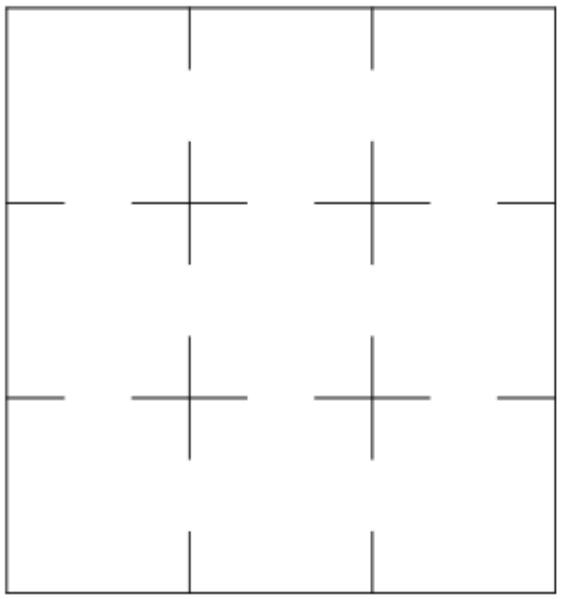


Figura 4.2: Laberinto

En esta paquetería `markovchains` las cadenas se manejan como objetos `markovchain`, a partir de la matriz de probabilidades de transición y el nombre de los estados. La forma en que hemos capturado la cadena del laberinto es

```
> Plab<-array(c(0,1/3,0,1/3,0,0,0,0,0,.5,0,.5,0,.25,0,0,0,0,0,1/3,0,
+             0,0,1/3,0,0,0,.5,0,0,0,.25,0,.5,0,0,0,1/3,0,1/3,0,1/3,
+             0,1/3,0,0,0,.5,0,.25,0,0,0,.5,0,0,0,1/3,0,0,0,1/3,0,0,
+             0,0,0,.25,0,.5,0,.5,0,0,0,0,0,1/3,0,1/3,0),dim=c(9,9))
> Lab<-new("markovchain",transitionMatrix=Plab)
```

A partir de este ejemplo podemos proponer varias preguntas que podemos responder con presición si existe solución analítica o por medio de herramientas de simulación en caso contrario.

1. ¿Cuánto tarda la rata en promedio en encontrar la comida si comienza en el cuarto i?

   Hemos utilizado la función `rmarkovchain` para componer una función que calcula el resultado numéricamente al determinar un número de trayectorias y la casilla de la cual se va a partir,

llamada `Lab.TLlegada` (ver B.3.3). Supongamos que la rata inicia en la casilla 1, los resultados se muestran en el Cuadro 4.1.

	Trayectorias	Real	Estimado
1	10	12.000000	10.600000
2	100	12.000000	15.460000
3	1000	12.000000	13.415000

Tabla 4.1: Estimados del tiempo para llegar a la comida desde 1

2. Si quitamos la comida y simplemente registramos la trayectoria de la rata, ¿cuál es la probabilidad de que se encuentre en el cuarto j en el paso k si comienza en i?

   De manera muy similar podemos simular trayectorias registrando el número de éxitos en los casos en que, partiendo de i, ocurrió que el k-ésimo estado fue j. La implementación sugerida puede realizarse con la función `Lab.Pijk` (ver B.3.4). Tomando por ejemplo $i = 1$, $j = 5$ y $k = 10$ pasos, obtenemos los resultados desplegados en el Cuadro 4.2.

	Trayectorias	Real	Estimado
1	10	0.333333	0.100000
2	100	0.333333	0.160000
3	1000	0.333333	0.185000

Tabla 4.2: Estimados de la probabilidad P(1,5)(10)

3. Si de nuevo seguimos solamente la trayectoria sin comida, ¿estamos seguros de regresar al punto inicial?

   Intuitivamente podemos pensar que la rata podría siempre recorrer todo el laberinto, la función `is.irreducible` nos puede hacer comprobar si la cadena es irreducible:

```
> is.irreducible(Lab)
```

```
[1] TRUE
```

   de modo que podemos concluir que todos los estados son recurrentes, y que $f_{j1} = 1$ para toda $i \in E$, donde $f_{ij}(n) = \mathbb{P}(X_n = j, X_{n-1} \neq j, \ldots, X_1 \neq j | X_0 = i)$ y $f_{ij} = \sum_{k=0}^{\infty} f_{ij}(k)$.

4. Si agregamos la comida, ¿cuántas veces regresará la rata al cuarto inicial antes de encontrar la comida?

   También a partir de trayectorias, podemos considerar la pregunta como un problema de primera visita, pero primero hay que saber cuántas veces paso por el inicio antes de llegar a la comida. Lo cual está implementado en `Lab.jpori` (ver B.3.5), mostramos entonces los resultados para el inicio en 1 y la comida en 9, los resultados se encuentran en el Cuadro 4.3.

De este modo se pueden dar las aproximaciones frecuentistas de las probabilidades deseables, que serán más precisas a medida que corramos más trayectorias en las funciones.

Ejemplo 4.1.3. *El problema de la lluvia.*

	Trayectorias	Recurrencias
1	10	0.600000
2	100	1.150000
3	1000	0.996000

Tabla 4.3: Estimados de los regresos a 1 hasta llegar a la comida

Suponga que un actuario se traslada todos los días de su casa a la oficina por las mañanas y regresa por las tardes. Cada vez que está a punto de trasladarse comprueba antes si está lloviendo (lo cual ocurre independientemente cada mañana o tarde con la misma probabilidad p) para buscar uno de los dos paraguas que posee para salir. Cuando no está lloviendo no se lleva ningún paraguas.

Una motivación natural para modelar este problema es conocer la probabilidad de que el actuario se moje debido a que no tiene paraguas disponibles para salir.

Notemos que la cantidad de paraguas que tendrá disponible al realizar un traslado, no dependerá de cuántos traslados haya realizado con anterioridad. De modo que el siguiente proceso resultará ser una cadena de Markov:

X_n = La cantidad de paraguas disponibles en el destino del n-ésimo traslado.

Notemos que el espacio de estados está dado por $E = \{0, 1, 2\}$ y la matriz de transición correspondiente a la dinámica es:

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1-p & p \\ 1-p & p & 0 \end{pmatrix}.$$

Una observación es que esta modelación puede en caso en que el actuario poseé r paraguas, y se lleva un paraguas cada vez que está lloviendo. En tal caso el espacio de estados es $E = \{0, 1, 2, \ldots, r\}$ y la matriz de transición es de tamaño $r + 1$.

Con la función `Par.markov` (ver B.3.6) podemos generar el objeto `markovchain` que representa esta cadena para r paraguas. Usaremos este modelo para ilustrar las principales funciones de la paquetería `markovchain`.

Supongamos que el caso de dos paraguas y $p = 1/2$, si llegamos a necesitar cosas sencillas de un objeto `markovchain` que tal vez no declaramos nosotros como la probabilidad $p_{i,j}$ o la distribución de $X_n|X_{n-1} = i$ (el renglón del estado i de la matriz), podemos usar las funciones `transitionProbability` y `conditionalDistribution` respectivamente:

```
> Par2<-Par.markov(2,.5)
> Par2
```

```
Unnamed Markov chain
 A  3 - dimensional discrete Markov Chain defined by the following states:
 0, 1, 2
 The transition matrix  (by rows)  is defined as follows:
    0   1   2
0 0.0 0.0 1.0
1 0.0 0.5 0.5
2 0.5 0.5 0.0
```

```
> tP<-transitionProbability(Par2,"1","1")
> tP
```

```
[1] 0.5
```

```
> cD<-conditionalDistribution(Par2,"0")
> cD
```

```
0 1 2
0 0 1
```

Al hablar de una cadena de Markov, tenemos como primer interés conocer características de la cadena como las clases de comunicación y el tipo de estados que tiene, podemos pensar que en este ejemplo no tiene estados absorbente ni transitorios y si todos los estados son recurrentes. Podemos aplicar `absorbingStates`, `transientStates` e `is.irreducible` respectivamente para conocer tales características.

```
> absorbingStates(Par2)
```

```
character(0)
```

```
> transientStates(Par2)
```

```
character(0)
```

```
> is.irreducible(Par2)
```

```
[1] TRUE
```

El output `character(0)` representa al conjunto como vacío. Ahora, de saber que es irreducible tenemos la certeza de que la función `steadyStates` nos proveerá de la distribución estacionaria de la cadena.

```
> DE<-steadyStates(Par2)
> DE
```

```
       0   1   2
[1,] 0.2 0.4 0.4
```

Las probabilidades de primera visita tienen la complicación de hablar de las posibles trayectorias donde estrictamente se empieza en i y que llegan por primera vez a j exáctamente en el paso n. Por lo que se necesita realizar una procesión de cálculos, para evitar j antes de haber realizado n transiciones, los cuales se implementa en `firstPassage` que calcula una matriz de tantos renglones como pasos especificados y tantas columnas como estados de la cadena.

Entonces para calcular $f_{ij}(n)$ habrá que pedir n renglones y escoger la columna representante del destino. Por ejemplo, buscamos la probabilidad de que en una mañana tengamos un paraguas en la casa, uno en la oficina y que hasta la mañana siguiente (dos pasos) se tenga la misma cantidad, representada por $f_{1,1}(2)$, tiene respuesta una repuesta analítica sencilla pues representa una trayectoria única

$$f_{1,1}(2) = p_{1,2}p_{2,1} = \frac{1}{2}\frac{1}{2} = \frac{1}{4}$$

se puede calcular con la función de la siguiente forma:

```
> Trans2<-firstPassage(Par2,"1",2)
> Trans2[2,2]#El estado 1 corresponde a la segunda columna
```

```
[1] 0.25
```

La utilidad principal de este modelo es lograr tener una idea de la cantidad de veces que el actuario va a mojarse. Lo cual puede ocurrir cuando no tiene sombrillas disponibles en algún paso. Para lo cual, se puede encontrar la distribución estacionaria π de la cadena, podemos decir que la probabilidad de estar sin paraguas transcurrido el tiempo es la entrada π_0 y que dada la probabilidad de lluvia p, entonces la proporción de veces que se moja es

$$p\pi_0 = \frac{p(1-p)}{3-p}$$

que en su caso, con ayuda de `markovchains`, para el ejemplo se calcularía

```
> moj<-.5*DE[1]
> moj
```

```
[1] 0.1
```

De manera similar se puede encontrar analíticamente la respuesta analítica para este problema con el caso de r paraguas, de modo que podemos comprobar la utilidad de la paquetería cuando podamos manejar una cadena de Markov con características más complejas.

Ejemplo 4.1.4. ***Cadena de Ehrenfest.*** *Se tienen dos urnas , entre ambas hay un total de N bolas. En cada paso se elige una de las bolas al azar y se cambia de urna. Si X_n = número de bolas en la urna A después de n ensayos, entonces $X = \{X_n\}_{n\geq 0}$ define una cadena de Markov.*

La distribución estacionaria de esta cadena es única y proporciona la forma en que se comporta desde el inicio pues la evolución se mantiene desde el principio. Con la función `Ehr.markov` *(ver B.3.7) nos ayudará a generar el objeto* `markovchain` *para N bolas.*

Por otro lado, podemos encontrar la forma en que la frecuencia de los estados en trayectorias simuladas de la cadena pueden aproximar la distribución estacionaria. Generando una cadena con $N = 4$ comparamos la distribución y la frecuencia de los estados en trayectorias. Dicha comparación se presenta en la Figura 4.3

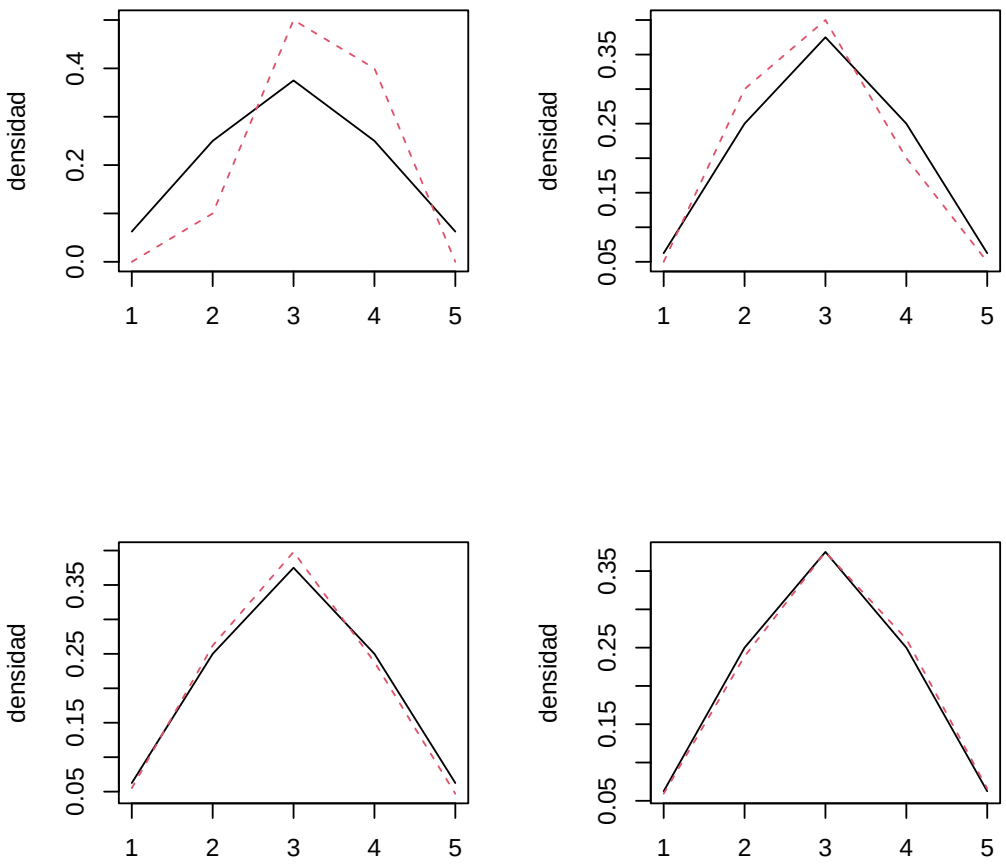


Figura 4.3: Aproximación de la distribución estacionaria para la cadena de Ehrenfest

4.2. Simulación de un proceso de Poisson

4.2.1. Simulación de un proceso de Poisson homogéneo

Suponga que se quieren generar los primeros n eventos de un proceso Poisson $\boldsymbol{N} = \{N_t\}_{t\geq 0}$ de intensidad $\lambda > 0$. Para ello nos apoyaremos del resultado de que el tiempo transcurrido entre dos eventos sucesivos cualesquiera son una sucesión de variables aleatorias independientes exponenciales de parámetro λ. De manera que para generar un proceso Poisson basta con generar esos tiempos de interarribo.

Si generamos n números aleatorios $U_1, U_2, \ldots, U_n$ y hacemos $X_i = -\dfrac{1}{\lambda}\log U_i$, entonces las X_i pueden considerarse como los tiempos entre el $(i-1)$-ésimo y el i-ésimo evento de un proceso Poisson. Supongamos que queremos generar la trayectoria de un proceso de Poisson en el intervalo finito $[0, T]$, podemos utilizar el siguiente algoritmo. En este algoritmo t se refiere al tiempo, I es el número de eventos que han ocurrido al tiempo t y $S(I)$ es el tiempo acumulado hasta el último evento.

Algoritmo 33 : Generación de trayectorias de un proceso de Poisson

1: Hacer $t = 0$, $I = 0$
2: Generar $U \sim U(0, 1)$
3: Hacer $t = t - \dfrac{1}{\lambda}\log U$. Si $t > T$, detenerse
4: En caso contrario, hacer $I = I + 1$ y $S(I) = t$
5: Ir al paso 2.

Con el Algoritmo 33 vamos a hacer estimaciones de la forma usual, con la simulación del proceso de Poisson, para el siguiente ejemplo. La implementación del Algoritmo 33 está disponible en la función `PPoisson.S` (ver B.3.8).

Ejemplo 4.2.1. ***Reclamaciones de una compañía aseguradora.*** *Suponga que las reclamaciones es una compañía de seguros ocurren de acuerdo a un proceso de Poisson con intensidad diaria $\lambda = 2$. Generando trayectorias del proceso de Poisson correspondiente, responderemos la siguientes preguntas.*

1. ¿Cuál es la probabilidad que en una semana se presenten 5 reclamaciones?
   Mediante el uso de la función `PPoisson.P` (ver B.3.9) damos el estimado numérico por simulaciones presentados en el Cuadro 4.4

	Trayectorias	Real	Estimado
1	10	0.003727	0.000000
2	100	0.003727	0.010000
3	1000	0.003727	0.002000
4	10000	0.003727	0.002900

Tabla 4.4: Estimados de la probabilidad P(N7=5)

2. ¿Cuál es el número de reclamaciones esperadas en un mes? La función `PPoisson.E` (ver B.3.10) nos ayuda a responder esta pregunta en el Cuadro 4.5.

3. ¿Cuál es el tiempo esperado hasta la octava reclamación? La función `PPoisson.TE` (ver B.3.11) nos ayuda a responder esta pregunta en el Cuadro 4.6.

	Trayectorias	Real	Estimado
1	10	60	60.800000
2	100	60	59.550000
3	1000	60	59.546000

Tabla 4.5: Estimados de E(N30)

	Trayectorias	Real	Estimado
1	10	4	3.704041
2	100	4	3.926290
3	1000	4	4.027999

Tabla 4.6: Estimados de E(S8)

Notemos que la metodología de la respuesta 1, con algo de espera, nos puede ayudar a aproximar probabilidades cuya respuesta analítica no es posible calcular cuando λt o x son muy grandes como para calcular $(\lambda t)^x$ o $x!$.

Ejemplo 4.2.2. ***Reclamaciones por monto de una compañía aseguradora*** *Suponga que las reclamaciones se han categorizado según el monto de reclamación. Si se tienen 5 diferentes categorias con vector de probabilidades* $p = (1/9, 1/18, 3/18, 1/18, 10/18)$ *para cada categoría. Si además suponemos que las reclamaciones se presentan de acuerdo a un proceso de Poisson con intensidad diaria* $\lambda = 3$*. Generamos trayectorias para dar solución a las siguinetes cuestiones.*

1. ¿Cuál es la probabilidad que en una semana se presenten 5 reclamaciones de la categoría 5? Sabiendo que los procesos son independientes entonces puede encontrarse como

$$\mathbb{P}[N_7^5 = 5] = e^{-\lambda p_5(7)} \frac{\lambda p_5(7))^5}{5!} = e^{-3(10/18)(7)} \frac{(3(10/18)(7))^5}{5!} = 3{,}8713 \times 10^{-43}$$

   Lo cuál nos da un ejemplo de una estimación por simulación de casos favorables altamente costosa en tiempo de implementación pues necesitaríamos más de 10^{43} simulaciones para notar algún resultado.

2. ¿Cuál es el número de reclamaciones esperadas en un mes de cada categoría? También por ser independientes se puede calcular con facilidad, por ejemplo para la categoría 1 sería:

$$\mathbb{E}[N_{30}^1] = \lambda p_1(30) = 3(1/9)(30) = 10$$

   Estimamos de esta forma con el parámetro nuevo en el Cuadro 4.7

	Trayectorias	Real	Estimado
1	10	10	10.300000
2	100	10	9.980000
3	1000	10	10.190000

Tabla 4.7: Estimados de E(N1,30)

3. ¿Cuál es el tiempo esperado hasta la octava reclamación de la categoría 1? De la misma forma, como proceso independiente, el tiempo de espera del N^1 distribuye como una variable aleatoria $Gamma(n, \lambda p_1)$, por lo que el tiempo de esperado es

$$\mathbb{E}[S_8^1] = \frac{8}{\lambda p_1} = \frac{8}{3/9} = 24$$

	Trayectorias	Real	Estimado
1	10	24	24.011851
2	100	24	24.377846
3	1000	24	23.899895

Tabla 4.8: Estimados de E(S8)

que podemos estimar en el Cuadro 4.8

4. Si en una semana se presentan 3 reclamaciones, ¿cuál es la probabilidad que todas sean del mismo tipo? Para responder esta pregunta necesitamos generar las 5 trayectorias de las categorías simultaneamente, lo cuál se dejará como ejercicio par el lector.

4.2.2. Simulación de Proceso de Poisson no homogéneo

El proceso de conteo extremadamente importante para propósitos de modelación es el proceso Poisson no homogéneo, el cual relaja el supuesto de incrementos estacionarios del proceso Poisson. Esto abre la posibilidad a que la tasa de arribo no necesariamente sea constante, sino que pueda variar en el tiempo, en esta sección se presenta un algoritmo para simular trayectorias de este proceso.

Proposición 4.1. *Consideremos un proceso de Poisson no homogéneo* $\boldsymbol{N} = \{N_t\}_{t\geq 0}$ *como en la Definición A.28. Supongamos que* $sup_{t>0}\lambda(t) \leq \lambda$. *Sea* $\boldsymbol{N}^* = \{N_t^*\}_{t\geq 0}$ *un proceso de Poisson con parámetro* λ *con tiempos de ocurrencia* $\{T_n^*\}_{n\geq 1}$ *e independiente de una sucesión de variables aleatorias* $\{U_n\}_{n\geq 1}$ *independientes e identicamente distribuidas uniformes en el intervalo* $(0, 1)$. *Entonces, se puede simular a* $\boldsymbol{N}$ *haciendo*

$$N_t = \sum_{n\geq 1} \mathbb{I}_{\{T_n^*\leq t, U_n\leq \lambda(T_n^*)\}}.$$

La demostración se deja como ejercicio para el lector.

Suponga que deseamos simular una trayectoria de un proceso de Poisson no homogéneo en un intervalo de tiempo $[0, T]$ con función de intensidad $\lambda(t)$. El método que revisaremos a continuación se conoce con el nombre de **muestreo aleatorio**.

Este comienza eligiendo un valor λ el cual es tal que

$$\lambda(t) \leq \lambda \quad \text{para toda} \quad t \leq T.$$

Tal proceso Poisson no homogéneo puede generarse seleccionando aleatoriamente el evento tiempos de un proceso Poisson con tasa λ. Es decir, si un evento de un proceso Poisson con tasa λ que ocurre al tiempo t es contabilizado con probabilidad $\lambda(t)/\lambda$, entonces el proceso de los eventos contabilizados es un proceso Poisson no homogéneo con función de intensidad $\lambda(t)$ para $0 \leq t \leq T$.

Por lo tanto, simulando un proceso Poisson y contabilizando aleatoriamente sus eventos, podemos generar un proceso Poisson no homogéneo. De esta forma podemos describir el algoritmo deseado como sigue.

Nota 4.1. *La justificación de que el Algoritmo 34 genera trayectorias de un proceso de Poisson no homogéneo se sigue de la Proposición 4.1.*

Algoritmo 34 : Proceso de Poisson no homogéneo

1: Hacer $t = 0$, $I = 0$
2: Generar un número aleatorio $U \sim U(0, 1)$
3: Hacer $t = t - \frac{1}{\lambda} \log U$. Si $t > T$, detenerse
4: En caso contrario, generar otro número aleatorio $U \sim U(0, 1)$
5: Si $U \leq \lambda(t)/\lambda$, hacer $I = I + 1$ y $S(I) = t$
6: Regresar al paso 2.

La implementación del Algoritmo 34 se presenta en la función `PPoissonNH.S` (ver B.3.12), la cuál puede funcionar con cualquier función de intensidad $\lambda : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}^+ \cup \{0\}$ acotada.

En la Figura 4.4 podremos apreciar la forma en que actúa la función de intensidad con la ocurrencia de saltos cuando tenemos $\lambda(t) = t/2$ y $\lambda(t) = 1 + sen(t)$.

Un resultado clásico de procesos estocásticos nos dice que, dado un proceso Poisson no homogéneo, $\{X_t : t \geq 0\}$, de intensidad $\Lambda(t) = \int_0^t \lambda(u)du$ invertible, si definimos al proceso $\{N_t : t \geq 0\}$ por

$$N_t := X_{\Lambda^{-1}(t)} \qquad \forall t \geq 0,$$

ocurre que $\{N_t : t \geq 0\}$ es un proceso Poisson homogeneo de parámetro 1. Sin embargo, con el propósito de simular un proceso Poisson no homogéneo, utilizaremos el resultado análogo siguiente:

Teorema 4.2. *Sea $\{N_t : t \geq 0\}$ un proceso Poisson de parámetro 1 y una función de intensidad $\Lambda(t) = \int_0^t \lambda(u)du$. Si definimos al proceso $\{X_t :\geq 0\}$ por*

$$X_t := N_{\Lambda(t)} \qquad t \geq 0,$$

entonces $\{X_t : t \geq 0\}$ es un proceso Poisson no homogéneo de intensidad $\Lambda(t)$.

La demostración de este resultado puede consultarse en el Teorema A.9 en el Apéndice.

4.2.3. Simulación de Proceso de Poisson compuesto

En esta sección formalizamos de manera algorítmica la simulación de un proceso de Poisson compuesto $\boldsymbol{X} = \{X_t\}_{t\geq 0}$ donde

$$X_t = \sum_{i=1}^{N_t} Y_i,$$

con $\boldsymbol{N} = \{N_t\}_{t\geq 0}$ un proceso de Poisson y $\{Y_i\}_{i\geq 1}$ una sucesión de variables aleatorias independientes e identicamente distribuidas con distribución F. El siguiente algoritmo describe la manera de simular trayectorias de $\boldsymbol{X}$ en el intervalo de tiempo $[0, T]$.

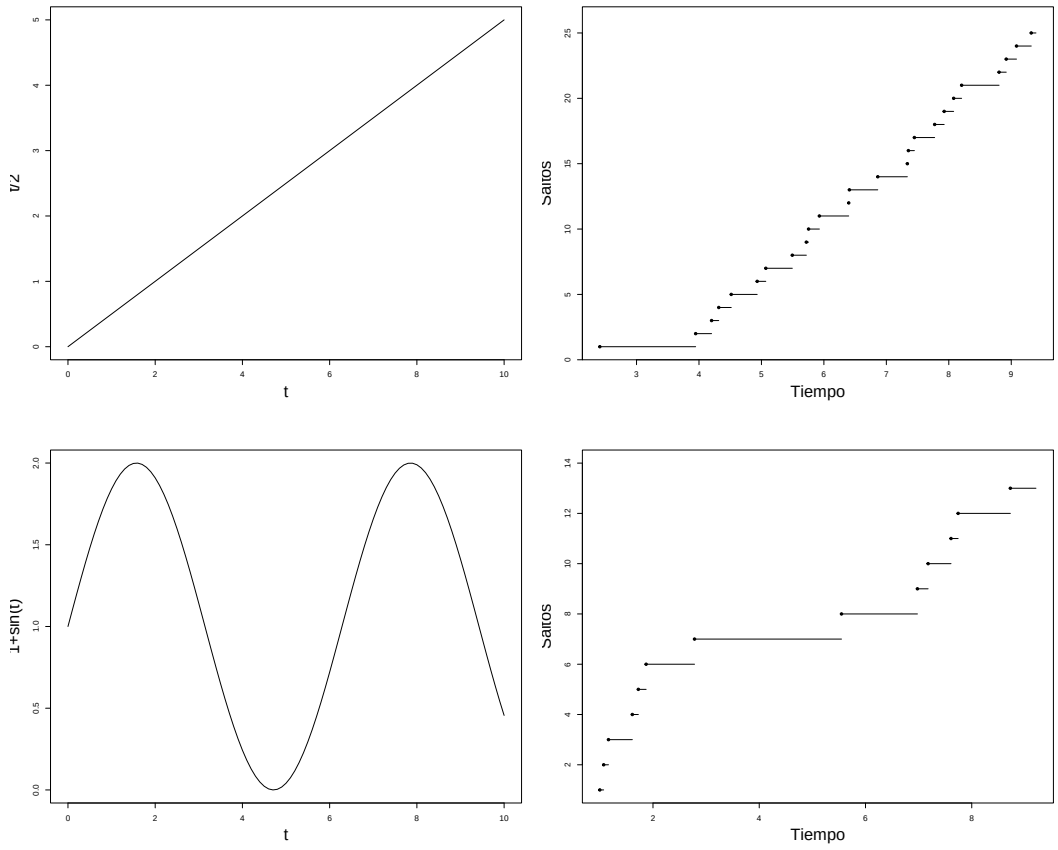


Figura 4.4: Proceso de Poisson no Homogéneo

Algoritmo 35 : Proceso de Poisson Compuesto

1: Hacer $t = 0$, $I = 0$.
2: Generar $U \sim U(0, 1)$
3: Hacer $t = t - \frac{1}{\lambda} \log U$. Si $t > T$, detenerse
4: En caso contrario, generar hacer $I = I + 1$,y $S(I) = t$
5: Generar $Y_I \sim F$.
6: Hacer $X_T = \sum_{i=1}^{I} Y_i$
7: Ir al paso 2.

La función `PPoiCE.S` (ver B.3.13) ejemplifica el caso en que las saltos siguen una distribución exponencial. Con lo anterior podemos también implementar la simulación del modelo de Cramér-Lundberg, con el que es posible obtener trayectorias del capital de una compañía de seguros a través del tiempo para estimar resultados de interés, como la probabilidad de ruina o tiempo esperado de ruina, como se muestra en la Figura 4.2.3 para el caso de arribos $exp(.4)$ y saltos $exp(.8)$ con un capital inicial de 10 unidades.

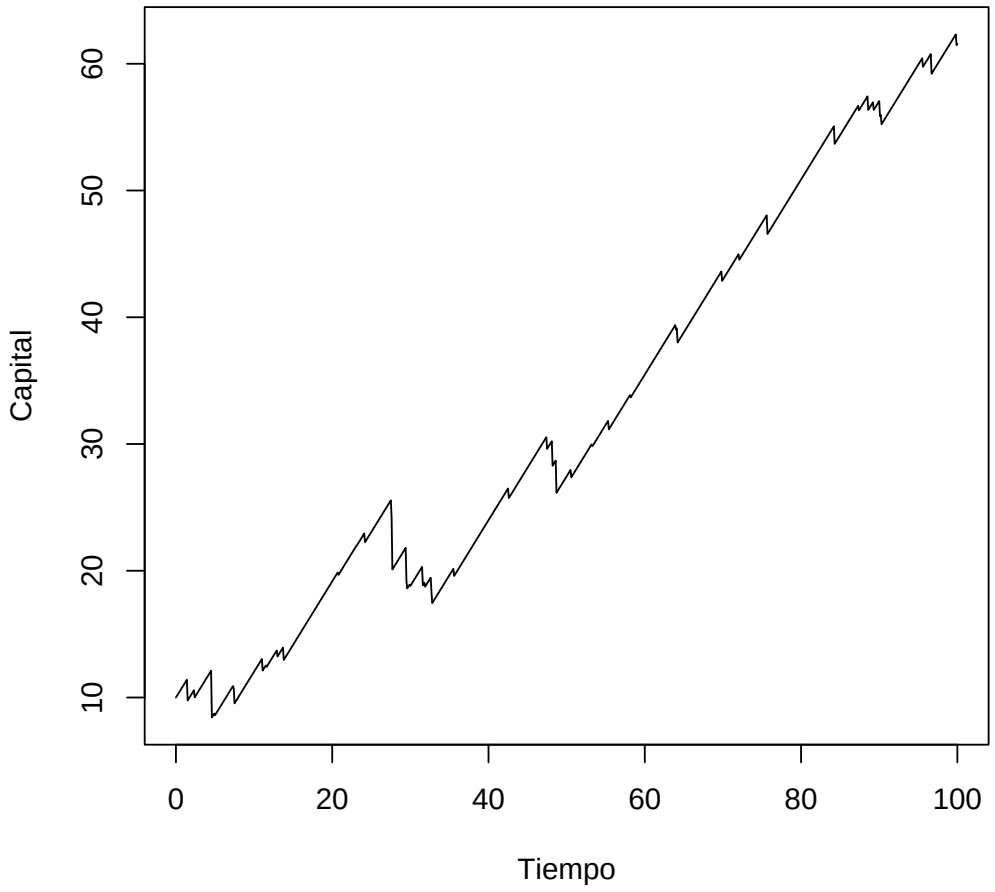


Figura 4.5: Modelo de Cramer-Lundberg

4.2.4. Simulación de Proceso de Poisson Espacial

En esta sección introduciremos un modelo para la distribución de puntos en el espacio bajo ciertas heterogeneidades, mejor conocido como Poisson Espacial.

Algunos conceptos previos se pueden consultar en A.3.5.

Proposición 4.2. *Un proceso Poisson espacial X con medida media μ satisface que*

1. *Para cada S, ocurre que $X(S) \sim Poisson(\mu(S))$.*
2. *Cuando $S_1, ..., S_r$ son conjuntos disjuntos, las variables aleatorias $\{X(S_i)\}_{i=1}^r$ son independientes.*

Demostración. Ver A.18. □

Ejemplo 4.2.3. *Tomemos el caso particular en que $\lambda(t) \equiv \lambda > 0$. En este caso*

$$\mu(S) = \int_S \lambda dt = \lambda \int_S dt = \lambda |S|,$$

para todo S integrable; donde $|S|$ denota al n-volumen del conjunto $/S/$ (en el caso en que $n = 1$, el n-volumen es la longitud, para $n = 2$ el área, para $n = 3$ el volumen usual). Al proceso Poisson que tiene por medida media $\mu(S) = \lambda|S|$ se le conoce como proceso Poisson espacial homogéneo. El caso $n = 1$ corresponde al proceso Poisson homogéneo usual (aunque definido en toda la recta real). La Ecuación (A.2), para una colección de intervalos disjuntos $\{(A_i, b_i]\}_{i=1}^r$, en este último caso se transforma en

$$\mathbb{P}(X(a_1, b_1] = k_1, ..., X(a_r, b_r] = k_r) = \prod_{i=1}^{r} \frac{[\lambda(b_i - a_i)]^{k_i} e^{-\lambda(b_i - a_i)}}{k_i!}.$$

Ejemplo 4.2.4. *Consideremos una función $\Lambda : \mathbb{R} \to \mathbb{R}$ no decreciente y derivable, con función derivada λ. Debido a que Λ es no decreciente, λ es no negativa. Definamos a μ a través de λ, por medio de la Ecuación A.1. Entonces el proceso Poisson espacial X, definido en la recta real, coincide con la definición de un proceso de Poisson no homogéneo con intensidad Λ. La Ecuación (A.2) toma la forma aquí:*

$$\mathbb{P}(X(a_1, b_1] = k_1, ..., X(a_r, b_r] = k_r) = \prod_{i=1}^{r} \frac{[\Lambda(b_i) - \Lambda(a_i)]^{k_i} e^{-(\Lambda(b_i) - \Lambda(a_i))}}{k_i},$$

para intervalos disjuntos.

Tenemos la siguiente proposición.

Proposición 4.3. *Sea X un proceso de Poisson espacial en $\mathbb{R}^n$ con medida media μ. Sea $S \subseteq \mathbb{R}^n$ un conjunto integrable tal que $\mu(S) < \infty$. Condicionado al evento $\{X(S) = n\}$, para todo $A_1, ..., A_n \subseteq S$ disjuntos e integrables ocurre que*

$$\mathbb{P}(X(A_1) = n_1, ..., X(A_k) = n_k | X(S) = n) = \frac{n}{n_0! n_1! ... n_k!} p(A_0)^{n_0} p(A_1)^{n_1} ... p(A_k)^{n_k},$$

para todas los valores $\{n_1, ..., n_k\}$ tales que $\sum_{i=1} k n_i = n$. Aquí p es la función de conjuntos (subconjuntos de S) definida por $p(A) := \frac{\mu(A)}{\mu(S)}$, el valor $n_0 := n - \sum_{i=1}^n n_i$ y $A_0 := S \backslash \left(\bigcup_{i=1}^n A_i \right)$. En otras palabras, condicionado al evento $\{X(S) = n\}$, el vector $(X(A_0), ..., X(A_r))$ distribuye $Multinomial(n, r, p(A_0), ..., p(A_r))$.

Demostración. Ver A.19. □

Una observación importante es que en el caso en que $\mu(\cdot) = \lambda|\cdot|$, la Proposición A.19 nos dice que $(X(A_0), ..., X(A_r))$ tiene distribución $Multinomial\Big(n, r, \frac{|A_0|}{|S|}), ..., \frac{|A_r|}{|S|}\Big)$. Esto implica que los n puntos (a los cuáles estamos condicionando al proceso X a tener en el conjunto S) están distribuidos de forma independiente y uniformemente sobre el conjunto S. Note como esto generaliza una propiedad similar para el proceso Poisson homogéneo en $[0, \infty)$.

La observación anterior nos permite simular con facilidad a un proceso Poisson homogéneo en $\mathbb{R}$ con tasa λ en un subconjunto S. El algoritmo es el siguiente.

Algoritmo 36 : Proceso de Poisson alternativo

1: Simular una variable aleatoria N con distribución $Poisson(\lambda|S)$.
2: Dado $N = n$, simular n variables aleatorias independientes uniformes en S.
3: Devolver las posiciones de las n variables aleatorias generadas.

Note que en el penúltimo paso, para simular una variable aleatoria uniforme en un conjunto integrable y acotado S basta simular variables aleatorias en una caja $[a_1, b_1] \times ... \times [a_n, b_n]$ que contenga a S y utilizar el método de aceptación-rechazo para asegurarnos que tal punto pertenece a S.

Habiendo implementado el algoritmo anterior, podemos generalizar el método propuesto para simular un Proceso Poisson no homogéneo en $[0, \infty)$ para simular un proceso Poisson espacial con tasa $\lambda(t)$. Supondremos que la tasa $\lambda(t)$ es acotada en S, es decir, existe λ^* tal que $\lambda(t) \leq \lambda^*$ para todo t en S.

Algoritmo 37 : Proceso de Poisson espacial

1: Simular una variable aleatoria N con distribución $Poisson(\lambda^*|S|)$.
2: Dado $N = n$, simular n variables aleatorias independientes uniformes en S, denotadas por $\{X_1, ..., X_n\}$.
3: Simular $\{U_1, ..., U_n\}$ variables aleatorias uniformes en $(0, 1)$ independientes.
4: Para cada i, conservar al punto X_i cuando ocurra que $U_i \leq \frac{\lambda(X_i)}{\lambda^*}$.
5: Devolver los puntos que fueron conservados.

La prueba de que el algoritmo anterior funciona es análoga a la prueba de que el algoritmo para Proceso Poisson no homogéneo en $[0, \infty)$ funciona.

Para ilustrar el algoritmo del Proceso Poisson Espacial, se deja como ejercicio al lector 19

4.3. Simulación del Proceso de Saltos de Markov

La característica crucial para la simulación de un Proceso de Saltos de Markov es la indepencia de los tiempos de estancia, cuya distribución es exponencial con parámetros correspondientes a la tasa de intensidad del estado. De este modo, al concluir un tiempo de estancia simulado sabemos con que probabilidad se dará un salto a un estado distinto necesariamente.

Por el Teorema A.11 se tiene que la forma de simular un Procesos de Saltos de Markov, a traves de una trayectoria por tiempo de estadía se presenta a continuación.

Algoritmo 38 : Proceso de Saltos de Markov

1: Inicializar t_0. Generar Y_0 de la distribución inicial de Y . Hacemos $X_0 = Y_0$ y $n = 0$.
2: Hacemos $Y_n = i$
3: Generamos $T_{n+1} \sim exp(q_i)$.
4: Hacemos $t_{n+1} = t_n + T_{n+1}$.
5: Ahora $X_n = Y_n$ para $t_n \leq t < t_{n+1}$
6: Generamos Y_{n+1} de la distribución correspondiente de K.
7: Ahora $n = n + 1$, regresamos al paso 2.

Para poder generar un trayectoria del Proceso de Saltos de Markov hay que considerar una matriz de intensidades compatible con las características del generador infinitesimal que representa los parámetros necesarios, en la función `PSM.S` (ver B.3.14) implementamos el algoritmo. Veamos en la Figura 4.6 usando la matriz de intensidades

$$Q = \begin{pmatrix} -0.5 & 0.25 & 0.25 \\ 0.4 & -1 & 0.6 \\ 0.1 & 0.1 & -0.2 \end{pmatrix}.$$

y una distribución inicial $\pi = (1/3, 1/3, 1/3)$.

4.4. Simulación de procesos con valores continuos

4.4.1. Simulación del Movimiento Browniano y sus transformaciones

Definición 4.1. *Un Movimiento Browniano Estándar (MB) es un proceso* $\boldsymbol{B} = \{B_t : t \geq 0\}$ *markoviano con incrementos estacionarios e independientes tal que*

1. $\mathbb{P}[B_0 = 0] = 1$,
2. B *tiene trayectorias continuas casi seguramente, y*
3. $B_t - B_s \sim N(0, t - s), \qquad \forall 0 \leq s \leq t.$

Este proceso es un proceso fundamental en la teoría de procesos estocásticos. La primera razón para ello es que es un proceso continuo análogo a la caminata aleatoria simple; su dinámica es simétrica en la posición y equiprobable. La segunda razón es su **universalidad**. En matemáticas, la universalidad de un modelo significa que aparece constantemente: así como la distribución normal aparece como un límite de sumas (centradas y escaladas) de variables aleatorias independientes y arbitrarias (bajo algunos supuestos generales), el Movimiento Browniano aparece como el límite (centrado y escalado) de procesos arbitrarios (bajo algunos supuestos generales).

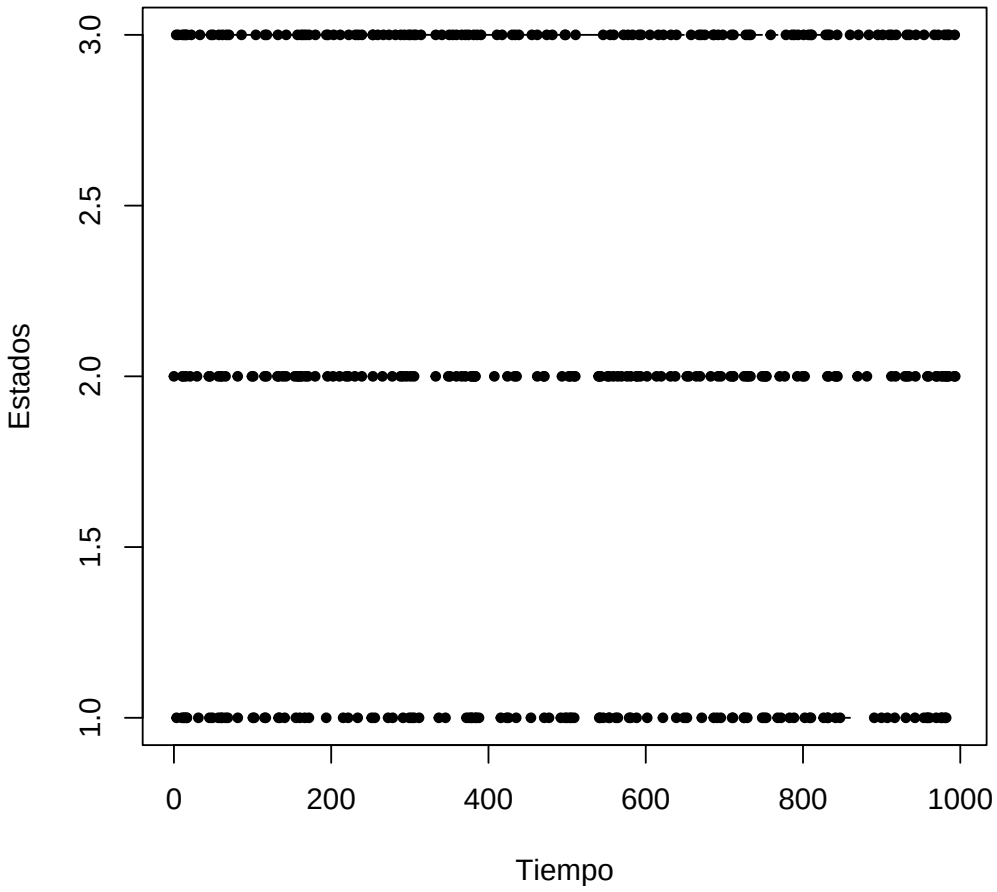


Figura 4.6: Trayectoria de un proceso de saltos de Markov

Las trayectorias del movimiento Browniano son no diferenciables en ninguna parte y satisfacen fractalidad, es decir, que un intervalo seleccionado de las valuaciones del Movimiento Browniano, tras escalarse, sigue teniendo la misma distribución que la trayectoria del Movimiento en $[0, 1]$. A pesar de tener tales extranñas propiedades, hay muchas cantidades que pueden calcularse debido a que los incrementos tienen la distribución normal y son independientes. En la literatura puede encontrarse una enciclopedia de resultados sobre las probabilidades de que las trayectorias satisfagan una característica dada. Esto afianza más aún su uso en modelos aplicados.

Un Movimiento Browniano (no estándard) de parámetros μ (media o deriva), y $\sigma^2 > 0$ (coeficiente de difusión o volatilidad) es un proceso que satisface las mismas propiedades que el Movimiento Browniano estándar pero tal que

$$B_t - B_s \sim N(\mu(t-s), \sigma^2(t-s)), \qquad 0 \leq s \leq t.$$

4.4.2. Construcción de Lévy del Movimiento Browniano

Esta construcción se basa en interpolaciones lineales entre puntos provenientes de una distribución Normal estándar.

El objetivo es construir un Movimiento Browniano en el intervalo [0,1] como un elemento aleatorio en el espacio C[0,1] de las funciones continuas en [0,1]. La idea es construir la distribución conjunta correcta del Movimiento Browniano paso a paso en los conjuntos finitos:

$$D_n = \left\{ \frac{k}{2^n} : 0 \leq k \leq 2^n \right\}$$

de puntos diádicos. Posteriormente interpolamos los valores en D_n de manera lineal. El resultado de Lévy asegura que el límite de estás funciones continuas existe y tiene la distribución de un Movimiento Browniano. Para ello, definamos $D = \bigcup_{n=1}^{\infty} D_n$ y sea $\{Z_t : t \in D\}$ una colección de variables aleatorias independientes con distribución normal estándar . Sea $B(0) := 0$ y $B(1) := Z_1$. Para cada $n \in \mathbb{N}$ podemos definir variables aleatorias $B(d)$ tales que:

1. Para $r < s < t$ en D_n, la variable aleatoria $B(t) - B(s)$ distribuye normal con media cero y varianza $t - s$, y $B(t) - B(s) \perp B(s) - B(r)$.

2. Las colecciones $(B(d) : d \in D_n)$ y $(Z_t \in D \setminus D_n)$ son independientes.

Notemos que si n=1, la construcción solo comprende al conjunto $D_0 = \{0, 1\}$. Cuando $n \geq 2$, definimos a $B(d)$ para $d \in D_n \setminus D_{n-1}$ como sigue:

$$B(d) = \frac{B(d - 2^{-n}) + B(d + 2^{-n}))}{2} + \frac{Z_d}{2^{\frac{n+1}{2}}}$$

Formalmente definimos:

$$F_0(t) = \begin{cases} Z_1 & t = 1 \\ 0 & t = 0 \\ \textit{interpolacion lineal} & t \in (0, 1), \end{cases}$$

y para $n \geq 1$

$$F_n(t) = \begin{cases} 2^{\frac{-(n+1)}{2}} Z_t & t \in D_n \setminus D_{n-1} \\ 0 & t \in D_{n-1} \\ \textit{interpolacion lineal} & \textit{en puntos consecutivos de } D_n. \end{cases}$$

A continuación mostramos una gráfica de un Movimiento Browniano generado por el algoritmo de la construcción de Lévy.

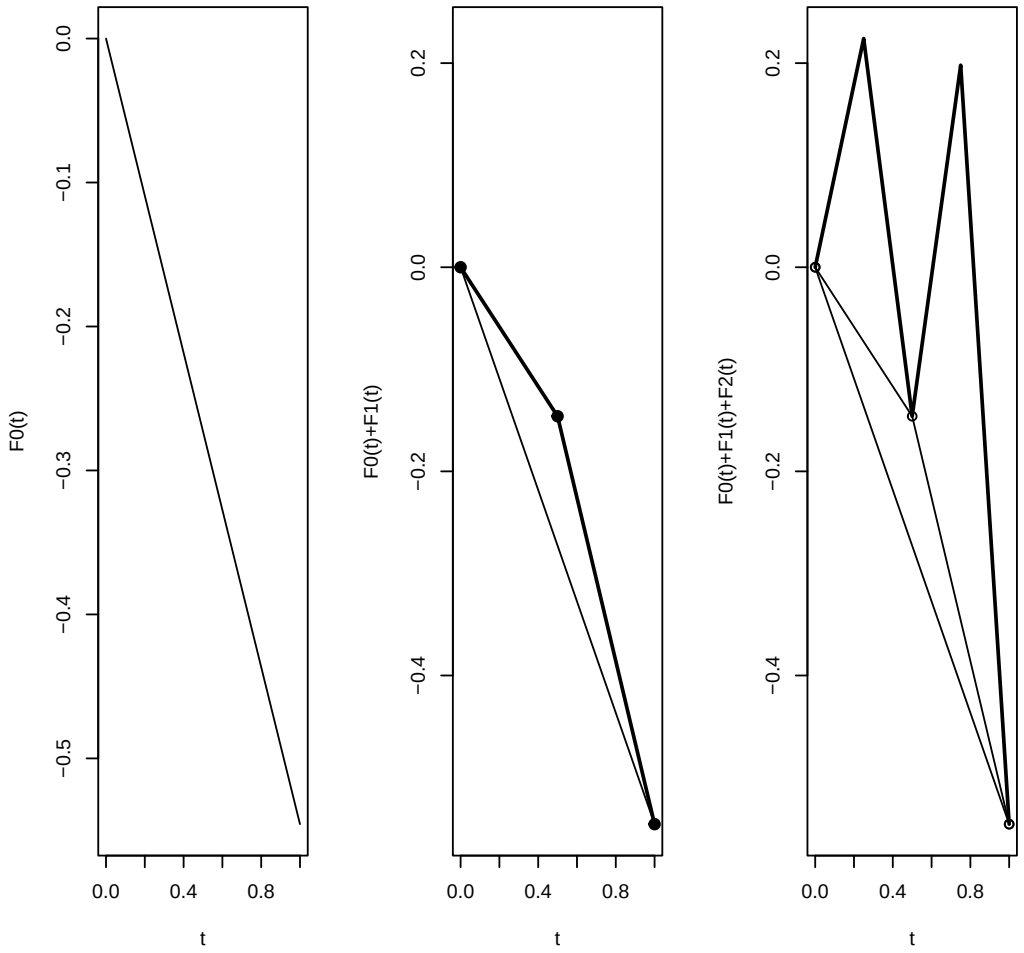


Figura 4.7: Movimiento Browniano mediante construcción de Lévy

Algoritmo 39 : Simulación del Movimiento Browniano mediante la construcción de Lévy

1: Generar los conjuntos D_i tal que $i \leq n$.

2: Para cada punto $d \in D_i \setminus D_{i+1}$ definimos a $B(d) = \frac{B(d-2^{-i})+B(d+2^{-i}))}{2} + \frac{Z_d}{2^{\frac{i+1}{2}}}$.

3: Regresar al punto anterior cuando $i \leq n$.

A partir del Movimiento Browniano podemos definir más procesos como sus transformaciones.

Definición 4.2. *Un movimiento Browniano geométrico G está definido como la siguiente transformación del Movimiento Browniano estándar B:*

$$G_t := G_0 \exp\Big\{\Big(\mu - \frac{\sigma^2}{2}\Big)t + \sigma B_t\Big\}.$$

Cabe mencionar, que en el modelo de Black-Scholes-Merton, se supone que el activo financiero tiene un comportamiento aleatorio modelado por un Movimiento Browniano Geométrico.

Simulemos al Movimiento Browniano Geométrico utilizando la transformación anterior siguiendo la idea de la construcción de Euler para el Movimiento Browniano.

Algoritmo 40 : Simulación del Movimiento Browniano Geométrico

1: Generar una partición equidistante de n puntos del intervalo $[0, t]$, $P = \left[t_i | t_i = \frac{it}{n}, t_0 = 0, t_n = t, 0 < i < n\right]$

2: Para cada punto en la partición t_i tal que $0 < i < n$, generar un punto del Movimiento Browniano correspondiente al intervalo $[t_i, t_{i-1}]$.

3: Notemos que multiplicar y sumar constantes afectan de manera determinista al Movimiento Browniano, pues son reescalamientos y traslaciones respectivamente, obteniendo así $(\mu - \frac{\sigma^2}{2})t_i + \sigma B(t_i)$.

4: Finalmente aplicar la transformación exponencial $G_0 \exp\left((\mu - \frac{\sigma^2}{2})t_i + \sigma B(t_i)\right)$, donde G_0 es un valor inicial dado por el usuario.

Ejemplo 4.4.1. *Generaremos un Movimiento Browniano Geométrico con los siguiente parámetros:* $G_0 = 0.5$, $\mu = 0$, $\sigma = 0.05$, $t = 1$ *y* $n = 1000$.

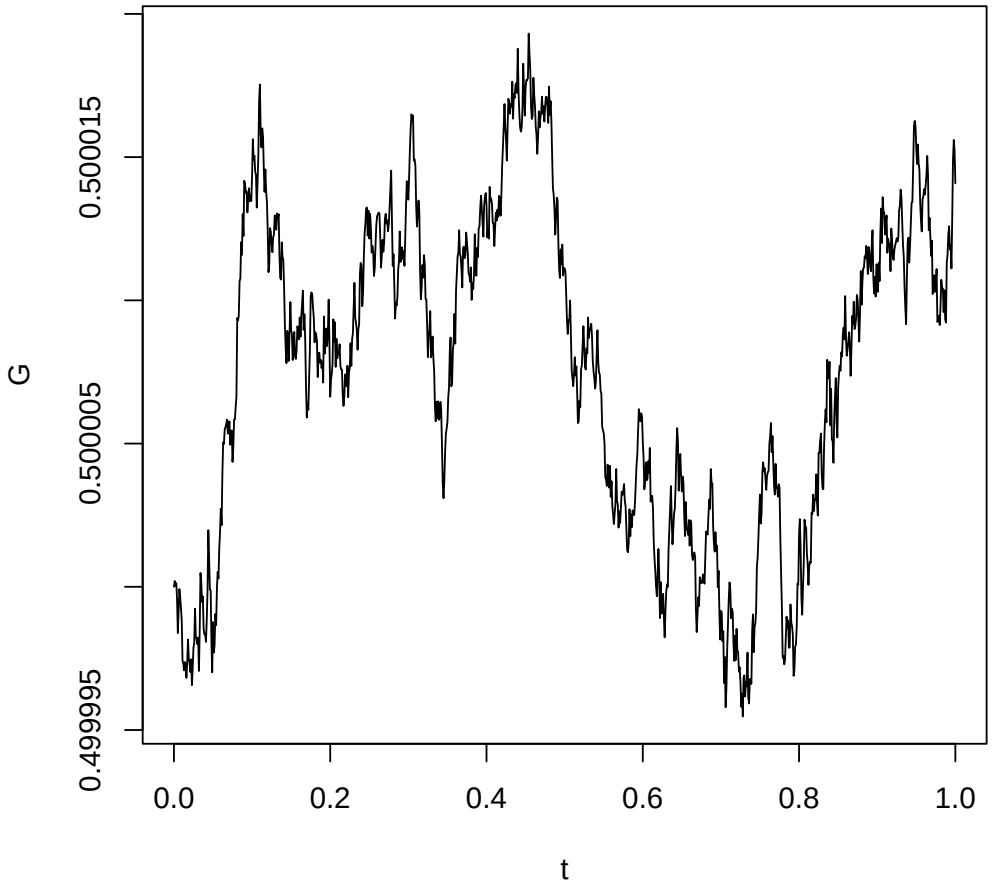


Figura 4.8: Aproximación del Movimiento Browniano Geométrico por Transformaciones del Movimiento Browniano

Definición 4.3. *Un proceso de Ornstein-Uhnlenbeck o proceso de Vasicek es un proceso* X *definido como*

la siguiente transformación del Movimiento Browniano estándar B:

$$X_t := X_0 e^{-at} + b(1 - e^{-at}) + \frac{\sigma e^{-at}}{\sqrt{2a}} B(e^{2at} - 1).$$

Este modelo se utiliza ampliamente para modelar tasas de interés o demográficas. Tiene la desventaja de que tiene posibilidad de tomar valores negativos. El parámetro b representa la media a largo plazo, al parámetro a se le conoce como velocidad de reversión hacia la media, y σ es la volatilidad del proceso.

Para llevar a cabo la simulación notemos que los primeros dos sumandos del proceso son deterministas, es decir, no tienen parte aleatoria. Sin embargo, el último término depende de un Movimiento Browniano con escala; una función del tiempo.

Para poder utilizar el algoritmo de Euler, tenemos que elegir una partición adecuada del intervalo $[0, t]$. Definamos la transformación $u(s) = \frac{ln(s+1)}{2a}$. Notemos que simular $B(s)$ en puntos de la forma $\{u(s) : s \in [0, t]\}$, es equivalente a simular $B(\exp(2as) - 1)$ en puntos de la forma $\{s : s \in [0, t]\}$.

Algoritmo 41 : Simulación del proceso de Ornstein-Uhnlenbeck.

1: Genera la partición $P = \left\{t_i | t_i = \frac{it}{n}, t_0 = 0, t_n = t, 0 < i < n\right\}$.
2: Generar la partición Q mediante la transformación $S(t_i) = \frac{\ln(t_i+1)}{2a}$.
3: Para cada $q_i \in Q$ y $0 \leq i \leq n$ generamos puntos del Movimiento Browniano.
4: Finalmente definimos $X_{q_i} = X_0 \exp(-aq_i) + b(1 - \exp(-aq_i) + \frac{\sigma \exp(-aq_i)}{\sqrt{2a}} B(\exp(2aq_i - 1)) = X_0 \exp(-aq_i) + b(1 - \exp(-aq_i) + \frac{\sigma \exp(-aq_i)}{\sqrt{2a}} B(\exp(t_i))$.

Ejemplo 4.4.2. *Generaremos un proceso de Ornstein-Uhnlenbeck con los siguiente parámetros:* $a = 0.01$ $b = 0.08$ $\sigma = 0.005$ *y* $x_0 = 0.04$.

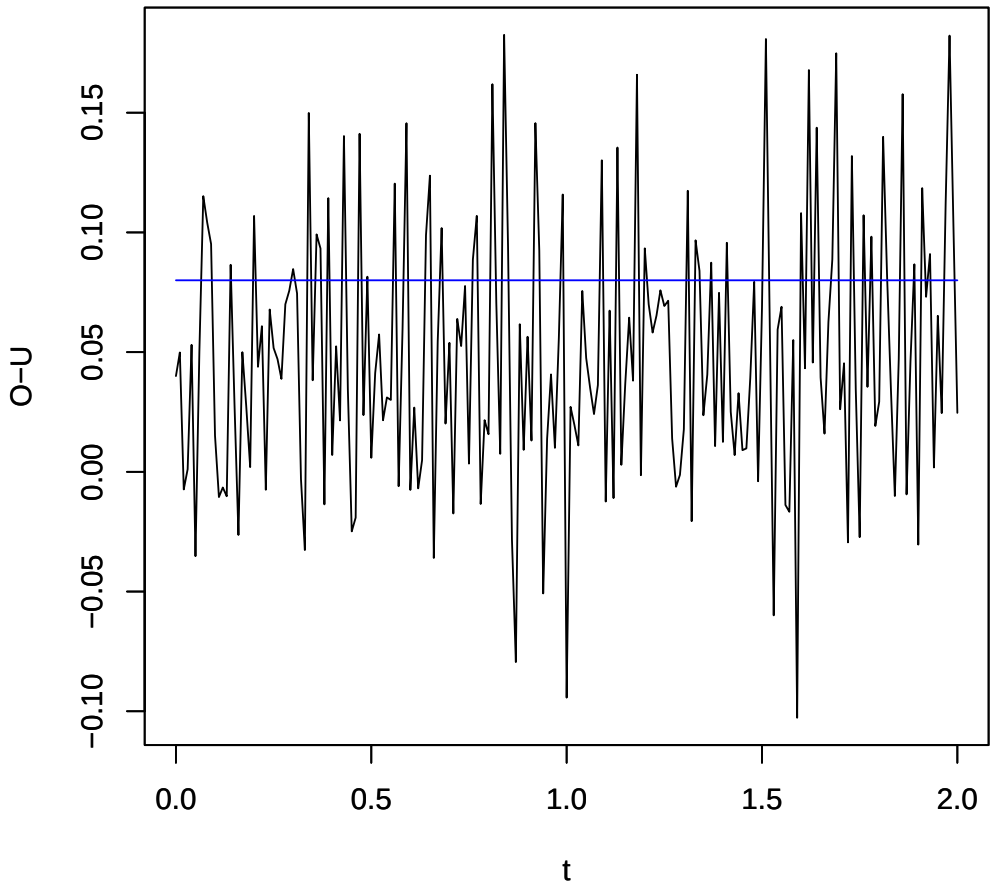


Figura 4.9: Aproximación del proceso de Ornstein-Uhlenbeck por Transformaciones del Moviento Browniano

4.4.3. Simulación de Ecuaciones Diferenciales Estocásticas

En esta sección desarrollaremos métodos para simular a las soluciones a ecuaciones diferenciales estocásticas. La mayoría de los métodos de simulación para este tipo de procesos están basados en aproximaciones discretas de la solución continua.

4.4.4. Método de Euler

Consideremos un proceso de difusión $X = \{X_t\}_{t\geq 0}$ que es solución a la ecuación diferencial estocástica

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t$$

con valor inicial determinista $X_{t_0} = x_0$ y un nivel de discretización

$$0 = t_0 < t_1 < ... < t_n = T$$

en el intervalo $[0, T]$. La aproximación de Euler de X es un proceso estocástico continuo Y, que satisface el siguiente sistema iterativo:

$$Y_{t_{i+1}} = Y_{t_i} + \mu(Y_{t_i}, t_i)\Delta t_i + \sigma(Y_{t_i}, t_i)\Delta W_i$$

para $i = 0, 1, \ldots, n-1$, con $Y_0 = x_0$ y

$$\Delta t_i = t_{i+1} - t_i,$$

$$\Delta W_i = W_{t_{i+1}} - W_{t_i},$$

por lo que $\Delta W_i \sim N(0, \Delta t_i)$.

Entre cualesquiera dos momentos de observación t_i y t_{i+1}, el proceso puede ser definido por una interpolación diferenciable. Una propuesta natural es considerar una interpolación lineal para Y_t, definida por

$$Y_t = Y_{t_i} + \frac{t - t_i}{t_{i+1}}(Y_{t_{i+1}} - Y_{t_i})$$

para $t \in [t_i, t_{i+1})$. Ocurre que el método de Euler tiene un orden fuerte de convergencia con $\tau = 1/2$ (ver **??**).

Ejemplo del método de Euler

Aquí, se presenta una simulación de un proceso de difusión con el esquema de Euler, consideramos el proceso de Ornstein-Uhlenbeck, que es una solución de la ecuación diferencial estocástica

$$dX_t = \alpha(b - X_t)dt + \sigma dW_t,$$

donde $\alpha > 0$, $\sigma > 0$ y $b \in \mathbb{R}$ son los parámetros del proceso y W es un proceso de Wiener estándar. Si suponemos que $X_0 = 0$ y consideremos un intervalo de realización $[0, 1]$ con un nivel de discretización $0 = t_0 < t_1 < ... < t_n = 1$, donde $n = 100$ y $\Delta t_i = 0.01$, para todo $i = 1, 2, ..., n-1$, entonces el método de Euler nos da la siguiente aproximación:

$$Y_{t_{i+1}} = Y_{t_i} + \alpha(b - Y_{t_i})\Delta t_i + \sigma\Delta W_i$$

En la siguiente Figura 4.10 se presenta una realización del proceso cuando $\alpha = 0.5$, $\sigma = 2.5$ y $b = 0$ en el intervalo $[0, 1]$.

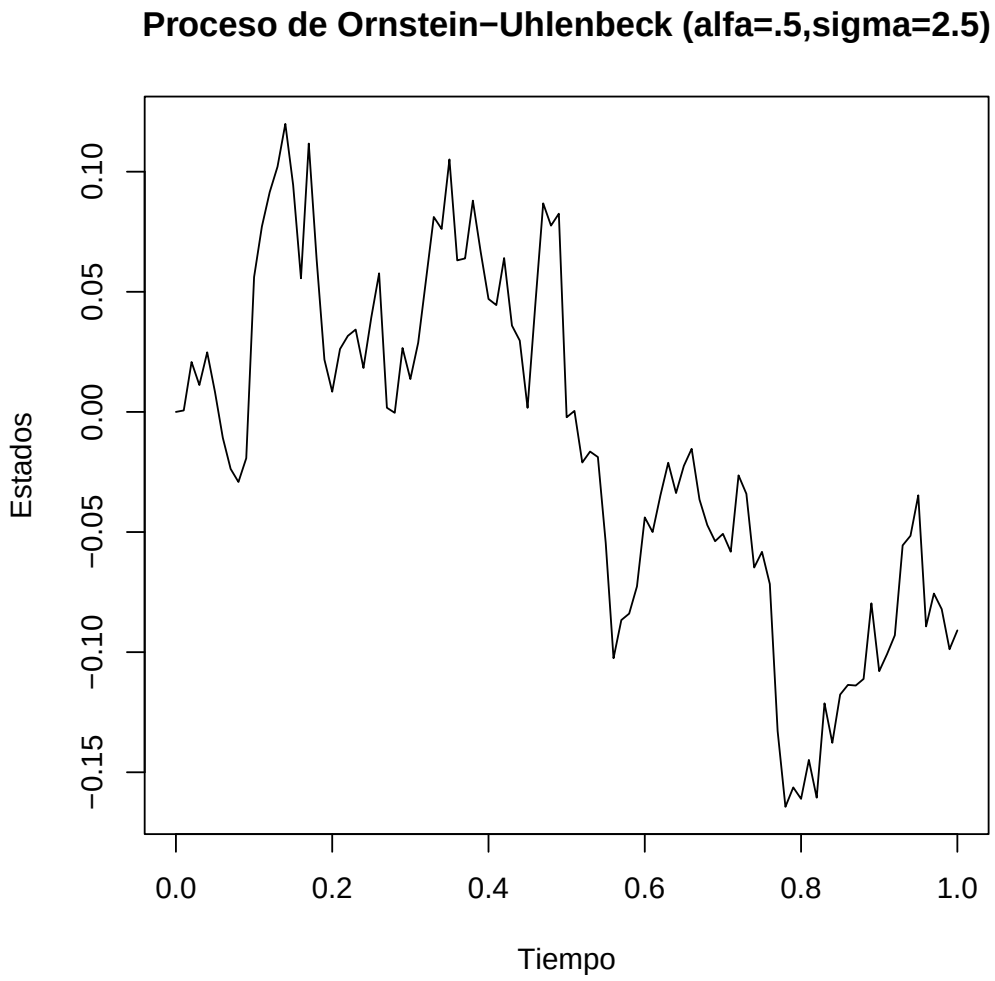


Figura 4.10: Aproximación del proceso de Ornstein-Uhlenbeck por el Método de Euler

4.4.5. Método de Milstein

El esquema de Milstein es un refinamiento del modelo de Euler cuyo objetivo radica en reducir el error de discretización y no se debe confundir con un modelo de reducción de varianza.

En la aproximación de Milstein se utiliza el lema de Itô, perteneciente a la teoría del Cálculo Estocástico, para aumentar la precisión en la aproximación mediante la incorporación del término de segundo orden.

Considerando la discretización del tiempo de la siguiente manera $0 = t_0 < t_1 < .... < t_n = T$. Para cada $i = 0, 1, ..., n-1$ esta aproximación se puede escribir como sigue:

$$Y_{t_{i+1}} = Y_{t_i} + \mu(Y_{t_i}, t_i)\Delta t_i + \sigma(Y_{t_i}, t_i)\Delta W_i + \frac{1}{2}\sigma(Y_{t_i}, t_i)\sigma'(Y_{t_i}, t_i)[(\Delta W_i)^2 - \Delta t_i].$$

Con

$$\Delta ti = t_{i+1} - t_i,$$

$$\Delta W_i = W_{t_{i+1}} - W_{t_i},$$

por lo que $\Delta W_i \sim N(0, \Delta t_i)$.

De donde no es difícil demostrar que $\Delta W_i = \sqrt{\Delta t_i} Z_{t_{i+1}}$, con $Z_{t_{i+1}} \sim N(0,1)$. Por lo que simular $Y_{t_{i+1}}$ se reduce a simular;

$$Y_{t_{i+1}} = Y_{t_i} + \mu(Y_{t_i}, t_i)\Delta t_i + \sigma(Y_{t_i}, t_i)\sqrt{\Delta t_i} Z_{t_{i+1}} + \frac{1}{2}\sigma(Y_{t_i}, t_i)\sigma'(Y_{t_i}, t_i)\Delta t_i[(Z_{t_{i+1}})^2 - 1].$$

Tal aproximación tiene convergencia débil y fuerte de orden uno. Para mas información consultar A.6.

Ejemplo de Método de Milstein

Para ejemplificar el uso de este método de simulación consideremos el movimiento Browniano geométrico, el cuál es solución a la ecuación diferencial estocástica

$$dX_t = \theta_1 X_t dt + \theta_2 X_t dW_t.$$

Para este proceso, tenemos que $\mu(x,t) = \theta_1 x$, $\sigma(x,t) = \theta_2 x$ y $\sigma'(x,t) = \theta_2$. Así, el esquema de Milstein para este proceso es:

$$\begin{aligned} Y_{t_{i+1}} &= Y_{t_i} + \theta_1 Y_{t_i}\Delta t_i + \theta_2 Y_{t_i}\Delta W_i + \frac{1}{2}\theta_2^2 Y_{t_i}[(\Delta W_i)^2 - \Delta t_i] \\ &= Y_{t_i}(1 + \theta_1\Delta t_i - \frac{\theta_2^2}{2}\Delta t_i + \theta_2\Delta W_i + \frac{\theta_2^2}{2}(\Delta W_i)^2) \\ &= Y_{t_i}(1 + \theta_1\Delta t_i - \frac{\theta_2^2}{2}\Delta t_i + \theta_2\sqrt{\Delta t_i} Z_{t_{i+1}} + \frac{\theta_2^2}{2}\Delta t_i Z_{t_{i+1}}^2) \end{aligned}$$

Con $Z_{t_{i+1}} \sim N(0,1)$.

Supongamos $X_0 = 5$ y consideremos un intervalo de realización $[0,1]$ con un nivel de discretización $0 = t_0 < t_1 < ... < t_n = 1$, para todo $i = 1, 2, \ldots, n-1$. Presentamos una simulación de este proceso usando $\theta = (1.0, 0.5)$ en el intervalo $[0,1]$. Presentamos una gráfica de tal simulación.

4.4.6. La paquetería `sde`

Ahora que conocemos la implementación bajo discretización, podemos encontrar algunas de la ecuaciones diferenciales estocásticas más comunes o usadas en la práctica implementadas en la paquetería `sde`.

En los ejemplos anteriores mencionamos al proceso de Ornstein-Uhlenbeck, también conocido como proceso de Vasicek, que se ha utilizado para modelos diversos: procesos biológicos y fisiológicos, tasas de interés, tasas mortalidad, etc. El Movimiento Browniano Geométrico, conocido también en la literatura como Black-Scholes-Merton (BS), modela el precio de activos financieros. Otro proceso muy útil, conocido como proceso de Cox-Ingersoll-Ross (CIR), está definido como solución de la siguiente ecuación diferencial estocástica,

$$dX_t = (\theta_1 - \theta_2 X_t)dt + \theta_3\sqrt{X_t} dB_t,$$

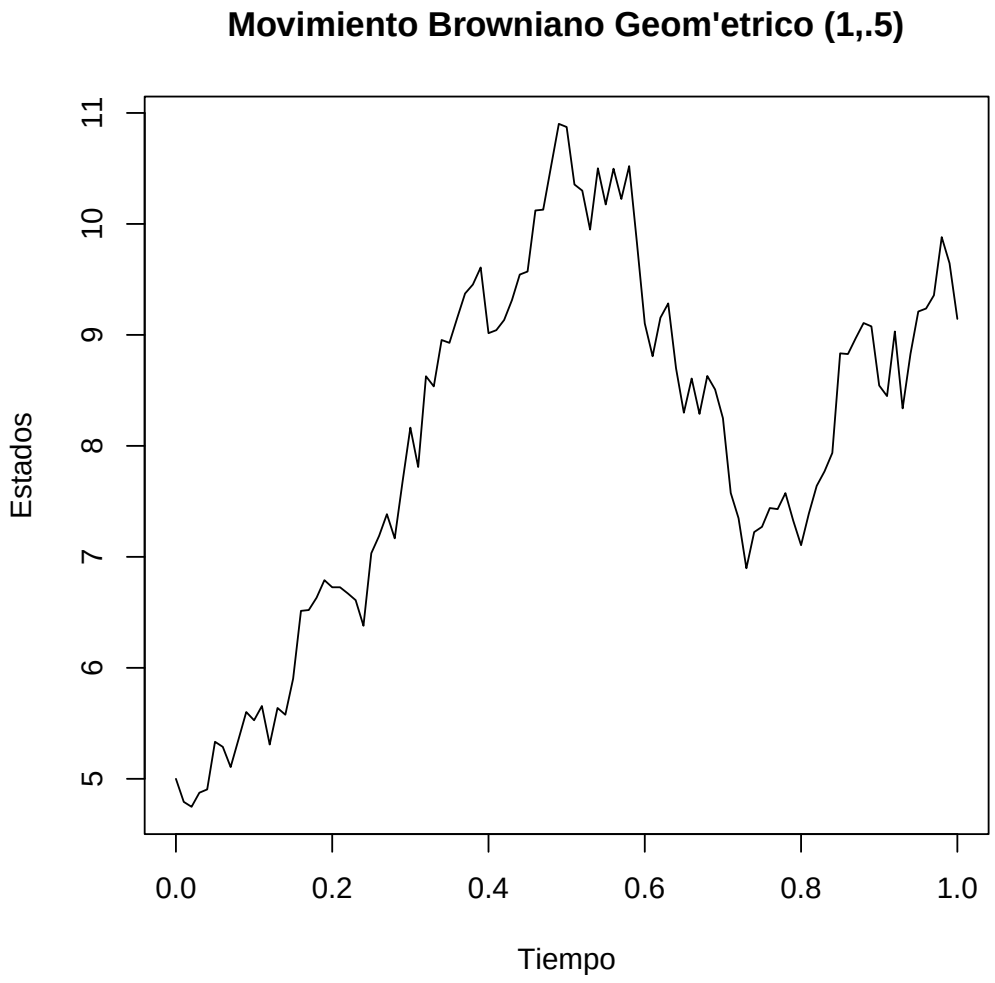


Figura 4.11: Aproximación del Movimiento Browniano Geométrico a través del Método de Milstein

donde $\theta_1, \theta_2, \theta_3 > 0$ y $2\theta_1 > \theta_3^2$.

Bajo el esquema me Milstein, tenemos que:

$$\mu(X_t, t) = \theta_1 - \theta_2 X_t.$$

$$\sigma(X_t, t) = \theta_3 \sqrt{X_t}.$$

$$\sigma^{'}(X_t, t) = \frac{1}{2} \frac{\theta_3}{\sqrt{X_t}}$$

Consideremos la siguiente partición del intervalo $[0, 1]$ como: $0 = t_0 < t_1 < .... < t_n = T$
Entonces, bajo el esquema de Milstein:

$$X_{t_{i+1}} = X_{t_i} + (\theta_1 - \theta_2 X_t)(t_{i+1} - t_i) + \theta_3 \sqrt{X_{t_i}} \sqrt{t_{i+1} - t_i} Z_{t_{i+1}} + \frac{\theta_3^2}{4}(t_{i+1} - t_i)(Z_{i+1}^2 - 1)$$

con $\{Z_{t_i}\}$ variables aleatorias normales independientes.

Simulamos el Modelo CIR utilizando lo anterior para $\theta_1 = 1.2$, $\theta_2 = 1$, $\theta_3 = 0.3$, $T = 1$ y $n = 100$ con valor inicial $X_0 = 0.1$

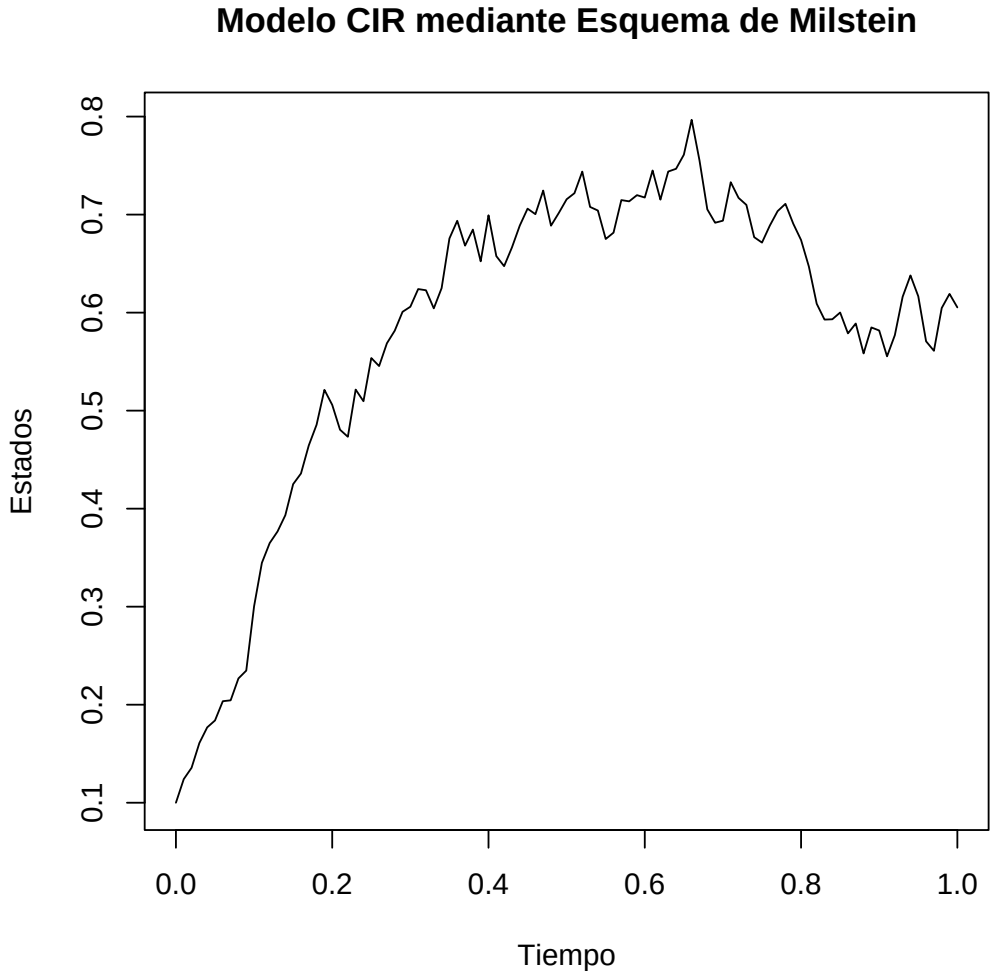


Figura 4.12: Modelo CIR bajo Esquema de Milstein

Ahora si realizamos la transformación $Y_t = 2cX_t$, obtenemos que el proceso Y tiene distribución marginal condicional al valor inicial conocida (χ^2 no central); a partir de esto se puede encontrar la distribución condicional del proceso original

$$X_t | X_0 \sim \chi^2_{(\nu,\eta)}$$

donde ν son los grados de libertad y η el parámetro no centralizado

$$c = \frac{2\theta_2}{\theta_3^2(1 - e^{-\theta_2 t})},$$

$$\nu = \frac{4\theta_1}{\theta_3^2},$$

$$\eta = x_0 e^{-\theta_2 t}.$$

Este proceso tiene la particularidad de mantenerse en valores positivos siempre, lo cuál es una característica deseable en modelos de tipos de cambio o de tasas.

En cada uno de los ejemplos anteriores, dado que se conoce la distribución condicional, puede simularse el proceso como variables aleatorias condicionados al valor de X_0 y para un intervalo de tiempo arbitrario. El lenguaje de programación R tiene precargadas la función de densidad (dc), función de distribución (pc), cuantíl (qc) y simulación (rc) de tales marginales, bajo la nomenclatura OU para Ornstein-Uhlenbeck, BS para el Browniano Geométrico y CIR para Cox-Ingersoll-Ross.

De modo que si desearamos realizar una simulación del proceso CIR de parámetro $\theta = (1, 2, 1)$, $\Delta = 1/100$, $T = 1$ y $X_0 = 1$, usando la paquetería `sde` podemos hacerlo de la siguiente forma:

```
> library(sde)
> proc<-numeric(101)
> proc[1]<-1
> for(i in 2:101){
+   proc[i]<-rcCIR(n=1,Dt=.01,x0=proc[i-1],theta=c(1,2,1))
+ }
```

Aún cuando no tengamos un proceso gaussiano, podemos seguir recurriendo a la discretización mediante Euler o Milstein, que no son precisamente los únicos métodos, para únicamente necesitar los parámetros de deriva y difusión. Según el método escogido se necesitará tener conocimiento de las derivadas parciales de ésas funciones, para lo cual en la función `sde.sim` se tiene disponible distintas implementaciones. Notemos la gran cantidad de parámetros que debe tener la función; se recomienda leer con cuidado la documentación de la paquetería.

Supongamos que queremos simular alguna ecuación diferencial estocástica en particular en $[0, 1]$ con 100 pasos. Tomemos como ejemplo el Proceso Hiperbólico Simple, entonces tenemos el parámetro de deriva $\mu(x, t) = -\theta X_t/\sqrt{1 - X_t^2}$ y de difusión $\sigma(x, t) = 1$, especificando el método de discretización de Milstein se requiere también especificar $\sigma_x(x, t) = 0$, supongamos $\theta = 1$, los parámetros del intervalo que necesitamos (t0=0,T=1,N=100) son los valores predeterminados y debemos especificar X0=0. Entonces la forma de implementar el ejemplo es el siguiente

```
> d<-expression(-1*x/sqrt(1-x^2))
> s<-expression(1)
> sx<-expression(0)
> set.seed(123)
> PH<-sde.sim(X0=0,drift=d,sigma=s,sigma.x=sx,method="milstein")
```

De manera muy simular la función `BBridge` está constituida para simular un Puente Browniano directamente, por lo que se requieren de menos especificaciones. Con lo que también se tiene disponible una para el Movimiento Browniano Estándar y para el movimiento Browniano Geométrico, `BM` y `GBM` respectivamente, que requieren menos parámetros. Con los valores determinados para estas funciones se pueden obtener las simulaciones directamente, a continuación mostramos un Puente Browniano usando `BBridge`.

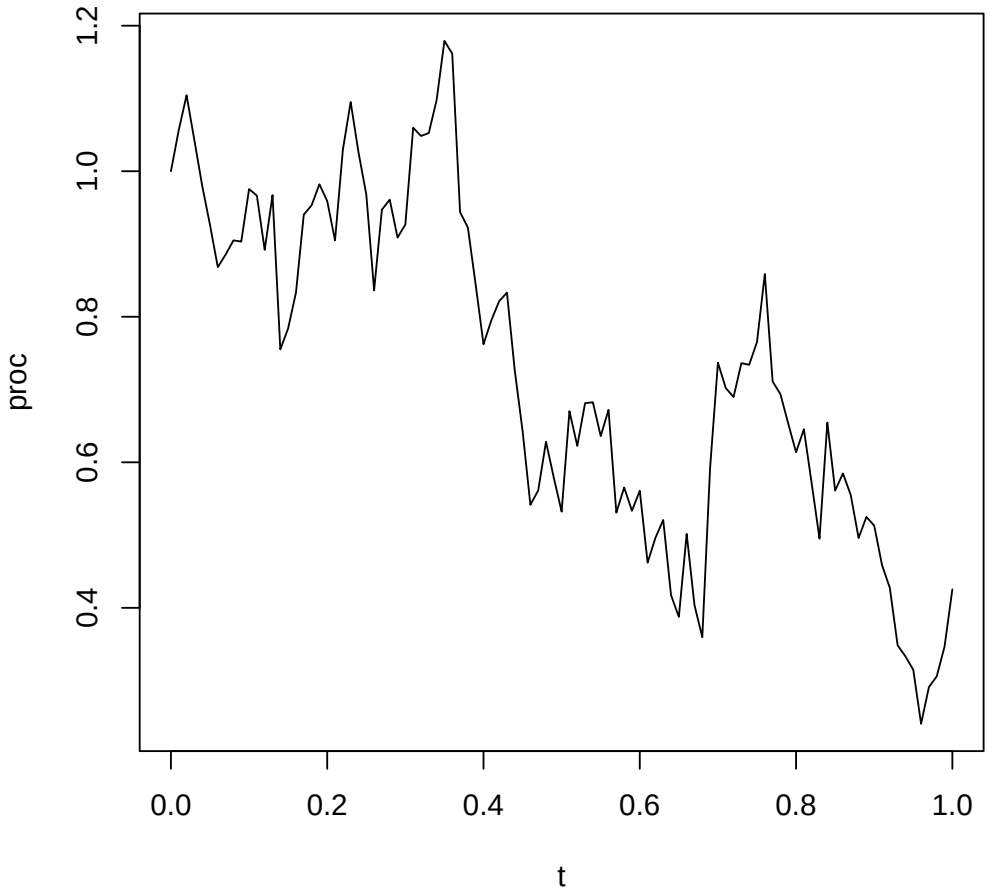


Figura 4.13: Proceso CIR por densidad condicional

```
> plot(BBridge())
```

Retomaremos las herramientas de la paquetería `sde` más adelante para aprovechar las funciones implementadas sobre métodos de Inferencia Estocástica.

Ejemplo de Transformada de Lamperti

Notemos que el Esquema de Milstein coincide con el esquema de Euler si σ es una función constante y la solución se reduce a una ecuación diferencial sencilla. La idea de esta tansformación es llevar a este proceso a uno con volatilidad constante para así poder aplicar el esquema de Euler y luego volver al proceso original. Entonces, considere el proceso de difusión X que satisface la ecuación diferencial

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t$$

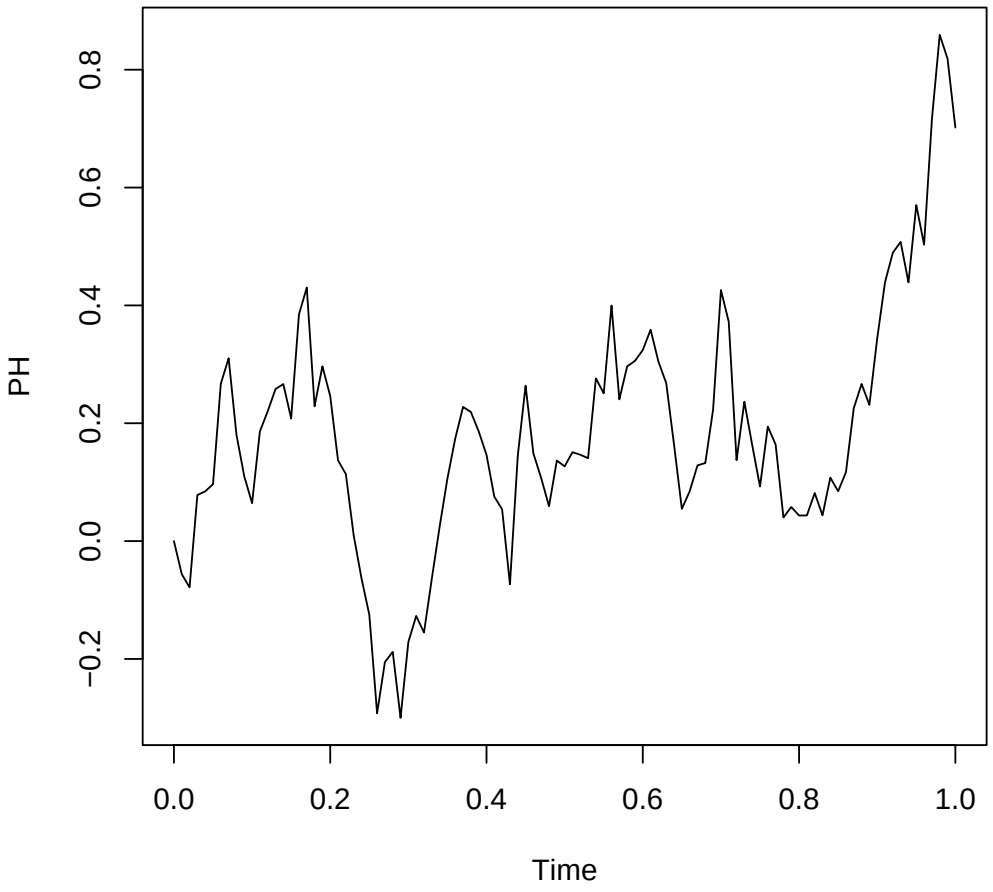


Figura 4.14: Proceso Hiperbólico (sde.sim milstein)

donde σ es continuamente diferenciable y estrictamente positiva. Definimos F y al proceso Y como:

$$Y_t = F(X_t), F(x) = \int \frac{1}{\sigma(x)} dx$$

Dado que $\sigma(x)$ es estrictamente positiva, F está bien definida y es estrictamente creciente. En particular, F^{-1} existe y $X_t = F^{-1}(Y_t)$. Al proceso Y se le conoce como la transformación de Lamperti de X.

$$F^{-1} = \frac{1}{\sigma^{'}}, F^{''}(x) = -\frac{\sigma^{'}(x)}{\sigma^2(x)}.$$

De la fórmula de Itô tenemos:

$$dY_t = F^{'}(X_t)dX_t + \frac{1}{2}F^{''}(X_t)(dX_t^2) = \left[\frac{b(X_t)}{\sigma(X_t)} - \frac{\sigma^{'}(X_t)}{2}\right] dt + dW_t$$

Sustituyendo $F^{-1}(Y_t)$ por X_t, tenemos que $dY_t = a(Y_t)dt + dW_t$, donde

$$a(y) = \frac{b(F^{-1}(y))}{\sigma(F^{-1}(y))} - \frac{\sigma^{'}(F^{-1}(y))}{2}.$$

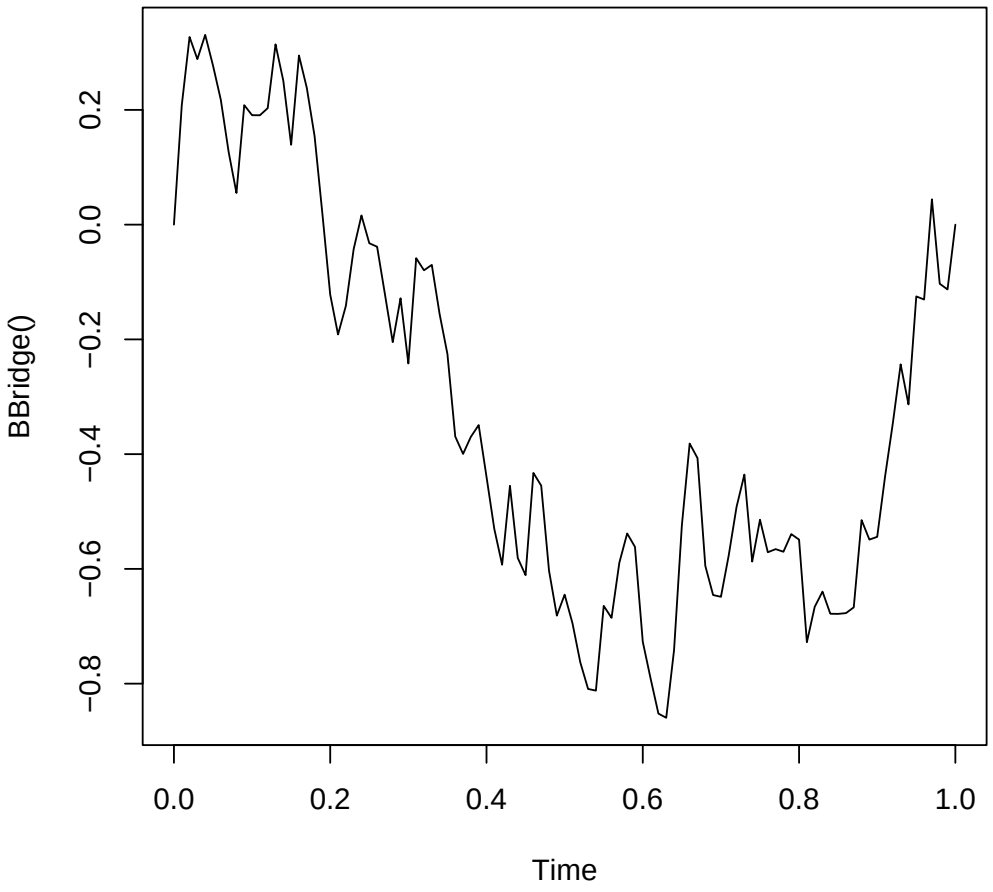


Figura 4.15: Puente Browniano

En otras palabras,Y es una difusión con volatilidad constante, lo que nos reduce a un esquema de Euler.

Se deja al lector la simulación del Modelo CIR bajo la transformada de Lamperti.

4.5. Ejercicios

1. Implementar el modelo matemático de ejemplo A.1.1.

2. Si suponemos que la partícula del ejemplo A.1.2 empieza en el nodo 0 y se mueve hasta que todos han sido visitados, ¿cuál es la probabilidad de que el nodo i con $i \in 0, 1, \ldots, m$, es el último que se visita?. La solución debe ser numérica.

3. En el ejemplo A.1.3, suponga que los arribos entre clientes siguen una distribución exponencial de parámetro α y que las variables aleatorias que modelan el tiempo de servicio son gobernadas por una distribución común y además son independientes (exponenciales de parámetro λ). Proponer un modelo y estimar los tiempos de servicio hasta atender a los clientes en el banco al tiempo t.

4. Responder (numéricamente) las preguntas planteadas en el ejemplo A.2.1. ♢

5. Considere una cadena de Markov con espacio de estados $\{0, 1, 2\}$ y matriz de transición

$$P = \begin{pmatrix} 0.6 & 0.3 & 0.1 \\ 0.3 & 0.3 & 0.4 \\ 0.4 & 0.1 & 0.5 \end{pmatrix}$$

y con distribución inicial $\pi = (0.2, 0.4, 0.4)$, simulando trayectorias, estimar las siguientes probabilidades:

   a) $\mathbb{P}(X_3 = 2|X_2 = 0)$,
   b) $\mathbb{P}(X_3 = 2, X_2 = 0, X_1 = 1, X_0 = 0)$,
   c) $\mathbb{P}(X_2 = 1)$
   d) $\mathbb{P}(X_{100} = 2|X_{99} = 0, X_1 = 1)$,
   e) $\mathbb{P}(X_3 = 2, X_1 = 0|X_0 = 2)$.
   f) Estimar la distribución invariante.

6. Estimar el número de pasos esperados para cambiar las bolas de urna en el ejemplo A.2.5.

7. Estimar la probabilidad de estar en el mismo punto de partida después de 100 pasos en el ejemplo A.2.3 para cualquier parametro p.

8. En el ejemplo A.2.4 suponga que el jugador decide apostar 6 pesos si tiene al menos 20pesos y 1 si tiene menos de 20. ¿Aumenta su probabilidad de ruina?. La solución debe ser númerica.

9. Considere una cadena de Markov con espacio de estados $\{1, \ldots, 5\}$ y matriz de transición

$$P = \begin{pmatrix} 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1/4 & 0 & 3/4 & 0 \\ 0 & 0 & 1/3 & 0 & 2/3 \\ 1/4 & 1/2 & 0 & 1/4 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \end{pmatrix}$$

Estimar tiempos medios de recurrencia y número esperado de visitas a cada estado.

10. Escribir un código para estimar el tiempo de llegada a un estado partiendo de otro en el ejemplo A.2.2.

11. Sea $\{X_n\}_{n\geq 0}$ una cadena de Markov con matriz de transición

$$P = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & 0 \end{pmatrix}$$

Hacer un programa que calcule los vectores de probabilidad invariantes. ♢

12. Generar un proceso de Poisson de parámetro $\lambda = 2.5$ utilizando el algoritmo implícito en el Teorema .♢

13. Resolver numericamente los ejemplos A.3.1 y A.3.2. ♢

14. Sea N_t un proceso de Poisson de parámetro $\lambda = 2$. Sea S_n el instante en el que ocurre el n-ésimo evento. Estime:

a) $E(N_5)$.

b) $E(N_5|N_2 = 1)$

c) $E(S_7)$.

d) $E(S_7|N_2 = 3)$

e) $E(S_7|N_5 = 4)$.

15. Las reclamaciones en un compañía de seguros se presentan de acuerdo a un proceso de Poisson de parámetro $\lambda = 4$. Sea N_t el número de clientes que han ingresado hasta el instante t. Calcule $\diamondsuit$

    a) La probabilidad que al tiempo $t = 2$ se tenga una sólo reclamación.

    b) La probabilidad que al tiempo $t = 1$ se tenga tres reclamaciones y $t = 1$ se tenga tres reclamaciones.

    c) La probabilidad que al tiempo $t = 1$ no se tengan reclamaciones sabiendo que al tiempo $t = 3$ se tienen cuatro reclamaciones.

    d) La probabilidad que al tiempo $t = 2$ se tengan cuatro reclamaciones sabiendo que al tiempo $t = 1$ se tienen dos reclamaciones.$P(N_2 = 4|N_1 = 2)$.

    e) El número esperado de reclamaciones al tiempo $t = 5$

    f) El número esperado de reclamaciones al tiempo $t = 5$ sabiendo que al tiempo $t = 2$ se tiene una reclamación.

    g) El tiempo esperado para que se presente la séptima reclamación.

    h) El tiempo esperado para que se presente la séptima reclamación sabiendo que al tiempo $t = 2$ se han presentado tres reclamaciones.

16. Implementar el algoritmo para el proceso de Poisson no homogéneo cuando $\lambda(t) = cos(t)$ en el intervalo de tiempo $[0, 2\pi]$.

17. Resolver numericamente los ejemplos A.3.1 y A.3.2 consideremos el supuesto que $\lambda(t) = t/2$.

18. Resolver numericamente los ejemplos A.3.3 y A.3.5.

19. Hacer un programa que simule (y grafique) un proceso de Poisson no homogéneo en $\mathbb{R}^3$ en la esfera de radio 5, donde la función de tasa espacial está dada por $\lambda(t) = \text{mín}\{10, \frac{1}{||t||}\}$ para $t \in \mathbb{R}^3$, para t tal que $||t|| \leq 5$.

20. Simule al Movimiento Browniano. Consideremos el método (de Euler) de hacer una malla con intervalos regulares y simular los incrementos de forma independiente. Obtendremos una aproximación discreta al MB.

21. Se deja como ejercicio simular un Movimiento Browniano mediante la Construcción de Lévy.

22. Simular al proceso de Vasicek utilizando la transformación anterior. Calcular $\mathbb{E}(X_t)$, $\mathbb{V}ar(X_t)$ y $\mathbb{C}ov(X_s, X_t)$ y tomar el límite cuando t va hacia infinito. ¿Qué significan los resultados en términos del modelo de tasas de interés?

Capítulo 5

Puentes Estocásticos

En el capítulo anterior, al simular un proceso de Markov, suponiamos conocida una distribución inicial para el valor X_0 (o un valor constante inicial) y procedíamos a simular la trayectoria del proceso hasta llegar al tiempo T en algún estado para X_T. Sin embargo, existe una gran cantidad de situaciones en donde tanto X_0 como X_T son valores conocidos y la trayectoria del proceso intermedia entre tales valores es desconocida (aunque posiblemente la ley del proceso sea conocida o estimada). La simulación de `puentes estocásticos`, es decir procesos estocásticos con trayectorias condicionadas a tomar valores fijos en X_0 y X_T, se convierte en una herramienta importante en tales situaciones. Una interpretación de tales procesos, es el de realizar interpolaciones aleatorias, dados los valores X_0 y X_T y una ley conocida (o estimada) del proceso. En este capítulo estudiaremos técnicas generales para procesos en valores discretos y continuos.

5.1. Método de la Bisección (Procesos de Salto Markovianos).

A continuación vamos a abordar la propuesta de Asmussen & Holbolth para simular Puentes de Markov. El algoritmo de la Bisección tiene la característica de proporcionar los estados intermedios en el intervalo de $(0, T)$ de un Puente de Markov. Bajo el supuesto de que un Proceso de Saltos de Markov cambia de estado con un tiempo de estadía τ, entonces se puede estudiar los casos en los que podemos encontrarnos, una primera observación es:

1. Si $X_0 = X_T = a$ y no hay saltos entonces conocemos todos los estados: $X_t = a$, para $0 \leq t \leq T$.
2. Si $X_0 = a$, $X_T = b \neq a$ y hay un único salto entonces sólo faltaría conocer a τ tal que $X_t = a$, para $0 \leq t \leq \tau$ y $X_t = b$, para $0 \leq \tau \leq T$.

Es entonces que estas conclusiones son fundamentales para la progresión del Algoritmo de la Bisección, ya que de no encontrarnos en los casos anteriores entonces necesariamente hay estados que no conocemos en el intervalo. De modo que el algoritmo busca llevar una revisión de los estados en un proceso de dividir el intervalo y comprobar la existencia de los saltos posibles que pudieron haber ocurrido.

Dicho lo anterior, notamos coherente la idea de realizar la metología sobre mitades de intervalo, y empezamos entonces establecer la mecánica del algoritmo.

Si biseccionamos un intervalo $(0,T)$ que tiene al menos un salto y encontramos que en $(0,T/2)$ o $(T/2,T)$ también hay al menos un salto entonces aún existen estados desconocidos y tendríamos que repetir el proceso.

Antes de describir la forma en que determinamos la existencia de saltos en intervalos del proceso tenemos que recurrir a algunos resultados.

Lema 5.1. *Sean a y b estados distintos de un PSM, y un intervalo de longitud T con $X_0 = a$. La probabilidad de que $X_t = b$ y que haya ocurrido un solo salto en el intervalo es*

$$R_{ab} = q_{ab} \begin{cases} \frac{e^{-q_a T} - e^{-q_b T}}{q_b - q_a} & q_a \neq q_b \\ T e^{-q_a T} & q_a = q_b \end{cases}$$

La densidad del tiempo de estadía es

$$f_{ab}(t;T) = \frac{q_{ab} e^{-q_b T}}{R_{ab}(T)} e^{-(q_a - q_b)t}, 0 \leq t \leq T.$$

Y además la probabilidad de que $X_T = b$ con al menos dos saltos en el intervalo es $P_{ab}(T) - R_{ab}(T)$.

Demostración. Sea $N(T)$ el número de saltos hasta el tiempo T, entonces

$$\begin{aligned} R_{ab} &= \mathbb{P}[X_T = b, N(T) = 1 | X_0 = a] \\ &= \int_0^T \frac{q_{ab}}{q_a} q_a e^{-q_a t} e^{-q_b (T-t)} dt \\ &= q_{ab} e^{-q_b T} \int_0^T e^{t(q_b - q_a)} dt \\ &= q_{ab} e^{-q_b T} \frac{e^{T(q_b - q_a)} - 1}{q_b - q_a} \end{aligned}$$

si $q_a \neq q_b$ y

$$R_{ab} = q_{ab} T e^{-T q_a}.$$

□

Proposición 5.1. *Distribuciones condicionales.*

- *Si $q_a = q_b$, el cambio de estado distribuye uniforme en $[0,T]$.*
- *Si $q_a < q_b$, el cambio de estado V distribuye de forma exponencial truncada a $[0,T]$.*
- *Si $q_a > q_b$, por simetría $f_{ab}(t)$ es la densidad de la variable $T - V$ con V distribuida de manera exponencial de parámetro $q_b - q_a$ truncada a $[0,T]$.*

Un elemento muy importante también es la matriz de probabilidades de transición $P(t) = exp(Qt)$, que puede ser calculada a través de determinar la diagonalización de Q para hacer $Q = UDU^{-1}$ con

$Caso$	$Saltos(0,T/2)$	$Saltos(T/2,T)$	$\mathbb{P}$	$Notacin$
1	0	0	$e_a e_a$	α_1
2	0	≥ 2	$e_a(p_{aa} - e_a)$	α_2
3	≥ 2	0	$(p_{aa} - e_a)e_a$	α_3
4	≥ 2	≥ 2	$(p_{aa} - e_a)(p_{aa} - e_a)$	α_4
5	1	1	$r_{ac} r_{ba}$	$\alpha_{5,c}$
6	1	≥ 2	$r_{ac}(p_{ca} - r_{ca})$	$\alpha_{6,c}$
7	≥ 2	1	$(p_{ac} - r_{ac})r_{ca}$	$\alpha_{7,c}$
8	≥ 2	≥ 2	$(p_{ac} - r_{ac})(p_{ca} - r_{ca})$	$\alpha_{8,c}$

Tabla 5.1: Escenarios posibles cuando $X_0 = X_T$

$D = diag(\lambda_i)$, entonces $P(t) = U diag(e^{\lambda_i}) U^{-1}$.

Caso de estados extremos iguales.

Supongamos que $X_0 = X_T = a$. Sea N(T) el número saltos en el intervalo [0,T], podemos escribir

$$P_{aa}(T) = P_{aa}(T/2)P_{aa}(T/2) + \sum_{c \neq a} P_{ac}(T/2)P_{ca}(T/2).$$

donde podemos ver que

$$\begin{aligned} P_{aa}(T/2) &= \mathbb{P}(X_{T/2} = a | X_0 = a) \\ &= \mathbb{P}(X_{T/2} = a, N(T/2) = 0 | X_0 = a) + \mathbb{P}(X_{T/2} = a, N(T/2) \geq 2 | X_0 = a) \\ &= e^{-q_a T/2} + [P_{aa}(T/2) - e^{-q_a T/2}], \end{aligned}$$

y también que

$$P_{ac} = R_{ac}(T/2) + [P_{ac}(T/2) - R_{ac}(T/2)].$$

Con lo cual podemos construir los escenarios posibles, tomando las abreviaturas $e_a = e^{-q_a T/2}$, $r_{ab} = R_{ab}(T/2)$ y $p_{ab} = P_{ab}(T/2)$, en la Tabla 5.1.

Es ahora que podemos reproducir el posible escenario que ocurre en el intervalo a traves de escoger alguno de forma proporcional a los α-valores. A partir del caso correspondiente tendremos que establecer si es posible hablar de saltos intermedios y tiempos de estadía en cada una de las bisecciones.

Caso de estados extremos distintos.

Ahora supongamos que $X_0 = a$ y que $X_T = b \neq a$, entonces tenemos la probabilidad de transición

$$P_{aa}(T) = P_{aa}(T/2)P_{ab}(T/2) + P_{ab}(T/2)P_{bb}(T/2) + \sum_{c\neq(a,b)} P_{ac}(T/2)P_{cb}(T/2).$$

Con lo cual también podemos construir los casos posibles en la Tabla 5.2.

$Caso$	$Saltos(0, T/2)$	$Saltos(T/2, T)$	$\mathbb{P}$	$Notacin$
1	0	1	$e_a r_{ab}$	β_1
2	0	≥ 2	$e_a(p_{ab} - e_{ab})$	β_2
3	≥ 2	1	$(p_{aa} - e_a)r_a$	β_3
4	≥ 2	≥ 2	$(p_{aa} - e_a)(p_{ab} - r_{ab})$	β_4
5	1	0	$r_{ab}e_b$	β_5
6	1	≥ 2	$r_{ab}(p_{bb} - e_b)$	β_6
7	≥ 2	0	$(p_{ab} - r_{ab})e_b$	β_7
8	≥ 2	≥ 2	$(p_{ab} - r_{ab})(p_{bb} - e_b)$	β_8
9	1	1	$r_{ac}r_{cb}$	$\beta_{9,c}$
10	1	≥ 2	$r_{ac}(p_{cb} - r_{cb})$	$\beta_{10,c}$
11	≥ 2	1	$(p_{ac} - r_{ac})r_{cb}$	$\beta_{11,c}$
12	≥ 2	≥ 2	$(p_{ac} - r_{ac})(p_{cb} - r_{cb})$	$\beta_{12,c}$

Tabla 5.2: Escenarios posibles cuando $X_0 \neq X_T$

De la misma manera hay que escoger alguno de los escenarios de manera propocional con los β-valores. Teniendo en cuenta que, estamos hablando de la posibilidad de que exista un estado c distinto de los extremos al cual se pudo llegar desde a para terminar en b antes del tiempo T, hay que realizar decisiones un poco distintas que cuando los extremos son iguales.

Todo lo anterior nos implica una recursión sobre la cual hay que decidir sobre cada bisección producida. Cuando produzcamos algún caso sobre un intervalo nuevo, con al menos un salto podremos registrar un cambio de estado y un tiempo de estadía para el Puente de Markov. Se deberá continuar el proceso hasta que esté determinado que en toda sección hay menos de dos saltos; lo cuál nos daría la información completa de toda la trayectoria del proceso.

5.2. Puentes Gaussianos

En esta sección abordaremos el caso en que tenemos un proceso gaussiano con valores continuos $\{X_t : t \geq 0\}$ condicionado a tomar un valor inicial de inicio y un valor final tiempo fijo T positivo. Al igual que en los puentes estocásticos tratados anteriormente nos interesa simular las posibles trayectorias que el proceso puede tomar.

5.2.1. Puente Browniano

Como primer acercamiento consideraremos el caso Browniano. Un puente Browniano puede ser construido de diversas maneras; las abordaremos a continuación sin hacer énfasis en los detalles de la construcción matemática. Posteriormente mostraremos la equivalencia de las definiciones.

Tenemos la siguiente definición.

Definición 5.1. *Un puente Browniano (estándar)* $\{X_t : t \in [0,1]\}$ *es un proceso estocástico con trayectorias continuas que tiene la distribución de un movimiento Browniano* $\{B_t : t \geq 0\}$ *condicionado a tomar el valor* 0 *en el tiempo* $T = 1$*, es decir:*

$$\{X_t : t \in [0,1]\} \stackrel{d}{=} \{B_t : t \in [0,1] \,|\, B_1 = 0\} \tag{5.1}$$

Presentamos la siguiente definición de puente Browniano, como un caso particular de proceso Gaussiano. Recordemos que un proceso $\{X_t : t \geq 0\}$ es Gaussiano si para cualquier colección de tiempos arbitrarios $0 \leq t_1, ..., t_n \leq 1$ y números reales arbitrarios $\lambda_1, ..., \lambda_n$ ocurre que la combinación lineal $\sum_{i=1}^{n} \lambda_i X_i$ tiene distribución normal.

Definición 5.2. *Un puente Browniano (estándar)* $\{X_t : t \in [0,1]\}$ *es un proceso estocástico Gaussiano con trayectorias continuas que cumple que* $X_0 = 0$ *y* $X_1 = 1$ *c.s. y tiene*

1. *función de esperanza* $\mathbf{E}[X_t] = 0$*, para* $0 \leq t \leq 1$*,*
2. *y función de covarianza* $Cov(X_s, X_t) = s(1-t)$ *para* $s \leq t$*.*

Finalmente presentamos una definición del puente Browniano como una transformación determinista de un movimiento Browniano.

Definición 5.3. *Representación anticipativa. Sea* $\{B_t : t \geq 0\}$ *un movimiento Browniano. Definimos a un puente browniano (estándar)* $\{X_t : t \geq 0\}$ *por*

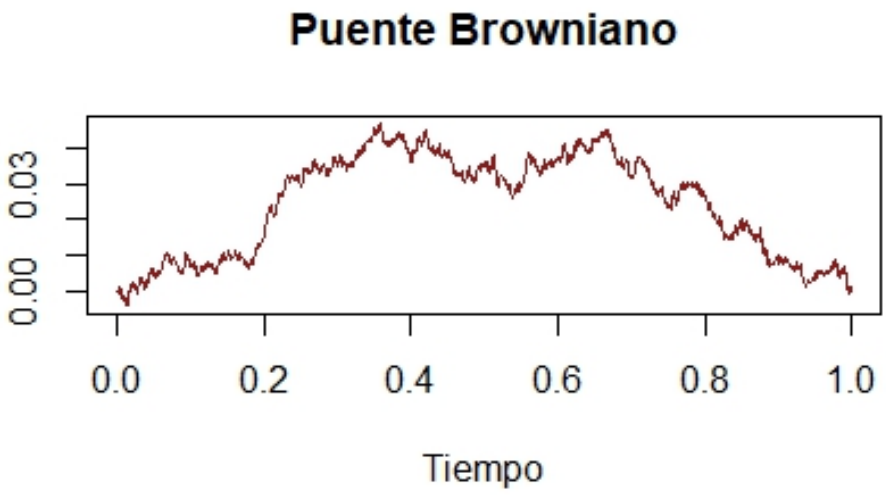
$$X_t := B_t - tB_1, \quad \forall t \in [0,1]. \tag{5.2}$$

A diferencia de las definiciones anteriores, suponiendo que tenemos una construcción explícita dada del movimiento Browniano, esta definición es explícita. Es por ello que será una de las posibilidades para simular un puente Browniano. Un esbozo de prueba de que las definiciones anteriores son equivalentes puede verse en el Apéndice.

Mostraremos a continuación la simulación de un puente Browniano a través de la representación anticipativa. Para ello generamos a un movimiento Browniano con algún método en una malla dada y aplicamos la transformación de la representación anticipativa.

Algoritmo 42 Representación adelantada

1: Generar una malla en el $[0,1]$.
2: Generar un movimiento Browniano $\{B_t : t \in [0,1]\}$ definido en la malla.
3: Para todo punto en la partición hacemos $X_t = B_t - tB_1$.
4: Devolvemos $\{X_t : t \in [0,1]\}$.



5.2.2. Puente Browniano como ecuación diferencial estocástica

Debido a resultados en el cálculo estocástico, existe otra definición alterna a las presentadas que utiliza la noción de ecuación diferencial estocástica. Consideramos ahora un puente Browniano más general que el caso estándar visto anteriormente: Un puente Browniano $\{X_t : t \in [0,T]\}$ es un proceso continuo que tiene la misma distribución que un movimiento Browniano condicionado a tomar el valor b al tiempo T.

Desde la perspectiva del cálculo estocástico, un puente Browniano puede verse como la solución a la ecuación diferencial estocástica

$$dX_t = \frac{b - X_t}{T - t}dt + dW_t;\ 0 \leq t \leq T,\ b \in \mathbb{R}\ y\ X_0 = T. \tag{5.3}$$

Simulación de un puente Browniano por el método de Euler

El Método de Euler nos brinda una aproximación de primer orden del sistema discreto de la ecuación (5.3) a la solución continua.

Algoritmo 43 Puente Browniano por el método de Euler

1: Generamos una discretización del intervalo.
2: Inicializamos $X_0 = 0$.
3: Iteramos mediante el método de Euler:

$$X_{t_{i+1}} = X_{t_i} + \frac{b - X_i}{T - t_i}\Delta t_i + \Delta W_i,$$

$$\Delta t_i = t_{i+1} - t_i,$$
$$\Delta W_i = W_{i+1} - W_i \sim N(0, \Delta t_i).$$

Simulación de un puente Browniano por el método de Milstein

Tal como indicamos en la sección de ecuaciones diferenciales estocásticas, el método de Euler tiene una velocidad de convergencia menor a la dada por el Método de Milstein. Presentamos ahora el algoritmo y la implementación de tal algoritmo cuando aproximamos la ecuación (5.3) utilizando el Método de Milstein.

Algoritmo 44 Puente Browniano por Milstein

1: Generamos una discretización del intervalo $[0, T]$

$$0 = t_1 \leq t_2 \leq \cdots \leq t_{n-1} \leq t_n = T.$$

2: Inicializamos $X_0 = a$.

3: Aplicamos la recurrencia del método de Milstein dada por:

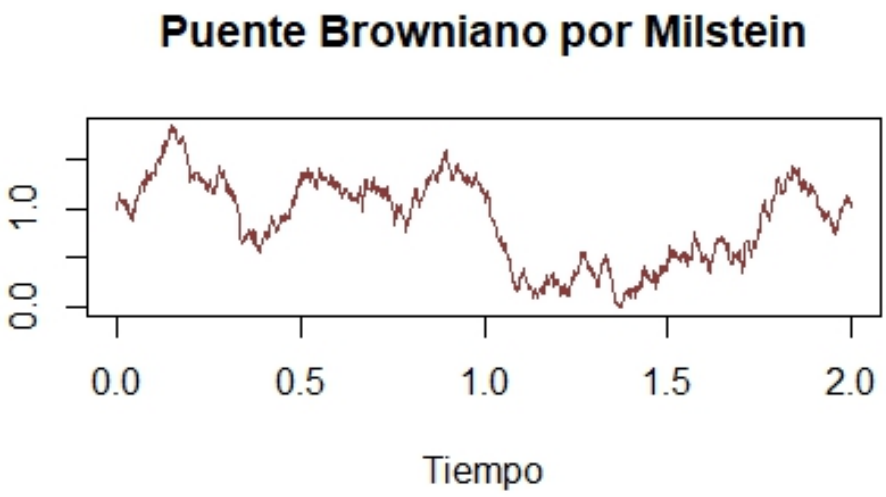
$$X_{t_{i+1}} = X_{t_i} + \frac{b - X_i}{T - t_i}\Delta t_i + \sqrt{\Delta t_i} Z_{i+1},$$

$$\Delta t_i = t_{i+1} - t_i,$$
$$\Delta W_i = \sqrt{\Delta t_i} Z_{i+1},\ Z_{i+1} \sim N(0, 1).$$

5.2.3. Puentes Gaussianos

En esta sección presentamos la definición de un puente Gaussiano y una representación anticipativa que nos permitirá simularlo.

Definición 5.4. *Decimos que un proceso* $\{X_t^{T,\theta} : t \in [0, T]\}$ *es un puente Gaussiano si es un proceso continuo que tiene la misma distribución que un proceso Gaussiano continuo* $\{X_t : t \in [0, T]\}$ *condicionado a que* $X_T = \theta$.



Como señalan Gasbarra, Sottinen y Valkeila, a diferencia del caso Browniano, no existe una representación general para los puentes gaussianos dada como la solución de una ecuación diferencial estocástica. Sin embargo, sí existe una representación anticipativa general para un puente Gaussiano. Este hecho nos permitirá la simulación de los puentes gaussianos.

La representación anticipativa es la siguiente. Considere un proceso Gaussiano $\{X_t : t \in [0,T]\}$ con función de covariancias dada por $\mathbb{C}ov(s,t)$. La representación anticipativa del puente $\{X_t^{T,\theta} : t \in [0,T]\}$ está dada por

$$X_t^{T,\theta} = \theta \frac{\mathbb{C}ov(T,t)}{\mathbb{C}ov(T,T)} + \left(X_t - \frac{\mathbb{C}ov(T,t)}{\mathbb{C}ov(T,T)} X_T \right).$$

Esta representación es una generalización para el puente Browniano estándar; para el caso de un puente browniano tenemos que $\theta = 0$, $T = 1$, $\mathbb{C}ov(t,1) = t$ y $\mathbb{C}ov(1,1) = 1$ recuperando la representación anticipativa presentada inicialmente

$$\begin{aligned} X_t^{1,0} &= 0 \cdot \frac{\mathbb{C}ov(1,t)}{\mathbb{C}ov(1,1)} + \left(X_t - \frac{\mathbb{C}ov(1,t)}{\mathbb{C}ov(1,1)} X_1 \right) \\ &= X_t - tX_1. \end{aligned}$$

Presentaremos ahora un ejemplo simple. Consideremos un puente derivado de un movimiento Browniano con deriva. El movimiento Browniano con deriva tiene la misma función de covarianza que un movimiento Browniano estándar, así que su representación anticipativa está dada por la misma transformación que la que se utiliza en la representación anticipativa de un puente Browniano estándar.

Algoritmo 45 Puente Browniano con deriva

1: Generar una malla en el $[0,1]$.
2: Generar un movimiento browniano con deriva $\{W_t : t \in [0,1]\}$.
3: Para todo punto en la partición hacemos $W_t = W_t - tW_1$.

5.2.4. Proceso Ornstein-Uhlenbeck

Una definición posible para el proceso de Ornstein-Uhlenbeck $\{X_t : t \geq 0\}$ que comienza en x_0 es como la transformación siguiente de un movimiento Browniano estándar $\{B_t : t \geq 0\}$:

$$X_t = x_0 e^{-at} + b(1 - e^{-at}) + \frac{\sigma e^{-at}}{\sqrt{2a}} B_{e^{2at}-1}.$$

Utilizando las propiedades del movimiento Browniano podemos calcular su función de esperanzas:

$$\begin{aligned}
\mathbf{E}(X_t) &= \mathbf{E}\left(x_0 e^{-at} + b(1 - e^{-at}) + \frac{\sigma e^{-at}}{\sqrt{2a}} B(e^{2at} - 1) \right) \\
&= x_0 e^{-at} + b(1 - e^{-at}) + \frac{\sigma e^{-at}}{\sqrt{2a}} \mathbf{E}(B(e^{2at} - 1)) \\
&= x_0 e^{-at} + b(1 - e^{-at}),
\end{aligned}$$

y también su función de covarianzas:

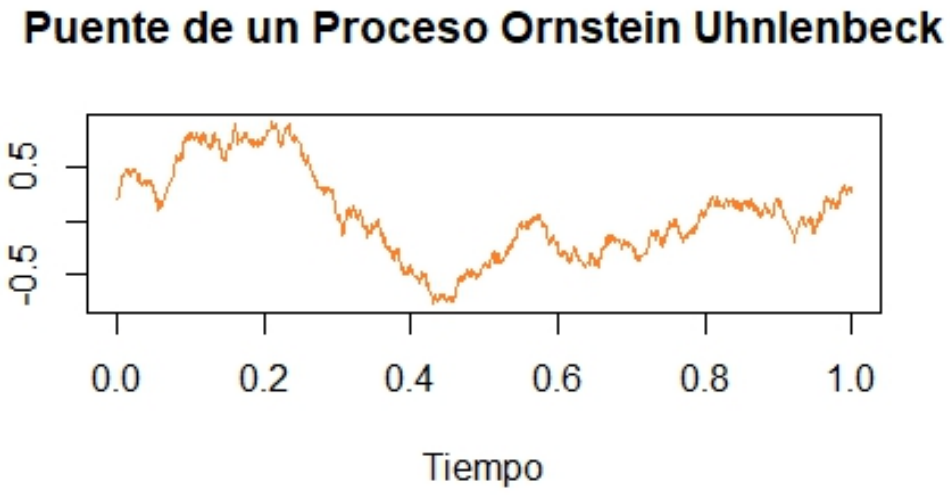
$$\begin{aligned}
\mathbb{C}ov(X_t, X_s) &= \mathbb{C}ov\left(\frac{\sigma e^{-at}}{\sqrt{2a}} B(e^{2at} - 1), \frac{\sigma e^{-as}}{\sqrt{2a}} B(e^{2as} - 1) \right) \\
&= \frac{\sigma^2 e^{-a(s+t)}}{2a} \mathbb{C}ov(B(e^{2at} - 1), B(e^{2as} - 1)) \\
&= \frac{\sigma^2 e^{-a(s+t)}}{2a} (e^{2a\ min\{t,s\}} - 1) \\
&= \frac{\sigma^2}{2a} (e^{-2a\ min\{t,s\} - a(t+s)} - e^{-a(t+s)}) \\
&= \frac{\sigma^2}{2a} (e^{-a|t-s|} - e^{-a(t+s)}).
\end{aligned}$$

Con las funciones anteriores, podemos tener explícitamente la representación anticipativa de un puente de Ornstein-Uhlenbeck.

Algoritmo 46 Puente Ornstein-Uhlenbeck

1: Generamos un proceso Ornstein-Uhlenbeck $\{X_t : t \in [0,T]\}$ en una malla.
2: Inicializamos $X_t^{T,\theta} = x_0$.
3: Para todo punto en la partición hacemos

$$X_t^{T,\theta} = \theta\frac{\mathbb{C}ov(T,t)}{\mathbb{C}ov(T,T)} + \left(X_t - \frac{\mathbb{C}ov(T,t)}{\mathbb{C}ov(T,T)}X_T\right) = \theta\frac{e^{-a(T-t)} - e^{-a(T+t)}}{1-e^{-2aT}} + \left(X_t - \frac{e^{-a(T-t)} - e^{-a(T+t)}}{1-e^{-2aT}}X_T\right).$$



5.2.5. Proceso Cox-Ingersoll-Ross

El proceso de Cox-Ingersoll-Ross $\{r(t) : t \geq 0\}$ puede definirse como solución de la siguiente ecuación diferencial estocástica:

$$dr(t) = k(\theta - r(t)) + \sigma\sqrt{r(t)}dW(t), \tag{5.4}$$

donde $\{W(t) : t \geq 0\}$ es un movimiento Browniano estándar. Se puede resolver por el método de variación de parámetros para encontrar una expresión explícita como transformación en términos del

$$\int_0^t e^{sk}dr(s) = \int_0^t k(\theta - r(s))e^{sk}ds + \int_0^t \sigma\sqrt{r(s)}e^{sk}dW(s)$$

$$r(t) = r_0e^{-tk} + e^{-tk}\int_0^t k(\theta - r(s))e^{sk}ds + e^{-tk}\int_0^t \sigma\sqrt{r(s)}e^{sk}dW(s)$$

Calculemos su esperanza:

$$\begin{aligned}\mathbf{E}\left(r(t)\right) &= \mathbb{E}\left(r_0e^{-tk} + e^{-tk}\int_0^t k(\theta - r(s))e^{sk}ds + e^{-tk}\int_0^t \sigma\sqrt{r(s)}e^{sk}dW(s)\right)\\ &= \mathbb{E}\left(r_0e^{-tk}\right) + \mathbb{E}\left(e^{-tk}\int_0^t k(\theta - r(s))e^{sk}ds\right) + \mathbb{E}\left(e^{-tk}\int_0^t \sigma\sqrt{r(s)}e^{sk}dW(s)\right)\\ &= r_0e^{-tk} + e^{-tk}\mathbb{E}\left(\int_0^t k(\theta - r(s))e^{sk}ds\right),\end{aligned}$$

donde utilizamos un resultado de cálculo estocástico que nos dice que una integral respecto a un movimiento Browniano es una martingala, por lo que tiene esperanza igual a 0.

Notemos que

$$\begin{aligned}\int_0^t k(\theta - r(s))e^{sk}ds &= (\theta - r(s))e^{sk}\Big|_0^t - \int_0^t e^{sk}dr(s)\\ &= (\theta - r(t))e^{tk} - (\theta - r_0) + r(t)e^{kt} - r_0\\ &= \theta(e^{tk} - 1).\end{aligned}$$

Por lo que

$$\begin{aligned}\mathbf{E}(r(t)) &= r_0e^{-tk} + e^{-tk}\mathbb{E}\left(\int_0^t k(\theta - r(s))e^{sk}ds\right)\\ &= r_0e^{-tk} + e^{-tk}\theta(e^{tk} - 1)\\ &= r_0e^{-tk} + \theta(1 - e^{-tk}).\end{aligned}$$

Calculemos ahora su función de covarianzas.

$$\begin{aligned}\mathbb{C}ov(r(t), r(s)) &= \mathbf{E}\left[\left(r(t) - \mathbf{E}(r(t))\right)\left(r(s) - \mathbf{E}(r(s))\right)()\right]\\ &= \mathbf{E}\left[e^{-tk}\left(\int_0^t \sigma\sqrt{r(u)}e^{uk}dW_u\right)\left(e^{-sk}\int_0^s \sigma\sqrt{r(v)}e^{vk}dW_v\right)\right]\\ &= \sigma^2e^{-k(t+s)}\int_0^s e^{2kv}\mathbf{E}(r(v))dv\\ &= \sigma^2e^{-k(t+s)}\int_0^s (r_0 - \theta)e^{vk} + \theta e^{2vk}dv\\ &= \sigma^2e^{-k(t+s)}\left[\frac{r_0 - \theta}{k}(e^{ks} - 1) + \frac{\theta}{2k}(e^{2ks} - 1)\right]\\ &= \sigma^2\frac{r_0 - \theta}{k}(e^{-kt} - e^{-k(s+t)}) + \sigma^2\frac{\theta}{2k}(e^{-k(t-s)} - e^{-k(t+s)}).\end{aligned}$$

Con los cálculos anteriores podemos hacer explícito el algoritmo de simulación de un puente de Cox-Ingersoll-Ross utilizando la representación anticipativa, de manera análoga al puente de Ornstein-Uhlenbeck. Dejamos al lector el cálculo explícito de la forma que toma el algoritmo en este caso. Presentamos a continuación el código asociado a tal simulación.


Puente de un Proceso CIR
0.0
0.2
0.4
0.0
0.5
1.0
1.5
2.0
Tiempo

Capítulo 6

Método Monte Carlo y reducción de varianza.

En este capítulo introduciremos un método importante en la simulación estocástica que permite dar solución a problemas matemáticos que, como ya se mencionó en el primer capítulo, resultan costosos o imposibles de resolver analíticamente. Además se presenta la importancia de la reducción de varianza en la implementación de algoritmos de simulación y algunas técnicas para conseguir este objetivo.

6.1. Método Monte Carlo

La simulación por el método de Monte Carlo (clásico o crudo) emplea números aleatorios para resolver ciertos problemas deterministas donde el transcurrir del tiempo no juega un papel sustancial.

El método se llamó así en referencia al Casino de Monte Carlo. Se originó y desarrolló durante la segunda guerra mundial cuando esta metodología fue aplicada a problemas relacionados al desarrollo de la bomba atómica.

Citando a uno de los grandes impulsores del desarrollo de este método [6] decimos que "el método de Monte Carlo consiste en representar la solución analítica de un problema como una propiedad de una población (hipotética) y hacer la estimación de esta propiedad (por ejemplo, a través de un parámetro) a partir de una muestra aleatoria".

En general podemos decir que la aplicación del método de Monte Carlo implica seguir los siguientes pasos:

1. Formular analíticamente el problema.
2. Diseñar un modelo probabilista adecuado al problema.
3. Simular dicho modelo para generar una muestra aleatoria que permita realizar una estimación paramétrica.

Existen dos problemas que se abordan con frecuencia: integración y optimización. Empecemos examinando

algunos ejemplos donde es importante la aplicación del método de Monte Carlo.

Ejemplo 6.1.1. *Integración*

Supongamos que tenemos el siguiente problema de integración. Sean $f : \mathbb{R}^m \to [0, \infty)$ y h una función real integrable. El problema consiste en computar

$$\Lambda = \int_A h(x)f(x)dx, \qquad A \subset \mathbb{R}^m.$$

Supongamos que la integral existe y es finita y que f es una función de densidad. Tambien supondremos que el soporte de f, dado por el conjunto

$$sop(f) := \{x \in \mathbb{R} \, : \, f(x) > 0\},$$

es decir el conjunto de valores en donde f no se anula, está contenido dentro del conjunto A. Entonces ocurre que Λ es el valor esperado de h. Así, por la ley fuerte de los grandes números, tenemos que

$$\mathbb{E}(h(X)) = \int_A h(x)f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} h(X_i) =: \hat{\Lambda},$$

para una muestra aleatoria $\{X_1, \ldots, X_n\}$ y una variable aleatoria X con densidad $f(x)$. Al valor $\hat{\Lambda}$ se le conoce como estimador de Monte Carlo de Λ.

Con este estimador, que se genera a partir de la simulación de varibles aleatorias, podemos tener una solución a un problema que no es viable de resolver análiticamente. Tenemos las siguientes observaciones importantes:

1. $\mathbb{E}_f(\hat{\Lambda}) = \Lambda$.
2. $\mathbb{V}ar_f(\hat{\Lambda}) = \frac{1}{n}\int_A (h(x) - \Lambda)^2 f(x)dx,$

es decir, el estimador de Monte Carlo es un estimador insesgado y la varianza de dicho estimador también se puede estimar por el método de Monte Carlo.

A la desviación estándar de $\hat{\Lambda}$

$$\sigma_f(\hat{\Lambda}) = \sqrt{\mathbb{V}ar_f(\hat{\Lambda})}$$

se le conoce como error estándar. Sabemos que la ley fuerte de los grandes números implica que $\hat{\Lambda} \approx \Lambda$ cuando n crece a infinito, y la desigualdad de Chebyshev nos da una cota general para la probabilidad de que Λ esté a distancia $\epsilon > 0$ de su estimador $\hat{\Lambda}$:

$$\mathbb{P}(|\hat{\Lambda} - \Lambda| \geq \epsilon) \leq \frac{\mathbb{V}ar(\hat{\Lambda})}{\epsilon^2}.$$

Así, si elegimos

$$\epsilon = \sqrt{\mathbb{V}ar(\hat{\Lambda})/\delta},$$

ocurre que

$$|\hat{\Lambda} - \Lambda| \leq \sqrt{\mathbb{V}ar(\hat{\Lambda})/\delta}$$

con probabilidad $1-\delta$. Como $\mathbb{V}ar(\hat{\Lambda}) = o(1/n)$, se sigue que la estimación de Λ por medio de $\hat{\Lambda}$ tiene un error de orden $n^{-1/2}$.

Ahora veamos un ejemplo concreto donde se utiliza el método Montecarlo para resolver un problema de integración.

Supongamos que queremos evaluar la integral

$$I = \int_a^b g(x)dx$$

donde $g(x)$ es una función que toma valores reales y que no es analíticamente integrable.

Para usar la simulación Monte Carlo, definimos $Y = (b-a)g(X)$ donde $X \sim U(a,b)$. Entonces ocurre que

$$\mathbb{E}(Y) = (b-a)\int_a^b g(x)f_X(x)dx = (b-a)\frac{\int_a^b g(x)dx}{(b-a)} = I$$

donde $f_X(x) = 1/(b-a)$ es la función de densidad de X. El problema de evaluar la integral se reduce a estimar el valor esperado $E(Y)$, que podemos hacer utilizando la media muestral de variables $Y_1, ..., Y_n$ independientes

$$\hat{I} = \frac{\sum_{i=1}^n Y_i}{n} = (b-a)\frac{\sum_{i=1}^n g(X_i)}{n},$$

donde $X_1, X_2, \ldots, X_n$ son v.a.i.i.d con distribución $U(a,b)$.

A continuación describimos otro ejemplo en donde podemos utilizar el método de Monte Carlo para variables aleatorias con un soporte infinito.

Ejemplo 6.1.2. *Para $\alpha > 0$ queremos calcular*

$$I_\alpha = \int_0^\infty x^{\alpha-1}e^{-x}dx.$$

Sea $f(x) = e^{-x}$ la densidad de X, es decir $X \sim exp(1)$. Notemos que

$$I_\alpha = E_f(X^{\alpha-1})$$

así I_α puede ser estimada por

$$\hat{I}_\alpha = \frac{1}{n}\sum_{i=1}^n X_i^{\alpha-1}$$

donde $X_1, \ldots, X_n$ es una muestra aleatorias con densidad f. A continuación se presentan resultados numéricos para el caso en que $\alpha = 1.9$. Recordemos que podemos generar de forma sencilla a las variables X_i's, definiendo

$$X_i = -ln(U_i)$$

donde las variables $U_i \sim U(0,1)$, $i = 1, 2, \ldots, n$ y son independientes. Presentamos los resultados obtenidos en la Tabla 6.1 a partir de la función `EMC.GAMMA` (ver B.3.15).

n	10	100	1,000	Tablas
$I_{1.9}$	0.8788	0.9312	0.9569	0.96177
Error Estándar	0.2302	0.0794	0.0277	

Tabla 6.1: Estimación para $\alpha = 1.9$

Ahora consideramos un ejemplo de optimización donde puede resultar de gran utilidad el método de Monte Carlo.

Ejemplo 6.1.3. *Sea* $h : [-1,1] \times [-1,1] \to \mathbb{R}$ *definida por*

$$h(x,y) = (xsen20y + ysen20x)^2 cosh(xsen10x) + (xcos10y - ysen10x)^2 cosh(ycos20y).$$

Utilizando el método de Monte Carlo encontraremos las posiciones de los valores máximo y mínimo y los valores máximo y mínimo alcanzados por la función h.

En este caso el método de Monte Carlo crudo se traduce en generar números aleatorios $\{u_1, \ldots, u_n\}$ en $[-1,1] \times [-1,1]$ y calcular $\{h(u_1), \ldots, h(u_n)\}$. De estos últimos, seleccionemos las parejas $(u^{max}, h(u^{max}))$ y $(u^{min}, h(u^{min}))$ tales que $h(u^{max}) = máx\{h(u_1), \ldots, h(u_n)\}$ y $h(u^{min}) = mín\{h(u_1), \ldots, h(u_n)\}$. En la Tabla 6.2 se presentan resultados numéricos para diferentes tamaños de muestra.

n	$h(u^{min})$	u^{min}	$h(u^{max})$	u^{max}
10	0.22×10^{-3}	(-0.0286,0.0439)	2.061	(0.749,0.875)
100	3.27×10^{-6}	(-0.0003,-0.0291)	5.301	(-0.083,0.091)
1000	6.46×10^{-8}	(-0.00061,-0.017)	5,218	(-0.072,0.099)
10000	4.08×10^{-8}	(-0.00012,-0.0055)	5.430	(-0.085,0.088)
100000	1.10×10^{-9}	(-0.00004,-0.0009)	5.739	(0.099,-0.098)

Tabla 6.2: Estimadores de máximos y mínimos de h.

Queremos remarcar que este método de optimización, por medio de Monte Carlo crudo, es poco eficiente: estamos haciendo una búsqueda aleatoria de forma independiente. Esto tiene algunas similitudes a realizar fuerza bruta computacional, que es realizar una búsqueda de diccionario (enlistar todos los elementos) para encontrar los valores deseados. Sin embargo su implementación es muy sencilla; el único requisito es poder evaluar la función f que estamos intentando optimizar.

6.2. Reducción de la varianza

En esta sección nos enfocaremos en algunos aspectos teóricos y prácticos de las técnicas de reducción de varianza, motivados en encontrar estimadores de Monte Carlo de varianza mínima. El objetivo es utilizar de mejor forma nuestro poder computacional (y por lo tanto nuestra precisión) al utilizar estimadores más precisos que los dados por el estimador de Monte Carlo crudo.

Definición 6.1. *Podemos definir a una técnica de reducción de la varianza como un medio para obtener estimadores más precisos utilizando información conocida de nuestro modelo.*

Por lo general, entre mayor sea la información con la que contemos sobre nuestro modelo, más efectiva será la reducción de la varianza. Las técnicas principales y más efectivas para la reducción de la varianza

son el muestreo por importancia y el método de Monte Carlo condicional; ambos serán discutidos en esta sección. Otras técnicas que pueden ayudar a la reducción de la varianza son el uso de variables antitéticas, de control y la estratificación.

Como motivación de la importancia del uso de las técnicas de reducción de varianza, comenzaremos considerando el siguiente ejemplo, tomado de [22], que ilustra la importancia de este concepto en el método de Monte Carlo.

Ejemplo 6.2.1. *Suponga que* $X \sim Cauchy(0,1)$ *y que estamos interesados en evaluar* $\mathbb{P}(X \geq 2)$*, es decir*

$$\theta = \mathbb{P}(X \geq 2) = \int_2^\infty \frac{dx}{\pi(1+x^2)}. \tag{6.1}$$

Como esta integral tiene solución analítica, encontraremos dicha solución y nos servirá de referencia para evaluar qué tan buenos son los estimadores que calcularemos. Para encontrar la solución, hacemos el cambio de variable

$$x(\phi) = \tan\phi.$$

Entonces

$$1 + x^2 = 1 + \tan^2\phi = \sec^2\phi$$

y

$$dx = \sec^2\phi d\phi.$$

Por otra parte, como $\phi(x) = \tan^{-1} x$, ocurre que $\phi(\infty) = \pi/2$, y se concluye

$$\theta = \int_2^\infty \frac{dx}{\pi(1+x^2)} = \frac{1}{\pi}\int_{\tan^{-1}2}^{\pi/2} d\phi = \frac{1}{2} - \frac{1}{\pi}\tan^{-1}2 \approx 0.147584.$$

Proponemos un primer estimador dado por

$$\tilde{\theta}_1 = \frac{1}{n}\sum_{i=1}^n I_{[2,\infty)}(X_i).$$

donde la muestra $\{X_1, \ldots, X_n\}$ tiene distribución $Cauchy(0,1)$. Es claro que $n\tilde{\theta}_1$ es una variable aleatoria Binomial con parámetros (n, θ) y entonces

$$\mathbb{V}ar(\tilde{\theta}_1) = \frac{\theta(1-\theta)}{n} \approx \frac{0.125803}{n}.$$

La varianza no parece tan pequeña.

Utilizando la simetría de la densidad Cauchy respecto al cero, podemos calcular θ como

$$2\theta = \mathbb{P}(|X| \geq 2) = 1 - \int_{-2}^{2} \frac{dx}{\pi(1+x^2)}$$

por lo que

$$\theta = \frac{1}{2} \int_{\mathbb{R}\setminus(-2,2)} \frac{dx}{\pi(1+x^2)}$$

y entonces proponemos un segundo estimador de θ por

$$\tilde{\theta}_2 = \frac{1}{2n} \sum_{i=1}^{n} I_{\mathbb{R}\setminus(-2,2)}(X_i)$$

con una muestra $\{X_1, \ldots, X_n\}$ con distribución $Cauchy(0,1)$. Como $2n\tilde{\theta}_2 \sim Bin(n, 2\theta)$, la varianza de $\tilde{\theta}_2$ es aproximadamente:

$$\mathbb{V}ar(\tilde{\theta}_2) \approx \frac{0.0520109}{n},$$

que es menor que la de $\tilde{\theta}_1$.

Hasta ahora hemos propuesto dos estimadores y tenemos un criterio de decisión saber cual es mejor, ambos están basados en la simulación de variables aleatorias Cauchy, pero un buen punto a resaltar es que no tenemos la certeza de haber encontrado el estimador de menor varianza.

Continuando en la busqueda de dicho estimador, notamos que

$$1 - 2\theta = \int_{-2}^{2} \frac{dx}{\pi(1+x^2)} = 2 \int_{0}^{2} \frac{dx}{\pi(1+x^2)}.$$

Entonces podemos generar números aleatorios $X_i \sim U(0,2)$ y proponer un tercer estimador dado por:

$$\tilde{\theta}_3 = \frac{1}{2} - \frac{2}{n} \sum_{i=1}^{n} \frac{1}{\pi(1+X_i^2)}$$

cuya varianza está dada por

$$\mathbb{V}ar(\tilde{\theta}_3) = \frac{4}{n} \left\{ \int_{0}^{2} \frac{1}{2} \left(\frac{1}{\pi(1+x^2)} \right)^2 dx - \left(\int_{0}^{2} \frac{1}{2\pi(1+x^2)} dx \right)^2 \right\}$$

que resulta ser

$$\mathbb{V}ar\left(\tilde{\theta}_3\right) \approx \frac{0.0285088}{n},$$

entonces tenemos un estimador de varianza menor que la de los dos anteriores que propusimos.

Finalmente, si hacemos $y = 1/x$ tenemos que

$$\theta = \int_2^\infty \frac{dx}{\pi(1+x^2)} = \int_0^{1/2} \frac{y^{-2}dy}{\pi(1+y^{-2})} = \int_0^{1/2} \frac{dy}{\pi(1+y^2)}.$$

Entonces, si generamos $Y_i \sim U(0, 1/2)$, tenemos un nuevo estimador

$$\tilde{\theta}_4 = \frac{1}{2\pi n} \sum_{i=1}^{n} \frac{1}{1+Y_i^2},$$

cuya varianza aproximada es

$$\mathbb{V}ar(\tilde{\theta}_4) \approx \frac{0.0000955253}{n}.$$

En este caso, el estimador $\tilde{\theta}_4$ construido con una sola observación tiene aproximadamente la misma precisión que $\tilde{\theta}_1$ construido con 1000 observaciones.

Este ejemplo pone en evidencia el hecho de que no es una tarea fácil encontrar el estimador de menor varianza simplemente proponiendo diferentes estimadores, motivando la necesidad de contar con técnicas y metodología que nos permita reducir la varianza de los estimadores.

6.2.1. Muestreo por importancia.

Regresemos al problema de resolver la integral dada por (6.1) y escribámosla de la siguiente manera:

$$\theta = \int_2^\infty \frac{dx}{\pi(1+x^2)} = \int_{-\infty}^{\infty} \psi(x) f(x) dx$$

donde $\psi(x) = I_{[2,\infty)}(x)$ y f es la densidad dada por $f(x) = \left[\pi(1+x^2)\right]^{-1}$. Supongamos que g es una función de densidad fácil de muestrear. Entonces

$$\theta = \mathbb{E}_f(\psi(X)) = \int_{-\infty}^{\infty} \frac{f(x)\psi(x)}{g(x)} g(x) dx = \int_{-\infty}^{\infty} \phi(x) g(x) dx = \mathbb{E}_g(\phi(X))$$

donde

$$\phi(x) = \frac{f(x)\psi(x)}{g(x)}$$

y la notación $\mathbb{E}_f$ significa que dentro de tal operador, la variable aleatoria X tiene la densidad f. Por lo tanto, un estimador posible para θ es

$$\tilde{\theta}_5 = \frac{1}{n} \sum_{i=1}^{n} \phi(X_i)$$

con $\{X_1, \ldots, X_n\}$ una muestra aleatoria con densidad g. La varianza de este estimador está dada por

$$\mathbb{V}ar(\tilde{\theta}_5) = \frac{1}{n}\mathbb{E}\left[\left(\tilde{\theta}_5 - \theta\right)^2\right] = \frac{1}{n}\int_{-\infty}^{\infty} \left(\frac{f(x)\psi(x)}{g(x)} - \theta\right)^2 g(x) dx,$$

la cual será mínima entre más similar sea g al integrando.

En el ejemplo de la densidad Cauchy, debemos buscar una función, $h(x)$, que tenga una forma similar y que sea fácil de simular. Para conseguir dicho objetivo, primero observemos que la función decrece de manera similar a $\frac{1}{x^2}$, entonces, es natural proponer a $h(x) = 1/ax^2$, para algún $a > 0$. Tomando h de la forma propuesta se tiene que

$$\int_2^\infty h(x)dx = \frac{1}{2a},$$

entonces

$$g(x) = \frac{2}{x^2} I_{(2,\infty)}(x),$$

es una función de densidad cuya función de distribución está dada por

$$G(x) = \int_2^x g(t)dt = 1 - \frac{2}{x}.$$

Ahora, utilizando el método de la transformada inversa, tenemos que si $U \sim U(0,1)$, entonces $X = 2/(1-U)$ es una variable aleatoria con densidad $g(x)$. Por ello podemos estimar a θ por medio de

$$\tilde{\theta}_6 = \frac{1}{n}\sum_{j=1}^{n} \frac{X_j^2}{2\pi(1+X_j^2)} I_{[2,\infty)}(X_j),$$

donde $\{X_1, \ldots, X_n\}$ es una muestra aleatoria con densidad $g(x)$. Podemos calcular explícitamente la varianza de $\tilde{\theta}_6$:

$$\mathbb{V}ar\left[\tilde{\theta}_6\right] = \frac{1}{4n\pi^2}\left\{\int_2^\infty \left(\frac{x^2}{1+x^2}\right)^2 \frac{2}{x^2}dx - \left[\int_2^\infty \left(\frac{x^2}{1+x^2}\right)\frac{2}{x^2}dx\right]^2\right\} = \frac{0.0003821}{n},$$

que representa una reducción en 329 veces de la varianza del estimador de Monte Carlo crudo. Es decir los tamaños de muestra correspondientes n_1 y n_6 de cada estimador deben satisfacer la relación $n_6 = 0.003n_1$ para tener la misma varianza.

Extrapolando lo anterior, podemos ver que el método de muestreo de importancia consiste en escribir un valor por estimar θ, de la forma

$$\theta = \mathbb{E}_f(\psi(X)) = \int_{-\infty}^{\infty} \frac{f(x)\psi(x)}{g(x)} g(x)dx = \int_{-\infty}^{\infty} \phi(x)g(x)dx = \mathbb{E}_g(\phi(X)).$$

A la función g se le llama **función de muestreo por importancia**.

En resumidas cuentas cuando se desea estimar θ por esta técnica se debe encontrar una función de densidad g cuyo soporte contenga al soporte de $\psi \cdot f$ y que sea fácil de simular. En tal caso

$$\theta = \mathbb{E}_g\Big(\psi(X)\frac{f(X)}{g(X)}\Big). \tag{6.2}$$

Por lo tanto, si $\{X_1, \ldots, X_n\}$ es una muestra aleatoria con densidad g entonces

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} \psi(X_i)\frac{f(X_i)}{g(X_i)}$$

es un estimador insesgado de θ, llamado **estimador de muestreo por importancia**. El cociente de densidades

$$\frac{f(x)}{g(x)}$$

es conocido como la razón o **cociente de verosimilitud**. Es importante hacer notar que una gran virtud de este procedimiento es que una función g dada puede ser usada para encontrar el valor esperado de cualquier función ψ.

Por otra parte, aunque la condición

$$\int_{-\infty}^{\infty} \frac{\psi(x)^2 f(x)^2}{g(x)} dx < \infty \tag{6.3}$$

no es una condición necesaria para la convergencia del estimador de muestreo de importancia a θ, sí es necesaria para la existencia de la varianza del estimador. Por lo tanto, podemos concluir que el muestreo por importancia tiene poca eficiencia cuando (6.3) no se cumple, pues produce inestabilidad en la generación de los números aleatorios y en consecuencia la estimación no es confiable, por lo que funciones g que no cumplan esta condición no son recomendables. Esto implica que g deberá tener colas más pesadas que f.

Minimización de varianza

Debido a la importancia que tiene la elección de la densidad g en el método de muestreo por importancia del estimador dado por (6.2), se puede plantear el problema de minimización siguiente:

$$\min_g \operatorname{Var}_g \left(\psi(X)\frac{f(X)}{g(X)} \right). \tag{6.4}$$

Se puede probar que la solución a este problema está dada por

$$g^*(x) = \frac{|\psi(x)|f(x)}{\int_{-\infty}^{\infty} |\psi(x)|f(x)dx}, \tag{6.5}$$

es decir g^* es la función $|\psi(x)|f(x)$ normalizada para ser una densidad.

Nota 6.1. *En el caso en que $\psi(x) \geq 0$ entonces*

$$g^*(x) = \frac{\psi(x)f(x)}{\theta}, \tag{6.6}$$

y

$$Var_{g^*}(\hat{\theta}) = Var_{g^*}\left(\frac{\psi(X)f(X)}{g^*(X)}\right) = Var_{g^*}(\theta) = 0.$$

A la densidad g^* de las expresiones (6.5) y (6.6) se le llama **densidad óptima de muestro por importancia**.

Ejemplo 6.2.2. *Sea $X \sim Exp(\lambda)$ y $\psi(x) = I_{\{x \geq \tau\}}$ para algún $\tau > 0$. Sea f la función de densidad de X y supongamos que queremos estimar:*

$$\theta = \mathbb{E}_f(\psi(X)).$$

Ocurre que

$$g^*(x) = I_{\{x \geq \tau\}} \lambda e^{-(x-\tau)\lambda}.$$

6.2.2. Monte Carlo condicional

Comenzaremos recordando algunas propiedades de esperanza condicional que servirán de motivación para este método de reducción de varianza, llamado Monte Carlo condicional que también se conoce como **Rao-Blackwellization**.

Consideremos a dos variables aleatorias cualesquiera, X e Y. Recordemos que la (variable aleatoria) varianza condicional $\mathbb{V}ar(X|Y)$ está definida por

$$\mathbb{V}ar(X|Y) := \mathbb{E}((X - \mathbb{E}(X|Y))^2|Y),$$

y que también se puede computar de la siguiente manera

$$\mathbb{V}ar\left(X|Y\right) = \mathbb{E}\left(X^2|Y\right) - \mathbb{E}\left(X|Y\right)^2.$$

Así que al tomar esperanza en la ecuación anterior ocurre que

$$\begin{aligned} \mathbb{E}\left(\mathbb{V}ar\left(X|Y\right)\right) &= \mathbb{E}\left(\mathbb{E}\left(X^2|Y\right)\right) - \mathbb{E}\left(\mathbb{E}\left(X|Y\right)^2\right) \\ &= \mathbb{E}\left(X^2\right) - \mathbb{E}\left(\mathbb{E}\left(X|Y\right)^2\right). \end{aligned}$$

Por otro lado,

$$\begin{aligned} \mathbb{V}ar\left(\mathbb{E}\left(X|Y\right)\right) &= \mathbb{E}\left(\mathbb{E}\left(X|Y\right)^2\right) - \left(\mathbb{E}\left(\mathbb{E}\left(X|Y\right)\right)\right)^2 \\ &= \mathbb{E}\left(\mathbb{E}\left(X|Y\right)^2\right) - \mathbb{E}\left(X\right)^2. \end{aligned}$$

Sumando ambas expresiones se tiene que

$$\mathbb{E}\left(\mathbb{V}ar\left(X|Y\right)\right) + \mathbb{V}ar\left(\mathbb{E}\left(X|Y\right)\right) = \mathbb{E}\left(X^2\right) - \mathbb{E}\left(X\right)^2 = \mathbb{V}ar\left(X\right)$$

,

de lo que se sigue que

$$\mathbb{V}ar\left(\mathbb{E}\left(X|Y\right)\right) \leq \mathbb{V}ar\left(X\right),$$

pues la variable aleatoria $\mathbb{V}ar\left(X|Y\right)$ es no negativa.

La desigualdad obtenida nos muestra que condicionar siempre conduce a la reducción de la varianza. Utilizaremos esta idea para proponer un método de reducción de varianza.

Supongamos que deseamos estimar

$$l = \mathbb{E}\left(h(X)\right) = \int h(x) f(x) dx,$$

donde X es una variable con función de densidad f y h es una función integrable. Supongamos que existe una variable aleatoria Y con función de densidad g (fácil de simular) donde $\mathbb{E}\left(h(X)|Y=y\right)$ es una expresión se puede calcular explícitamente. Entonces, debido a que

$$l = \mathbb{E}\left(h(X)\right) = \mathbb{E}\left(\mathbb{E}\left(h(X)|Y\right)\right),$$

se tiene que $\mathbb{E}\left(h(X)|Y\right)$ es un estimador insesgado de l y, por lo visto al inicio de esta sección, ocurre que

$$\mathbb{V}ar\left(\mathbb{E}\left(h(X)|Y\right)\right) \leq \mathbb{V}ar\left(h\left(X\right)\right).$$

Esto nos dice que usar como estimador a la variable aleatoria $\mathbb{E}\left(h(X)|Y\right)$, en lugar de la variable $h(X)$, conduce a la reducción de la varianza.

Ahora estamos listos para proponer el algoritmo formalmente.

Algoritmo 47 : Monte Carlo condicional

1: Generar una muestra $Y_1, \ldots, Y_n$ de la densidad g.
2: Calcular $\mathbb{E}\left(h(X)|Y_k\right)$, $k = 1, \ldots, n$ analíticamente.
3: Estimar $l = \mathbb{E}\left(h(X)\right)$ mediante $\hat{l} = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \mathbb{E}\left(h(X)|Y_k\right)$.

Como podemos ver, este algoritmo requiere encontrar una variable aleatoria Y con las siguientes características:

- Debe ser fácil de simular.
- Debe poderse calcular analíticamente $\mathbb{E}\left(h(X)|Y\right)$.
- Es deseable que el valor $\mathbb{E}(\mathbb{V}ar(h(X)|Y))$ sea lo más grande posible en comparación con $Var(\mathbb{E}(h(X)|Y))$.

Veamos un ejemplo del uso del Algoritmo que acabamos de presentar.

Ejemplo 6.2.3. *Consideremos la siguiente integral*

$$G = \int_0^1 \int_0^1 g(x,y)dxdy,$$

donde

$$g(x,y) = \begin{cases} 1 & \text{si } x^2+y^2 \leq 1 \\ 0 & \text{si } x^2+y^2 > 1 \end{cases} \tag{6.7}$$

Si muestreamos uniformemente y de manera independiente X e Y en $[0,1]$, es decir, $(X,Y) \sim Unif([0,1]^2)$ podemos definimos al estimador de G (que es el estimador de Monte Carlo crudo con una sola observación) de la siguiente manera:

$$\hat{G} = g(X,Y) = \mathbb{1}_{\{x^2+y^2\}}(X,Y).$$

Ocurre que

$$E(\hat{G}) = G = \frac{\pi}{4}$$

y se puede calcular analíticamente su varianza

$$\mathbb{V}ar(\hat{G}) = \frac{\pi}{4} - \left(\frac{\pi}{4}\right)^2 = 0.168.$$

Utilizando propiedades de la esperanza condicional tenemos que:

$$\mathbb{E}(g(X,Y)) = \mathbb{E}(\mathbb{E}(g(X,Y)|Y)).$$

Dado y, podemos computar:

$$\mathbb{E}(g(X,Y)|Y=y) = \int_0^1 \mathbb{1}_{\left\{x \leq \sqrt{1-y^2}\right\}} f_{X|Y}(x|y)dx.$$

Como X es independiente de Y y X tiene distribución $Unif([0,1])$ se tiene que,

$$f_{X|Y}(x|y) = f_X(x) = \mathbb{1}_{[0,1]}(x)$$

por lo que sustituyendo en la expresión para $\mathbb{E}(g(X,Y)|Y=y)$, se sigue que

$$\mathbb{E}(g(X,Y)|Y=y) = \int_0^1 \mathbb{1}_{\left\{x \leq \sqrt{1-y^2}\right\}} dx = \int_0^{\sqrt{1-y^2}} dx = \sqrt{1-y^2}.$$

Concluimos que

$$\mathbb{E}(g(X,Y)|Y)) = \sqrt{1-Y^2}$$

y

$$\mathbb{E}(g(X,Y)) = \mathbb{E}(\sqrt{1-Y^2}).$$

Por lo tanto, un estimador de Monte Carlo condicional (de una única observación) está dado por $\sqrt{1-Y^2}$, donde Y tiene distribución uniforme en $[0,1]$. La varianza de este nuevo estimador está dada por

$$\int_0^1 (1-y^2)dy - \left(\frac{\pi}{4}\right)^2 = 0.050.$$

Obtuvimos una reducción de la varianza cercana a un tercio de la del estimador de Monte Carlo crudo.

Presentamos otro ejemplo de Monte Carlo condicional.

Ejemplo 6.2.4. *Sea $\{X_i\}_{i=1}^{\infty}$ una sucesión de variables aleatorias independientes e idénticamente distribuidas tales que X_i tiene distribución F y una variable aleatoria M, que toma valores en los naturales, independiente de la sucesión $\{X_i\}_{i=1}^{\infty}$. Definamos $S_M = \sum_{i=1}^{M} X_i$. Queremos estimar $l = \mathbb{P}(S_M \leq x)$, para un x dado.*

Denotemos por F^m a la función de distribución de $S_m = \sum_{i=1}^{m} X_i$, para un número natural m. Entonces tenemos que

$$F^m(x) = \mathbb{P}\Big(\sum_{i=1}^{m} X_i \leq x\Big) = F\Big(x - \sum_{i=2}^{m} X_i\Big).$$

Por otro lado, como

$$l = \mathbb{E}(I_{\{S_M \leq x\}}) = \mathbb{E}\Big[\mathbb{E}\Big(I_{\{S_M \leq x\}} \Big| \sum_{i=2}^{M} X_i\Big)\Big] = \mathbb{E}\Big[\Big(F(x - \sum_{i=2}^{M} X_i\Big)\Big],$$

podemos estimar a esta esperanza por medio de

$$\hat{l} = \frac{1}{n}\sum_{i=1}^{n} F\Big(x - \sum_{j=2}^{M} X_j^i\Big).$$

6.2.3. Variables de control

Supongamos que deseamos estimar $\theta := \mathbb{E}(Y)$ donde $Y = h(X)$ es el resultado de un experimento de simulación. Además supongamos que Z es otro resultado (que posiblemente resultó del proceso de simular) que puede ser fácilmente calculado. Finalmente asumamos que conocemos $\mathbb{E}[Z]$. Con esto en consideración podemos construir varios estimadores insesgados de θ:

$$\hat{\theta}_c = Y + c(Z - \mathbb{E}[Z]),$$

donde $c \in \mathbb{R}$ (en el caso en que $c = 0$, recuperamos el estimador de Monte Carlo crudo). Es claro que $\mathbb{E}[\hat{\theta}_c] = \theta$. ¿Cuándo ocurre que $\hat{\theta}_c$ tiene menor varianza que $\hat{\theta}$?

Notemos que:

$$\mathbb{V}ar(\hat{\theta}_c) = \mathbb{V}ar(Y) + c^2\,\mathbb{V}ar(Z) + 2\,c\,Cov(Y, Z).$$

Como esta expresión es válida para cualquier c, podemos minimizar esta varianza como una función de c. Obtenemos el mínimo de la varianza cuando:

$$c^* = -\frac{\mathbb{C}ov(Y, Z)}{\mathbb{V}ar(Z)}.$$

Así, ocurre que

$$\mathbb{V}ar(\hat{\theta}_{c^*}) = \mathbb{V}ar(Y) - \frac{\mathbb{C}ov(Y, Z)^2}{\mathbb{V}ar(Z)} = \mathbb{V}ar(\hat{\theta}) - \frac{\mathbb{C}ov(Y, Z)^2}{\mathbb{V}ar(Z)},$$

de modo que para reducir la varianza basta que el valor $\mathbb{C}ov(Y, Z)$ sea distinto de cero.

Lo anterior nos sugiere el uso del siguiente estimador:

$$\hat{\theta}_{c^*} = \frac{\sum_{i=1}^{n} (Y_i + c^*(Z_i - \mathbb{E}[Z]))}{n},$$

donde Z es conocida como la **variable de control del estimador**. Debido a que necesitamos conocer el valor c^* para implementar tal estimador, lo que podemos hacerlo es estimarlo, por medio de estimadores estadísticos usuales:

$$\hat{\mathbb{C}ov}(Y, Z) = \frac{\sum_{i=1}^{n} (Y_i - \bar{Y})(Z_i - \mathbb{E}(Z)}{n - 1},$$

$$\hat{\mathbb{V}ar}(Z) = \frac{\sum_{i=1}^{n} (Z_i - \mathbb{E}(Z)^2}{n - 1},$$

para aproximar c^* por

$$\hat{c}^* = -\frac{\hat{\mathbb{C}ov}(Y, Z)}{\hat{\mathbb{V}ar}(Z)}.$$

Ejemplo 6.2.5. *Supongamos que deseamos estimar $\theta = \mathbb{E}[e^{(U+W)^2}]$ donde U y $W \sim U(0, 1)$ son variables aleatorias $Unif(0, 1)$ independientes. Utilizemos a $Z := (U + W)^2$ como variable de control. Notemos que en este caso*

$$\mathbb{E}(Z) = \mathbb{E}(U^2 + 2UW + W^2) = \frac{1}{3} + \frac{1}{2} + \frac{1}{3} = \frac{7}{6}.$$

6.2.4. Variables antitéticas

En esta subsección abordaremos un método para reducir la varianza de los estimadores de Monte Carlo en donde se busca no añadir esfuerzo computacional, y que puede ser útil en diversas situaciones.

Suponga que

$$I = \int_{-\infty}^{\infty} f(x)g(x)dx,$$

donde f es una función de densidad, y donde tenemos dos estimadores insesgados de I basados en las muestras $\{X_1, ..., X_n\}$ e $\{Y_1, ..., Y_m\}$, identificados por $I_1 = I_1(X_1, ..., X_n)$ e $I_2 = I_2(Y_1, ..., Y_m)$ respectivamente. Además, supongamos que las respectivas varianzas son $\mathbb{V}ar(I_1)$ y $\mathbb{V}ar(I_2)$. Entonces

$$\hat{I} := \frac{I_1 + I_2}{2}$$

es un estimador insesgado de I con varianza dada por

$$\mathbb{V}ar(\hat{I}) = \frac{1}{4}\mathbb{V}ar(I_1) + \frac{1}{4}\mathbb{V}ar(I_2) + \frac{1}{2}\mathbb{C}ov(I_1, I_2).$$

Si $\{X_1, ..., X_n\}$ e $\{Y_1, ...Y_m\}$ son independientes y provienen de la misma distribución, entonces habremos reducido la varianza a la mitad del promedio de las varianzas originales. Entonces, uno podría pensar que en este proceso redujimos la varianza. Esto es sin embargo sólo una ilusión: por ejemplo, supongamos que $n = m$ y que las muestras son independientes. Entonces el estimador $\hat{I}$ utiliza $2n$ observaciones para reducir la varianza a la mitad, que es la varianza correspondiente al estimador de Monte Carlo crudo con $2n$ observaciones.

Pero, ¿qué pasa si las muestras $\{X_1, ..., X_n\}$ e $\{Y_1, ..., Y_m\}$ no son independientes? , en este caso, $\mathbb{C}ov(I_1, I_2)$ no es cero y su valor puede ser positivo o negativo. Si las muestras son correlacionadas negativamente, la varianza del estimador $\hat{I}$ se reducirá. Concluimos que en la búsqueda de la reducción de la varianza es importante utilizar muestras correlacionadas negativamente.

Para fijar ideas, supongamos que $\mathbb{V}ar(I_1) = \mathbb{V}ar(I_2)$. Entonces

$$\begin{aligned}
\mathbb{V}ar(\hat{I}) &= \frac{1}{2}\mathbb{V}ar(I_1) + \frac{1}{2}\mathbb{C}ov(I_1, I_2) \\
&= \frac{1}{2}\mathbb{V}ar(I_1)\left\{1 + \frac{\mathbb{C}ov(I_1, I_2)}{\sqrt{\mathbb{V}ar(I_1)\mathbb{V}ar(I_2)}}\right\} \\
&= \frac{1}{2}\mathbb{V}ar(I_1)\{1 + \rho(I_1, I_2)\},
\end{aligned}$$

donde $\rho(X, Y)$ denota al coeficiente de correlación entre las variables aleatorias X e Y. Entonces, la varianza de $\hat{I}$ será mucho menor que la de I_1 si la correlación es negativa y cercana a -1.

En este punto, la pregunta natural es, ¿Es posible generar variables correlacionadas negativamente sin aumentar demasiado el tamaño de la muestra original? La respuesta es afirmativa y además, asombrosamente, no se generan observaciones adicionales, al menos teóricamente. Veamos un ejemplo de esto.

Ejemplo 6.2.6. *Supongamos que queremos calcular*

$$I = \int_0^1 \sqrt{1-x^2}dx.$$

En este caso, esta integral tiene solución analítica. Si hacemos $x = \sin\theta$ y por lo tanto, $dx = \cos\theta d\theta$, tenemos que

$$\begin{aligned} I &= \int_0^{\pi/2} \cos^2\theta d\theta = \int_0^{\pi/2} \left(\frac{1}{2} + \frac{1}{2}\cos 2\theta\right) d\theta \\ &= \left[\frac{\theta}{2} + \frac{1}{4}\sin 2\theta\right]_0^{\pi/2} = \frac{\pi}{4}. \end{aligned}$$

El estimador por Monte Carlo simple es

$$I_1 = \frac{1}{n}\sum_{i=1}^{n}\sqrt{1-U_i^2}$$

donde $\{U_1, \ldots, U_n\}$ es una muestra de $U(0,1)$ y cuya varianza resulta es

$$\mathbb{V}ar(I_1) = \frac{1}{n}\left[\int_0^1 (1-u^2)du - I^2\right] \approx \frac{0.0498}{n}$$

Ahora, consideremos el estimador (de variables antitéticas) dado por

$$\hat{I} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left[\sqrt{1-U_i^2} + \sqrt{1-(1-U_i)^2}\right]$$

donde $\{U_1, ..., U_n\}$ es una muestra proveniente de la distribución $U(0,1)$. Entonces ocurre que

$$\mathbb{V}ar(\hat{I}) = \frac{1}{4n}\int_0^1 \left(\sqrt{1-u^2} + \sqrt{1-(1-u)^2} - 2I\right)^2 du \approx \frac{0.0068}{n}.$$

Hubo una reducción aproximada de la varianza original del 73 %.

6.3. Ejercicios

1. A partir de la expresión
$$\pi = 4\sum_{n=0}^{\infty}\frac{(-1)^n}{2n+1},$$
estimar el valor de π por el método de Monte Carlo.

2. Implementar el Ejemplo 7.1.2 y estimar el valor k.

3. Aproximar la integral de la función de densidad de una v.a. Normal(0,1) con límite inferior igual a -3 y límite superior igual a 5 y calcular la desviación estándar del estimador.

4. Aproximar la integral del kernel de una v.a. gamma y calcular la desviación estándar del estimador.

5. Calcular los 4 estimadores para la probabilidad de que $X > 2$, donde $X \sim Cauchy(0, 1)$ y sus varianzas

   a) Usando Cauchy y la indicadora de 2 a infinito.

   b) Pegándole una uniforme, con una transformación del estimador aprovechando la simetría.

   c) Usando la Cauchy, con una indicadora de (-2,2) complemento.

   d) Haciendo el cambio de variable $x = 1/y$.

6. Para esa misma Cauchy pero ahora con muestreo por importancia y su varianza

7. Monte carlo condicional

   a) X distribuye Exp(10) .

   b) M sea va Poisson (2).

8. Ejemplo 2 de reducción de varianza , implementar en R (reducción de varianza con dos muestras correlacionadas negativamente, para la integral de 0 a 1 de la función $\sqrt{(1 + x^2)}$.

9. Probar que solución al problema de minimización de varianza en el método de muestro por importancia es el de la expresión 6.5.

10. Estimar el parámetro del Ejemplo 6.2.2.

11. Del ejercicio anterior, analizar el caso $V(I1) < V(I2)$ en R.

12. En el ejemplo de las diferentes varianzas, analizar el caso en que $\mathbb{V}ar(I_1) > \mathbb{V}ar(I_2)$.

Capítulo 7

Monte Carlo vía Cadenas de Markov

En este capítulo presentamos una herramienta de simulación genérica, conocida como Monte Carlo vía cadenas de Markov (MCMC), la cuál permite generar muestras (aproximadas) de cualquier distribución. Las técnicas de Monte Carlo vía cadenas de Markov permiten generar, de manera iterativa, observaciones de distribuciones multivariadas que difícilmente podrían simularse utilizando métodos como los estudiados hasta ahora. La idea básica es construir una cadena de Markov que sea fácil de simular y cuya distribución estacionaria corresponda a la distribución objetivo que nos interesa.

Las ideas de esta técnica han tenido su desarrollo histórico desde diferentes ramas de las mateáticas: de la probabilidad, de la mecánica estadística y de la estadística bayesiana. Cada enfoque tiene su utilidad conceptual y de implementación, y se enriquecen entre ellos. Es por ello que presentaremos una introducción a estos diferentes enfoques en el capítulo.

Antes de presentar los algoritmos, presentaremos el resultado que motiva y fundamenta matemáticamente el uso de los algoritmos MCMC.

Tenemos el siguiente resultado fundamental para Cadenas de Markov.

Proposición 7.1. *Sea $\{X_n\}_{n\geq 0}$ una cadena de Markov homogénea, irreducible y aperiódica con espacio de estados E y distribución estacionaria π, se tiene que cuando $n \to \infty$ ocurre que*

$$X_n \xrightarrow{d} X$$

donde X tiene distribución π y

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) \xrightarrow{c.s.} E(g(X)).$$

La primera afirmación del resultado es el teorema clásico de convergencia de cadenas de Markov. Sin embargo, la segunda afirmación requiere un estudio más detallado. En la teoría de sistemas dinámicos, se prueba que para ciertos sistemas ocurre que los promedios temporales son iguales a los promedios espaciales. Este resultado nos permitirá tomar promedios de valores temporales en una cadena de Markov que son claramente dependientes pero que al promediarlos se comportan como si fuesen independientes; de forma similar a la ley fuerte de los grandes números. Este resultado es conocido como el teorema ergódico para Cadenas de Markov.

7.1. Metropolis-Hastings

En esta sección estudiaremos el principal algoritmo de Monte Carlo vías cadenas de Markov. La idea de este algoritmo, como se mencionó anteriormente, es simular una cadena de Markov tal que su distribución estacionaria coincida con la distribución a simular.

Como motivación para el desarrollo de este algoritmo, supongamos que queremos generar valores de una variable aleatoria X con soporte en $E = \{1, \ldots, N\}$ y distribución $\pi = (\pi_1, \ldots, \pi_N)$, donde $\pi_i = b_i/C$, $i \in E$, N es grande y C es la constante de normalización (difícil de calcular). Construiremos un método que solucione el problema.

Construimos una cadena de Markov $\{X_n\}_{n\geq 0}$ con espacio de estados E, cuya evolución depende de la matriz de transición $Q = \{q_{ij}\}$ de otra cadena de Markov irreducible. Dicha evolución está definida de la siguiente manera:

1. Cuando $X_n = i$, generamos una variable aleatoria Y tal que
$$P(Y = j) = q_{ij}$$
para todo $i, j \in E$.
2. Si $Y = j$, hacemos
$$X_{n+1} = \begin{cases} j & \text{con probabilidad } \alpha_{ij} \\ i & \text{con probabilidad } 1 - \alpha_{ij} \end{cases}$$
donde
$$\alpha_{ij} = \text{mín}\left\{\frac{\pi_j q_{ji}}{\pi_i q_{ij}}, 1\right\} = \text{mín}\left\{\frac{b_j q_{ji}}{b_i q_{ij}}, 1\right\}.$$

La evolución anterior se traduce en que el proceso $\{X_n\}_{n\geq 0}$ tiene como matriz de transición a $P = \{p_{ij}\}_{\{i,j\in E\}}$ donde

$$p_{ij} = \begin{cases} q_{ij}\alpha_{ij} & \text{si } i \neq j \\ q_{ii} + \sum_{k\neq i} q_{ik}(1 - \alpha_{ik}) = 1 - \sum_{k\neq i} q_{ik}\alpha_{ik} & \text{si } i = j. \end{cases}$$

A α_{ij} se le conoce como la **probabilidad de aceptación**.

El siguiente resultado prueba la eficacia de la construcción. Presentamos la prueba por ser simple.

Proposición 7.2. *La cadena de Markov $\{X_n\}_{n\geq 0}$ tiene como distribución estacionaria a π.*

Demostración. Demostraremos que la cadena de Markov con matriz de transición $P = \{p_{ij}\}_{i,j\in E}$ es reversible (con respecto al tiempo) y tiene como distribución estacionaria a π. Para ello basta verificar que se satisfacen las ecuaciones de balance local:

$$\pi_i p_{ij} = \pi_j p_{ji},$$

para toda $i, j \in E$. Esto es equivalente a que se cumpla:

$$\pi_i q_{ij}\alpha_{ij} = \pi_j q_{ji}\alpha_{ji}.$$

Para verificar esta última igualdad basta notar que si

$$\alpha_{ij} = \frac{\pi_j q_{ji}}{\pi_i q_{ij}}$$

entonces $\alpha_{ji} = 1$ y viceversa. □

Corolario 7.1. *Si la cadena de Markov $\{X_n\}_{n\geq 0}$ es irreducible y aperiódica, entonces π es la distribución límite.*

Este resultado es inmediato de los resultados de convergencia de cadenas de Markov. Las hipótesis de irreducibilidad y aperiodicidad se siguen directamente de que la matriz Q es irreducible y de la construcción de P.

Tenemos dos observaciones importantes:

- La elección de la cadena Q es arbitraria, siempre que satisfaga la condición de irreducibilidad.
- No fue necesario calcular a la constante C para definir a la cadena de Markov $\{X_n\}_{n\geq 0}$.

La idea anterior (y su prueba) es suficientemente robusta y se puede extender a cadenas de Markov con espacio de valores continuos (por ejemplo $\mathbb{R}$ o $\mathbb{R}^n$) bajo algunas condiciones generales (que involucran la interpretación de irreducibilidad y convergencia de Cadenas de Markov en espacios de valores continuos). Por supuesto, hay muchos detalles técnicos involucrados, por lo cuál no abordaremos la teoría correspondiente y simplemente daremos como efectiva la construcción análoga de la cadena de Markov $\{X_n\}_{n\geq 0}$ con valores continuos.

A partir de ahora cuando escribamos a x y y como estados en E, estaremos pensando que x y y son números reales o vectores. Imaginemos que queremos generar una muestra de una función de densidad (que podría ser multidimensional) objetivo $f(x)$.

El algoritmo siguiente usa una construcción análoga a la de la cadena de Markov en espacio de estados discreto, y se denomina algoritmo **Metropolis-Hastings**.

Se define la evolución de la cadena a continuación, dado que la cadena se encuentra en el estado X_n:

Algoritmo 48 : Metropolis-Hastings

1: Generar $Y \sim q(X_n, y)$, es decir generamos Y con el kernel de transición a un valor y dado que estamos en el valor X_n (la función de densidad de y al siguiente paso dado el valor anterior).
2: Generar $U \sim U(0,1)$ y hacemos

$$X_{n+1} = \begin{cases} Y & \text{si } U \leq \alpha(X_n, Y), \\ X_n & \text{en otro caso} \end{cases}$$

donde

$$\alpha(x,y) = \text{mín}\left\{\frac{f(y)q(y,x)}{f(x)q(x,y)}, 1\right\}.$$

De esta forma tenemos que, repitiendo los pasos del algoritmo anterior, obtenemos una sucesión de variables aleatorias dependientes $X_1, X_2, \ldots X_n$, donde X_m tiene aproximadamente la función de densidad

$f(x)$, para toda m suficientemente grande.

Observemos que en el fondo, este es un algoritmo dinámico de aceptación y rechazo. La eficiencia del algoritmo dependerá de la probabilidad de aceptación $\alpha(x, y)$.

Una cuestión importante en este algoritmo es la selección de matriz de transición Q. Existen diferentes alternativas para esta matriz, que dependen del problema a resolver. Las más usadas son:

- **Simetría:** Consiste en elegir q tal que $q(x, y) = q(y, x)$. En este caso tendríamos que la probabilidad de aceptación es:
$$\alpha(x, y) = \text{mín}\left\{\frac{f(y)}{f(x)}, 1\right\},$$
es decir, solamente depende de la función de densidad de la variable aleatoria a simular, entonces tenemos un método de aceptación-rechazo similar al método original de aceptación-rechazo.

- **Independencia:** Podemos elegir q tal que $q(x, y) = g(y)$, para alguna función de densidad g. Esto quiere decir que la transición no está dependiendo del estado actual, por lo que en cada paso de la evolución simplemente estamos sorteando un candidato (con densidad g) independiente. El candidato generado será aceptado con probabilidad
$$\alpha(x, y) = \text{mín}\left\{\frac{f(y)g(x)}{f(x)g(y)}, 1\right\}.$$
Este método es muy parecido al de aceptación-rechazo original por lo que es importante que la densidad propuesta g esté cercana a f. Una observación importante es que a diferencia del método de aceptación-rechazo original, éste produce muestras dependientes (por ejemplo si se rechaza una vez, tendremos dos variables consecutivas generadas con valores idénticos).

- **Caminata aleatoria:** Bajo esta elección, el candidato es obtenido con la siguiente expresión: $y = x + Z$, donde Z es una variable con densidad g simulable y simétrica (alrededor del origen). Así, la transición está dada por $q(x, y) = g(y - x)$, y la probabilidad de aceptación es:
$$\alpha(x, y) = \text{mín}\left\{\frac{f(y)}{f(x)}, 1\right\}.$$

Ahora veremos algunos ejemplos del uso del algoritmo para la simulación de vectores aleatorios dependientes.

Ejemplo 7.1.1. *Sea (X, Y) un vector aleatorio cuya densidad es*

$$f(x, y) = c * exp(-((xy)^2 + x^2 + y^2 - 8x - 8y)/2),$$

para x e y reales.

Imaginemos que queremos estimar $\lambda = \mathbb{E}(X)$ por medio del estimador de Monte Carlo

$$\hat{\lambda} = \frac{\sum_{i=1}^{M} X_i}{M},$$

donde la muestra $\{(X_i, Y_i)\}_{i=1}^n$ que aproxima a la distribución f se genera usando una matriz de transición Q dada por un algoritmo de Metropolis-Hastings **caminata aleatoria**.

Elegiremos que el incremento $Z = (Z_1, Z_2)$ esté dado por variables aleatorias $Z_1, Z_2 \sim N(0, a^2)$ independientes para alguna $a > 0$. Algunas observaciones en la selección de la constante a son las siguientes:

- Si a es demasiado pequeña, los componentes serán muy correlacionados positivamente, por lo que necesitaremos una muestra grande para la convergencia.
- Si a es demasiado grande, muchas de las muestras serán rechazadas, por lo que el algoritmo es poco eficiente.

Entonces, nuestro algoritmo para generar números provenientes de f tiene la forma explícita siguiente.

Algoritmo 49 : Metropolis-Hastings caminata aleatoria

1: Iniciamos en un valor arbitrario (X_0, Y_0), $n = 0$.
2: Dado el vector (X_n, Y_n), generamos $Z = (Z_1, Z_2)$ donde $Z_1, Z_2 \sim N(0, 1)$ independientes. Hacemos $C = (X_n, Y_n) + a * Z$.
3: Calculamos $\alpha((X_n, Y_n), C) = \text{mín}\{\frac{f(C)}{f(X_n, Y_n)}, 1\}$, donde f es la función de densidad objetivo.
4: Generamos $U \sim U(0, 1)$. Si $U < \alpha((X_n, Y_n), Y)$, hacemos $(X_{n+1}, Y_{n+1}) = C$ y $n = n + 1$. Volvemos al paso 2.
5: Paramos cuando $n = N$, un valor dado N de iteraciones totales.

A continuación presentamos un ejemplo concreto en donde estimar la distribución estacionaria nos puede ayudar a estimar la solución de un problema de optimización relacionado con una evolución markoviana de un sistema a través del tiempo.

Ejemplo 7.1.2. *Una persona tiene el proyecto de poner una pensión para autos con K cajones. Supongamos que Y_n es el número de personas que acuden a rentar un cajón el n-ésimo día (el contrato es por día) y que $\{Y_n\}_{n\geq 0}$ son v.a.i.i.d. Poisson(λ). Si hay cajones disponibles se consideran como contratos nuevos, en otro caso cada cliente toma su lugar ya asignado. La probabilidad de que una persona desocupe el cajón al día siguiente es p (los cajones se regresan simpre al inicio del día). Sea X_n el número de cajones en renta al final del n-ésimo día y Z_n el número de nuevos contratos durante el n-ésimo día. Cada nuevo contrato genera una ganancia de g pesos y un ingreso de r pesos por día (o fracción del día) y cada cajón se puede comprar sólo al principio con un costo de c pesos por cajón y por día. El objetivo es encontrar k que maximice la ganancia.*

Notemos que

$$X_{n+1} = \text{mín}\{k, Bin(X_n, 1-p) + Y_{n+1}\},$$

y

$$Z_{n+1} = X_{n+1} - Bin(X_n, 1-p).$$

Además $\{X_n\}_{n\geq 0}$ es una cadena de Markov homogénea a tiempo discreto con probabilidades de transición

$$\mathbb{P}(X_{n+1} = j | X_n = i) = \sum_{l=1}^{\text{mín}\{i,j\}} \binom{i}{l} (1-p)^l p^{i-l} \frac{\lambda^{j-l} e^{-\lambda}}{(j-l)!},$$

con $0 \leq i \leq k$ y $0 \leq j \leq k-1$. Si $j = k$ entonces $p_{ik} = 1 - \sum_{j=0}^{k-1} p_{ij}$.

Para maximizar la ganancia promedio por día (a largo plazo), tenemos que maximizar

$$M(k) = g\mathbb{E}[Z(k)] + r\mathbb{E}[X(k)] - ck \tag{7.1}$$

donde

$$\mathbb{E}[Z(k)] = \lim_{n\to\infty} \mathbb{E}[Z_n(k)] = \lim_{n\to\infty} \mathbb{E}[X_n(k)] - \mathbb{E}[Bin(X_{n-1}(k), 1-p)] = p\mathbb{E}[X(k)].$$

Entonces podemos reescribir a la función objetivo (7.1) como

$$M(k) = (g+r)\mathbb{E}[X(k)] - sk,$$

donde

$$\mathbb{E}[X(k)] = \sum_{x=0}^{k} x\pi_x$$

y π es la distribución estacionaria de la cadena $\{X_n\}_{n\geq 0}$. Si usamos la media muestral tendremos un estimador para k.

7.2. Muestreo de Gibbs

En esta sección estudiaremos un caso particular del algoritmo de Metropolis-Hastings que es muy utilizado para generar muestras de vectores aleatorios y ha sido extensamente usado en estadística bayesiana. Este método es conocido como el muestreo de Gibbs. La principal característica del muestreo de Gibbs es que la Cadena de Markov subyacente es construida de una sucesión de distribuciones condicionales.

Supongamos que queremos obtener una muestra un vector aleatorio $X = (X_1, X_2, \ldots, X_k)$ con función de densidad conjunta $f(X)$. Denotemos por $X_{[i]}$ al vector que se obtiene al considerar el vector X tras eliminar la i-ésima entrada. Consideremos la densidad condicional de X_i dado $X_{[i]}$, dada por

$$f(x_i|x_{[i]}) = \frac{f(x_i, x_{[i]})}{f(x_{[i]})} = \frac{f(x)}{\int_{-\infty}^{\infty} f(x)dx_i},$$

donde hemos extendido la notación de X, X_i y $X_{[i]}$ a vectores deterministas x, x_i y $x_{[i]}$.

Supongamos que las distribuciones condicionales son conocidas y fáciles de simular. Consideremos el algoritmo de Metropolis-Hastings eligiendo a la probabilidad de transición q de la siguiente manera particular: Elegimos una entrada i de manera equiprobable entre las coordenadas de X. Construimos Y como un vector idéntico a X salvo en la coordenada i. La coordenada i-ésima del vector Y se elige con la densidad de probabilidad:

$$q(x_i|x) := f(x_i|x_{[i]}).$$

Entonces, tendremos que

$$\alpha(x_i, x) = \text{mín}\left\{1, \frac{f(x_i)f(x_{[i]}|x_i)}{f(x_{[i]})f(x_i|x_{[i]})}\right\} = 1,$$

es decir, se acepta siempre al candidato; no hay rechazo.

Así, podemos escribir el algoritmo del Muestreo de Gibbs de forma explícita.

Algoritmo 50 : Muestreo de Gibbs

1: Inicializar $X^0 = (X_1^{(0)}, \dots X_k^{(0)})$.
2: Elegimos un índice i de forma equiprobable en $1, ..., k$.
3: Generamos $Z \sim f(X_i^{(n)} | X_{[i]}^{(n)})$.
4: Hacemos $X^{(n+1)}$ como la concatenación de $X_{[i]}^{(n)}$ y Z (poniendo Z como la i-ésima coordenada).
5: Repetir los paseos anteriores.

Ejemplo 7.2.1. *Distribución Normal Bivariada.*

Recordemos que un vector aleatorio (X, Y) continuo que toma valores en el plano $\mathbb{R}^2$ tiene distribución Normal Bivariada si su función de densidad está dada por

$$f_{X,Y}(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y} \exp\Big\{ -\frac{1}{2(1-\rho^2)}\Big[\frac{(x-\mu_X)^2}{\sigma_X^2} - \frac{2\rho(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y} + \frac{(y-\mu_Y)^2}{\sigma_Y^2}\Big]\Big\},$$

para x, y reales, donde ρ es el coeficiente de correlación. Consideremos el caso cuando $\sigma_X = \sigma_Y = 1$ y $\mu_X = \mu_Y = 0$, entonces tenemos que la función de densidad correspondiente es:

$$f_{X,Y}(x,y) = \frac{1}{2\pi} \exp\Big\{ -\frac{1}{2(1-\rho^2)}[x^2 - 2\rho xy + y^2]\Big\}.$$

Es fácil ver que $X, Y \sim N(0,1)$, y también verificar que

$$X|Y \sim N(\rho Y, 1-\rho^2)$$

y

$$Y|X \sim N(\rho X, 1-\rho^2),$$

por lo que podemos simular las densidades condicionales utilizando cualquiera de los métodos presentados antes para simular una densidad normal.

Ejemplo 7.2.2. *Implementemos el algoritmo anterior en un programa.*

A continuación presentamos la implementación de tal código. La siguiente función implementa el Muestreo de Gibbs donde los vectores generados tienen aproximadamente la densidad de una normal bivariada con el coeficiente de correlación deseado.

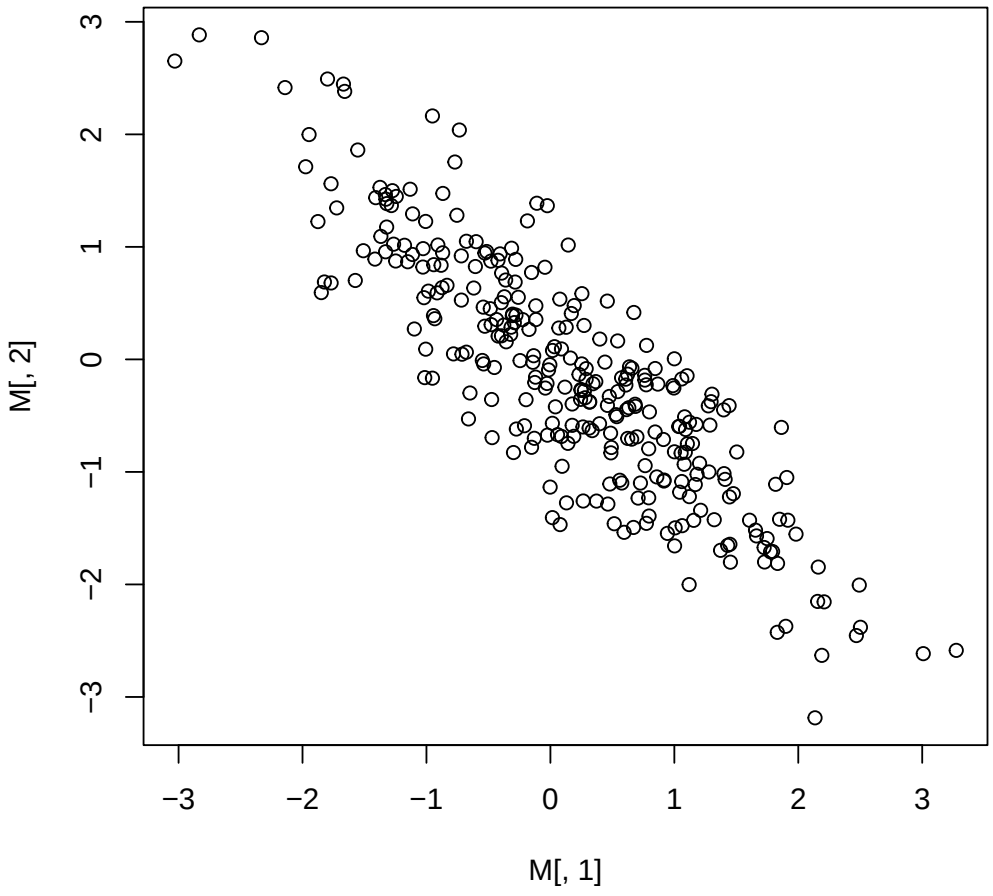
```
Gibbs<-function(k, rho){

  A=matrix(0,k,2)
  x=rnorm(1)  #Valores iniciales
  y=rnorm(1)  #Valores iniciales
  A[1,1]=x*sqrt(1-rho^2)+rho*y
  #Generamos otro parametro con la condicional en Xo
  A[1,2]<-x*sqrt(1-rho^2)+rho*A[1,1]
  #Generamos el otro con la misma condicional pero ahora en A[1,1]
```

```
  for (i in 2:k){ #Repetimos k veces
    A[i,1]<-rnorm(1)*sqrt(1-rho^2)+rho*A[i-1,2]
    A[i,2]<-rnorm(1)*sqrt(1-rho^2)+rho*A[i,1]
  }
  return (A)
}
```

Podemos graficar los valores obtenidos.

```
M=Gibbs(300,-.85)
plot( M[,1],M[,2])
```



Un ejercicio sugerido es utilizar nuestro programa para resolver un pequeño problema de estimación: Supongamos que deseamos estimar el valor de la probabilidad

$$\mathbb{P}(X^2 + Y^2 > 1.5)$$

cuando (X, Y) tiene distribución normal estándar bivariada con coeficiente de correlacion $\rho = -0.85$. Considere que el programa contemple un tamaño de muestra n y un período de calentamiento k.

Ejemplo 7.2.3. *Distribución Autoexponencial.*

Este modelo fue introducido por Besag (1974) en el área de estadística espacial. Supongamos que

$$f(y_1, y_2, y_3) \propto \exp\left\{-(y_1 + y_2 + y_3 + \theta_{12}y_1y_2 + \theta_{13}y_1y_3 + \theta_{23}y_2y_3)\right\},$$

para valores $y_1, y_2, y_3 \in [0, \infty)$, en donde θ_{12}, θ_{13} y θ_{23} son conocidos (recordemos que la notación $f \propto g$ indica que f y g difieren en una constante positiva, es decir, son directamente proporcionales).

Una propuesta inicial para simular observaciones de esta distribución es usando la relación

$$f(y_1, y_2, y_3) = f(y_1)f(y_2|y_1)f(y_3|y_1, y_2).$$

Para poder usar esta representación calculemos la marginal de (y_1, y_2),

$$\begin{aligned} f(y_1, y_2) &\propto \int_0^\infty f(y_1, y_2, y_3)dy_3 \\ &\propto \exp\left\{-(y_1 + y_2 + \theta_{12}y_1y_2)\right\} \int_0^\infty \exp\left\{-y_3\left(1 + \theta_{23}y_2 + \theta_{31}y_1\right)\right\} dy_3 \\ &\propto \frac{\exp\left\{-(y_1 + y_2 + \theta_{12}y_1y_2)\right\}}{1 + \theta_{23}y_2 + \theta_{31}y_1}. \end{aligned}$$

Entonces, la marginal de y_1 está dada por

$$f(y_1) \propto e^{-y_1} \int_0^\infty \frac{\exp\left\{-(y_2 + \theta_{12}y_1y_2)\right\}}{1 + \theta_{23}y_2 + \theta_{31}y_1} dy_2,$$

cuya solución analítica se dificulta. Sin embargo, es fácil mostrar que las densidades condicionales están dadas por

$$\begin{aligned} f(y_1|y_2, y_3) &= \exp(y_1|1 + \theta_{12}y_2 + \theta_{13}y_3), \\ f(y_2|y_1, y_3) &= \exp(y_2|1 + \theta_{12}y_1 + \theta_{23}y_3), \\ f(y_3|y_1, y_2) &= \exp(y_3|1 + \theta_{13}y_1 + \theta_{23}y_2), \end{aligned}$$

donde estamos utilizando la notación $\exp(x|\lambda)$ para denotar a la densidad dada por $f(x) = \lambda e^{-\lambda x}$. Así que estas densidades son muy sencillas de simular y por lo tanto un muestreo de Gibbs es fácil de implementar. Un ejercicio sugerido es la implementación del algoritmo de muestreo de Gibbs para simular observaciones provenientes de la distribución Autoexponencial y graficarlas.

7.2.1. Aplicación en estadística bayesiana

Sea $X = (X_1, \ldots, X_n)$ una muestra de una distribución Poisson con un punto de cambio aleatorio, es decir, supongamos que el punto de cambio se presenta en un valor $m \in \{1, \ldots, n\} = J_n$ y ocurre que, condicional al valor que toma m, tenemos que

$$X_i \sim Poisson(\lambda)$$

para $i \in \{1, ..., m\}$ y

$$X_i \sim Poisson(\tau)$$

para $i \in \{m+1, ..., n\}$. Por otro lado, supongamos que λ, τ y m tienen distribuciones a priori (es decir, no condicionales) independientes donde $\lambda \sim Gamma(\alpha, \beta)$, $\tau \sim Gamma(\sigma, \delta)$ y $m \sim Unif\{1, \dots n\}$, con α, β, σ y δ constantes conocidas.

La interpretación usual es la siguiente. Tenemos un conjunto de parámetros con cierta distribución a priori, que representa nuestro conocimiento previo de la distribución de tales parámetros. Esa información puede provenir de experiencia estadística anterior o de la información que nos proporcione un experto en el fenómeno. Por otro lado, recibimos datos, provenientes de información factual (es decir, trabajo de campo, experimentos en un laboratorio, etc). Nos gustaría recalcular la distribución de los parámetros, dados los datos que obtuvimos; esto servirá para tomar en cuenta la información inicial dada por la distribución a priori, contemplando la certeza que nos dieron los datos recibidos.

La densidad condicional anterior se puede derivar a través de la fórmula de Bayes. En nuestro caso toma la forma específica:

$$f(\lambda, \tau, m|X) = \frac{f(X|\lambda, \tau, m) f(\lambda, \tau, m)}{f(X)}.$$

Podemos verificar, de forma relativamente sencilla, que la densidad conjunta a posteriori (es decir condicional a los datos X) es

$$f(\lambda, \tau, m|X) \propto \lambda^{\alpha + y_m - 1} \tau^{\sigma + z_m - 1} \exp\{-(\beta + m)\lambda\} \exp -(\delta + n - m)\tau\},$$

donde

$$y_m = \sum_{i=1}^{m} X_i$$

y

$$z_m = \sum_{i=m+1}^{n} X_i,$$

expresión que no parece fácil de simular. Para hacer uso del muestreo de Gibbs tenemos que encontrar todas las condicionales a posteriori correspondientes, las cuales están dadas por:

$$\lambda|\tau, m, X \sim Gamma(\alpha + y_m, \beta + m),$$

$$\tau|\lambda, m, X \sim Gamma(\sigma + z_m, \delta + n - m)$$

y

$$f(m|\lambda, \tau, X) = \frac{\lambda^{\alpha + y_m - 1} \tau^{\sigma + z_m - 1} \exp\{-(\beta + m)\lambda\} \exp -(\delta + n - m)\tau\}}{\sum_{i=1}^{m} \lambda^{\alpha + y_i - 1} \tau^{\sigma + z_i - 1} \exp\{-(\beta + i)\lambda\} \exp -(\delta + n - i)\tau\}},$$

para toda $m \in \{1, ..., n\}$. Con lo anterior, somos capaces de programar un muestreo de Gibbs que simule a la distribución conjunta a posteriori $f(\lambda, \tau, m|X)$.

El algoritmo es el siguiente.

Algoritmo 51 : Muestreo de Gibbs 1

1: Inicializar τ, λ, m.
2: Elegir τ o λ o m, cada uno con probabilidad $1/3$.
3: Dado que elegimos el parámetro θ, simularlo de nuevo utilizando la densidad condicional $f(\theta|(\lambda, \tau, \theta) \setminus \{\theta\})$.
4: Repetir desde el paso 2.

Este algoritmo nos permite hacer lo siguiente: dado un conjunto de datos X, podríamos estimar los valores λ, τ y μ. Este es un ejercicio del capítulo y es un ejemplo típico de estimación utilizando el muestreo de Gibbs.

7.2.2. El muestreo de Gibbs dentro del contexto de la Física

Muchas de las ideas del muestreo de Gibbs tienen sus orígenes en la mecánica estadística. Para mostrar la versatilidad de tales ideas y profundizar en su comprensión presentamos en esta sección un breve esbozo de los conceptos utilizados y la forma específica que toma el muestreo de Gibbs en este contexto. Este lenguaje es utilizado en también en otras áreas como la estadística bayesiana, donde se utilizan ideas similares.

Campos de Markov

Sea S un conjunto finito o numerable con elementos a los que llamaremos sitios y Λ un conjunto finito al cual nos referiremos como espacio fase. Un campo aleatorio con sitios S y espacio fase Λ es una colección $\{X(s)_{s\in S}\}$ de variables aleatorias con valores en Λ.

Alternativamente, un campo aleatorio es una variable aleatoria que toma sus valores en el espacio Λ^S. Al espacio Λ^S le llamaremos espacio de configuraciones. Para una configuración fija x y dado un subconjunto $A \subset S$, definimos:

$$x(A) = (x(s), s \in A)$$

como la restricción de la configuración x al subconjunto A. Por comodidad, algunas veces escribiremos a toda la configuración x como la concatenación de la configuración x restringida a A con la configuración x restringida al complemento de A, es decir

$$x = (x(A), x(S \setminus A)).$$

Veamos un ejemplo. Consideremos $S = \mathbb{Z}_5 \times \mathbb{Z}_5$ y $\Lambda = \{0, 1\}$. En este caso, una configuración posible puede representarse gráficamente de la siguiente manera:

$$\begin{array}{ccccc} + & - & - & - & + \\ - & + & - & + & - \\ + & + & + & - & + \\ - & + & - & + & + \\ + & + & + & - & + \end{array}$$

Ahora procederemos a definir la dependencia posible entre las variables que conforman al campo aleatorio. Para ello definimos un sistema de vecindades en el conjunto de sitios S.

Definición 7.1. *Un sistema de vecindades de S es una familia $\{\mathcal{N}_\int\}_{s\in S}$ de subconjuntos de S tales que para toda $s \in S$:*

1. *$s \notin S$,*
2. *Si $t \in \mathcal{N}_s$ entonces $s \in \mathcal{N}_t$.*

Denotaremos por $\mathcal{N}_s$ a la vecindad de s. Y definiremos como $\widetilde{\mathcal{N}_s} = \mathcal{N}_s \cup \{s\}$ a la vecindad no agujereada de s. Cuando S consiste en un malla cuadralada (finita o infinita) un ejemplo clásico de vecindades son las α (vecindad de von Neumann) y β (vecindad de Moore):

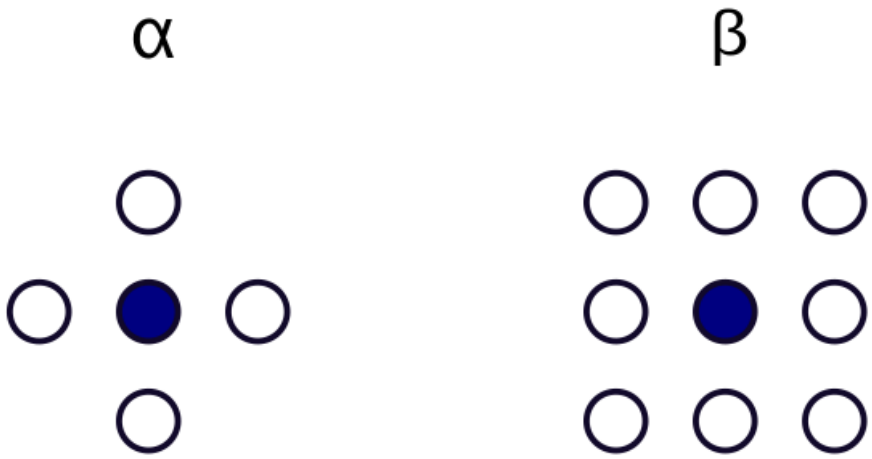


Figura 7.1: Tipos de vecindades.

Ahora explicaremos cómo se puede definir a la propiedad de Markov en este contexto.

Notemos que si tenemos una cadena de Markov $\{X_n\}_{n\geq 0}$ se puede probar (utilizando la propiedad de Markov y el teorema de Bayes) que para todo n natural ocurre lo siguiente:

$$(X_1, ..., X_{n-1})|_{X_n} \perp (X_{n+1}, ..., X_{n+m})|_{X_n},$$

es decir, que condicionalmente al valor de X_n, los vectores $(X_1, ..., X_{n-1})$ y $(X_{n+1}, ..., X_{n+m})$ son independientes. Esta propiedad, que es de hecho equivalente a la propiedad de Markov, nos da una intuición espacial: dado un sitio n, si conocemos el valor de la variable X_n, ocurre que lo que queda a la izquierda es independiente a lo que queda a la derecha de tal posición.

Siguiendo tal intuición espacial hacemos una extensión para definir la propiedad de Markov en campos aleatorios.

Definición 7.2. *Decimos que X es un campo aleatorio de Markov con respecto al sistema de vecindades $\{\mathcal{N}\}_{s\in S}$, si para todos los sitios de S las variables aleatorias $X(s)$ y $X(S \setminus \widetilde{\mathcal{N}_s})$ son independientes condicionadas a $X(\mathcal{N}_s)$, es decir:*

$$\mathbb{P}(X(s) = x(s)|X(S \setminus s) = x(S \setminus s)) = \mathbb{P}(X(s) = x(s)|X(\mathcal{N}_s) = x(\mathcal{N}_s)),$$

para todo $s \in S$ y $x \in \Lambda^s$.

El concepto análogo a las probabilidades de transición para cadenas de Markov está dado por las especificaciones locales para campos de Markov.

Definición 7.3. *La especificación local de un campo aleatorio de Markov en el estado s es la función $\pi^s : \Lambda^s \to [0, 1]$ definida por:*

$$\pi^s(x) = \mathbb{P}(X(s) = x(s)|X(\mathcal{N}_s) = x(\mathcal{N}_s))$$

La familia $\{\pi^s\}_{s\in S}$ es llamada la especificación local del campo aleatorio de Markov.

Campos de Gibbs

En esta sección definiremos a los campos de Gibbs, que son un tipo de campos aleatorios con una forma específica para su distribución conjunta. Su importancia radica en que, bajo ciertas condiciones generales, un campo de Markov es un campo de Gibbs y viceversa. La forma específica de los campos de Gibbs nos permite hacer cálculos explícitos, por lo que nos da una manera de implementar el muestreo de Gibbs en una forma detallada.

Comenzaremos con los conceptos utilizados para caracterizar la dependencia de las variables aleatorias de un campo de Gibbs (que definiremos después). Continuamos en el contexto de un campo aleatorio con espacio de sitios S y espacio fase Λ.

Definición 7.4. *Un subconjunto $C \subset S$ es un clique si y solo si:*

1. *$C = \{s\}$ ó*
2. *Para todo par de sitios s, t tales que $s, t \in C$ ocurre que s y t son vecinos.*

Notemos que para las vecindades α y β presentadas anteriormente tenemos que los posibles cliques, sin contar rotaciones, son de la forma:

Definimos ahora al potencial para un campo de Gibbs, que nos es una colección de funciones en donde cada función asociada a un clique C depende exclusivamente de los valores de la configuración en tal clique C.

Definición 7.5. *Un potencial de Gibbs en el espacio Λ^S relacionado a un sistema de vecindades $\{\mathcal{N}_s\}_{s\in S}$ es una colección de funciones $\{V_C\}_{C\subset S}$, $V_C : \Lambda^S \to \mathbb{R} \cup \{\infty\}$ tales que:*

1. *$V_C \equiv 0$ si C no es un clique.*
2. *$\forall x, x' \in \Lambda^S$ y $\forall C \subset S$ si $x(C) = x'(C) \implies V_C(x) = V_C(x')$*

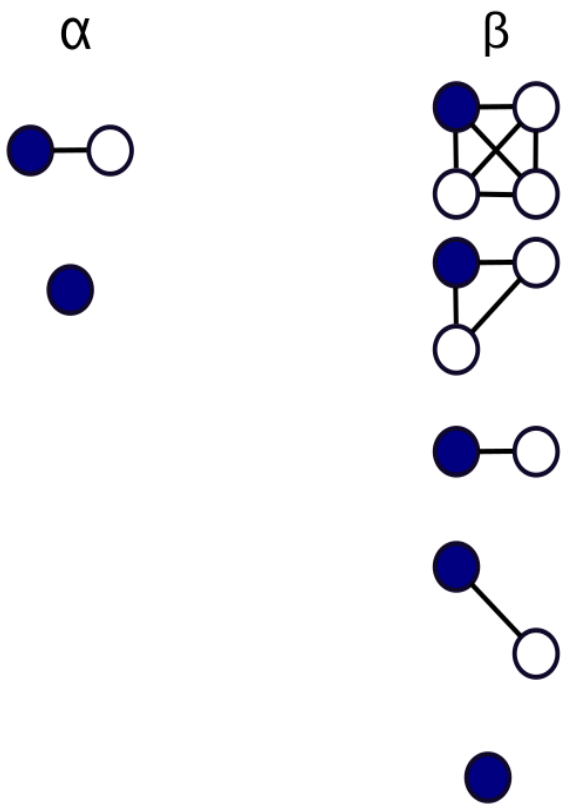


Figura 7.2: Todos los cliques de cada vecindad (salvo rotaciones).

A través del potencial de Gibbs, podemos definir a la energía de una configuración x como la suma de todos las funciones potenciales (sumando sobre todos los cliques) de la configuración x.

Definición 7.6. *La función de energía* $\varepsilon : \Lambda^S \to \mathbb{R} \cup \{\infty\}$ *está dada por*

$$\varepsilon(x) = \sum_C V_C(x).$$

Tras definir estos conceptos, podemos definir a una distribución de Gibbs. Un campo aleatorio tiene una distribución de Gibbs si tiene la forma:

$$\pi_T(x) = \frac{1}{Z_T} e^{-\frac{1}{T}\varepsilon(x)},$$

donde Z_T es una constante de normalización llamada función de partición, que hace que $\{\pi_T(x)\}_{x\in\Lambda^S}$ sea una función de masa de probabilidades. La función de energía proviene de un potencial de Gibbs, para un sistema de cliques. Al parámetro $T > 0$ se le conoce como la temperatura del sistema, y tiene la siguiente intuición (correspondiente a la física): cuando T crece a infinito, la distribución de Gibbs se acerca a una distribución uniforme sobre el espacio de configuraciones Λ^S. En el caso en que T decrece a 0, la distribución de Gibbs tiende a la distribución de probabilidad de una variable aleatoria que toma con probabilidad 1 a la configuración x^* que maximiza $\pi_T(x)$.

En cuanto a la relación entre campos de Gibbs y campos de Markov nos contentaremos en enunciar el siguiente resultado, que nos indica que los campos de Gibbs son de hecho campos de Markov. El inverso es cierto: bajo ciertas condiciones muy generales, un campo de Markov puede expresarse como un campo de Gibbs.

Teorema 7.1. *Si X es una v.a. con distribución π donde su función de energía proviene de un potencial de Gibbs $\{V_C\}_{C\subset S}$ relativo al sistema de vecindades de $\{\mathcal{N}_s\}_{s\in S}$ entonces X es un campo aleatorio de markov respecto a las mismas vecindades $\{\mathcal{N}_s\}_{s\in S}$ y sus especificaciones están dadas por:*

$$\pi^s(x) = \frac{e^{-\sum_{C\ni s} V_C(x)}}{e^{-\sum_{\lambda\in\Lambda} V_C(\lambda, x(S\setminus s))}}.$$

La notación $\sum_{C\ni s}$ significa la suma sobre todos los conjuntos C tales que contengan al estado s.

Veamos ahora un ejemplo específico de todos los conceptos presentados. Consideremos el modelo de Ising, un ejemplo clásico de la física estadística en donde se modela el ferromagnetismo. En este caso podemos considerar $S = \mathbb{Z}_m^2$ un toro discreto, y un espacio fase $\{+,-\}$ donde los valores del espacio fase representan si un sitio s está imantado positiva o negativamente.

Sabemos que, físicamente, un cuerpo tiende a imantarse con el mismo signo que los cuerpos que tiene cercanos. Eso lo podemos expresar de la siguiente manera: consideremos un sistema de vecindades α y cliques dados por tales vecindades. Consideremos que los únicos cliques con función potencial distinto de cero tienen la forma $\{s,t\}$ donde s y t difieren exactamente en una coordenada, por $+1$ o -1 y el potencial está dado por:

$$V_{\{s,t\}} = -x(s)x(t).$$

Así, la energía está dada por

$$\varepsilon(x) = -\sum_{\{s,t\}\text{ clique}} x(s)x(t)$$

y la medida de Gibbs inducida es la siguiente:

$$\pi_T(x) = \frac{1}{Z_T} e^{\frac{1}{T}\sum_{\{s,t\}\text{clique}} x(s)x(t)}.$$

Notemos que esta medida codifica el fenómeno que queremos capturar. La configuración

$$\begin{array}{ccccc} + & + & + & + & + \\ + & + & + & + & + \\ + & + & + & + & + \\ + & + & + & + & + \\ + & + & + & + & + \end{array}$$

tiene todos los vecinos con fases del mismo signo, por lo que su energía será pequeña. Eso implica que tal configuración tiene gran probabilidad. Por otro lado, la configuración

$$\begin{array}{ccccc} + & - & + & - & + \\ - & + & - & + & - \\ + & - & + & - & + \\ - & + & - & + & - \\ + & - & + & - & +. \end{array}$$

tiene muchos más vecinos con fases de diferente signo, por lo que su energía será mayor y por lo tanto su probabilidad será menor que la de la configuración de arriba.

Utilizando el Teorema 7.1 tenemos que las especificaciones locales para nuestro ejemplo están dadas por

$$\pi^s(x) = \frac{g(x(s))}{g(x(s)) + g((1 - x(s)))},$$

donde

$$g(i) = e^{-(ix(s+\mathbf{e}_1)+ix(s-\mathbf{e}_1)x(s)+ix(s+\mathbf{e}_2)+ix(s-\mathbf{e}_2))} \tag{7.2}$$

para valores $i = +1, -1$, donde $\mathbf{e}_1$ y $\mathbf{e}_2$ representan a los vectores canónicos.

Utilizando tales especificaciones locales (funciones de transición), podemos utilizar el muestreo de Gibbs para simular a la distribución estacionaria de un campo aleatorio de Ising en el toro finito discreto. El algoritmo es el siguiente:

Algoritmo 52 : Modelo de Ising

1: Comenzar con una configuración arbitraria $x_0 \in \{+, -\}^{\mathbb{Z}^2_m}$.
2: Elegir al azar y de forma equiprobable un sitio s in $\mathbb{Z}^2_m$.
3: Actualizar el valor de la configuración en el sitio s: $x(s)$ toma el valor $+$ con probabilidad $\pi^s(1)$, y el valor $-$ con probabilidad $\pi^s(-1)$, donde π^s está definida por (7.2).
4: Volvemos al paso 2.

La implementación de tal algoritmo nos ayudaría a estudiar la evolución del magnetismo en este modelo simple.

7.3. Muestreo de Slice

En esta sección analizaremos otra forma particular del muestreo de Metropolis-Hastings, que tiene una interpretación geométrica muy directa.

Supongamos que queremos simular valores de una variable aleatoria X con función de densidad $f(x)$. Consideremos al conjunto

$$B = \{(x, y) : 0 < y < f(x)\}.$$

La idea básica es la siguiente: si conseguimos generar un vector (X, Y) que tenga distribución uniforme sobre el conjunto B, entonces la marginal X tendrá la densidad deseada. Esto se puede verificar fácilmente:

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x, y)dy = \int_0^{f(x)} cdy = cf(x),$$

por lo que X tiene a f como densidad (y además la constante de normalización c es igual a 1).

Sin embargo, generar un vector uniforme (X, Y) en el conjunto B no necesariamente es sencillo. Por lo cual se propone el siguiente algoritmo, que como veremos después, es un algoritmo similar al muestreo de Gibbs:

El nombre del algoritmo Slice (rebanar, en inglés) toma su nombre de la interpretación geométrica del algoritmo: Dado el valor $X = x$, se elige el valor Y como un nivel uniforme en la línea que une al punto $(x, 0)$ con el punto $(f(x), 0)$, es decir, como un valor uniforme en una rebanada vertical. Dado el valor

Algoritmo 53 : Muestreo Slice

1: Comenzar con un valor arbitrario $X = x$, tal que $f(x) > 0$.
2: Dado $X = x$, elegir Y como un valor uniforme entre 0 y $f(x)$.
3: Dado $Y = y$, elegir X de manera uniforme en el conjunto S, donde $S = \{x : y < f(x)\}$.
4: Volver al paso 2.

$Y = y$ se elige al punto X como un valor uniforme en el segmento (o unión de segmentos) de los valores x que al evaluarlos llegan a la línea de nivel y (en el conjunto $\{x : f(x) = y\}$), es decir, como un valor uniforme en una rebanada horizontal.

Este algoritmo tiene particularidades a observar. Por un lado, a diferencia del algoritmo de Metropolis, por ejemplo con el esquema de caminata aleatoria, hay una autoregulación de los valores que va tomando X: la misma densidad f regula qué tan probable es saltar al valor siguiente, haciendo improbable ir a un valor donde la densidad es pequeña y haciendo más probable ir a valores donde la densidad es grande. Por otro lado, los valores consecutivos generados por el algoritmo tienen alta dependencia.

A continuación probaremos la eficacia del algoritmo. Para ello basta verificar que este algoritmo es una variación del muestreo de Gibbs, donde la distribución invariante es uniforme sobre el conjunto B.

Notemos que debido a que las actualizaciones son deterministas (primero en X, luego en Y, luego en X, luego en Y, etc.) tenemos un esquema de Metropolis-Hastings, con matriz de transición:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Ahora, para verificar que en efecto tenemos como distribución estacionaria a un vector (X, Y) uniforme sobre el conjunto B, nos basta ver que las transiciones dadas por el algoritmo coinciden con las densidades marginales $f_{Y|X}$ y $f_{X|Y}$ de tal vector (y por lo tanto la probabilidad de aceptación es igual a 1). Lo probamos a continuación.

Proposición 7.3. *Sea (X, Y) con distribución uniforme sobre el conjunto $B = \{(x, y) : 0 < y < f(x)\}$. Entonces ocurre que*

1. *$Y|_{X=x}$ tiene distribución uniforme en el intervalo $(0, f(x))$.*
2. *$X|_{Y=y}$ tiene distribución uniforme sobre el conjunto S, donde $S = \{x : y < f(x)\}$.*

Demostración. 1) Notemos que para x tal que $f(x) > 0$ ocurre que

$$f_{Y|X}(y|x) = \frac{f_{X,Y}(x, y)}{f_X(x)} = \frac{1_B(x, y)}{f(x)} = \frac{1_{\{0<y<f(x)\}}(y)}{f(x)},$$

donde utilizamos lo que presentamos al principio de esta sección: la marginal de X es la función $f(x)$.

2)Definamos $S = \{x : y < f(x)\}$. Calculemos primero la densidad de Y:

$$f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x, y)dx = \int_S 1dx = \lambda(S)$$

donde $\lambda(S)$ denota a la longitud del conjunto S. Entonces, para $0 \leq y \leq \max_{x\in\mathbb{R}} f(x)$ ocurre que

$$f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} = \frac{1_B(x,y)}{\lambda(S)} = \frac{1_S(x)}{\lambda(S)},$$

que es lo que se quería demostrar.

□

7.3.1. Implementación y calibración

Es importante aclarar que en la implementación de simulaciones Monte Carlo a través de cadenas de Markov pueden aparecer varios problemas. En esta sección discutimos algunos de estos problemas y las soluciones prácticas que se suelen utilizar para remediarlos.

Uno de los problemas es el siguiente: aunque la existencia de una distribución estacionaria y la convergencia a ella pueden ser demostrados formalmente, las condiciones para que se cumpla la convergencia no son fáciles de demostrar en problemas concretos y que aún si pudiera hacerse, la convergencia puede ser muy lenta. Por ejemplo, en el caso del algoritmo de Metropolis-Hastings, la probabilidad de moverse a una nueva posición, puede ser muy pequeña y la cadena podría tardarse mucho en visitar todo el espacio.

Es entonces muy importante corroborar si efectivamente la cadena alcanza a la distribución estacionaria. Debemos dar un tiempo a que la cadena se estabilice, es decir, tomar un cierto número de iteraciones como un "período de calentamiento" y luego de este período analizar la convergencia y estabilidad de la cadena. Para determinar el período de calentamiento una sugerencia es hacer gráficas de los estados visitados por la cadena. También se pueden graficar funciones de interés (por ejemplo la media muestral). Lo fundamental en este sentido es monitorear la convergencia de las características estadísticas de los datos. Por ejemplo, se pueden observar los coeficientes de correlación serial, para tener idea de la velocidad de convergencia (correlación alta implica convergencia lenta). Una propuesta para medir la correlación serial de orden k está dada por:

$$\rho_k = \frac{\sum_{j=1}^{n-k}(x_j - \bar{x})(x_{j+k} - \bar{x})}{\sum_{j=1}^{n}(x_j - \bar{x})^2}$$

El monitoreo consiste en verificar si el estimador anterior tiende a cero conforme k crece.

Un problema fundamental subyacente es que las observaciones generadas no son independientes (pues son construidos a través de la dinámica de una cadena de Markov) a pesar de que el objetivo es generar una muestra aleatoria.

Existen diversas sugerencias para resolver el problema de la dependencia de los datos. Gelfand & Smith (1990), sugieren hacer corridas simultaneas de n cadenas de forma independientemente hasta obtener la convergencia en cada una. El problema en esta metodología es que si la convergencia se alcanza después de m iteraciones, entonces para obtener una muestra de tamaño n, es necesario generar mn observaciones. Por otro lado, Geyer (1992), sugiere que una vez que se alcanza la convergencia, basta con generar n observaciones más y tomar estas como muestra, pues aunque no son independientes la teoría ergódica sustenta la inferencia. Otra propuesta es la de Raftery & Lewis (1992) quienes sugieren que, tras un período de calentamiento, se seleccione un elemento para la muestra períodicamente, es decir, cada k observaciones. Por su parte, Gelman & Rubin (1992) proponen que es recomendable correr, de manera simultánea, l cadenas independientes, con l no mayor a diez, y de cada una tomar n/l observaciones,

ya sea de manera consecutiva o cada k pasos. Se recomienda realizar distintas metodologías como las sugeridas.

7.4. Ejercicios

1. Implementar el algoritmo propuesto para la distribución Autoexponencial.

2. Implementar un algoritmo de Metropolis-Hastings para generar valores provenientes de una variable aleatoria $X \sim N(0,1)$. Utilice el método de la caminata aleatoria simétrica, eligiendo a una transición Q de la forma:

$$Y = X_n + U,$$

donde $U \sim Unif[-a, a]$ para $a > 0$. Hacer un histograma con una muestra grande generada por el algoritmo.

3. Hagamos un algoritmo de Metropolis-Hastings para generar valores provenientes de una variable aleatoria con densidad:

$$f(t) = Csen(x)^2 sen(2t)^2 \phi(t),$$

donde $C > 0$ y $\phi(t)$ denota a la densidad de una normal estándar. Utilice el método de la caminata aleatoria simétrica, eligiendo a una transición Q de la forma:

$$Y = X_n + U,$$

donde $U \sim Unif[-a, a]$ para $a > 0$. Permita que el algoritmo tenga un período de calentamiento k, genere después una muestra de tamaño n grande y grafíquela con un histograma.

4. Para una normal multivariada estándar con coeficiente de correlacion $\rho = -0.85$, estimar el valor

$$\mathbb{P}(X^2 + Y^2 > 1.5)$$

utilizando el muestreo de Gibbs. Considere que el programa contemple un tamaño de muestra n y un período de calentamiento k.

5. En la sección del ejemplo de aplicación bayesiana, considere una muestra de datos dada por

$$X = \{7, 3, 6, 5, 3, 2, 0, 6, 4, 5, 2, 2, 4, 4, 3, 4, 3, 3, 5, 0\},$$

y $\alpha = 4$, $\beta = \frac{3}{2}$, $\sigma = 2$, $\delta = 1$. Implementar el algoritmo y estimar los parámetros λ, τ y m. Comparar con los parámetros reales que resultaron al hacer la simulación de la muestra: $\lambda = 4.2729$, $\tau = 2.9205$, $m = 14$.

6. Hacer un programa que simule y grafique el modelo de Ising en el toro $\mathbb{Z}_5 \times \mathbb{Z}_5$. Hacer que el programa imprima la configuración inicial, al paso 50, al paso 1000, y al paso 10,000.¿Qué pasa a largo plazo? ¿Hay una o varias configuraciones finales deterministas o converge hacia una distribución aleatoria estacionaria?

7. Utilizar el muestreo de slice para simular una muestra proveniente de la densidad

$$f(x) = \frac{1}{2}\exp(-\sqrt{x}), \qquad x > 0.$$

a) Simular, mediante el muestreo de Slice, una colección de variables $\{X_{k+1}, ..., X_{k+n}\}$, para algún período de calentamiento k, y n fijo. Calcular la distribución empírica:

$$\hat{F}(t) := \frac{1}{n} \sum_{i=1}^{n} 1_{\{X_i \leq t\}},$$

y graficarla.

b) Probar que si X tiene la densidad f y se hace la transformación $Y = \sqrt{X}$ entonces $Y \sim Gamma(\frac{3}{2}, 1)$.

c) Utilizar el inciso anterior para simular una muestra de v.a. $\{X_1, ..., X_n\}$ con densidad f (utilizando el mismo n que en el primer inciso). Calcular la distribución empírica y graficarla.

d) Comparar ambas aproximaciones con la densidad teórica.¿Qué tan buenas son? ¿Cuál es mejor? ¿Qué tanto difieren, alterando k y n ?

Capítulo 8

Optimización

8.1. Algoritmos de optimización

En esta sección analizaremos una clase de algoritmos de optimización (o de búsqueda) agrupados bajo el término de *enfriamiento estocástico*.

8.1.1. Esquema frío

Consideremos un conjunto finito o infinito E de estados (o soluciones) y una función $C : E \to \mathbb{R}$ una función de costo (utilidad) a ser minimizada (maximizada). Buscamos algún valor $i^* \in E$ tal que

$$C(i^*) \leq C(i) \qquad \forall i \in E,$$

es decir la mejor solución. Cuando el espacio E es muy grande o infinito, es poco práctico imposible enumerar todos los valores $\{C(i)\}_{i \in E}$ para elegir al valor mínimo y al estado minimizante. Describimos a continuación un método simple para intentar encontrarlo.

Definimos un sistema de vecindades, que son una colección de conjuntos $\{N(i)\}_{i \in E}$ (donde $N(i) \subseteq E$), tales que

1. $i \notin N(i)$, es decir i no es vecino de sí mismo,
2. y si $i \in N(j)$ ocurre que $j \in N(i)$, es decir que el que i sea vecino de j implica que j sea vecino de i.

Pediremos además que el sistema de vecindades $\{N(i)\}_{i \in E}$ sea comunicante, es decir, que para todo i, j en E, existe una sucesión $i_1, ..., i_m$ en E tales que

$$i_1 \in N(i),\ i_2 \in N(i_1),\ ...,\ j \in N(i_m),$$

en otras palabras, de un estado i a otro j, se puede llegar a través de un número finito de vecinos.

El *esquema frío* consiste en el siguiente algoritmo.

Tenemos dos observaciones. La primera es: en el caso en que tenemos un espacio E no numerable y elegimos como sistema de vecindades a $N(i) := E \setminus \{i\}$ (es decir todos los estados diferentes i, j son vecinos)

Algoritmo 54 :Esquema frío

1: Iniciamos con un estado o solución i.
2: Elegimos un vecino $j \in N(i)$ aleatoriamente (no necesariamente de forma equiprobable).
3: Comparamos a $C(i)$ y $C(j)$ y elegimos a i o j como aquél que tiene el menor costo.
4: Repetimos el algoritmo desde el segundo paso

y eligimos a un vecino de forma equiprobable, el Algoritmo 54 con n iteraciones, es equivalente a sortear $n-1$ estados o soluciones de forma independiente, añadir la solución inicial, y elegir entre esos estados a aquel que minimiza la función C.

La segunda observación nos habla de la gran limitación de este algoritmo. En el caso en que tengamos una solución i tal que

$$C(i) \leq C(j) \qquad \forall j \in N(i),$$

es decir un estado que es un mínimo local, el algoritmo se queda atrapado en ese valor, cuando posiblemente exista una mejor solución global.

8.1.2. Algoritmos evolutivos

Los algoritmos evolutivos se pueden conceptualizar dentro de los algoritmos de optimización o de búsqueda. Presentamos a continuación una versión simple, que es una generalización del esquema frío.

Algoritmo 55 Evolutivo

1: Iniciamos con un estado o solución i.
2: Con probabilidad α (muy baja) elegimos a j uniformemente sobre todo el espacio de soluciones E (*mutación*). Con probabilidad $1-\alpha$ (muy alta) elegimos a un vecino $j \in N(i)$ aleatoriamente (*descendencia*).
3: Comparamos a $C(i)$ y $C(j)$ y elegimos a i o j como aquél que tiene el menor costo.
4: Repetimos el algoritmo desde el segundo paso.

La interpretación es la siguiente: Teniendo una solución o individuo i podemos, o bien tener un descendiente (que debe tener características cercanas a i) con probabilidad muy alta, o bien encontramos a otro individuo proveniente de una mutación con probabilidad muy baja. Ambos compiten entre sí, y el que tiene mejor desempeño sobrevive a la siguiente generación.

La gran ventaja de este algoritmo sobre el esquema frío es que no nos quedamos atascados en mínimos locales, pero a diferencia de la búsqueda independiente, la descendencia nos ayuda a utilizar las iteraciones pasadas para un mejor desempeño del algoritmo.

8.1.3. Enfriamiento Estocástico

Este algoritmo generaliza el esquema frío y a su vez, es un caso particular del algoritmo del Metropolis-Hastings.

Dado un sistema de vecindades $\{N(i)\}_{i\in E}$ comunicante, consideremos una cadena de Markov definida sobre E con transición $\{q_{ij}\}_{i,j\in E}$ tal que $q_{ij} > 0$ si y sólo si $i = j$ o bien j está en $N(i)$. Definamos un valor $T > 0$ que representará en este caso la temperatura.

Definimos P_{ij} por

$$P_{ij}(T) := \mathbb{I}_{E\setminus\{i\}}(j)q_{ij}\alpha_{ij}(T) + \mathbb{I}_{\{i\}}(j)\sum_{k\neq i} q_{ik}\alpha_{ik}(T)$$

donde

$$\alpha_{ij}(T) := \text{mín}\left(1, \frac{e^{\frac{C(i)}{T}}}{e^{\frac{C(j)}{T}}}\right).$$

Desde la perspectiva de los algoritmos de Hastings, el parámetro de la temperatura T no influye, es decir la prueba utilizada ahí sigue funcionando en este caso.

Analicemos ahora a las probabilidades de aceptación $\alpha_{i,j}(T)$. En el caso en que $C(i) \geq C(j)$ ocurre que $\alpha_{i,j}(T) = 1$, por lo que aceptamos j (lo cuál es consistente con que queremos minimizar). En el caso en que $C(i) < C(j)$, a diferencia del algoritmo frío, no rechazamos directamente a j. Aceptamos j con probabilidad

$$e^{\frac{C(i)-C(j)}{T}},$$

aunque j sea peor solución que i.

La idea es que, con temperatura positiva, permitimos al sistema de búsqueda explorar regiones aún cuando las soluciones no estén mejorando de forma monónota en cada iteración. La maginitud del parámetro T nos dice que tan posible es para el algoritmo explorar regiones en donde es poco verosímil encontrar el mínimo deseado.

El método de enfriamiento estocástico consiste en ir *disminuyendo* lentamente la temperatura T, de tal forma que al principio exploramos muy diversas regiones, pero a medida en que las iteraciones avanzan buscamos cáda vez más localmente. El algoritmo es el siguiente:

Algoritmo 56 : Enfriamiento Estocástico

1: Iniciamos $X = i$, temperatura $T = t$, iteraciones realizadas $k = 0$.
2: Elegimos a un vecino $j \in N(i)$ con probabilidad $q_{i,j}$.
3: Con probabilidad $\alpha_{i,j}(T)$ hacemos $X = j$.
4: Hacemos $T = T - \frac{t}{n}$, $k = k + 1$.
5: Repetimos hasta que $k = n$.
6: Devolvemos X.

8.2. Algoritmo EM (Optimización en Estadística)

Un problema importante tanto en el enfoque clásico como bayesiano de la estadística es el de maximización, donde desde el punto de vista clásico se buscan estimadores máximo verosímiles y en Bayesiana se desea la maximizacíon de la utilidad esperada en la toma de decisiones por ejemplo.

El algoritmo EM (Esperanza-Maximización) es un algoritmo determinista que busca resolver el problema de maximizar la función de verosimilitud a través de una sucesión relativamente sencilla de problemas de maximización cuyas soluciones convergen al máximo verosímil.

Si bien, el nombre y formulación general del algoritmo EM se le atribuye al paper realizado por Dempster, Laird y Rubin en 1977, la idea de este algoritmo como tal ya había surgido y sido expuesta por varios autores en una amplia gama de artículos que precedieron a este paper. Para saber más sobre la historia del algoritmo, ir a [**?**].

Una de las principales ventajas de este algoritmo es que es un algoritmo iterativo para resolver problemas de máxima verosimilitud cuando cierta información de las variables aleatorias no está observado (información incompleta o faltante). Podemos describir la idea general de este algoritmo de la siguiente forma:

1. Reemplazar los datos faltantes por valores estimados.
2. Estimar los parámetros considerando la información completa con los valores estimados.
3. Repetir los pasos anteriores de la siguiente forma: para 1) se usa como parámetros al estimado y en 2) se usan los valores estimados como observados y se sigue iterativamente hasta la convergencia.

Supongamos que tenemos un vector aleatorio Y con función de densidad $f(Y;\theta)$ donde $\theta \in \Theta \subset R^p$. Si se tiene observado completamente al vector Y, existen diferentes técnicas estadísticas para hacer estimación parametral. Por otro lado, cuando se tienen datos falatantes, se puede considerar que se tiene observada solamente una función de la información completa. Esta situación se puede denotar por

$$Y = (Y_{obs}, Y_{fal})$$

donde Y_{obs} corresponde a los datos observados e Y_{fal} a los datos faltantes. Entonces podemos reescribir a la función de densidad como:

$$f(Y;\theta) = f(Y_{obs}, Y_{fal};\theta) = f_1(Y_{obs};\theta) f_2(Y_{fal} \mid Y_{obs})$$

donde f_1 es la densidad del vector aleatorio Y_{obs} y f_2 es la función de densidad de $Y_{fal} \mid Y_{obs}$ respectivamente. Entonces tenemos que:

$$l_{obs}(\theta; Y_{obs}) = l(\theta; Y) - \log(f_2(Y_{fal} \mid Y_{obs};\theta)),$$

donde $l_{obs}(\theta; Y_{obs})$ es la log-verosimilitud de los datos observados y $l(\theta; Y)$ la log-verosimilitud correspondiente a los datos completos. El algoritmo EM es utilizado en los casos cuando maximizar $l_{obs}(\theta; Y_{obs})$ resulrta tarea complicada pero maximizar $l(\theta; Y)$ es simple. El punto aquí es que al no tener observado a Y no se puede evaluar $l(\theta; Y)$ y mucho menos maximizar.

En el contexto de este problema, el algoritmo EM propone una solución al maximizar $l(\theta; Y)$ de manera iterativa reemplazando los datos faltantes por el valor esperado de éstos dados los datos observados. Esta

esperanza es calculada con respecto a la distribución de los datos completos evaluada en el valor actual del parámetro θ, es decir, si $\theta^{(0)}$ es el valor inicial de θ, entonces la primera iteración se debe calcular como:

$$Q(\theta;\theta^{(0)}) = E_{\theta^{(0)}}[l(\theta;Y) \mid Y_{obs}]$$

y luego $Q(\theta;\theta^{(0)})$ es maximizada como función de θ, es decir, se encuentra $\theta^{(1)}$ tal que

$$Q(\theta;\theta^{(0)}) \leq Q(\theta^{(1)};\theta^{(0)})$$

para todo $\theta \in \Theta$. Ahora estamos listos para enunciar formalmente el algoritmo.

Algoritmo 57 Algoritmo EM

1: Inicializar parámetros $\boldsymbol{\theta}^{(0)}$, $n = 0$.
2: **Paso E** Calcular $Q(\theta;\theta^{(n)}) = E_{\theta^{(n)}}[l(\theta;Y)\,|\,Y_{obs}]$.
3: **Paso M** Maximizar $Q(\theta;\theta^{(n)})$ y obtener siguiente estimador como

$$\theta^{(n+1)} = \arg\max_{\theta\in\Theta} Q(\theta;\theta^{(n)}).$$

4: $t = t + 1$ y regresar a 2.

Estos pasos se repiten hasta la convergencia, es decir, hasta que $L(\theta^{(n+1)}) - L(\theta^{(n)}) < \delta$ para δ lo suficientemente pequeña.

Cuando la distribución de Y pertenece a la familia exponencial, el paso E se reduce a calcular la esperanza de los estadísticos suficientes de la información completa dada la información observada. También el paso M se simplifica, pues implica maximizar la esperanza de la log-verosimilitud calculada en el paso E. En el caso de la familia exponencial, el paso M, puede reemplazarse por sustituir directamente las esperanzas condicionales de los estadísticos suficientes calculados en el paso E, por los estadísticos suficientes resultantes de las expresiones obtenidas para los estimadores máximo verosímiles de la información completa de θ. Veamos algunos ejemplos.

8.2.1. Convergencia del algoritmo

Monotonicidad del algoritmo EM

Una de las propiedades más importantes de este algoritmo demostradas por Dempster, Laird y Rubin, es el comportamiento no decreciente de la función de verosimilitud después de una iteración, es decir:

$$L(\theta^{(k+1)}) \geq L(\theta^{(k)}) \tag{8.1}$$

para $k = 1, 2, ...$ y con $L(\theta^{(k)}) = l_{obs}(\theta; Y_{obs})$. Partimos de definir la función

$$H(\theta' \,|\, \theta) = \mathbb{E}\left[\log f_2(Y_{fal} \,|\, Y_{obs};\theta') \,|\, Y_{obs};\theta\right]. \tag{8.2}$$

Con esto, procedemos a demostrar el siguiente lema, necesario para demostrar la desigualdad en (8.1).

Lema 8.1. *Para cualesquiera* $(\theta', \theta) \in \boldsymbol{\Theta} \times \boldsymbol{\Theta}$,

$$H(\theta \,|\, \theta) \geq H(\theta' \,|\, \theta).$$

Demostración. Sean $(\theta', \theta) \in \boldsymbol{\Theta} \times \boldsymbol{\Theta}$. Entonces,

$$\begin{aligned} H(\theta \,|\, \theta) - H(\theta' \,|\, \theta) &= \mathbb{E}\left[\log(f_2(Y_{fal} \,|\, Y_{obs}; \theta)) \,|\, Y_{obs}; \theta\right] - \mathbb{E}\left[\log(f_2(Y_{fal} \,|\, Y_{obs}; \theta')) \,|\, Y_{obs}; \theta\right] \\ &= \int_{y \in \mathcal{Y}(y_{obs})} \log(f_2(y_{fal} \,|\, Y_{obs}; \theta)) f_2(y_{fal} \,|\, Y_{obs}; \theta) dy_{fal} \\ &- \int_{y_{fal} \in \mathcal{Y}(y_{obs})} \log(f_2(y_{fal} \,|\, Y_{obs}; \theta')) f_2(y_{fal} \,|\, Y_{obs}; \theta) dy_{fal} \\ &= \int_{y_{fal} \in \mathcal{Y}(y_{obs})} f_2(y_{fal} \,|\, Y_{obs}; \theta) \log(\frac{f_2(y_{fal} \,|\, Y_{obs}; \theta)}{f_2(y_{fal} \,|\, Y_{obs}; \theta')}) dy_{fal} \\ &= -\int_{y_{fal} \in \mathcal{Y}(y_{obs})} f_2(y_{fal} \,|\, Y; \theta) \log(\frac{f_2(y_{fal} \,|\, Y_{obs}; \theta')}{f_2(y_{fal} \,|\, Y_{obs}; \theta)}) dy_{fal} \end{aligned}$$

Por la desigualdad de Jensen y la concavidad de la función logaritmo:

$$\begin{aligned} &\geq -\log \int_{y_{fal} \in \mathcal{Y}(y_{obs})} f_2(y_{fal} \,|\, Y_{obs}; \theta) \frac{f_2(y_{fal} \,|\, Y_{obs}; \theta')}{f_2(y_{fal} \,|\, Y_{obs}; \theta)} dy_{fal} \\ &= 0. \end{aligned}$$

□

A la secuencia de iteraciones definida por el algoritmo $\theta^{(0)} \to \theta^{(1)} \to \theta^{(2)}$... la podemos pensar como un mapeo $M : \Theta \to \Theta$ tal que

$$\theta^{(k+1)} = M(\theta^{(k)}),$$

para $k = 0, 1, 2, ...$

Definición 8.1. Un algoritmo iterativo con mapeo $M(\theta)$ es un algoritmo EM generalizado (GEM por sus siglas en inglés) si

$$Q(M(\theta) \,|\, \theta) \geq Q(\theta \,|\, \theta).$$

Emplear un algoritmo EM generalizado puede resultar de gran utilidad, cuando la solución del paso M del algoritmo no es cerrada y el valor θ tal que maximiza la función $Q(\theta \,|\, \theta^{(k)})$ resulta difícil de obtener.

Teorema 8.1. *Para todo algoritmo EM generalizado (GEM),*

$$L(M(\theta)) \geq L(\theta),$$

para todo θ *en* Θ*, cumpliéndose la iguladad si y sólo si*

$$Q(M(\theta) \,|\, \theta) = Q(\theta \,|\, \theta)$$

y

$$f_2(Y_{fal} \,|\, Y_{obs}; M(\theta)) = f_2(Y_{fal} \,|\, Y_{obs}; \theta).$$

Demostración. Notemos que

$$\begin{aligned} L(\theta) &= l_{obs}(\theta; Y_{obs}) \\ &= l(Y; \theta) - \log(f_2(Y_{fal} \,|\, Y_{obs}; \theta)) \end{aligned}$$

cuya esperanza con respecto de la distribución condicional de X dado Y con los parámetros $\theta^{(k)}$ es

$$\begin{aligned} L(\theta) &= \mathbb{E}\left[\log(f(Y; \theta)) \,|\, Y_{obs}; \theta^{(k)}\right] - \mathbb{E}\left[\log(f_2(Y_{fal} \,|\, Y_{obs}; \theta)) \,|\, Y_{obs}; \theta^{(k)}\right] \\ &= Q(\theta \,|\, \theta^{(k)}) - H(\theta \,|\, \theta^{(k)}). \end{aligned}$$

Entonces,

$$\begin{aligned} L(\theta^{(k+1)}) - L(\theta^{(k)}) &= Q(\theta^{(k+1)} \,|\, \theta^{(k)}) - H(\theta^{(k+1)} \,|\, \theta^{(k)}) - Q(\theta^{(k)} \,|\, \theta^{(k)}) + H(\theta^{(k)} \,|\, \theta^{(k)}) \\ &= \{Q(\theta^{(k+1)} \,|\, \theta^{(k)}) - Q(\theta^{(k)} \,|\, \theta^{(k)})\} - \{H(\theta^{(k+1)} \,|\, \theta^{(k)}) - H(\theta^{(k)} \,|\, \theta^{(k)})\} \end{aligned}$$

Claramente la primera diferencia del lado derecho de la igualdad es no negativa, por definición del paso M del algoritmo. Por 8.1, la segunda diferencia es no positiva y por lo tanto, la log-verosimilitud es no decreciente y claramente se da la igualdad si y sólo si $Q(M(\theta) \,|\, \theta) = Q(\theta \,|\, \theta)$ y $f_2(Y_{fal} \,|\, Y_{obs}; M(\theta)) = f_2(Y_{fal} \,|\, Y_{obs}; \theta)$.

□

Corolario 8.1. *Sea $\theta^* \in \Theta$ tal que $L(\theta^*) \geq L(\theta)$ para todo $\theta \in \Theta$. Entonces para todo algoritmo EM generalizado,*

1. $L(M(\theta^*)) = L(\theta^*)$
2. $Q(M(\theta^*) \,|\, \theta^*) = Q(\theta^* \,|\, \theta^*)$
3. $f_2(Y_{fal} \,|\, Y_{obs}; M(\theta^*)) = f_2(Y_{fal} \,|\, Y_{obs}; \theta^*)$

Demostración. Del Teorema 8.1 junto con los supuestos tenemos lo siguiente:

1. $L(\theta^*) \geq L(\theta)$, para todo $\theta \in \Theta$
2. $L(M(\theta^*)) \geq L(\theta)$, para todo $\theta \in \Theta$,

lo cual implica que tanto $L(M(\theta^*)) \geq L(\theta^*)$ como $L(M(\theta^*)) \leq L(\theta^*)$ se cumpla. Por lo tanto, $L(M(\theta^*)) = L(\theta^*)$. Además del Teorema 8.1 también tenemos $Q(M(\theta^*) \,|\, \theta^*) = Q(\theta^* \,|\, \theta^*)$ y $f_2(Y_{fal} \,|\, Y_{obs}; M(\theta^*)) = f_2(Y_{fal} \,|\, Y_{obs}; \theta^*)$.

□

Corolario 8.2. *Sea $\theta^* \in \Theta$ tal que $L(\theta^*) > L(\theta)$ para todo $\theta \in \Theta$ tal que $\theta^* \neq \theta$, entonces paraa todo algoritmo EM generalizado,*

$$M(\theta^*) = \theta^*$$

Demostración. Supongamos que $M(\theta^*) \neq \theta^*$

Del Corolario 8.1 tenemos que

$$L(M(\theta^*)) = L(\theta^*), \tag{8.3}$$

y de la hipótesis tenemos

$$L(\theta^*) > L(\theta), \tag{8.4}$$

para todo $\theta \in \Theta$ tal que $\theta^* \neq \theta$. Pero supusimos que $M(\theta^*) \neq \theta^*$ lo cual contradice que tanto (8.3) como (8.4) se cumplan. Por lo tanto, $M(\theta^*) = \theta^*$.

□

Los corolarios (8.1) y (8.2) nos dicen que si el algoritmo llega a un maximizador global, la log-verosimilitud ya no cambia en las siguientes iteraciones. También nos muestran que si contamos un único maximizador global, el algoritmo tiene un punto fijo.

8.2.2. Error estándar

Una de las principales desventajas del algoritmo EM es que no calcula de manera directa la matriz de covarianzas del estimador máximo verosímil.

Denotamos como

$$\boldsymbol{S}_{obs}(\theta) = \frac{\partial}{\partial\theta} l(\theta; Y_{obs}) \tag{8.5}$$

$$\boldsymbol{S}_{c}(\theta) = \frac{\partial}{\partial\theta} l(\theta; Y) \tag{8.6}$$

al score de los datos observados y al de los datos completos, respectivamente, donde $l_{obs}(\theta; Y_{obs})$ hace referencia a la log-verosimilitud de los datos observados y $l(\theta; Y)$ es la log-versimilitud con respecto de los datos completos.

Podemos expresar a $\boldsymbol{S}_{obs}$ de la siguiente manera:

$$
\begin{aligned}
\boldsymbol{S}_{obs}(\theta) &= \frac{\partial}{\partial\theta} l_{obs}(\theta; Y_{obs}) && (8.7)\\
&= \frac{\partial}{\partial\theta} \log(f_1(Y_{obs};\theta)) && (8.8)\\
&= \frac{\frac{\partial}{\partial\theta} f_1(Y_{obs};\theta)}{f_1(Y_{obs};\theta)} && (8.9)\\
&= \frac{\frac{\partial}{\partial\theta} \int_{y\in\mathcal{Y}} f(y;\theta)dy}{f_1(y_{obs};\theta)} && (8.10)\\
&= \frac{\int_{y\in\mathcal{Y}} \frac{\partial}{\partial\theta} f(y;\theta)dy}{f_1(y_{obs};\theta)} && (8.11)\\
&= \int_{y\in\mathcal{Y}} \frac{\partial}{\partial\theta} \log(f(y;\theta)) \frac{f(y;\theta)}{f_1(y_{obs};\theta)} dy && (8.12)\\
&= \mathbb{E}\left[\frac{\partial}{\partial\theta} \log(f(y;\theta)) \,|\, y_{obs} \right] && (8.13)\\
&= \mathbb{E}\left[\boldsymbol{S}_c(\theta) \,|\, y_{obs} \right] && (8.14)\\
& && (8.15)
\end{aligned}
$$

con $\mathcal{Y} = \{y : y_{obs}(y) = y_{obs}\}$, $f_1(Y_{obs};\theta)$ la función de densidad correspondiente al vector de datos observados y $f(Y;\theta)$ la función de densidad de los datos completos.

Por otro lado, denotemos como I_{obs} e I_c a la información de Fisher de los datos observados y a la de los datos completos, respectivamente; es decir

$$
\begin{aligned}
I_{obs}(\theta) &= -\frac{\partial^2}{\partial\theta} l_{obs}(\theta; Y_{obs}) && (8.16)\\
I_c(\theta) &= -\frac{\partial^2}{\partial\theta} l(\theta; Y) && (8.17)
\end{aligned}
$$

Notemos que

$$
l_{obs}(\theta; Y_{obs}) = l(\theta; Y) - \log(f_2(Y_{fal} \,|\, Y_{obs};\theta)).
$$

con $f_2(Y_{fal} \,|\, Y_{obs};\theta) = \frac{f(Y;\theta)}{f_1(Y_{obs};\theta)}$, la densidad condicional de $Y_{fal} \mid Y_{obs}$.

Entonces,

$$
I_{obs}(\theta) = I_c(\theta) + \frac{\partial^2}{\partial\theta} \log(f_2(Y_{fal} \,|\, Y_{obs};\theta)).
$$

Tomando esperanza sobre la distribución condicional $f_2(Y_{fal} \,|\, Y_{obs};\theta)$ obtenemos la siguiente expresión

$$
I_{obs}(\theta) = \mathcal{I}_c(\theta \,|\, Y_{obs}) - \mathcal{I}_m(\theta \,|\, Y_{obs}),
$$

donde $\mathcal{I}_c(\theta \,|\, Y_{obs}) = -\mathbb{E}\left[\dfrac{\partial^2}{\partial\theta} l(\theta; Y) \,|\, Y_{obs}\right]$ y $\mathcal{I}_m(\theta \,|\, y_{obs}) = -\mathbb{E}\left[\dfrac{\partial^2}{\partial\theta} \log(f_2(y_{fal} \,|\, Y_{obs}; \theta)) \,|\, Y_{obs}\right]$.

Como se demuestra en **??**, $\mathcal{I}_m(\theta \,|\, Y_{obs}) = \text{Var}\left(\boldsymbol{S}_c(\theta) \,|\, Y_{obs}\right)$ y por lo tanto

$$I_{obs}(\theta) = \mathcal{I}_c(\theta \,|\, Y_{obs}) - \text{Var}\left(\boldsymbol{S}_c(\theta) \,|\, Y_{obs}\right). \tag{8.18}$$

Ejemplo 8.2.1. Muestra Normal Univariada

Consideremos a $Y = (Y_1, ..., Y_n)$ una muestra aleatoria tal que $Y_i \sim N(\mu, \sigma^2)$, entonces

$$f(Y; \mu, \sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left[-\frac{1}{2\sigma^2}\left(\sum_i Y_i^2 - 2\mu \sum_i Y_i + n\mu^2\right)\right].$$

Claramente $\sum_i Y_i^2$ y $\sum_i Y_i$ son estadísticos suficientes y se tiene que

$$l(\mu, \sigma^2 \,|\, Y) = -\frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_i Y_i^2 + \frac{\mu}{\sigma^2}\sum_i Y_i - \frac{n\mu^2}{\sigma^2}.$$

Ahora, supongamos que de Y solamente observamos m elementos de la muestra ($m < n$), es decir, $Y_{obs} = (Y_1, ..., Y_m)$. Dado que la distribución normal pertence a la familia exponencial, el algoritmo se reduce a lo siguiente:

Algoritmo 58 Algoritmo EM para muestra normal univariada

1: Inicializar parámetros $(\mu^{(0)}, \sigma^{2(0)})$, $n = 0$.

2: **Paso E** Calcular

$$s_1^{(n)} := E_{\mu^{(n)}, \sigma^{2(n)}}\left(\sum_{i=1}^n Y_i \,|\, Y_{obs}\right) = \sum_{i=1}^m Y_i + (n-m)\mu^{(n)}$$

.

$$s_2^{(n)} := E_{\mu^{(n)}, \sigma^{2(n)}}\left(\sum_{i=1}^n Y_i^2 \,|\, Y_{obs}\right) = \sum_{i=1}^m Y_i^2 + (n-m)(\sigma^{2(n)} + \mu^{2(n)})$$

3: **Paso M** Actualizar estimadores

$$\mu^{(n+1)} = \frac{s_1^{(n)}}{n}.$$

$$\sigma^{2(n+1)} = \frac{s_2^{(n)}}{n} - \left(\mu^{(n+1)}\right)^2$$

4: $n = n + 1$ y regresar a 2.

Se implementó un programa en R, simulando una muestra de tamaño $n = 1000$ de la cual generamos una submuestra de tamaño $m = 500$, la cual hace referencia a los datos observados del problema. Se inicializaron los valores de los parámetros (μ, σ^2) de tres maneras distintas:

1. $\mu_0 = 0.5$, $\sigma_0^2 = 0.5$
2. $\mu_0 = 5$, $\sigma_0^2 = 1$
3. $\mu_0 = 10$, $\sigma_0^2 = 2$.

Después de 20 iteraciones se obtuvieron los siguientes valores:

1. $\mu^{(20)} = 2.989279$, $\sigma^{2(20)} = 0.03984177$
2. $\mu^{(20)} = 2.989284$, $\sigma^{2(20)} = 0.03984019$
3. $\mu^{(20)} = 2.989288$, $\sigma^{2(20)} = 0.03988416$

Comparando los resultados con los estimadores máximo verosímiles de (μ, σ^2) de los datos observados Y_{obs}, cuyos valores son $(2.989282, 0.03983542)$, podemos notar que los resultados se aproximan bastante a los valores reales, como se muestra también en la siguiente gráfica, donde la línea punteada representa el valor del estimador máximo verosímil.

En la Figura 8.1 se ilustra la evolución de los estimadores.

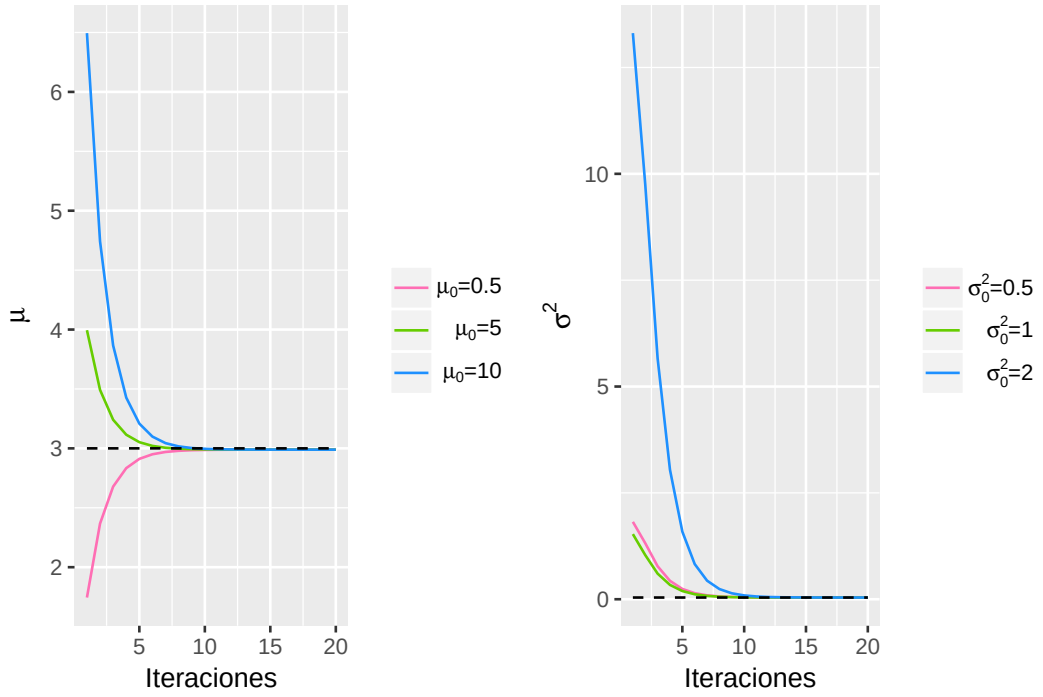


Figura 8.1: Convergencia de parámetros. Software: R

Ejemplo 8.2.2. Muestra de población multinomial

Ahora presentamos un ejemplo ilustrativo de datos observados provenientes de una población multinomial. Dichas observaciones son las frecuencias $Y_{obs} = (38, 34, 125)$ con vector de probabilidades $\left(\frac{1}{2} - \frac{1}{2}\theta, \frac{1}{4}\theta, \frac{1}{4}\theta + \frac{1}{2}\right)$. Supondremos que los datos están incompletos y que los datos completos corresponden a una población con distribución multinomial con cuatro categorías $Y = (Y_1, Y_2, Y_3, Y_4)$ y con probabilidades $\boldsymbol{p} = \left(\frac{1}{2} - \frac{1}{2}\theta, \frac{1}{4}\theta, \frac{1}{4}\theta, \frac{1}{2}\right)$, respectivamente. Notemos que los datos observados están en función de los datos completos del siguiente modo:

$$Y_{obs_1} = Y_1 \tag{8.19}$$
$$Y_{obs_2} = Y_3 \tag{8.20}$$
$$Y_{obs_3} = Y_3 + Y_4 \tag{8.21}$$

Entonces, la log-verosimilitud de los datos completos está dada por

$$l(\theta; Y) = \sum_{i=1}^{4} Y_i \log(p_i) + \text{cte.}$$

Claramente, los estadísticos suficientes están dados por Y_1, Y_2, Y_3, Y_4 y dado que la distribución multinomial pertenece a la familia exponencial el paso E se reduce a obtener:

$$\begin{aligned}
\mathbb{E}\left[Y_1 \,|\, Y_{obs}\right] &= Y_1 \\
\mathbb{E}\left[Y_2 \,|\, Y_{obs}\right] &= Y_2 \\
\mathbb{E}\left[Y_3 \,|\, Y_{obs}\right] &= 125\frac{\frac{1}{4}\theta}{\frac{1}{4}\theta + \frac{1}{2}} \\
\mathbb{E}\left[Y_4 \,|\, Y_{obs}\right] &= 125\frac{\frac{1}{2}}{\frac{1}{4}\theta + \frac{1}{2}}.
\end{aligned}$$

Las últimas dos esperanzas se deben a que los valores Y_3 y Y_4 condicionados a los datos observados, que es la suma de $Y_{obs_3} + Y_{obs_4}$, siguen una distribución binomial con parámetros $(125, p)$, donde p vale

$$p = \frac{\frac{1}{4}\theta}{\frac{1}{4}\theta + \frac{1}{2}}$$

para Y_3 y $1 - p$ para Y_4.

Del paso M obtenemos que el estimador máximo verosímil de θ de los datos completos es

$$\hat{\theta} = \frac{Y_2 + Y_3}{Y_1 + Y_2 + Y_3^{(n)}}, \tag{8.22}$$

pues

$$\frac{\partial Q(\theta;\theta^{(n)})}{\partial\theta} = -\frac{Y_1}{1-\theta} + \frac{Y_2}{\theta} + \frac{Y_3^{(n)}}{\theta}$$

con $Y_3^{(t)} = \mathbb{E}\left[Y_3 \,|\, Y_{obs}\right] = 125\frac{\frac{1}{4}\theta}{\frac{1}{4}\theta+\frac{1}{2}}$. Por lo tanto, maximizando sobre θ en $\Theta = (0,1)$, obtenemos el valor dado en (8.22). Entonces, el algoritmo para este ejemplo es:

Algoritmo 59 Algoritmo EM para ejemplo de Población Multinomial

1: Inicializar parámetro $\theta^{(0)}$, $n = 0$.

2: **Paso E** Calcular $Y_3^{(t)} = 125\frac{\frac{1}{4}\theta^{(n)}}{\frac{1}{4}\theta^{(n)}+\frac{1}{2}}$.

3: **Paso M** Obtener

$$\theta^{(n+1)} = \frac{Y_2 + Y_3^{(n)}}{Y_1 + Y_2 + Y_3^{(n)}}$$

4: $n = n + 1$ y regresar a 2.

En la Figura 8.2 mostramos los resultados de convergencia al estimador máximo verosímil dado por $\theta_{MLE} = 0.626821497$, después de 9 iteraciones y con cuatro parámetros iniciales distintos.

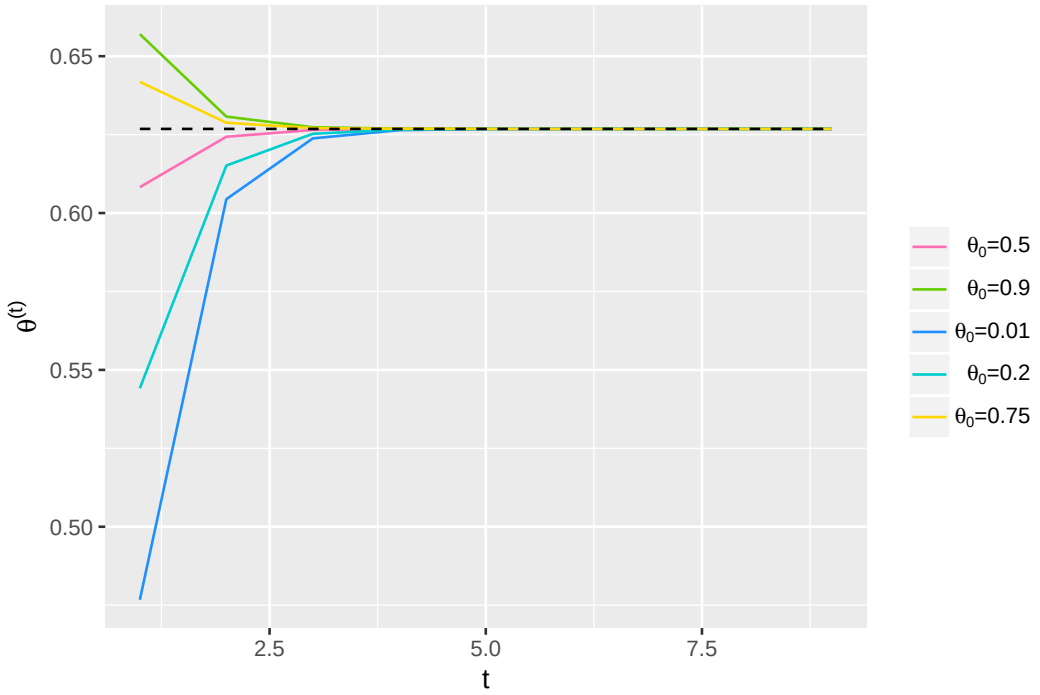


Figura 8.2: Convergencia de θ con distintos valores iniciales . Software: R

Ejemplo 8.2.3. Mezcla de distribuciones Binomial/Poisson

La siguiente tabla muestra el número de hijos de N viudas que recibirán cierta pensión.

Número de hijos:	0	1	2	3	4	5	6
# Observado de viudas:	x_0	x_1	x_2	x_3	x_4	x_5	x_6

Dado que la información obtenida resultó no ser consistente con el hecho de venir de una distribución Poisson (siendo muy grande el número de viudas que no tenían hijos), se propone adoptar el siguiente modelo como alternativa. Supongamos que una variable aleatoria discreta se distribuye como la mezcla de dos poblaciones, por lo que definimos a las poblaciones A y B como:

- **Población A:** Con probabilidad ξ, la variable aleatoria toma el valor de cero, y
- **Población B:** Con probabilidad $1-\xi$, la variable aleatoria sigue una distribución Poisson con media λ.

Reformularemos el problema a uno con datos incompletos, suponiendo que el número de viudas sin hijos resulta de la suma de los casos que vienen de cada una de las poblaciones A o B, por lo que:

- $y_0 : x_A + x_B$
- x_A: Número de viudas sin hijos de la población A.
- x_B: Número de viudas sin hijos de la población B

En este caso, $Y_{fal} = (x_A, x_B)$ y $Y_{obs} = (x_1, x_2, ..., x_6)$ Notamos que este ejemplo es muy parecido al anterior en el sentido que uno de nuestros datos observados (y_0) proviene de una suma de dos de los valores de los datos completos $Y = (x_A, x_B, x_1, x_2, x_3, x_4, x_5, x_6)$, con lo cual la función de probabilidad de los datos completos queda del siguiente modo:

$$f(Y;\boldsymbol{\theta}) = \binom{N}{x_A x_B x_1 x_2 x_3 x_4 x_5 x_6} (\xi + (1-\xi)e^{-\lambda})^{x_A + x_B} \prod_{i=1}^{6} \left((1-\xi)e^{-\lambda}\frac{\lambda^i}{i!} \right)^{x_i}$$

con $\boldsymbol{\theta} = (\xi, \lambda)$.

Por lo tanto, la función de log-verosimilitud de los datos completos está dada por:

$$l(\boldsymbol{\theta};Y) = \log(N) - \log(x_A!) - \log(x_B!) - \sum_{i=1}^{6} \log(x_i!) + (x_A + x_B)\log\left(\xi + (1-\xi)e^{-\lambda}\right) \tag{8.23}$$

$$+ \sum_{i=1}^{6} x_i \left(\log(1-\xi) - \lambda\right) + \sum_{i=1}^{6} x_i \left(i\log(\lambda) - \sum_{k=1}^{i} \log(k) \right), \tag{8.24}$$

con $\boldsymbol{\theta} = (\xi, \lambda)$.

Obteniendo de (8.23) los estadísticos suficientes $(x_A, x_B, x_1, x_2, x_3, x_4, x_5, x_6)$ y como podemos expresar la función de (8.23) de la forma mostrada en (sección familias exponenciales) el paso E se define como

$$\mathbb{E}\left[x_i \,|\, \boldsymbol{Y_{obs}}\right] = x_i,$$

para $i = 1, 2, ..., 6$ y

$$\mathbb{E}\left[x_A \,|\, \boldsymbol{Y_{obs}}\right] = \frac{x_0 \xi}{\xi + (1-\xi)e-\lambda},$$

pues $x_A \,|\, \boldsymbol{Y_{obs}} \sim Bin(x_0, p_1)$, con $p_1 = \frac{p_A}{p_A + p_B}$ con $p_A = \xi$ y $p_B = (1-\xi)\exp -\lambda$. Análogamente, $x_B \,|\, \boldsymbol{Y_{obs}} \sim Bin(x_0, p_2)$ con $p_2 = \frac{p_B}{p_A + p_B}$, por lo que a cada iteración, el paso E se reduce al cálculo de:

$$x_A^{(t)} = \frac{x_0 \xi^{(t)}}{\xi^{(t)} + (1-\xi^{(t)})e-\lambda^{(t)}}.$$

Ahora, para el paso M, necesitamos obtener los estimadores máximo verosímiles para ξ y λ. Para ello, observemos que $x_A \sim Bin(N, \xi)$ y los valores $(x_B, x_1, x_2, x_3, x_4, x_5, x_6)$ son frecuencias observadas de una distribución Poisson de parámetro λ. Por lo tanto,

$$\xi^{(t+1)} = \frac{x_A^{(t)}}{N}$$

$$\lambda^{(t+1)} = \frac{\sum_i i x_i}{x_B^{(t)} + \sum_i x_i},$$

con $x_B = y_0 - x_A$. El algoritmo para este ejemplo queda expresado del siguiente modo:

Algoritmo 60 Algoritmo EM para ejemplo de Mezclas Binomial/Poisson

1: Inicializar parámetros $\xi^{(0)}$, $\lambda^{(0)}$ $t = 0$.
2: **Paso E** Calcular

$$x_A^{(t)} = \frac{y_0 \xi^{(t)}}{\xi^{(t)} + (1-\xi^{(t)})e-\lambda^{(t)}}$$

.
3: **Paso M** Obtener

$$\xi^{(t+1)} = \frac{x_A^{(t)}}{N},$$
$$\lambda^{(t+1)} = \frac{\sum_i i x_i}{x_B^{(t)} + \sum_i x_i}$$

4: $t = t + 1$ y regresar a 2.

Implementamos el algoritmo en R con los datos mostrados en la Tabla 8.1 y mostramos los resultados obtenidos usando distintos valores iniciales en la Tabla 8.2 (ver código **??** en Apéndice).

Número de hijos:	0	1	2	3	4	5	6
# Observado de viudas:	3,062	587	284	103	33	4	2

Tabla 8.1: Observaciones utilizadas en la implementación

Valor inicial	Convergencia en iteración	Valor a iteración t
$\xi^{(0)}$	t	$\xi^{(t)}$
0.75	36	0.6150566
0.4	63	0.6150566
0.1	74	0.6150566
0.86	52	0.6150568
$\lambda^{(0)}$	t	$\lambda^{(t)}$
0.4	36	1.037839
0.3	63	1.0378388
0.7	74	1.0378388
0.8	52	1.037839

Tabla 8.2: Convergencia de estimadores para ξ y λ

En la Figura 8.3 se presentan el valor de los estiamdores y de la función de verosimiltud para 50 iteraciones.

8.2.3. Formulación del algoritmo EMMC

A diferencia del algoritmo EM original, el algoritmo EM Monte Carlo (al cual nos referimos como EMMC), sustituye a a esperanza que se calculaba en el paso E del algoritmo EM original, por el estimador Monte Carlo de dicha esperanza, dada por

$$\hat{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) = \frac{1}{m} \sum_{i=1}^{m} l_c(\theta; x) \tag{8.25}$$

donde m es el tamaño de la muestra generada por la función de densidad condicional correspondiente a los datos faltantes dados los datos observados. Una vez obtenido dicho estimador, se procede al paso M y actualizamos $\boldsymbol{\theta}$.

Algoritmo 61 Algoritmo EM Monte Carlo

1: Inicializar parámetros $\boldsymbol{\theta}_i$ para i en $\{1, 2, ..., M\}$, $t = 0$
2: **Paso MC E** Simulamos variables aleatorias $Y_{fal}^1, ..., Y_{fal}^m$ de tamaño m adecuado, de la distribución condicional de los datos faltantes dados los datos observados.
3: Calcular $\hat{Q}(\theta; \theta^{(t)}) = \frac{1}{m} \sum_{i=1}^{m} l_c(\theta; Y_{obs}, Y_{fal}^i)$.
4: **Paso M** Maximizar $\hat{Q}(\theta; \theta^{(t)})$ y actualizar parámetros.
5: Repetir hasta que se estabilice el proceso.

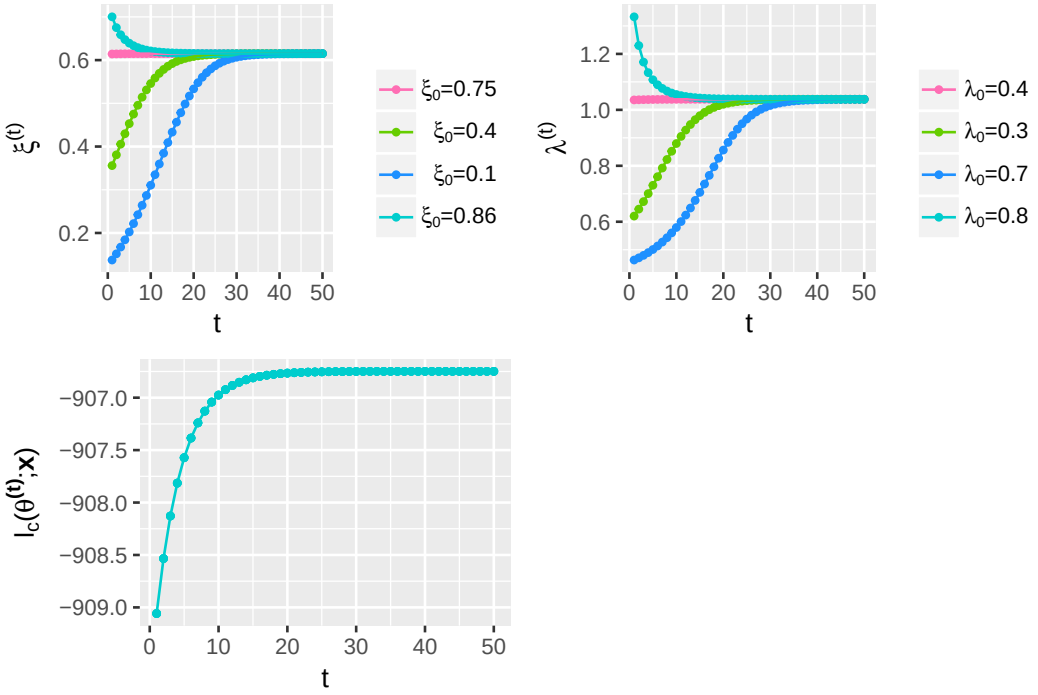


Figura 8.3: Arriba: convergencia de estimadores para ξ, λ con distintos valores iniciales. Abajo: convergencia de la función de log-verosimilitud. Software: R

Notemos que en el paso 5 mencionamos la estabilidad de la secuencia generada por el algoritmo, la cual explicaremos más adelante.

8.2.4. Algoritmo EM estocástico (EME)

El algoritmo EM estocástico (al cual nos referimos como SME por sus siglas en inglÃ©s) difiere del EMMC, o más bien, puede pensarse como un caso específico del EMMC en el cual el tamaño de muestra m es 1, es decir, solamente simulamos una vez a los datos faltantes. El algoritmo queda expresado del siguiente modo:

Algoritmo 62 Algoritmo EM Estocástico

1: Inicializar parámetros $\boldsymbol{\theta}_i$ para i en $\{1, 2, ..., M\}$, $t = 0$
2: **Paso MC E** Generamos un valor Y_{fal} de la distribución condicional de los datos faltantes dados los datos observados.
3: Calcular $\hat{Q}(\theta; \theta^{(t)}) = l(\theta; Y_{obs}, Y_{fal})$.
4: **Paso M** Maximizar $\hat{Q}(\theta; \theta^{(t)})$ y actualizar parámetros.
5: Repetir hasta que se estabilice el proceso.

Definición 8.2. Definimos al proceso $\tilde{\theta}_n(k)$ como el parámetro estimado de la muestra completa X de tamaño n en la k-ésima iteración del SEM.

Intuitivamente, podemos notar que se trata de una cadena de Markov y que es necesario también pensar en un tamaño de muestra factible para tener suficiencia de información y bajo costo de implementación. El

ojetivo principal es hacer uso de las propiedades de la cadena de Markov antes mencionada. Consideremos primero la siguiente cadena de Markov:

Definición 8.3. Definimos al proceso $\tilde{X}(k)$ como las observaciones simuladas faltantes en la k-ésima iteración del SEM dado el valor de $\tilde{\theta}_n(k)$.

La forma en que se encuentran definidos ambos procesos implican el resultado crucial para la convergnecia, la cadena de Markov $\tilde{X}(k)$ proporciona ciertas propiedades a la cadena de Markov $\tilde{\theta}_n(k)$. Los siguientes resultados nos permiten justificar la aplicación del algoritmo.

Lema 8.2. *La cadena de Markov $\tilde{X}(k)$ es irreducible y aperiódica.*

Lema 8.3. *La cadena de Markov $\tilde{\theta}_n(k)$ cumple la propiedad de Feller.*

Teorema 8.2. *Sea $c = \log \int k_{\tilde{\theta}}(\tilde{x} \,|\, y) dv_y(\tilde{x})$, donde $\tilde{\theta}$ es el valor máximo verosímil para $\tilde{x}$. Supongamos que $c < \infty$ y que la función*

$$\theta \to \Delta(\theta) = \tilde{E}(\log f_{\tilde{\theta}(k+1)}(\tilde{X}) - \log f_{\tilde{\theta}(k)}(\tilde{X}) \,|\, \tilde{\theta}(k) = \theta),$$

donde $\tilde{X}$ tiene la distribución condicional a las observaciones conocidas, es más grande que $(1+\delta)c$ para algún $\delta > 0$.

Entonces la cadena de Markov $\tilde{\theta}_n(k)$ tiene distribución estacionaria, y además, si $\Delta(\theta)$ se distribuye normalmente entonces la cadena es ergódica.

Además, como lo demuestra (Nielsen 2000), la distribución estacionaria converge a la distribución normal, con lo cual podemos usar como estimador final el promedio de los valores de la cadena, excluyendo en dicho promedio los valores del período de calentamiento.

Ejemplo 8.2.4. Población multinomial

Se presenta el Ejemplo 8.2.2 pero esta vez haciendo uso de del algoritmo EM Monte Carlo. La log-verosimilitud de los datos completos es

$$\begin{aligned} l_c(\theta; x) &= f(x; \theta) && (8.26) \\ &= \text{cte} + y_1 \log\left(\frac{1}{2} - \frac{1}{2}\theta\right) + (y_2 + y_{31}) \log(\theta), && (8.27) \end{aligned}$$

Dado que $Y_{31} \sim \text{Bin}(125, \frac{\theta}{2+\theta})$, procedemos a generar una muestra $z_1, ..., z_m$ proveniente de dicha distibución para obtener el estimador MonteCarlo de la función $Q(\theta; \theta^{(k)})$ como

$$\hat{Q}(\theta; \theta^{(k)}) = y_1 \log\left(\frac{1}{2} - \frac{1}{2}\theta\right) + (y_2 + z_{31}) \log(\theta), \tag{8.28}$$

donde $z_{31} = \frac{1}{m} \sum_{i=1}^{m} z_i$. Una vez completado el paso E, realizamos el paso M del mismo modo que se hizo en el Ejemplo 8.2.2.

	$m = 1$	$m = 10$	$m = 100$
$\hat{\mu}_{\hat{\theta}_{MCEM}}$	0.6216783	0.624364	0.6244651
$\hat{\sigma}_{\hat{\theta}_{MCEM}}$	0.004267926	0.00433769	0.004560546
Tiempo CPU promedio	0.0009733268	0.0008787428	0.002172402

Tabla 8.3: Promedios y desviaciones estándar de $\hat{\theta}_{multMCEM}$ para cada tamaño de muestra y tiempo CPU promedio

Se implementó el código **??** con 7 distintos valores iniciales, como se muestra en la Figura 8.4, para tamaños de muestra de $m = 1, 10$ y 100. Es decir, en total se corrió 21 veces el algoritmo, haciendo 50 iteraciones y de las cuales las primeras 30 las consideramos para el período de calentamiento de la cadena. Por el resultado dado en (Nielsen 2000) que nos dice que la distribución estacionaria converge a una distribución normal, aplicamos para cada secuencia generada la prueba de Anderson-Darling para probar normalidad, con lo cual confirmamos con un nivel de $\alpha = 5\,\%$ que 20 de las 21 secuencias generadas cumplen que siguen una distribución normal, con lo cual se obtuvieron los estimadores finales definidos como:

$$\hat{\theta}_{MCEM} = \frac{1}{20} \sum_{t=30}^{50} \theta^{(t)}$$

De la Tabla 8.3 podemos notar que los promedios obtenidos de los 7 distintos valores de $\hat{\theta}_{MCEM}$ para cada tamaño de muestra, son muy parecidos al valor original $\theta_{MLE} = 0.626821497$. Como podemos notar también de la Figura 8.4, conforme aumenta el tamaño de muestra, claramente la varianza se reduce.

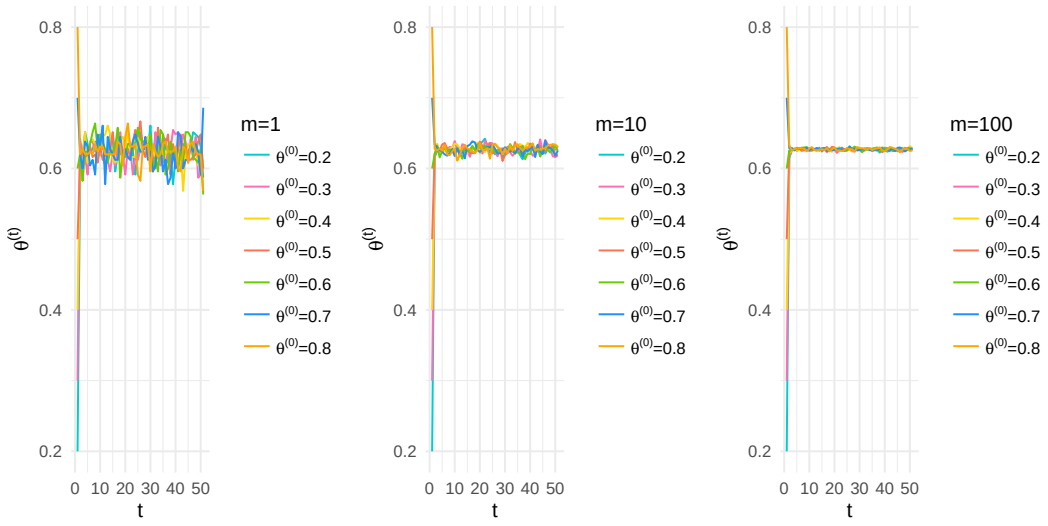


Figura 8.4: Convergencia de parámetros. Software: R

Ejemplo de las viudas

Presentamos el Ejemplo 8.2.3 pero ahora implementando el algoritmo SEM. La única diferencia entre este ejemplo y el Ejemplo 8.2.3 es que simulamos una variable aleatoria con distribución binomial de

parámetros y_0 y p_1, definidas en dicho ejemplo. Implementamos el código **??** iniciazlizando los parámetros con 7 valores distintos como se ve en la Figura 8.5. Para esta implementación corrimos el algoritmo con 100 iteraciones, con un período de calentamiento de 70 iteraciones. La media de los estimadores finales $\hat{\theta}_{MCEM}$ resultó ser de 0.613675 para ξ y de 1.03457 para λ y desviaciones estándar de 0.004045165 y 0.1513798, respectivamente, y un tiempo CPU promedio de 0.001997858.

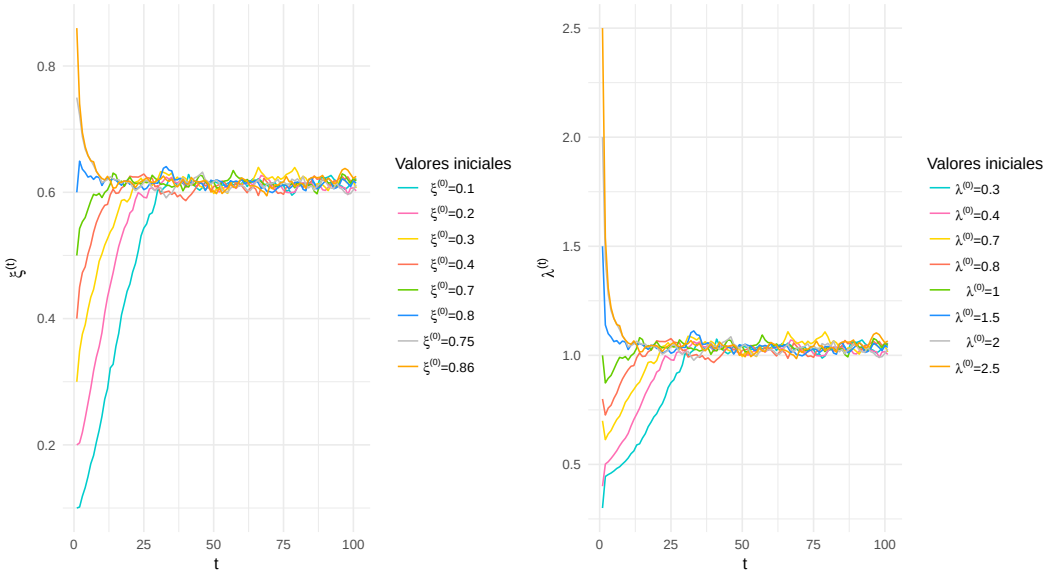


Figura 8.5: Convergencia de parámetros. Software: R

8.3. Ejercicios

1. Hacer en R un programa que resuelva el problema del agente viajero, utilizando **enfriamiento estocástico**, para el siguiente diagrama:

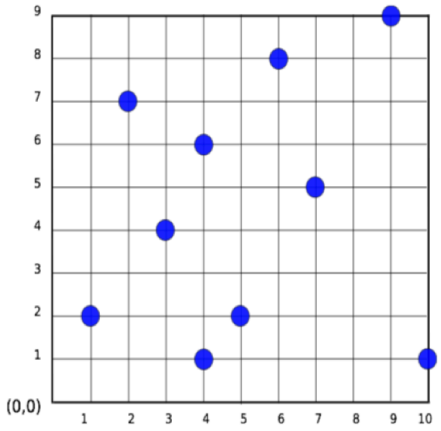


Figura 8.6: Ciudades.

Cada punto en la gráfica representa la locación geográfica de la ciudad. El costo del viaje entre

dos ciudades está dado por la distancia entre ellas: $d((x_1, x_2), (y_1, y_2)) = |x_1 - y_1| + |x_2 - y_2|$. El programa debe encontrar una solución particular al problema, calcular el costo asociado a esa solución y graficar la trayectoria dada por la solución.

2. Definamos a la funcion `Cam` (introducida por Dixon y Szego en 1978) como

$$f(x, y) = 100(x^2 - y)^2 + (1 - x)^2,$$

para (x, y) en el conjunto E definido por las restricciones:

- $0 \leq x \leq 10$,
- $0 \leq y \leq 13$,
- $x + y \leq 12$

Minimizar la función con el algoritmo de esquema frío (elegir un sistema de vecindades, una forma de sortear a un vecino y encontrar una solución inicial). Graficar a la función. La solución encontrada parece ser el mínimo global?

3. Optimizar a la función `Cam` definida en el ejercicio anterior, utilizando un algoritmo evolutivo eligiendo el mismo sistema de vecindades, distribución para sortear a un vecino y la misma distribución para la solución inicial. Comparar el resultado con el algoritmo de esquema frío tras usar el mismo número de iteraciones.

4. Implementar el Algoritmo 58.

5. Implementar el Algoritmo 59.

6. Implementar el Algoritmo 60.

7. Suponga que n_i es el número de viudas con i hijos, donde $\sum_{i=0}^{6} n_i = N$ es la cantidad de viudas que tiene derecho a pensión por hijo. Asumiendo que el número de viudas sin hijos es grande y modelar por una distribución de Poisson no resulta adecuado debemos adoptar un modelo de mezclas de dos poblaciones. Sea A la población donde la variable aleatoria toma valor 0 (con probabilidad p) y B la población donde la variable aleatoria sigue una distribución Poisson de parámetro λ (con probabilidad $1 - p$). Obtener el estimador máximo verosimil de (p, λ) e implementar algoritmo EM en R.

8. Implementar el ejemplo del Algoritmo SEM, suponga que la muestra que trabajaremos tiene censura a partir de $c = 1.5$ y que la información completa es de 100 observaciones.

Capítulo 9

Otros modelos

En este capítulo mostramos algunos otros modelos utilizados en simulación estocástica.

9.1. Muestreo de bootstrap

La palabra bootstrap en inglés, tiene como significado literal a la tira que sale de la parte de atrás de las botas (cerca de donde se coloca al talón). La alegoría es la siguiente: al igual que uno utiliza esa tira de la bota para calzarse las botas, el muestreo de bootstrap utiliza un conjunto de datos dados para generar más datos.

La idea básica se describe a continuación. Dada una muestra de datos $\{x_1, ..., x_n\}$ podemos calcular a su distribución empírica $\hat{F}$ de la forma usual:

$$\hat{F}(t) := \frac{1}{n} \sum_{i=1}^{n} 1_{x_i \leq t} \qquad \forall t \in \mathbb{R}.$$

Debido al teorema de Glivenko-Cantelli, sabemos que cuando n crece a infinito, ocurre que $\hat{F}$ converge a F. Eso nos da un indicio de que tener una colección de datos grande es algo deseable para poder aproximar a F correctamente. Sin embargo, en muchas situaciones es imposible obtener más datos que los existentes; esto puede ser debido a la imposibilidad de repetir un experimento bajo condiciones similares, o un acontecimiento en condiciones específicas, o bien, porque es posible pero es demasiado costoso (como repetir una encuesta).

Una alternativa es la siguiente, utilizando directamente a la distribución empírica $\hat{F}$ construir más datos, a través de la simulación. Mostramos a continuación algunos ejemplos.

Supongamos que tenemos una muestra pequeña; que para fines didácticos generaremos con el generador de números normales estándar de `R`. Graficamos también su distribución empírica:

```
> muestra=rnorm(20,0,1)
```

Debido a que son exclusivamente 20 datos, podría ser aventurado realizar intervalos de confianza y pruebas

estadísticas concernientes a la distribución. Una alternativa es generar más datos, utilizando la distribución empírica de los datos que ya tenemos:

```
> muestraBootstrap=sample(muestra, size=1000, replace=TRUE)
```

Con el comando anterior, generamos 1000 datos provenientes de distribución empŕica de la muestra original (el valor *TRUE* indica que hicimos muestreo con reemplazo al generar datos de la muestra dada). Ahora podemos calcular los cuantiles para la muestra generada:

```
> quantile(muestraBootstrap,probs=c(0.025,0.975))
  2.5%     97.5%
-1.916235  1.722044
```

Análogamente, podemos realizar intervalos de confianza para la media, al obtener 1000 muestras (cada una de tamano 20), calcular la media para cada muestra, y obtener un intervalo de confianza para las 1000 medias obtenidas:

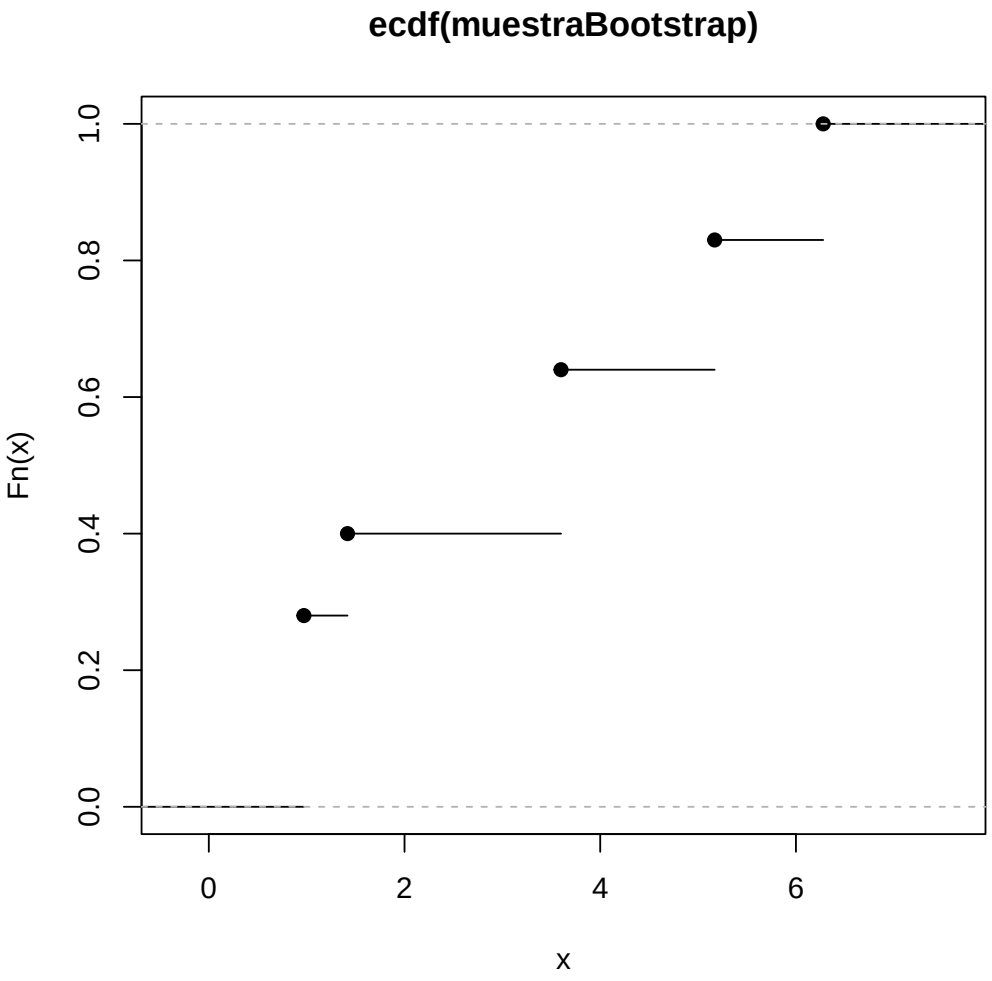
```
> quantile(replicate(1000,mean(sample(muestra,replace=TRUE))) ,probs=c(0.025,0.975))
   2.5%      97.5%
-0.2613799  0.4960472
```

Debemos advertir, que a pesar de ser una técnica que permite reducir la variancia (al aumentar el tamaño de la muestra), tiene muchas limitaciones. La primera es que, debido a que los datos son generados a partir de una muestra dada, los datos generados por bootstrapping son sesgados, pues tienen como media a la media de la muestra original. En nuestro ejemplo:

```
> mean(muestra)
[1] 0.1193076
```

Otra gran limitación es la siguiente. En el caso en que generemos datos con una distribución F continua, al aumentar el tamaño de muestra generado por bootstrapping no tenemos una convergencia a la distribución F, sino a la distribución empírica inicial $\hat{F}$ (porque los datos son generados a partir de ella). En términos prácticos, esto se traduce en que al aumentar el tamano de muestra, la distribución empírica no se suaviza. Veamos la distribución empírica de otra muestra generada por boostrapping, utilizando el comando `ecdf`:

```
muestra=rnorm(5,3,2)
muestraBootstrap=sample(muestra, size=100, replace=TRUE)
d=ecdf(muestraBootstrap)
plot(d)
```



9.1.1. Bootstrapping en modelos

El muestreo por bootstrap puede utilizarse en distintos modelos estadísticos. En esta subsección mostramos como puede ser utilizado en uno de los modelos más simples: la regresión lineal simple. Nuestro modelo es el siguiente. Tenemos datos $\{x_i\}$ de la variable independiente, y datos $\{y_{i,j}\}$ de la variable dependiente (el índice j representa la oportunidad de haber realizado múltiples mediciones de tal variable). Suponemos que la dependencia de las variables tiene la forma

$$y_{i,j} = \alpha + \beta x_i + \varepsilon_{i,j}$$

donde las variables $\{\varepsilon_{i,j}\}$ son variables aleatorias normales equidistribuidas e independientes. El ajuste para estimar a α y β se realiza por medio de mínimos cuadrados, obteniendo los estimadores $\hat{\alpha}$ y $\hat{\beta}$. Con ello se obtienen los residuales (o errores estimados) dados por

$$\hat{\varepsilon} = y_{ij} - \hat{\alpha} - \hat{\beta} x_i.$$

Veamos un ejemplo. Primero, suponiendo que tenemos la relación de dependencia

$$Y = 2 + 4X + \varepsilon,$$

definimos a la muestra de X, como el conjunto $\{-3, -2, -1, 0, 1, 2, 3\}$. En `R` lo escribimos así:

```
> x=seq(-3,3,le=7)
```

Ahora, por practicidad, generemos a una muestra aleatoria correspondiente a nuestra relación de dependencia:

```
> y=2+ 4* x + rnorm(7)
```

Podemos utilizar el comando de regresión lineal, definido por `lm`, para obtener a $\hat{\alpha}$ y $\hat{\beta}$:

```
> lm(y~x)
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      2.307        4.013
```

Estamos realizando regresión lineal utilizando exclusivamente 7 observaciones. Deseamos realizar intervalos de confianza para α y β y una de nuestras opciones es realizar muestreo de bootstrap para ello. Primero, obtenemos los residuales asociados a los datos:

```
> fit=lm(y~x)
> residuales=fit$residuals
```

Después, hacemos n muestras de la regresión lineal utilizando los mismos datos observados $\{y_{i,j}\}$ pero generando en cada muestra a los residuales a través de la distribución empírica de los residuales originales. En cada paso, guardamos a $\hat{\alpha}$ y $\hat{\beta}$.

```
> n=2000
> B=array(0,dim=c(n,2))
> for (i in 1:n){
ystar=y + sample (residuales, replace=TRUE)
Bfit=lm(ystar~x)
B[i,]=Bfit$coefficients
}
```

Tenemos una matriz B de n renglones en donde en cada renglón está la entrada $\hat{\alpha}$ y $\hat{\beta}$ correspondientes a una muestra obtenida por bootstrap y su regresión lineal. Ahora podemos obtener los intervalos de confianza para α y β:

```
> quantile(B[,1], probs=c(0.025,0.975))
    2.5%    97.5%
2.026132 2.582437
quantile(B[,2], probs=c(0.025,0.975))
    2.5%    97.5%
3.879805 4.152703
```

Obtuvimos intervalos pequeños para los coeficientes, gracias al bootstrapping. Sin embargo, el sesgo de la muestra inicial puede manifestarse. En nuestros resultados, por ejemplo, el primer coeficiente (que sabemos que es 2) no pertenece al intervalo de confianza estimado.

Apéndice A

Procesos Estocásticos

A.1. Introducción

En este apéndice se presentan los resultados fundamentales de procesos estocásticos. Se describen sus características principales, se realiza una clasificación básica y se dan algunos ejemplos.

Consideremos un sistema cuya evolución en el tiempo entre diferentes estados (determinados previamente) está modelada por una ley de movimiento. Si X_t representa el estado del sistema en el tiempo t y suponemos que la evolución de éste no es determinista, sino regida por algún componente aleatorio, resulta natural considerar a X_t como una variable aleatoria para cada valor t. Hasta aquí, tenemos una colección de variables aleatorias indexadas por el tiempo, a esta colección se le conoce como un proceso estocástico. Resulta de gran relevancia estudiar las relaciones entre las variables aleatorias de dicha colección que permiten clasificar a los procesos estocásticos.

Comenzaremos dando una definición formal de un proceso estocástico.

Definición A.1. *Un proceso estocástico $X = \{X_t\}_{t\in T}$ es una colección de variables aleatorias (definidas en el mismo espacio de probabilidad) que toman valores en un conjunto E con parámetro T, es decir, para cada $t \in T$, X_t es una variable aleatoria.*

En esta definición a T se le conoce como el espacio parametral y es interpretado como el tiempo, de esta forma a X_t se le entenderá como el estado del proceso al tiempo t. Además se considera que las variables aleatorias X_t para toda $t \in T$, están definidas sobre el mismo espacio de probabilidad $(\Omega, \mathscr{F}, P)$. Un proceso estocástico puede ser interpretado como la evolución en el tiempo de algún fenomeno cuya dinámica se rige por el azar.

Siguiendo la definición de proceso estocástico, se le puede ver como una función de dos variables

$$X : T \times \Omega \to E,$$

tal que a cada pareja $(t, \omega) \in (T, \Omega)$ se le asocia $X(t, \omega) = X_t(\omega)$ y de esta manera para cada $t \in T$ la

función de

$$\omega \to X_t(\omega)$$

es una variable aleatoria y para cada $\omega \in \Omega$, la función

$$t \to X_t(\omega)$$

es una trayectoria o realización del proceso.

A.1.1. Clasificación general.

Según las caracteristicas del espacio parametral se puede hacer la primera distinción entre procesos estocásticos.

Definición A.2. ***Proceso estocástico a tiempo discreto*** *Si T es un conjunto a lo más numerable, se dice que X es un proceso estocástico a tiempo discreto, para nosotros será $T = M$, donde $M \subset \mathbb{N}$ y en general se denota por $X = \{X_n\}_{n\geq 0}$.*

Definición A.3. ***Proceso estocástico a tiempo continuo*** *Cuando T sea un conjunto infinito no numerable, se dice que X es un proceso estocástico a tiempo continuo, para este caso $T = [0, \tau]$ ó $[0, \infty)$. y en será denotado por $X = \{X_t\}_{t\geq 0}$.*

En cuanto al conjunto donde las variables aleatorias toman valores, se le conoce como espacio de estados del proceso y de manera análoga este también puede ser discreto o continuo.

A.1.2. Algunos características posibles de procesos estocásticos.

En esta sección hablaremos sobre características que pueden tener los procesos estocásticos.

Procesos de variables independientes.

Definición A.4. ***Procesos de variables independientes.*** *Sea $X = \{X_n\}_{n\geq 0}$ se dice que es un proceso estocástico de variables aleatorias independientes si X_m y X_n son independientes para toda $n, m \geq 0$ con $m \neq n$.*

Procesos con incrementos independientes.

Definición A.5. ***Procesos con incrementos independientes.*** *Un proceso estocástico $X = \{X_t\}_{t\in T}$, se dice que tiene incrementos independientes si para cualesquiera $t_0 < t_1 < \ldots < t_n$, $t_i \in T$, las variables aleatorias*

$$X_{t_1} - X_{t_0}, \ldots, X_{t_n} - X_{t_{n-1}}$$

son independientes.

Procesos con incrementos estacionarios.

Definición A.6. ***Procesos con incrementos estacionarios.*** *Un proceso estocástico $X = \{X_t\}_{t\in T}$, se dice que es estacionario si para cualesquiera $t_0 < t_1 < \ldots < t_n$, , $t_i \in T$, la distribución del vector aleatorio $(X_{t_0}, X_{t_1} \ldots, X_{t_n})$ es la misma que la del vector aleatorio $(X_{t_0+h}, X_{t_1+h} \ldots, X_{t_n+h})$ para toda $h > 0$ y entonces tiene incrementos estacionarios si para cada $s < t$ y $h > 0$ se tiene que $X_{t+h} - X_{s+h}$ y $X_t - X_s$ tienen la misma distribución.*

Procesos Gaussianos.

Definición A.7. ***Procesos Gaussianos.*** *Se dice que un proceso estocástico $X = \{X_t\}_{t\in T}$ es un procesos de Gaussiano si para cualquier colección finita de tiempos $t_1, \ldots, t_n$ el vector $(X_{t_1}, \ldots, X_{t_n})$ tiene distribución Gaussiana multivariada.*

Procesos de Markov.

Definición A.8. ***Procesos de Markov.*** *Se dice que un proceso estocástico $X = \{X_t\}_{t\in T}$ es un proceso de Markov si satisface la propiedad de Markov, es decir, si*

$$P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}, \ldots, X_{t_0} = x_0) = P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1})$$

con $t_0 < t_1 < \ldots < t_n$, $t_i \in T$ y $x_i \in E$ para $i = 0, 1, \ldots n$. Este tipo de procesos son aquellos cuyo evolución futura depende solamente del estado proesente del proceso.

A.1.3. Características principales de los procesos estocásticos

Esperanza de un proceso estocástico

Para un proceso estocástico $X = \{X_t\}_{t\in T}$ se define la esperanza de X_t como el valor esperado de la variable aleatoria observando el proceso en algún tiempo t, es decir

$$E(X_t) = \int_E x dF_{X_t}(x),$$

donde F_{X_t}es la función de distribución de X_t .

Autocorrelación de un proceso estocástico

La función de autocorrelación del proceso $X = \{X_t\}_{t\in T}$ se define como el valor esperado del producto de las variables aleatorias X_{t_1} yX_{t_2} en los momentos t_1 y t_2 respectivamente, es decir,

$$E(X_{t_1} X_{t_2}) = \int_E \int_E x_1 x_2 dF_{X_{t_1}, X_{t_2}}(x_1, x_2)$$

donde $F_{X_{t_1}, X_{t_2}}$es la función de densidad conjunta de las variables aleatorias X_{t_1} yX_{t_2}.

Autocovarianza de un proceso estocástico

La función de autocovarianza del proceso $X = \{X_t\}_{t\in T}$ se define por:

$$R_X(t_1, t_2) = E[(X_{t_1} - E(X_{t_1}))(X_{t_2} - E(X_{t_2})].$$

Correlación cruzada de procesos estocásticos

Sean $X = \{X_t\}_{t\in T}$ e $Y = \{Y_t\}_{t\in T}$ dos procesos estocásticos se define la correlación cruzada entre estos procesos por:

$$R_{XY}(t_1, t_2) = E(X_{t_1} Y_{t_2})$$

o

$$R_{YX}(t_1, t_2) = E(Y_{t_1} X_{t_2})$$

para cualesquiera tiempos t_1 y t_2.

Covarianza cruzada de procesos estocásticos

Sean $X = \{X_t\}_{t\in T}$ e $Y = \{Y_t\}_{t\in T}$ dos procesos estocásticos se define la covarianza cruzada entre estos procesos por:

$$E[(X_{t_1} - E(X_{t_1}))(Y_{t_2} - E(Y_{t_2}))]$$

para cualesquiera tiempos t_1 y t_2.

A.1.4. Ejemplos

Presentaremos algunos ejemplos de procesos estocásticos.

Ejemplo A.1.1. *(**Caminatas aleatorias simples y el problema de la ruina.**) Consideremos la siguiente situación: teniendo un capital de k pesos al tiempo cero, a cada instante de tiempo se apuesta un peso en un lanzamiento de moneda, ganando si cae águila. Si X_n denota el capital acumulado después del n-ésimo volado entonces $X = \{X_n\}_{n\geq 0}$ es un proceso estocástico que modela la evolución del capital en el tiempo.*

Consideremos un modelo matemático preciso para describir al proceso X definido arriba. Definimos $U_1, U_2, \ldots$ como una colección de variables aleatorias Uniformes en $(0, 1)$ e independientes. Notemos que la variable aleatoria $\mathbb{I}_{\{U_i \leq 1/2\}}$ toma los valores 0 y 1, cada uno con probabilidad $\frac{1}{2}$, que pueden asociarse a las salidas posibles de un lanzamiento de moneda. Notemos que si consideramos a la variable aleatoria

$Y_i = 2\mathbb{I}_{\{U_i \leq 1/2\}} - 1$, ocurre que tal variable toma los valores 1 y -1 de forma equiprobable, que se pueden asociar a la paga recibida tras lanzar una moneda. De esta forma tenemos que

$$X_n = \sum_{i=0}^{n} Y_i.$$

Ejemplo A.1.2. *Considere una partícula que se mueve en un conjunto de $m+1$ nodos, etiquetados por $0, 1, \ldots, m$, distribuidos en un círculo. Sea X_n la posición de la partícula después de n pasos y supongamos que se mueve a los nodos vecinos con igual probabilidad.*

Ejemplo A.1.3. ***Tiempos de espera*** *Consideremos la fila en un banco y supongamos que los clientes van llegando a tiempos aleatorios y cada uno requiere un servicio que es también una variable aleatoria. Supongamos que un cliente llega a un tiempo t, nos interesa saber ¿cuánto tiempo tarda en salir del banco?*

Ejemplo A.1.4. *(**Valor de un activo financiero en el tiempo**). Supongamos que tenemos un proceso estocástico a tiempo continuo $X = \{X_t\}_{t \geq 0}$ donde X_t representa el precio de una acción al tiempo t.*

Estudiando los elementos de este proceso podemos conocer características del comportamiento futuro del precio del activo.

Ejemplo A.1.5. ***El modelo clásico de Cramér-Lundberg.***

Este modelo tiene sus orígenes en la tesis doctoral de Filip Lundberg defendida en el año de 1903. En este trabajo Lundberg analiza el reaseguro de riesgos colectivos y presenta el proceso de Poisson compuesto. En 1930 Harald Cramér retoma las ideas originales de Lundberg y las pone en el contexto de los procesos estocástico. El modelo ha sido muy estudiado y se han propuesto varias formas de generalizarlo.

El modelo que estudiaremos es el proceso a tiempo continuo $\{C_t\}_{t \geq 0}$ dado por

$$C_t = u + ct - \sum_{j=1}^{N(t)} Y_j$$

donde:

- u es el capital inicial de la compañía aseguradora,
- ct es la entrada por primas hasta el tiempo t con c una constante positiva,
- Y_j es el monto de la $j-$ésima reclamación, y
- N_t es un proceso de conteo, es decir $N(t)$ representa el número de reclamaciones que ha habido entre el tiempo 0 y el tiempo t..

C_t representa el balance más sencillo de ingresos menos egresos de una compañía aseguradora. Al proceso C_t se le llama proceso de riesgo o proceso de superávit.

La variable aleatoria C_t se puede interpretar como el capital de la compañía aseguradora al tiempo t y por razones naturales y legales es importante que C_t permanezca por arriba de cierto nivel mínimo. Cuando $C_t < 0$ se dice que hay ruina.

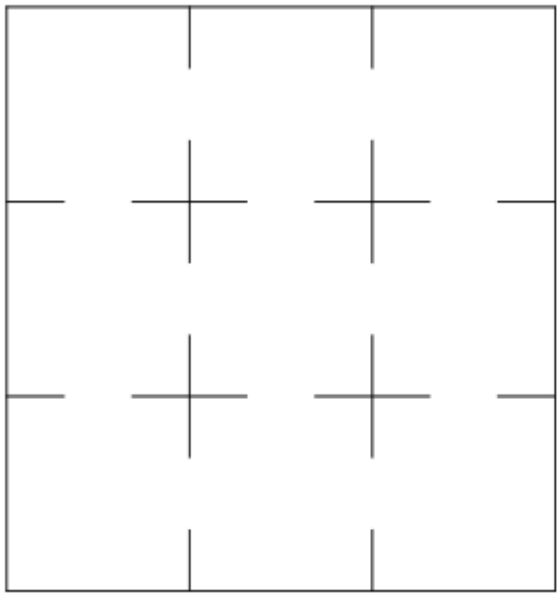


Figura A.1: Laberinto

La ruina casi nunca sucede en la práctica, es solamente un término técnico que produce alguna toma de decisión.Por ejemplo, si el capital de una compa?ía aseguradora asignado a una cartera decrece en forma significativa, ésta automáticamente puede tomar ciertas medidas para subsanar esta situación y no se trata de un evento insalvable.

A.2. Cadenas de Markov a tiempo discreto

Como motivación al estudio de las cadenas de Markov a tiempo discreto consideremos el siguiente ejemplo:

Ejemplo A.2.1. *Laberinto*

Si colocamos una rata en la esquina inferior izquierda del laberinto de la Figura A.2 y comida en la esquina superior derecha y supongamos que estando en cualquiera de los cuartos del laberinto, la rata puede ir a cualquiera de los cuartos vecinos con la misma probabilidad, entonces para modelar la trayectoria que la rata sigue hasta la comida, podemos seguir el siguiente modelo matemático. Enumerando los cuadros de izquierda a derecha y de abajo a arriba, definimos p_{ij} como la probabilidad con la que la rata pasa del cuarto i al j para $i, j \in \{1, 2, \ldots, n\}$.

Podemos organizar a estas probabilidades en la siguiente matriz:

$$P = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 \end{pmatrix}$$

Tenemos, por ejemplo, que la probabilidad de que la rata siga la trayectoria 1,2,3,6,9 hasta la comida es: $p_{12}p_{23}p_{36}p_{69}$. Hay dos características importantes que debemos notar en la matriz:

1. $p_{ij} \geq 0$ para todo i y j y
2. $\sum_j p_{ij} = 1$ para todo i.

A matrices con estas propiedades se le llama matrices estocásticas y en particular podemos ver que en cada renglón de dichas matrices está definida una distribución de probabilidad discreta.

A partir de este ejemplo podemos proponer varias preguntas para las cuales la teoría subsecuente nos ayudará a encontrar respuestas.

1. ¿Cuánto tarda la rata en promedio en encontrar la comida si comienza en el cuarto i?
2. Si quitamos la comida y nada más seguimos la trayectoria de la rata, ¿cuál es la probabilidad de que se encuentre en el cuarto j en el paso n si comienza en i?
3. Si de nuevo seguimos solamente la trayectoria sin comida, ¿estamos seguros de regresar al punto inicial?
4. Si agregamos la comida, ¿cuántas veces regresará la rata al cuarto inicial antes de encontrar la comida?

A.2.1. Conceptos básicos

Consideremos una colección de variables aleatorias $X_0, X_1, \ldots$, interpretando a X_n como el estado de un sistema al tiempo n y supongamos que el conjunto de posibles valores de X_n es a lo más numerable, enumerándolos, tenemos que $E = \{1, 2, \ldots, N\}$ para el caso finito y $E = \{1, 2, \ldots\}$ para el caso numerable, E es llamado espacio de estados. Si este proceso satisface la **propiedad de Markov**

$$P(X_{n+1} = x_{n+1} | X_n = x_n, \ldots, X_0 = x_0) = P(X_{n+1} = x_{n+1} | X_n = x_n)$$

con $x_0, \ldots, x_{n+1} \in E$, decimos que $\{X_n\}_{n\geq 0}$ es **una cadena de Markov.**

A la distribución de X_0 se le llama distribución inicial y la denotaremos por $\pi = (\pi_1, \ldots, \pi_N)$, es decir, si $\pi_i = P(X_0 = i)$ para toda $i \in E$.

Algunas observaciones importantes:

1. Si $X_n = x_n, \ldots, X_0 = x_0$ entonces decimos que se ha presentado la trayectoria $x_0, \ldots, x_n$.
2. A la familia $P(X_{n+1} = j | X_n = i)$ se le conoce como familia de probabilidades de transición de la cadena de Markov y describe la evolución de la cadena en el tiempo.
3. En caso de que exista independencia al tiempo se le llama homogénea y

$$P(X_{n+1} = j | X_n = i) = p_{ij}$$

son las probabilidades de transición y para $m \geq 1$

$$p_{ij}^{(m)} = P(X_{m+n} = j | X_n = i),$$

que representa la probabilidad de pasar del estado i al estado j en m unidades de tiempo.

Definición A.9. *Matriz de transición:*

Es la matriz cuadrada de orden N formada por las probabilidades de transición

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix}$$

con entradas no negativas y la suma de cada renglón es uno (matriz estocástica).

A.2.2. Ejemplos

Veamos algunos ejemplos de cadenas de Markov

Ejemplo A.2.2. *Cadena de rachas.*

Considere la observación de un experimento con propabilidad p de ocurrencia, definiendo como transición un ensayo y a $X_n =$ como los estados de estar en racha de éxitos o no, entonces $X = \{X_n\}_{n \geq 0}$ define una cadena de Markov.

Ejemplo A.2.3. *Caminata aleatoria simple.*

Un individuo está parado e inica una caminata en el cual sólo se mueve hacia adelante con probabilidad p o hacia atrás con probabilidad $1-p$. Definiendo a X_n como la posición del individuo después de $n-$pasos, se tiene entonces que $E = \mathbb{Z}$ y las probabilidades de transición para cualquier $n \geq 0$ e $i, j \in E$ están dadas por:

$$p_{ij} = \begin{cases} p & j = i+1 \\ 1-p & j = i-1 \\ 0 & e.o.c. \end{cases}$$

Ejemplo A.2.4. *Problema del jugador.*

Suponga que un jugador A apuesta \$1 sucesivamente contra otro jugador B. A inicia con k unidades y B con $N-k$. En cada apuesta, A gana con probabilidad p y pierde con probabilidad $1-p$. Si definimos X_n como la fortuna de A al tiempo n, entonces $X = \{X_n\}_{n\geq 0}$ define una cadena de Markov.

Ejemplo A.2.5. *Cadena de Ehrenfest.*

Se tienen dos urnas , entre ambas hay un total de N bolas. En cada paso se elige una de las bolas al azar y se cambia de urna. Si $X_n =$ número de bolas en la urna A después de n ensayos, entonces $X = \{X_n\}_{n\geq 0}$ define una cadena de Markov.

A.2.3. Resultados Importantes

A continuación presentaremos algunos resultados importantes.

Proposición A.1. *Ecuación de Chapman-Kolmogorov.*

Sea $X = \{X_n\}_{n\geq 0}$ una cadena de Markov con matriz de transición
$P = \{p_{ij}\}_{ij\in E}$ entonces

$$p_{ij}^{(n+m)} = \sum_{k\in E} p_{ik}^{(n)} p_{kj}^{(m)}.$$

Este resultado nos dice que $p_{ij}^{(n)}$ es la entrada ij de la $n-$ésima potencia de P. En general, no es posible calcular explícitamente las potecias de la matriz de transición. Sin embargo, nos podemos auxiliar de paquetes computacionales para esta tarea.

Proposición A.2. *Propiedad de corrimiento.*

Sean $n, k \in \mathbb{N}$ fijos y $x_0, x_1, \ldots, x_n, \ldots, x_{n+k} \in E$ entonces

$$P(X_{n+k} = x_{n+k}, \ldots, X_{n+1} = x_{n+1} | X_n = x_n, \ldots, X_0 = x_0) = P(X_k = x_{n+k}, \ldots, X_1 = x_{n+1} | X_0 = x_n).$$

A.2.4. Clases de comunicación

Sean x e y dos estados de E, se dice que el estado y es accesible desde el estado x si existe $n \in \mathbb{N}$, tal que $p_{xy}^{(n)} > 0$ $(x \to y)$. Si $x \to y$ e $y \to x$ entonces, x e y se **comunican**, $(x \leftrightarrow y)$, en particular, se tiene el siguiente resultado sobre la comunicación.

Proposición A.3. *La relación $x \leftrightarrow y$ (comunicación) es una relación de equivalencia, por lo tanto, induce a una partición en el espacio de estados.*

A las clases de equivalencia inducidas por esta relación les llamaremos clases de comunicación. En particular diremos que una cadena de Markov es irreducible si tiene una sola clase de comunicación.

Sea $x \in E$, a su clase de cominicación la dentaremos por C_x, es decir

$$C_x = \{y \in E : x \leftrightarrow y\}.$$

En cuanto a la partición inducida por la comunicación de estados decimos que, tomando $C \subset E$, es un conjunto cerrado si para toda $y \in E - C$, $x \in C$ no puede acceder a y, en caso contrario decimos que es una clase abierta.

A.2.5. Propiedad Fuerte de Markov

Una extensión de la propiedad de Markov es la presentada a tiempos aleatorios conocida como propiedad fuerte de Markov. Comenzaremos esta sección dando la definición de tiempos aleatorios.

Definición A.10. *Un **tiempo aleatorio** es una variable aleatoria*

$$T : \Omega \to \mathbb{N} \cap \{\infty\}.$$

Definición A.11. *Un tiempo aleatorio T es un **tiempo de paro** si para $n \in \mathbb{N}$ existe $A_n \subset E^{n+1}$ tal que*

$$\{T = n\} = \{(X_0, \ldots, X_n) \in A_n\}.$$

A la variable aleatoria tiempo de paro la podemos interpretar como el tiempo que se obtiene de observar una trayectoria hasta que se cumpla cierta condición. Veamos algunos ejemplos de tiempo de paro.

Ejemplo A.2.6. *Sea T_1 el tiempo en el que una cadena de Markov regresa a su estado inicial, es decir*

$$T = \begin{cases} \infty & X_n \neq X_0 \text{ para toda } n \\ min\{n \geq 1 | X_n = X_0\} & e.o.c \end{cases}$$

Ejemplo A.2.7. *Sea T_n el tiempo en el ocurre la enésima visita al estado inicial es un tiempo de paro.*

Teorema A.1. ***Propiedad Fuerte de Markov.*** *Sea $A \in E^{n+1}$, T un tiempo de paro tales que $P(A, X_n = j, T = n) > 0$. Entonces, condicionado a $A \cap \{T = n, X_n = j\}$, el proceso $\{X_{n+m}\}_{m \geq 0}$ es una cadena de Markov que comienza en j y tiene matriz de P.*

A.2.6. Tiempos de llegada y tiempos de absorción

En el estudio de las cadenas de Markov resulta de particular interés el estudio de los tiempos aleatorios de la ocurrencia de eventos relacionados con la llegada a un estado o un conjunto de estados del espacio de la cadena. En está sección estudiaremos este tipo de variables aleatorias.

Definición A.12. *Sea T_A primera vez que la cadena accede a A un subconjunto del espacio de estados, es decir*

$$T_A = \begin{cases} \infty & X_n \in E - A \text{ para toda } n \\ min\{n \geq 1 | X_n \in A\} & e.o.c \end{cases}$$

Nota A.1. *En general, no resulta tarea fácil encontrar la distribución de este tipo de variables aleatorias.*

En el caso que el conjunto A consta de un solo estado (j) y la cadena empieza en el estado i denotaremos por T_{ij} la primera vez que la cadena visita el estado j iniciando en i si $i \neq j$. Si $i = j$ lo denotamos por T_i.

Veamos un ejemplo donde podemos ilustrar esta definición.

Ejemplo A.2.8. ***Racha de éxitos.*** *Sea $E_1, E_2, \ldots,$ una sucesión de ensayos Bernoulli con probabilidad de éxito p y probabilidad de fracaso $q = 1 - p$. Si X_n es el número de éxitos consecutivos previos al tiempo n (incluido el tiempo n). Se dice que una cadena de r éxitos ocurre al tiempo n si en el ensayo $n - r$ ocurre un fracaso y los resultados de los ensayos $n - r + 1$ al n son éxitos.*

Definición A.13. *Para cada $n \geq 1$, denotamos por $f_{ij}^{(n)}$ a la probabilidad de que una cadena que inicia en el estado i, llegue al estado j por primera vez en exactamente n pasos, es decir,*

$$f_{ij}^{(n)} = P(X_n = j, X_{n-1} \neq j \ldots, X_1 \neq j, X_0 = i).$$

Además definimos

$$f_{ij}^{(0)} = \begin{cases} 1 & si \ i = j \\ 0 & e.o.c \end{cases}$$

Para la cadena de Markov de racha de éxitos tenemos (por ejemplo)

1. $f_{01}^{(n)} = q^{n-1}p.$
2. $f_{00}^{(n)} = p^{n-1}q.$

Un resultado importante es el que permite descomponer la probabilidad de visitar el estado j iniciando en i en términos de todas las posibilidades de tiempos que pueden ser la primera visita.

Proposición A.4. *Para cada $n \geq 1$,*

$$p_{ij}^{(n)} = \sum_{k=1}^{n} f_{ij}^{(k)} p_{ij}^{(n-k)}.$$

A.2.7. Clasificación de estados

De acuerdo a las propiedades que tiene cada estado en una cadena de Markov en cuanto a la probabilidad de regresar a él, los estados pueden ser clasificados de la siguiente manera.

Definición A.14. *Se dice que un estado i es* ***recurrente*** *si la probabilidad de (eventualmente) regresar a i, es uno, es decir, si*

$$P(X_n = i | X_0 = i) = 1,$$

para alguna $n \geq 1$. Un estado que no es recurrente se llama ***transitorio****.*

Tratar de clasificar los estado en recurrentes y transitorios partiendo de la definición no resulta, en general, tarea fácil. Una manera distinta de enunciar estos conceptos es la siguiente forma.

Definición A.15. *Definimos a*

$$f_i = \sum_{n=1}^{\infty} f_{ii}^{(n)},$$

que es la probabilidad de un eventual regreso al estado i.

Nota A.2. *Decimos que:*

- *El estado i es **recurrente** si $f_i = 1$, en particular si $f_{xx}^{(1)} = 1$ el estado se le llama **absorbente**.*
- *El estado x es **transitorio** si $f_x < 1$.*

Podemos ya formular el siguiente importante teorema.

Teorema A.2. ***(Teorema de descomposición de las Cadenas de Markov)*** *El espacio de estados E de una cadena de Markov, tiene la siguiente partición única:*

$$E = T \cap C_1 \cap C_2 \cap \ldots$$

donde T es un conjunto de estados transitorios, y C_i son clases cerradas e irreducibles de estados recurrente.

Nota A.3. *El teorema de descomposición nos muestra las posibilidades que pueden darse en una cadena de Markov. Esto es, si $X_0 \in C_k$, entonces la cadena nunca abandonará la clase C_k y entonces, podemos considerar el espacio de estados $E = C_k$. Por otra parte, si $X_0 \in T$ entonces, o la cadena permanece por siempre en T o se mueve a una clase C_k y permanece ahí por siempre. Así, o la cadena siempre toma valores en el conjunto de estados transitorios o acaba en un conjunto cerrado persistente de estados donde permanecerá por siempre. Veamos ahora que en el caso en el que E sea finito la primera situación no puede darse.*

Definición A.16. *Definiendo al número de visitas al estado i como la variable aleatoria*

$$V_i = \sum_{n=0}^{\infty} \mathbb{I}_{\{X_n = i\}},$$

Nota A.4. *La variable aleatoria anterior toma valor infinito con probabilidad cero o uno.*

Decimos que si V_i es infinita con probabilidad 1, partiendo de i, entonces i es estado recurrente, en caso contrario es un estado transitorio.

Motivados en este criterio tenemos el siguiente resultado:

Proposición A.5.

1. Un estado i es recurrente si y sólo si $\sum_n p_{ii}^{(n)} = \infty$.
2. Un estado i es transitorio si y sólo si $\sum_n p_{ii}^{(n)} < \infty$.

Este resultado nos proporciona una equivalencia, en términos de las matrices de transición, para que un estado sea recurrente.

A continuación presentamos una serie de resultados respecto a la clasificación de estados y la comunicación.

Proposición A.6. *Sean $i, j \in E$ tales que se comunican, si i es transitorio entonces j también es transitorio.*

Este resultado tiene como consecuencia que la transitoriedad o recurrencia son propiedades de clase.

Un criterio fácil para saber si una clase es transitoria es el siguiente.

Proposición A.7. *Sea $C \subset E$ una clase abierta. Entonces C es transitoria.*

Usando estos resultados tenemos la siguiente conclusión.

Proposición A.8. *Si el espacio de estados es finito, una clase es recurrente si y sólo si es cerrada.*

Corolario A.1. *En una cadena de Markov irreducible con espacio de estados finito todos sus estados son recurrente.*

Corolario A.2. *Toda cadena de Markov con espacio de estados finito tiene por lo menos un estado recurrente.*

Nota A.5. *En general, el análisis de recurrencia y transitoriedad, en cadenas con espacio de estados infinito, es más complicado.*

Siguiendo la clasificación de estados, resulta importante conocer el número esperado de pasos para regresar a i, es decir,

$$\mu_i = \sum_{n=1}^{\infty} n f_{ii}^{(n)},$$

a la cual llamaremos **tiempo medio de recurrencia**.

Nota A.6. *Puede darse el caso de que siendo i un estado recurrente el tiempo medio de recurrencia, μ_i, sea infinito; siendo éste el caso en el que aunque el regreso a i es cierto se necesita un tiempo infinito. Así, realizamos la siguiente distinción entre los estados recurrentes obteniendo una subclasificación de estos.*

Definición A.17. *Decimos que*

1. *El estado i es **recurrente positivo** si $\mu_i < \infty$.*
2. *El estado i es **recurrente nulo** si $\mu_i = \infty$.*

Se tiene directamente el siguiente resultado.

Proposición A.9. *En una cadena irreducible con espacio de estados finito, todos los estados son recurrentes positivos.*

Ejemplo A.2.9. *Veamos que la cadena de Markov de racha de éxitos es una cadena recurrente positiva.*

$$\mu_0 = \sum_{n=1}^{\infty} n f_{00}^{(n)} = \sum_{n=1}^{\infty} n(1-p)p^{n-1} = (1-p)\sum_{n=1}^{\infty} np^{n-1} = \frac{1}{1-p} < \infty.$$

Otro resultado de gran utilidad es el siguiente.

Proposición A.10. *No existen estados recurrentes nulos en cadenas de Markov finitas.*

El siguiente resultado se sigue de la teoría que los promedios temporales son iguales a los promedios espaciales en los sistemas dinámicos.

Teorema A.3. ***Teorema ergódico para cadenas de Markov*** *Sean i, j cualesquiera estados de una cadena de Markov irreducible se cumple que*

$$lím_{n\to\infty} \frac{V_{ij}}{n} = \frac{1}{\mu_j}$$

casi seguramente.

Ejemplo A.2.10. *Considere una cadena de Markov con espacio de estados $\{0, \ldots, 5\}$ y matriz de transición*

$$P = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{8} & 0 & \frac{7}{8} & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & \frac{3}{4} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{5} & 0 & \frac{1}{5} & \frac{1}{5} & \frac{2}{5} \end{pmatrix}$$

Determine que estados son transitorios y cuales recurrentes. Calcular y estimar tiempos medios de recurrencia y número esperado de visitas a cada estado.

A.2.8. Distribuciones invariantes

Definición A.18. *Sea $\nu \in R^{|E|}$, decimos que ν es una* ***distribución invariante o estacionaria*** *de la cadena de Markov con matriz de transición P, si satisface*

1. *Tiene todas sus entradas no negativas.*
2. $\sum_{i\in E} \nu_i = 1$.
3. $\sum_{j\in E} \nu_i p_{ij} = \nu_j$.

En términos matriciales, la distribución de probabilidad ν es invariante si

$$\nu P = \nu.$$

Nota A.7. *La interpretación de una distribución invariante es que una variable aleatoria con dicha distribución mantendrá su distribución en la evolución de la cadena de Markov en el tiempo.*

Desde que encontrar una distribución estacionaria en una cadena de Markov consiste en resolver un sistema de $|E|+1$ ecuaciones y $|E|$ incógnitas podemos no tener solución, una infinidad de soluciones o una única solución.

Los siguientes resultados son de gran utilidad en el cálculo de la distribución estacionaria de una cadena de Markov.

Proposición A.11. *Sea ν una distribución estacionaria para una cadena de Markov. Si i es un estado transitorio o recurrente nulo, entonces $\nu_i = 0$.*

Una propiedad importante de una distribución invariante es que si X_0 tiene distribución ν, entonces X_n tendrá la misma distribución para toda n.

Proposición A.12. *Toda cadena de Markov que es irreducible y recurrente positiva tiene una única distribución estacionaria dada por*

$$\nu_i = \frac{1}{\mu_i} > 0.$$

Teorema A.4. *Para una cadena irreducible las siguientes condiciones son equivalentes.*

1. *Todos los estados son recurrentes positivos.*
2. *Algún estado es recurrente positivo.*
3. *La cadena tiene una única distribución invariante*

Ejemplo A.2.11.

$$P = \begin{pmatrix} 1/2 & 1/2 \\ 1/4 & 3/4 \end{pmatrix}$$

$\nu = (1/3, 2/3)$.

Ejemplo A.2.12. ***Cadena de Ehrenfest****. Como esta cadena de Markov es irreducible y con espacio de estados finito se tiene que es recurrente positiva y por los resultados anteriores tiene una única distribución invariante ν.*

A.2.9. Cadenas regulares, absorbentes y convergentes.

En esta sección presentaremos resultados del comportamiento de una cadena de Markov recurrente cuando la trayectoria es lo suficientemente grande. Otro concepto importante en el estudio de las cadenas de Markov es el relacionado con la posible periodicidad con la que se visita a cada estado.

Definición A.19. *Sea P la matriz de transición, definimos al período de un estado i como el máximo común divisor del conjunto $\{n \in \mathbb{N} : p_{ii}^{(n)}\}$.*

Ahora combinando esta definición con el concepto de recurrencia positiva tenemos que un estado es **ergódico** si es recurrente positivo y tiene período uno (aperiódico).

Definición A.20. *Decimos que una cadena de Markov $\{X_n\}$ con espacio de estados finito es **regular** si existe una potencia de la matriz de transición con todas las entradas positivas.*

Teorema A.5. *Sea P la matriz de transición de una cadena de Markov regular con espacio de estados finito E, se cumple que:*

1. *Si $n \to \infty$, las potencias P^n se aproximan a una matriz W tal que todos sus renglones son iguales a un mismo vector de probabilidad w con componentes estrictamente positivos.*
2. *Se tiene que w es el vector de probabilidad invariante y cualquier otro vector v tal que $vP = v$ es un escalar multiplicado por w (como vector de probabilidad es único).*
3. $$lím_{n\to\infty} P(X_n = x) = w_x$$
   para todo $x \in E$ y
   $$lím_{n\to\infty} P(X_n = x | X_0 = y) = w_x$$
   para todo $x, y \in E$.

Ahora presentamos resultados fundamentales en el estudio de la convergencia de las cadenas de Markov.

Definición A.21. *Si los límites*
$$\pi_j = lim_{n\to\infty} p_{ij}^{(n)}$$
existen para cada estado j en el espacio de estados entonces π, es la distribución límite.

Nota A.8. *Las siguientes observaciones son consecuencia de la definición anterior.*

1. *La distribución límite es inducida por la matriz de transición P.*
2. *La distribución límite no depende de la distribución inicial.*
3. *El límite de las potencias de P es una matriz estocástica para la cual todos sus renglones son la distribución límite.*
4. *La distribución límite es una distribución estacionaria.*

Teorema A.6. *(Existencia de la distribución límite.) Si la cadena de Markov es irreducible, aperiódica y con distribución invariante π, entonces*
$$\pi_j = \lim_{n\to\infty} p_{ij}^{(n)}$$

Teorema A.7. *Si la cadena de Markov es irreducible, aperiódica y recurrente positiva existen las probabilidades límite y son la única distribución invariante.*

Por último estudiaremos una clase particular de cadenas de Markov, las cadenas de Markov absorbentes.

Definición A.22. *Una cadena absorbente es una cadena de Markov con espacio de estados finita, con al menos un estado absorbente y que desde cualquier estado se puede llegar a algún estado absorbente.*

Para una cadena de Markov absorbente es importante estudiar conceptos como el tiempo medio de absorción o ¿cual será el estado más probable de absorción?

Ejemplo A.2.13. *Calificaciones crediticias*

	Aaa	Aa	A	Baa	Ba	B	C	D
Aaa	0.95	0.02	0.015	0.009	0.006	0	0	0
Aa	0.06	0.91	0.01	0.007	0.005	0.004	0.004	0
A	0.01	0.04	0.88	0.03	0.019	0.011	0.01	0.0
Baa	0.002	0.003	0.005	0.88	0.05	0.03	0.02	0.01
Ba	0.001	0.005	0.005	0.009	0.85	0.05	0.05	0.04
B	0	0	0.003	0.007	0.02	0.88	0.05	0.04
C	0	0	0.002	0.003	0.005	0.09	0.79	0.11
D	0	0	0	0	0	0	0	1

Tabla A.1: Matriz de Transición de Calificación Crediticia Moody's 2000.

En los contratos de crédito, las posibles pérdidas económicas involucran la estimación de las probabilidades de incumplimiento por parte de los acreditados.

El acreditado es clasificado según la capacidad de cumplir con sus obligaciones en un contrato de crédito. En el campo de incumplimiento por parte de las empresas, en un contrato de crédito, las matrices de transición representan las probabilidades de pasar de una calificación a otra en un intervalo de tiempo.

Las clasificaciones son emitidas por las compa?ías calificadoras. El papel de estas compa?ías en los mercados globales de capital es tener una medición del riesgo de crédito de manera independiente, creíble y objetiva; una cobertura comprensible y consistencia global; una transparencia crediticia y un aumento de la eficiencia y la liquidez.

El mercado de calificadoras de crédito está dominado por dos compa?ías: **Standard and Poors (S&P)** y **Moody's Investor Services (Moody's)**. Una calificación de crédito representa una tasa global del cumplimiento de o e las obligaciones por parte del acreditado.

La tabla muestra una matriz de transición anual con las calificaciones de crédito otorgadas por Moody's.

A.3. Proceso de Poisson

En esta sección estudiaremos las principales características del proceso de conteo más usado en el contexto actuarial, el proceso de Poisson.

A.3.1. Definiciones

Podemos definir, en general, a un proceso de conteo $N = \{N_t\}_{t\geq 0}$ como un procesos estocástico con trayectorias constantes por pedazos con valores en los naturales y que va incrementando de uno en uno en tiempos aleatorios. Entonces, interpretamos a N_t como el número de eventos ocurridos hasta el tiempo t.

Si denotamos por $0 = S_0 < S_1 < \ldots$ a los tiempos en los que el proceso incrementa su valor, entonces

tenemos que

$$N_t = máx\{n \in \mathbb{N} : S_n \leq t\}.$$

Nota A.9. *Las siguientes equivalencias de eventos son importantes:*

- $\{N_t = n\} = \{S_n \leq t < S_{n+1}\}$.
- $\{N_t \geq n\} = \{S_n \leq t\}$.

Nota A.10. *A los tiempos $T_i = S_i - S_{i-1}$ les llamaremos tiempos de interarribo que son los tiempo aleatorios entre la ocurrencia de dos eventos seguidos*

Ahora daremos una definición formal de un proceso de conteo.

Definición A.23. *Un proceso estocástico $N = \{N_t\}_{t\geq 0}$ se dice que es un proceso de conteo si N_t representa el número total de eventos que han ocurrido hasta el tiempo t.*

Nota A.11. *De la definición se tiene que:*

1. $N_t \geq 0$.
2. N_t *es un entero no negativo.*
3. *Si* $s < t$ *entonces* $N_s \leq N_t$.
4. *Para* $s < t$, $N_t - N_s$ *es el número de eventos que ocurren en el intervalo de tiempo* (s, t).

Ahora daremos un par de definiciones del proceso de conteo que es nuestro foco de interés.

Definición A.24. ***(Definición 1 P.P.)*** *Un proceso de contéo $N = \{N_t\}_{t\geq 0}$ es de Poisson con parámetro $\lambda > 0$ (intensidad del proceso) si satisface las siguientes condiciones:*

1. $N_0 = 0$.
2. *Tiene incrementos independientes.*
3. *El número de eventos en cualquier intervalo de longitud t se distribuye Poisson de parámetro λt.*

Definición A.25. ***(Definición 2 P.P.)*** *Un proceso de contéo $N = \{N_t\}_{t\geq 0}$ es de Poisson con parámetro $\lambda > 0$ (intensidad del proceso) si satisface las siguientes condiciones:*

1. $N_0 = 0$.
2. *Tiene incrementos independientes.*
3. *Tiene incrementos estacionarios.*
4. $P(N_{t+h} - N_t \geq 1) = \lambda h + o(h)$.
5. $P(N_{t+h} - N_t \geq 2) = o(h)$ *para* $t \geq 0$ *cuando* $h \to 0$.

Nota A.12. *Las dos definiciones son equivalentes.*

Veamos algunos resultados importantes del proceso de Poisson.

Proposición A.13. *La sucesión de variables aleatorias $\{T_n\}$ (tiempos de interarribo entre eventos) de un proceso de Poisson de parámetro λ son independientes y con la misma distribución exponencial de parámetro λ.*

Consecuencia inmediata de esta proposición se tiene el siguiente corolario.

Corolario A.3. *Para sucesión de variables aleatorias $\{S_n\}$ (tiempos de ocurrencia de eventos) definidas anteriormente se tiene que:*

$$S_n \sim Gamma(n, \lambda).$$

Hay una definición alternativa de proceso de Poisson que facilita hacer cálculos.

Definición A.26. *Un proceso de conteo $N = \{N_t\}_{t\geq 0}$ es de Poisson si y sólo si existe $\lambda > 0$ tal que los tiempos de interarribo son variables aleatorias exponenciales independientes de parámetro λ.*

Consecuencias inmediatas de esta definición son que se pueden calcular explicitamente las distribuciones de tiempos de interarribo, los tiempos de ocurrencia y que se tiene conocida la distribución exacta del proceso de conteo.

Veamos un ejemplo.

Ejemplo A.3.1. *Suponga que las reclamaciones es una compa?ia de seguros ocurren de acuerdo a un proceso de Poisson con intensidad diaria $\lambda = 2$*

1. ¿Cuál es la probabilidad que en una semana se presenten 5 reclamaciones?

$$P(N_7 = 5) = \frac{e^{-14}14^5}{5!}.$$

2. ¿Cuál es el número de reclamaciones esperadas en un mes?

$$E[N_{30}] = 60.$$

3. ¿Cuál es el tiempo esperado hasta la octava reclamación?

$$E[S_8] = 4.$$

A.3.2. Propiedades principales

Sea $\{N_t\}_{t\geq 0}$ un proceso de Poisson con parámetro λ, supongamos que cada tiempo de ocurrencia de un evento está clasificado como un evento tipo I o tipo II. Además supongamos que es tipo I con probabilidad p y tipo II con probabilidad $1-p$. Si N_t^1 y N_t^2 denotan el número de eventos tipo I y II respectivamente que ocurren en $[0, t]$. El siguiente resultado modela el comportamiento en el tiempo de un escenario como el que acabamos de describir.

Proposición A.14. *$\{N_t^1\}_{t\geq 0}$ y $\{N_t^2\}_{t\geq}$ son procesos de Poisson independientes de parámetros $p\lambda$ y $(1-p)\lambda$ respectivamente.*

Nota A.13. *Es fácil generalizar este resultado cuando se tienen k tipos de evento, cada uno con probabilidad $p_i \in (0,1)$ de ocurrencia y $\sum_{i=1}^{k} p_i = 1$. Entonces se tiene que $\{N_t^1\}_{t\geq 0}, \{N_t^2\}_{t\geq}, \ldots, , \{N_t^k\}_{t\geq}$ procesos de Poisson independientes de parámetros $p_1\lambda, p_2\lambda, \ldots, p_k\lambda$ respectivamente.*

Ejemplo A.3.2. *Suponga que las reclamaciones se han categorizado según el monto de reclamación. Si se tienen 5 diferentes categorias con vector de probabilidades $p = (1/9, 1/18, 3/18, 1/18, 10/18)$ para cada categoría. Si además suponemos que las reclamaciones se presentan de acuerdo a un proceso de Poisson con intensidad diaria $\lambda = 3$.*

1. ¿Cuál es la probabilidad que en una semana se presenten 5 reclamaciones de la categoría 5?
2. ¿Cuál es el número de reclamaciones esperadas en un mes de cada categoría?
3. ¿Cuál es el tiempo esperado hasta la octava reclamación de la categoría 1?
4. Si en una semana se presentan 3 reclamaciones, ¿cuál es la probabilidad que todas sean del mismo tipo?

Ahora, supongamos que sabemos que exactamente un evento ha ocurrido hasta el tiempo t y estamos interesados en la distribución del tiempo al cual éste ha ocurrido. Es razonable pensar que el tiempo está uniformemente distribuido en $[0,t]$. Supongamos que $s \leq t$, entonces

$$P(T_1 \leq s | N_t = 1) = \frac{P(N_s = 1, N_{t-s} = 0)}{P(N_t = 1)} = \frac{s}{t}.$$

Generalizando esta idea tenemos el siguiente resultado.

Teorema A.8. *Dado que $N_t = n$, los n tiempos de ocurrencia tienen la misma distribución que las estadísticas de orden correspondientes a n variables aleatorias independientes e identicamente distribuidas uniformes en $[0,t]$, es decir,*

$$f_{S_1,\ldots,S_n|N_t}(s_1, \ldots, s_n | n) = \frac{n!}{t^n}.$$

A.3.3. El proceso Poisson compuesto

Definición A.27. *Sea $\{N_t\}_{t\geq 0}$ un proceso Poisson con tasa λ y sean $\{Y_i,\ i = 1, 2, \ldots\}$ una familia de variables aleatorias independientes entre si e idénticamente distribuidas. Suponga que el proceso Poisson $\{N_t | t \geq 0\}$ y la sucesión $\{Y_i,\ i = 1, 2, \ldots\}$ son independientes. Definimos al proceso aleatorio $\{X_t | t \geq 0\}$ como un **proceso Poisson compuesto** si para $t \geq 0$,*

$$X_t = \sum_{i=1}^{N_t} Y_i.$$

Entonces, $X(t)$ es una suma Poisson de variables aleatorias.

Proposición A.15. $E[X_t] = \lambda t E[Y]$.

Veamos algunos ejemplos.

Ejemplo A.3.3. *Retiro de efectivo del banco*

Suponga que los clientes de cierto banco ingresan a una sucursal para realizar retiros de efectivo en ventanilla de acuerdo a un proceso Poisson de tasa $\lambda = 1/3$ (tres clientes por minuto). Suponga que cada cliente retira efectivo de dicho banco en sólo una ocasión al día y que el monto retirado es una variable aleatoria independiente e idénticamente distribuida log-normal con media $\mu = 7$ y desviación típica $\sigma = 1.5$. El monto total de los retiros al tiempo t es un proceso Poisson compuesto

$$X(t) = \sum_{i=0}^{N_t} Y_i, \qquad \text{con} \qquad Y_0 = 0.$$

Ahora, supongamos que en este caso se desea calcular el monto total de los retiros en el transcurso de una hora, $T = 60$, es decir, $E[X_{60}]$. Sabemos que la distribución log-normal tiene función de densidad

$$f(x) = \frac{1}{\sqrt{2\pi\sigma x}} exp\left\{-\frac{(\log(x)-\mu)^2}{2\sigma^2}\right\},$$

donde μ y σ son la media y la desviación típica del logarítmo. Para cualquier variable aleatoria log-normal:

- la media es: $E(X) = exp\left\{\mu + \sigma^2/2\right\}$, y
- la varianza es: $Var(X) = \left(e^{\sigma^2} - 1\right) e^{2\mu+\sigma^2}$

Entonces

$$\begin{aligned} E(X(t)) &= E\left[\sum_{i=1}^{N(t)} Y_i\right] \\ &= E\left[E\left[\sum_{i=1}^{N(t)} Y_i | N(t)\right]\right] \\ &= E[N(t)E[Y_i]] = E[Y_i]\,E[N(t)]\,. \end{aligned}$$

Por otra parte, sabemos que: $N(t) \sim Poi(\lambda t)$, entonces $E[N(t)] = \lambda t$ y $Var[N(t)] = \lambda t$ y también que $E[Y_i] = e^{\mu+\sigma^2/2}$ y $Var[Y_i] = \left(e^{\sigma^2} - 1\right) e^{2\mu+\sigma^2}$ entonces,

$$\begin{aligned} E[X(t)] &= E[Y_i]\,E[N(t)] \\ &= e^{7+(1.5)^2/2}\lambda t \\ &= e^{8.125}(1/3)60 \\ &\approx 67557.36. \end{aligned}$$

Ejemplo A.3.4. ***El modelo clásico de Cramér-Lundberg.***

Este modelo tiene sus orígenes en la tesis doctoral de Filip Lundberg defendida en el a?o de 1903. En este trabajo Lundberg analiza el reaseguro de riesgos colectivos y presenta el proceso de Poisson compuesto.

En 1930 Harald Cramér retoma las ideas originales de Lundberg y las pone en el contexto de los procesos estocástico. El modelo ha sido estudiado en extenso y se han propuesto varias formas de generalizarlo.

El modelo que estudiaremos es el proceso a tiempo continuo $\{C_t\}_{t\geq 0}$ dado por

$$C_t = u + ct - \sum_{j=1}^{N_t} Y_j$$

donde:

- u es el capital inicial de la compa?ía aseguradora
- ct es la entrada por primas hasta el tiempo t con c una constante positiva
- Y_j es el moneto de la $j-$ésima reclamación, y
- N_t es un proceso de Poisson de parámetro λ.

C_t representa el balance más sencillo de ingresos menos egresos de una compañía aseguradora. Al proceso C_t se le llama proceso de riesgo o proceso de superávit.

La variable aleatoria C_t se puede interpretar como el capital de la compa?ía aseguradora al tiempo t y por razones naturales y legales es importante que C_t permanezca por arriba de cierto nivel mínimo. Cuando $C_t < 0$ se dice que hay ruina. La ruina casi nunca sucede en la práctica, es solamente un término técnico que produce alguna toma de decisión.

Por ejemplo, si el capital de una compa?ía aseguradora asignado a una cartera decrece en forma significativa, ésta automáticamente puede tomar ciertas medidas para subsanar esta situación y no se trata de un evento insalvable.

Ejemplo A.3.5. *Supongamos que en el modelo de Cramer Lumdberg el monto de reclamaciones siguen una distribución uniforme en* $(0, 100)$ *y éstas se presentan con una intensidad* $\lambda = 3$*. Modelar el capital esperado de la compa?ia aseguradora en el tiempo.*

A.3.4. Proceso de Poisson no homogéneo

Un proceso de conteo extremadamente importante para propósitos de modelación es el proceso Poisson no homogéneo, el cual omite la suposición de incrementos estacionarios del proceso Poisson. Esto abre la posibilidad a que la tasa de arribo no necesariamente sea constante, sino que pueda variar en el tiempo.

Definición A.28. *Un* ***Proceso de Poisson no homogéneo*** *es un proceso a tiempo continuo* $\{N_t\}_{t\geq 0}$ *y espacio de estados* $E = \{0, 1, 2, \ldots\}$ *con parámetro una función positiva y localmente integrable* $\lambda(t)$*, que satisface:*

1. $N_0 = 0$
2. *Tiene incremento independientes*

3. *Para cualquier $t \geq 0$ y $h > 0$ decreciente a cero se tiene que*

   a) $P(N_{t+h} - N_t \geq 1) = \lambda(t)h + o(h)$

   b) $P(N_{t+h} - N_t \geq 2) = o(h)$

El resultado análogo en cuanto a la distribución del número de eventos en un proceso de Poisson no homogéneo es el siguiente:

Proposición A.16. *La variable aleatoria N_t en un proceso de Poisson tiene distribución Poisson$(\Lambda(t))$, donde $\Lambda(t)) = \int_0^t \lambda(s)ds$ donde $\lambda(t)$ es la intensidad al tiempo t.*

De este resultado se sigue inmediatamente el siguiente resultado.

Proposición A.17. $N_{t+s} - N_t \sim Poisson(\Lambda(t+s) - \Lambda(t))$.

Ejemplo A.3.6. *Para los ejemplos A.3.1 y A.3.2 consideremos el supuesto que $\lambda(t) = t/2$. Resolver numericamente los ejemplos A.3.1 y A.3.2.*

Teorema A.9. *Sea $\{N_t : t \geq 0\}$ un proceso Poisson de parámetro 1 y una función de intensidad $\Lambda(t) = \int_0^t \lambda(u)du$. Si definimos al proceso $\{X_t :\geq 0\}$ por*

$$X_t := N_{\Lambda(t)} \qquad t \geq 0,$$

entonces $\{X_t : t \geq 0\}$ es un proceso Poisson no homogéneo de intensidad $\Lambda(t)$.

Demostración. 1. Primero notemos $X_0 = 0$. Esto ocurre pues $\Lambda(0) = 0$ y $N_0 = 0$, por ser $\{N_t : t \geq 0\}$ un proceso Poisson.

2. Probaremos que el proceso $\{X_t : t \geq 0\}$ tiene incrementos independientes. Recordemos que $\lambda(t) \geq 0$, lo que implica que $\Lambda(t)$ es una función no decreciente. Así, si consideramos una colección arbitraria de tiempos $0 \leq t_1 \leq ... \leq t_n$, ocurre que las evaluaciones de tales tiempos bajo la función Λ, dados por $0 \leq \Lambda(t_1) \leq ... \leq \Lambda(t_n)$, mantienen el orden. Entonces, para una colección arbitraria de tiempos $0 \leq t_1 \leq ... \leq t_n$, las variables aleatorias $\{X_{t_{i+1}} - X_{t_i}\}_{i=1}^{n-1} = \{N_{\Lambda(t_{i+1})} - N_{\Lambda(t_i)}\}_{i=1}^{n-1}$ son independientes, debido a la propiedad de incrementos independientes del proceso Poisson $\{N_t : t \geq 0\}$, evaluado en los tiempos $0 \leq \Lambda(t_1) \leq ... \leq \Lambda(t_n)$.

3. Demostraremos que la variable aleatoria $X_t - X_s$ tiene distribución $Poisson(\Lambda(t) - \Lambda(s))$, a través de su función generadora de momentos $m_{X_t - X_t}(\theta)$. Debido al punto anterior ocurre que $X_t - X_s$ es independiente a X_s por lo que

$$\begin{aligned} m_{X_t}(\theta) &= \mathbf{E}(e^{\theta X_t}) \\ &= \mathbf{E}(e^{\theta(X_s + X_t - X_t)}) \\ &= \mathbf{E}(e^{\theta X_s})\mathbf{E}(e^{\theta(X_t - X_t)}) \\ &= m_{X_s}(\theta) m_{X_t - X_t}(\theta). \end{aligned}$$

   Se sigue que

$$\begin{aligned} m_{X_t - X_t}(\theta) &= \frac{m_{X_t}(\theta)}{m_{X_t}(\theta)} \\ &= \frac{e^{\Lambda(t)(e^\theta - 1)}}{e^{\Lambda(s)(e^\theta - 1)}} \\ &= e^{(\Lambda(t) - \Lambda(s)(e^\theta - 1))} \end{aligned}$$

Concluimos que $X_t - X_s$ tiene distribución $Poisson(\Lambda(t) - \Lambda(s))$.

□

A.3.5. Proceso de Poisson espacial

Consideremos el espacio $\mathbb{R}^n$ y a una función $\lambda : \mathbb{R}^n \to [0, \infty)$ integrable. Para los conjuntos $S \subseteq \mathbb{R}^n$ integrables, definamos a la función de conjuntos:

$$\mu(S) := \int_S \lambda(t)dt, \tag{A.1}$$

donde estamos utilizando la notación $dt = dt_1 \dots dt_n$, y $t = (t_1, ..., t_n)$.
Ocurre que $\mu(\cdot)$ satisface las siguientes propiedades intuitivas, que presentaremos sin demostración:

1. $\mu(A) \geq 0$ para todo conjunto A integrable.
2. $\mu(\emptyset) = 0$.
3. Si tenemos una colección de conjuntos $A_1, A_2, ...$ que son disjuntos a pares, ocurre que

$$\mu\Big(\bigcup_{i=1}^{\infty} A_i\Big) = \sum_{i=1}^{\infty} \mu(A_i).$$

Definición A.29. *Decimos que X es un proceso Poisson (no homogéneo) con medida media μ (y tasa λ) si X es un subconjunto aleatorio discreto (es decir numerable y sin puntos de acumulación) de $\mathbb{R}^d$, tal que al definir la función de conjuntos:*

$$X(A) := \#(X \cap A)$$

se cumple que para cualquier colección de conjuntos $S_1, ..., S_r$ integrables y enteros no negativos $k_1, ..., k_r$ ocurre que

$$\mathbb{P}(X(S_1) = k_1, ..., X(S_r) = k_r) = \prod_{i=1}^{r} \frac{\mu(S_i)^{k_i} e^{-\mu(S_i)}}{k_i!}, \tag{A.2}$$

donde μ está definido a través de λ por la Ecuación (A.1).

Tenemos el siguiente resultado, que es un corolario directo de la definición.

Proposición A.18. *Un proceso Poisson espacial X con medida media μ satisface que*

1. *Para cada S, ocurre que $X(S) \sim Poisson(\mu(S))$.*
2. *Cuando $S_1, ..., S_r$ son conjuntos disjuntos, las variables aleatorias $\{X(S_i)\}_{i=1}^r$ son independientes.*

Demostración. Para probar 1) basta tomar $r = 1$ en la Ecuación (A.2) que nos dice

$$\mathbb{P}(X(S) = k) = \frac{\mu(S)^k}{k} e^{\mu(S)} \qquad k = 0, 1, 2, ...,$$

lo que significa que $X(S)$ tiene distribución Poisson de parámetro $\mu(S)$. Utilizando 1) sabemos que cada $X(S_i)$ tiene distribución Poisson de parámetro $\mu(S_i)$ y la Ecuación (A.2) nos dice que en el caso en que los conjuntos $\{S_i\}_{i=1}^r$ son disjuntos a pares ocurre que la densidad conjunta de las variables aleatorias $\{X(S_i\}_{i=1}^r$ es el producto de sus densidades marginales, lo que implica la independencia de las variables.
□

Veamos algunos ejemplos de caso Poisson.

Ejemplo A.3.7. *Tomemos el caso particular en que $\lambda(t) \equiv \lambda > 0$. En este caso*

$$\mu(S) = \int_S \lambda dt = \lambda \int_S dt = \lambda |S|,$$

para todo S integrable; donde $|S|$ denota al n-volumen del conjunto $|S|$ (en el caso en que $n = 1$, el n-volumen es la longitud, para $n = 2$ el área, para $n = 3$ el volumen usual). Al proceso Poisson que tiene por medida media $\mu(S) = \lambda|S|$ se le conoce como proceso Poisson espacial homogéneo. El caso $n = 1$ corresponde al proceso Poisson homogeneo usual (aunque definido en toda la recta real). La Ecuación (A.2), para una colección de intervalos disjuntos $\{(A_i, b_i]\}_{i=1}^r$, en este último caso se transforma en

$$\mathbb{P}(X(a_1, b_1] = k_1, ..., X(a_r, b_r] = k_r) = \prod_{i=1}^r \frac{[\lambda(b_i - a_i)]^{k_i} e^{-\lambda(b_i - a_i)}}{k_i!}.$$

Ejemplo A.3.8. *Consideremos una función $\Lambda : \mathbb{R} \to \mathbb{R}$ no decreciente y derivable, con función derivada λ. Debido a que Λ es no decreciente, λ es no negativa. Definamos a μ a través de λ, por medio de la Ecuación A.1. Entonces el proceso de Poisson espacial X, definido en la recta real, coincide con la definición de un proceso de Poisson no homogéneo con intensidad Λ. La Ecuación (A.2) toma la forma aquí:*

$$\mathbb{P}(X(a_1, b_1] = k_1, ..., X(a_r, b_r] = k_r) = \prod_{i=1}^r \frac{[\Lambda(b_i) - \Lambda(a_i)]^{k_i} e^{-(\Lambda(b_i) - \Lambda(a_i))}}{k_i},$$

para intervalos disjuntos.

Tenemos la siguiente proposición.

Proposición A.19. *Sea X un proceso de Poisson espacial en $\mathbb{R}^n$ con medida media μ. Sea $S \subseteq \mathbb{R}^n$ un conjunto integrable tal que $\mu(S) < \infty$. Condicionado al evento $\{X(S) = n\}$, para todo $A_1, ..., A_n \subseteq S$ disjuntos e integrables ocurre que*

$$\mathbb{P}(X(A_1) = n_1, ..., X(A_k) = n_k | X(S) = n) = \frac{n}{n_0! n_1! ... n_k!} p(A_0)^{n_0} p(A_1)^{n_1} ... p(A_k)^{n_k},$$

para todas los valores $\{n_1, ..., n_k\}$ tales que $\sum_{i=1} k n_i = n$. Aquí p es la función de conjuntos (subconjuntos de S) definida por $p(A) := \frac{\mu(A)}{\mu(S)}$, el valor $n_0 := n - \sum_{i=1}^n n_i$ y $A_0 := S \backslash \left(\bigcup_{i=1}^n A_i \right)$. En otras palabras, condicionado al evento $\{X(S) = n\}$, el vector $(X(A_0), ..., X(A_r))$ distribuye $Multinomial(n, r, p(A_0), ..., p(A_r))$.

Demostración. Notemos que, bajo el condicional, ocurre que

$$P(X(A_1) = n_1, ..., X(A_k) = n_k | X(S) = n) = P(X(A_0) = n_0, X(A_1) = n_1, ..., X(A_k) = n_k | X(S) = n),$$

que por definición es igual a

$$\frac{\mathbb{P}(X(A_0) = n_0, X(A_1) = n_1, ..., X(A_k) = n_k)}{\mathbb{P}(X(S) = n)}.$$

Debido a que los conjuntos $\{A_i\}_{i=0}^n$ son disjuntos, podemos sustituir la fórmula (A.2) en el numerador y denominador de la expresión anterior, para obtener

$$\begin{aligned} P(X(A_1) = n_1, ..., X(A_k) = n_k | X(S) = n) &= \frac{\prod_{i=0}^k \frac{\mu(A_i)^{n_i} e^{-\mu(A_i)}}{n_i!}}{\frac{\mu(S)^n e^{-\mu(S)}}{n!}} \\ &= \frac{n!}{n_0! n_1! ... n_k!} \left(\frac{\mu(A_0)}{\mu(S)} \right)^{n_0} \left(\frac{\mu(A_1)}{\mu(S)} \right)^{n_1} ... \left(\frac{\mu(A_k)}{\mu(S)} \right)^{n_k}, \end{aligned}$$

lo que queríamos probar. □

Una observación importante es que en el caso en que $\mu(\cdot) = \lambda|\cdot|$, la Proposición A.19 nos dice que $(X(A_0), ..., X(A_r))$ tiene distribución $Multinomial\Big(n, r, \frac{|A_0|}{|S|}), ..., \frac{|A_r|}{|S|}\Big)$. Esto implica que los n puntos (a los cuáles estamos condicionando al proceso X a tener en el conjunto S) están distribuidos de forma independiente y uniformemente distribuidos sobre el conjunto S. Note cómo esto generaliza una propiedad similar para el proceso Poisson homogéneo en $[0, \infty)$.

La observación anterior nos permite simular con facilidad a un proceso Poisson homogéneo en $\mathbb{R}$ con tasa λ en un subconjunto S. El algoritmo es el siguiente.

Algoritmo 63 : Proceso Poisson homogéneo

1: Simular una variable aleatoria N con distribución $Poisson(\lambda|S)$.
2: Dado $N = n$, simular n variables aleatorias independientes uniformes en S.
3: Devolver las posiciones de las n variables aleatorias generadas.

Note que en el penúltimo paso, para simular una variable aleatoria uniforme en un conjunto integrable y acotado S basta simular variables aleatorias en una caja $[a_1, b_1] \times ... \times [a_n, b_n]$ que contenga a S y utilizar el método de aceptación-rechazo para asegurarnos que tal punto pertenece a S.

Habiendo implementado el algoritmo anterior, podemos generalizar el método propuesto para simular un Proceso Poisson no homogéneo en $[0, \infty)$ para simular un proceso Poisson espacial con tasa $\lambda(t)$. Supondremos que la tasa $\lambda(t)$ es acotada en S, es decir, existe λ^* tal que $\lambda(t) \leq \lambda^*$ para todo t en S.

Algoritmo 64 : Proceso Poisson homogéneo alternativa

1: Simular una variable aleatoria N con distribución $Poisson(\lambda^*|S|)$.
2: Dado $N = n$, simular n variables aleatorias independientes uniformes en S, denotadas por $\{X_1, ..., X_n\}$.
3: Simular $\{U_1, ..., U_n\}$ variables aleatorias uniformes en $(0, 1)$ independientes.
4: Para cada i, conservar al punto X_i cuando ocurra que $U_i \leq \frac{\lambda(X_i)}{\lambda^*}$.
5: Devolver los puntos que fueron conservados.

La prueba de que el Algoritmo 64 funciona es análoga a la prueba de que el algoritmo para Proceso Poisson no homogéneo en $[0, \infty)$ funciona.

A.4. Procesos de Saltos de Markov

En esta sección consideremos un proceso de Markov $X = \{X_t\}_{t\geq 0}$ a tiempo continuo, homogéneo en el tiempo y con espacio de estados finito $E = \{1, 2, \ldots, m\}$.

Definición A.30. *Una matriz $P(t) = \{p_{ij}(t)\}_{i,j\in E}$ (para cada $t \geq 0$) es llamada matriz de transición de Markov asociada al proceso de Markov $X = \{X_t\}_{t\geq 0}$.*

A la familia $\{P(t)\}_{t\geq 0}$ se le llama semigrupo de transición del proceso de Markov.

Teorema A.10. *La familia $\{P(t)\}_{t\geq 0}$ es un semigrupo estocástico, es decir, satisface:*

1. $P(0) = I$,

2. *$P(t)$ es una matriz estocástica y*

3. *La ecuación Chapman-Kolmogorov $P(s+t) = P(s)P(t)$.*

Cuando el espacio de estados es finito el comportamiento aleatorio de un proceso de Markov $X = \{X_t\}_{t\geq 0}$ está determinado por su semigrupo estocástico $\{P(t)\}_{t\geq 0}$ y la distribución de X_0.

Definición A.31. *El semigrupo $\{P(t)\}_{t\geq 0}$ es estándar si*

$$P(t)_{t\downarrow 0} \to 0$$

Nosotros consideramos sólo procesos de Markov con semigrupos estándar lo cual garantiza que las realizaciones de éstos son funciones continuas por la derecha, con mayor precisión, se tiene que el proceso es estocasticamente continuo, separable y medible en intervalos compactos. Además, existe una versión separable que es estocásticamente equivalente a este proceso. De hecho, con el espacio de estados finitos se tiene que sus realizaciones son funciones escalonadas, es decir, para casi todo $w \in \Omega$ y para todo $t \geq 0$ existen $\Delta(t, w) > 0$ tal que

$$X_{t+\tau}(w) = X_t(w)$$

para $\tau \in [0, \Delta(t, w))$. Esto motiva el concepto de procesos de saltos de Markov (PSM).

Definición A.32. *Un PSM es un proceso de Markov $X = \{X_t\}_{t\geq 0}$ a tiempo continuo, con espacio de estados finito (a lo más numerable en general) que empieza en un estado x_0 al tiempo $\tau_0 = 0$ y permanece en ese estado hasta un tiempo τ_1 cuando realiza una transición (salto) a otro estado x_1. Permanece en ese nuevo estado hasta un tiempo $\tau_2 > \tau_1$, momento en el cual salta a otro estado x_2 y así sucesivamente.*

De acuerdo a esta definición, resulta de interés estudiar la ley de probabilidad que gobierna el tiempo de estancia de un PSM en un estado i y la transición a otro estado. Con esos objetivos consideramos a la variable aleatoria

$$\Delta(t) = \inf\{s > 0 | X_{t+s} \neq i, X_t = i\}, \tag{A.3}$$

es decir, $\Delta(t)$ es la variable aleatoria que modela el tiempo de estancia en el estado i desde el tiempo t. $\Delta(t)$ es un tiempo de paro.

Proposición A.20. *Sea $X = \{X_t\}_{t\geq 0}$ un PSM, entonces*

$$P(\Delta(t) > s | X_t = i) = e^{-\lambda_i S}.$$

para todo $s, t > 0$ e $i \in E$.

Con este resultado tenemos que el tiempo de estancia en un estado para un PSM sigue una distribución exponencial de parámetro dependiente del estado. Nosotros nos restringimos al estudio de PSM tales que $\lambda_i < \infty$, es decir, PSM que no dan saltos instantaneos.

Ahora introducimos notación útil para determinar la transición entre estados en un PSM.

Definición A.33. *Sea $X = \{X_t\}_{t\geq 0}$ un PSM entonces definimos*

1. *$\tau_0 = 0$,*

2. *$\tau_k =$ el tiempo del $k-$ésimo salto,*

3. x_k = *el estado visitado en el intervalo de tiempo* $[\tau_k, \tau_{k+1})$,

4. $\Delta_k = \tau_{k+1} - \tau_k$, *es decir, el tiempo de permanencia en* x_k *y*

5. $N(t) = max\{n \geq 0 | \tau_n < t\}$, *es decir, el número de saltos hasta el tiempo* t.

Definición A.34. *Sea* $X = \{X_t\}_{t\geq 0}$ *un PSM decimos que es regular si*

$$\tau_\infty := sup\tau_k = \infty.$$

El siguiente resultado relaciona el tiempo de estancia en un estado con la probabilidad de transición.

Proposición A.21. *Sea* $X = \{X_t\}_{t\geq 0}$ *un PSM regular y* $\tau_{k+1} < \infty$. *Entonces, condicionada a* $X_{\tau_k} = i$, *las variables aleatorias* Δ_{k+1} *y* $X_{\tau_{k+1}}$ *son independientes.*

A.4.1. Generador Infinitesimal

Sea $X = \{X_t\}_{t\geq 0}$ un PSM y que $X_t = i$. Ahora, estamos interesados en analizar que pasa en intervalos de tiempo infinitesimales $(t, t+h)$, tenemos los siguientes casos.

1. El proceso puede estar en el mismo estado al tiempo $t+h$ y esto ocurre con probabilidad $p_{ii}(h)+o(h)$, donde $o(h)$ representa la posibilidad que el proceso se salga y regrese al estado i en el intervalo de tiempo.

2. El proceso se mueva al estado j $t+h$ y esto ocurre con probabilidad $p_{ij}(h) + o(h)$, donde $o(h)$ representa la posibilidad que el proceso realize dos o más transiciones en el intervalo de tiempo.

Proposición A.22. *Conssideremos el semigrupo* $\{P(t)\}_{t\geq 0}$ *de un proceso de saltos de Markov. Entonces, el límite*

$$\lim_{h\to o^+} \frac{P(h) - I}{h} = Q$$

existe.

A la matriz $Q = \{q_{ij}\}_{i,j\in E}$ la llamaremos generador infinitesimal y es fácil ver que su correspondiente matriz de transición al tiempo t está dada por

$$P(t) = exp^{tQ}.$$

Teorema A.11. *Sea* Q *el generador infinitesimal de un proceso de saltos de Markov* $\{X_t\}_{t\geq 0}$, *entonces*

1. *Dado que el proceso está en el estado* i *al tiempo* t, *la variable aleatoria* $\Delta(t)$ *tiene distribución exponencial de parámetro* $-q_{ii}$ *y*

2. *si el proceso está en el estado* i *al tiempo* t *y* $-q_{ii}$, *con probabilidad 1 existe una función discontinua para algún* $t > 0$ *y de hecho la primera disconinuidad es un salto. Si* $0 < s <\leq \infty$, *la probabilidad condicional de que la primera discontinuidad en el intervalo* $[t, t+s)$ *sea un salto al estado* j $(i \neq j)$ *es* $\frac{q_{ij}}{q_i}$, $(q_i = -q_{ii})$.

Definición A.35. *Puente de Markov.*
Se define como Puente de Markov, con parámetros T, a, b, *un proceso estocástico indezado por* $t \in [0, T]$ *determinado por la distribución de* $\{X(t) : 0 \leq t \leq T\}$ *condicionado con* $X(0) = a$ *y* $X(T) = b$.

Ejemplo A.4.1. *Proceso de nacimiento y muerte.*

En este proceso X_t representa la población de una cierta entidad al tiempo t, donde a tasa de nacimiento es λ y la tasa de mortadidad es β. Si suponemos que la población está acotada superiormete, éste es un ejemplo de un proceso de saltos de Markov que puede ser facilmente simulado.

A.5. Procesos con valores continuos

A.5.1. El Movimiento Browniano y sus transformaciones

Definición A.36. *Movimiento Browniano Estándar*

Un Movimiento Browniano Estándar (MB) es un proceso B markoviano a tiempo continuo con incrementos estacionarios e independientes tal que

1. $\mathbb{P}[B_0 = 0] = 1$,
2. B tiene trayectorias continuas casi seguramente, y
3. $B_t - B_s \sim Normal(0, t-s), \qquad \forall 0 \leq s \leq t.$

Este proceso es un proceso fundamental en la teoría de procesos estocásticos. La primera razón para ello es que es un proceso continuo análogo a la caminata aleatoria simple; su dinámica es simétrica en la posición y equiprobable. La segunda razón es su **universalidad**. En matemáticas, la universalidad de un modelo significa que aparece constantemente: así como la distribución normal aparece como un límite de sumas (centradas y escaladas) de variables aleatorias independientes y arbitrarias (bajo algunos supuestos generales), el Movimiento Browniano aparece como el límite (centrado y escalado) de procesos arbitrarios (bajo algunos supuestos generales).

Las trayectorias del movimiento Browniano son no diferenciables en ninguna parte y satisfacen fractalidad, es decir, que un intervalo seleccionado de las valuaciones del Movimiento Browniano, tras escalarse, sigue teniendo la misma distribución que la trayectoria del Movimiento en $[0, 1]$. A pesar de tener tales extranñas propiedades, hay muchas cantidades que pueden calcularse debido a que los incrementos tienen la distribución normal y son independientes. En la literatura puede encontrarse una enciclopedia de resultados sobre las probabilidades de que las trayectorias satisfagan una característica dada. Esto afianza más aún su uso en modelos aplicados.

Un Movimiento Browniano (no estándard) de parámetros μ (media o deriva), y σ^2 (coeficiente de difusión o volatilidad) es un proceso que satisface las mismas propiedades que el Movimiento Browniano estándar pero tal que

$$B_t - B_s \sim Normal(\mu(t-s), \sigma^2(t-s)), \qquad \forall 0 \leq s \leq t.$$

A partir del Movimiento Browniano podemos definir más procesos como transformaciones del MB.

Definición A.37. *Un movimiento Browniano geométrico G está definido como la siguiente transformación del Movimiento Browniano estándar B:*

$$G_t := G_0 \exp\{(\mu - \frac{\sigma^2}{2})t + \sigma B_t\}.$$

Cabe mencionar, que en el modelo de Black-Scholes-Merton, se supone que el activo financiero tiene un comportamiento aleatorio modelado por un Movimiento Browniano Geométrico.

Definición A.38. *Un proceso de Orstein-Uhnlenbeck o proceso de Vasicek es un proceso X definido como la siguiente transformación del Movimiento Browniano estándar B:*

$$X_t := X_0 e^{-at} + b(1 - e^{-at}) + \frac{\sigma e^{-at}}{\sqrt{2a}} B(e^{2at} - 1).$$

Este modelo se utiliza ampliamente para modelar tasas de interés o demográficas. Tiene la desventaja de que tiene posibilidad de tomar valores negativos. El parámetro b representa la media a largo plazo, al parámetro a se le conoce como velocidad de reversión hacia la media, y σ es la volatilidad del proceso.

A.6. Ecuaciones diferenciales estocásticas

En esta sección presentaremos de forma no rigurosa el concepto de una Ecuación Diferencial Estocástica (SDE, por sus siglas en inglés). Presentemos primero a los procesos Markovianos con valores continuos en un cierto nivel de generalidad.

Definición A.39. *Proceso de difusión*

Un proceso de difusión $\{X_t\}_{t\geq 0}$ unidimensional es un proceso estocástico a tiempo continuo con propiedad fuerte de Markov y trayectorias continuas casi en todas partes.

Si $I = [r, s]$, es el espacio de estados de $\{X_t\}_{t\geq 0}$ entonces

$$lim_{h\downarrow 0} \frac{1}{h} \mathbb{P}[|X_{t+h} - x| > \epsilon | X_t = x] = 0$$

$\forall x \in I$ y $\epsilon > 0$.

Casi todos los procesos de difusión son caracterizados por dos condiciones básicas que describen su media y varianza infinitesimales. Sea $\Delta h X_t = X_{t+h} - X_t$ y $x \in I$, dadas por

$$lim_{h\downarrow 0} \frac{1}{h} \mathbb{E}[\Delta h X_t | X_t = x] = \mu(x, t)$$

y

$$lim_{h\downarrow 0} \frac{1}{h} \mathbb{P}[(\Delta h X_t)^2 | X_t = x] = \sigma(x, t).$$

Asociados a los procesos estocásticos anteriores, tenemos ecuaciones diferenciales a través de los cuáles podemos definirlos (tal como algunas funciones reales pueden definirse como soluciones de ecuaciones diferenciales ordinarias).

Definición A.40. *Ecuación Diferencial Estocástica*

Sean $\mu(x,t)$ y $\sigma(x,t)$ dos funciones $\mathbb{R}\times[0,T]\to\mathbb{R}$. Una SDE es una ecuación de la forma

$$dX_t = \mu(X_t,t)dt + \sigma(X_t,t)dB_t,$$

definida para los valores de t en $[0,T]$ y con condición inicial X_0 independiente del Movimiento Browniano, cuya solución representa un proceso de difusión con tales condiciones infinitesimales.

Notemos que en la ecuación diferencial aparece un diferencial relativo al Movimiento Browniano. Este diferencial no tiene sentido dentro de la teoría de integración de cálculo, ni de la teoría de integración de Stieljes o de Lebesgue, debido a la gran oscilación que tienen las trayectorias de un Movimiento Browniano. El **cálculo estocástico**, desarrollado principalmente por Kiyoshi Itô, desarrolla una teoría en donde tal diferencial adquiere sentido. Su profundidad es demasiada para exponerse aquí; tan sólo mencionaremos que es una teoría adecuada para funciones aleatorias (procesos estocásticos) que generaliza al cálculo diferencial e integral estándard al incluir la posibilidad de integrar y derivar funciones que tienen una gran oscilación (como el movimiento Browniano).

Definición A.41. *Tiempo de Llegada*

Sea $\{X_t\}_{t\geq 0}$ un Proceso de Difusión, definimos a la variable aleatoria T_z, para $z\in I$, como

$$T_z \quad = \quad \begin{cases} inf\{t\geq 0 \| X_t = z\} & \{t\geq 0\|X_t=z\}\neq\varnothing \\ \infty & e.o.c \end{cases}$$

Definición A.42. *Proceso regular*

Un proceso de difusión $\{X_t\}_{t\geq 0}$ es regular si

$$\mathbb{P}[T_z<\infty | X_0 = x] > 0, \forall x,z\in I.$$

Teorema A.12. *Transformaciones de Procesos de Difusión*

Sea $\{X_t\}_{t\geq 0}$ un Proceso de Difusión regular en el espacio de estados I y parámetros $\mu(x,t)$ y $\sigma^2(x,t)$. Sea g estricta monótona en I con segunda derivada continua, entonces $Y_t = g(X_t)$ define un Proceso de Difusión Regular en $I' = [g(r), g(s)]$ y tiene parámetros

$$\mu_Y(x,t) = \frac{1}{2}\sigma^2(x,t)g''(x) + \mu(x,t)g'(x)$$

y

$$\sigma_Y^2(x,t) = \sigma^2(x,t)[g'(x)]^2$$

Es importante mencionar algunos conceptos que son importantes para abordar de manera más amplia el cálculo estocástico,

Lema A.1. *Lema de Itô.*
Sea $dX_t = b(t,X_t)dt + \sigma(t,X_t)dW_t$ entonces $Y_t = f(X_t,t)$ satisface la EDE:

$$dY_t = \frac{\partial f}{\partial t}dt + \frac{\partial f}{\partial x}dX_t + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}dX_t dX_t = (\frac{\partial f}{\partial t} + b\frac{\partial f}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2 f}{\partial x^2})dt + \sigma\frac{\partial f}{\partial x}dW_t$$

A.7. Esquema de Milstein

El esquema de Milstein es un refinamiento del modelo de Euler cuyo objetivo radica en reducir el error de discretización. No se debe confundir con un modelo de reducción de varianza.
Sea X un proceso de difusión de la forma:

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t$$

dónde b y σ son funciones doblemente diferenciables. Considere la discretización del tiempo de la siguiente manera $0 = t_0 < t_1 < .... < t_n = T$. Para cada $i = 0, 1, ..., n-1$ definimos:

$$X(t_{i+1}) = X(t_i) + \int_{t_i}^{t_{i+1}} b(X_t)dt + \int_{t_i}^{t_{i+1}} \sigma(X_t)dW_t$$

En este esquema se examina de manera más detallada a las funciones $b(X_t)$ y $\sigma(X_t)$ con ayuda de la fórmula de Itô. Sea $f(X_t)$, función doblemente diferenciable, por el Lema de Itô tenemos que:

$$f(X_t) = f(X_{t_i}) + \int_{t_i}^{i} L^0 f(X_s)ds + \int_{t_i}^{t} L^1 f(X_s)dW_s$$

donde

$$L^0 f(X) = f^{'}(x)b(x) + \frac{1}{2}f^{''}(x)\sigma^2(x), L^1 f(x) = f^{'}(x)\sigma(x)$$

Entonces,

$$X_{t_{i+1}} = X_{t_i} + b(X_{t_i})(t_{i+1} - t_i) + \sigma(X_{t_i})(W_{t_{i+1}} - W_{t_i}) + R$$

donde

$$R = \int_{t_i}^{t_{i+1}} \left[\int_{t_i}^{t} L^0 b(X_s)ds + \int_{t_i}^{t} L^1 b(X_s)dW_s \right] dt + \int_{t_i}^{t_{i+1}} \left[\int_{t_i}^{t} L^0 \sigma(X_s)ds + \int_{t_i}^{t} L^1 \sigma(X_s)dW_s \right] dW_t$$

conocido como el residuo. Este refinamiento mejora el Esquema de Euler pues aporta algunos de los términos de orden superior en R. Lo que se realiza es la aproximación de la doble integral como:

$$\begin{aligned} L^1\sigma(X_t) \int_{t_i}^{t_{i+1}} \int_{t_i}^{t} dW_s dW_t &= L^1\sigma(X_{t_i}) \int_{t_i}^{t_{i+1}} (W_t - W_{t_i})dW_t \\ &= \frac{1}{2} L^1\sigma(X_{t_i}) \left[(W_{t_{i+1}} - W_{t_i})^2 - (t_{i+1} - t_i) \right] \end{aligned}$$

Finalmente, el esquema de Milstein está dado por:

$$X(t_{i+1}) = X_{t_i} + b(X_{ti})(t_{i+1} - t_i) + \sigma(X_{t_i})\sqrt{t_{i+1} - t_i}Z_{i+1} + \frac{1}{2}\sigma^{'}(X_{t_i})\sigma(X_{t_i})(t_{i+1} - t_i)(Z_{i+1}^2 - 1)$$

con $\{Z_{t_i}\}$ variables aleatorias normales independientes.

Definición A.43. *Puente Browniano*

Se define como Puente Browniano aquel proceso $\{X_t : t \in [0, 1]\}$ solución de la ecuación diferencial estocástica

$$dX_t = -\frac{X_t}{1-t}dt + dB_t$$

con condición inicial $X_0 = 0$.

éste proceso tiene distintas caracterizaciones y cumple que

$$lim_{t \to 1-} X_t = 0 c.s.$$

Apéndice B

FUNCIONES

B.1. Generadores Lineales Congruenciales

B.1.1. Generador Lineal Congruencial

Descripción

`Gen.Lin.Cong` nos regresa un vector de números pseudo-aleatorios generados por la sucesión recursiva.

Código

```
> Gen.Lin.Cong<-function(n,a=7^5,m=2^31-1,c=0,
+                        sem=as.numeric(Sys.time())){
+   V<-numeric(n)
+   V[1]<-sem
+   for(i in 1:(n-1)){
+     V[i+1]<-(a*V[i]+c)%%m
+   }
+   return(V/m)
+ }
```

Argumentos

- n, número de observaciones.
- a, valor en los reales positivos del multiplicador (por default 7^5).
- c, valor en los reales no negativos del incremento (por default 0, llamado entonces Generador Congruencial Multiplicativo).
- m, valor en los reales positivos del módulo (por default $2^{31} - 1$).
- sem, valor real positivo inicial o semilla (por default la hora de la máquina).

B.1.2. Generador Congruencial Semilla-Controlado Simple

Descripción

`Gen.Cong.SemCont` nos regresa un vector de números pseudo-aleatorios generados por la sucesión recursiva, iniciando por un número propuesto en (0,1) que puede considerarse como la primera variable aleatoria del total de la muestra.

Código

```
> Gen.Cong.SemCont<-function(n,a=7^5,m=2^31-1,
+                            sem=as.numeric(Sys.time())){
+   V<-numeric(n)
+   V[1]<-sem/m
+   for(i in 2:n){
+     V[i]<-((floor(m*a*V[i-1]))%%m)/m
+   }
+   return(V)
+ }
```

Argumentos

- n, número de observaciones mayor que 1.
- a, valor en los reales positivos del multiplicador (por default 7^5).
- m, valor en los reales positivos del módulo (por default $2^{31} - 1$).
- sem, valor real positivo inicial o semilla (por default la hora de la máquina), que debe ser menor a m si es propuesta.

B.1.3. Pruebas estadísticas

Wald-Wolfowitz

```
> library(adehabitatLT)#Cargamos la libreria
> y<-factor()
> for(i in 1:500){
+   a<-0
+   U<-runif(1)
+   if(U>.5){a<-1}
+   y<-c(y,a)
+ }
> wawotest(y)
```

```
          a           ea           va           za            p
-32.6249860  -1.0000000 497.9947988  -1.4171569   0.9217815
```

```
>
```

Anderson-Darling

```
> library(ADGofTest) #Cargamos la paqueteria
> y<-runif(100)
> ad.test(y) #Llamamos a hacer una prueba AD para uniformes (0,1)
```

```
        Anderson-Darling GoF Test

data:  y
AD = 0.8164, p-value = 0.469
alternative hypothesis: NA
```

```
> # Generamos la muestra por GCM
> muestra<-Gen.Lin.Cong(100)
> ad.test(muestra)
```

```
        Anderson-Darling GoF Test

data:  muestra
AD = 0.95354, p-value = 0.3824
alternative hypothesis: NA
```

B.2. Generadores de variables aleatorias

B.2.1. Generador de variables aleatorias discretas (Transformada Inversa).

Descripción

rTInv nos regresa un vector de observaciones generadas a partir de un vector que represente la distribución de una variable aleatoria.

Código

```
> rTInv<-function(n,D){
+   if(sum(D)!=1)stop("El vector de densidades debe sumar 1")
+   muestra<-numeric(n)
+   for(i in 1:n){
```

```
+   k<-0
+   U<-runif(1)
+   F<-0
+   while(F<U){
+     F<-F+D[k+1]
+     if(F<U){k<-k+1}else{F<-1}
+   }
+   muestra[i]<-k
+   }
+   return(muestra)
+ }
```

Argumentos

- n, número de observaciones.
- D, vector de la distribución (refleja entonces la cardinalidad del soporte y las entradas suman uno).

B.2.2. Generador de observaciones exponenciales por inversión.

Descripción

`G.exp.inv` aplica el método de transformada inversa para la generación de variables aleatorias con distribución exponencial, i.e. con función de densidad

$$f(x) = \lambda e^{-\lambda x}$$

y entonces como vimos la distribución tiene función inversa.

Código

```
> G.exp.inv<-function(n,lambda=1){
+   V<-numeric(n)
+   for(i in 1:n){U<-runif(1);V[i]<--log(1-U)/lambda}
+   return(V)
+ }
```

Argumentos

- n, número de observaciones.
- lambda, valor real positivo del parámetro λ (por default 1).

B.2.3. Generador de observaciones Poisson por Inversa Generalizada.

Descripción

`G.poi.inv` aplica el método de transformada inversa en el caso discreto para generación de variables aleatorias con distribución Poisson, haciendo uso de la propiedad de multiplicaciones iterativas.

Código

```
> G.poi.inv<-function(n,lambda=1){
+   V<-numeric(n)
+   for(j in 1:n){
+     p<-exp(-lambda)
+     F<-p
+     i<-0
+     U<-runif(1)
+     if(F<U){
+     while(F<U){
+       i<-i+1
+       p<-p*lambda/i
+       F<-F+p
+       }
+     }
+     V[j]<-i
+   }
+   return(V)
+ }
```

Argumentos

- n, número de observaciones.
- lambda, valor real positivo del parámetro λ (por default 1).

B.2.4. Generador de observaciones Poisson Modificado.

Descripción

`G.poi.mod` aplica el método de transformada inversa en el caso discreto para generación de variables aleatorias con distribución Poisson, con la modificación conveniente de iniciar el valor a escoger de la variable aleatoria en el valor de λ. Para uso ilustrativo también registra las iteraciones realizadas para obtener cada observación.

Código

```
> G.poi.mod<-function(n,lam){
+   M<-numeric(n)
+   PoisCont<-function(x)eval((lam^x)*exp(-lam)/gamma(x+1))
+   it<-numeric(n)
+   for(i in 1:n){
+   U<-runif(1)
+   k<-floor(lam)
+   F<-sum(PoisCont(0:k))
+     if(F<U){
+       while(F<U){
+         F<-F+PoisCont(k+1)
+         if(F<U){k<-k+1;it[i]<-it[i]+1}
+       }
+     }else{
+       while(F>U){
+         F<-F-PoisCont(k)
+         if(F>U){k<-k-1;it[i]<-it[i]+1}
+       }
+     }
+    M[i]<-k
+   }
+   info<-data.frame(muestra=M,iteraciones=it)
+ }
```

Argumentos

- n, número de observaciones.
- lambda, valor real positivo del parámetro λ.

B.2.5. Generador de observaciones por Aceptación-Rechazo Continuo.

Descripción

`AR.cont` aplica el método de aceptación rechazo para la función

$$f(x) = 20x(1-x)^3$$

con la cual también se calcula el promedio de iteraciones que refleja la eficiencia según la proximidad a la constante c. La información se encuentra en un objeto tipo `list()`.

Código

```
> AR.cont<-function(n){
+ M<-numeric(n)
```

```
+ I<-numeric(n)
+ c<-135/64
+ for(i in 1:n){
+   it<-1
+   ac<-0
+   while(ac==0){
+     U<-runif(2)
+     if(U[2]<=256*U[1]*((1-U[1])^3)/27){ac<-1}else{it<-it+1}
+   }
+   M[i]<-U[1]
+   I[i]<-it
+ }
+ prom<-sum(I)/n
+ info<-list(muestra=M,promedio=prom,constante=c)
+ }
```

Argumentos

- n, número de observaciones (obtenibles llamando con $muestra).

B.2.6. Generador de observaciones por Aceptación-Rechazo Discreto.

Descripción

`AR.dis` aplica el método de aceptación rechazo para la densidad discreta

$$p = (0.2, 0.15, 0.35, 0.2, 0.1)$$

con la cual también se calcula el promedio de iteraciones que refleja la eficiencia según la proximidad a la constante c. La información se encuentra en un objeto tipo `list()`, se incluye también la función de densidad para complementar el método.

Código

```
> f.dis<-function(x){
+   y<-0
+   if(x==1){y<-.2}
+   if(x==2){y<-.15}
+   if(x==3){y<-.35}
+   if(x==4){y<-.2}
+   if(x==5){y<-.1}
+   return(y)
+ }
> AR.dis<-function(n){
+ M<-numeric(n)
+ I<-numeric(n)
```

```
+ c<-.35/.2
+ for(i in 1:n){
+   it<-1
+   ac<-0
+   while(ac==0){
+   U<-runif(2)
+   Y<-floor(5*U[1])+1
+   if(U[2]<=5*f.dis(Y)/c){ac<-1}else{it<-it+1}
+   }
+   M[i]<-Y
+   I[i]<-it
+ }
+ prom<-sum(I)/n
+ info<-list(muestra=M,promedio=prom,constante=c)
+ }
```

Argumentos

- n, número de observaciones (obtenibles llamando con $muestra).

B.2.7. Generador de observaciones Exponenciales por Cociente de Uniformes.

Descripción

`G.exp.counif` aplica el método de cociente de uniformes desarrollado anteriormente para la distribución exponencial dos observaciones uniformes por cada exponencial requerida. Se debe tener consideración que, por la naturaleza de los cálculo realizados, el error de cálculo en la máquina nos puede generar observaciones con valor muy alto.

Código

```
> G.exp.counif<-function(n,lambda=1){
+   M<-numeric(n)
+   for(i in 1:n){
+   u<-runif(1,0,sqrt(lambda))
+   v<-runif(1,0,-(u/lambda)*(2*log(u)-log(lambda)))
+   M[i]<-v/u
+   }
+   return(M)
+ }
```

Argumentos

- n, número de observaciones.

- lambda, valor real positivo del parámetro λ (por default 1).

B.2.8. Generador de Normales truncadas por Marsaglia.

Descripción

`Normal.Trun` utiliza el método de aceptación-rechazo de Marsaglia para variables aleatorias Normales truncadas en un semirayo en los positivos.

Código

```
> Normal.Trun<-function(n,a){
+   M<-1:n
+   for(i in 1:n){
+     r=0
+     while(r==0){
+     u<-runif(1)
+     v<-runif(1)
+     if( v<a*( a^2 - 2*log(u))^(-.5)  ) {
+       x<-(a^2 - 2*log(u))^(.5)
+       r=1
+     }
+     }
+     M[i]=x
+   }
+   return(M)
+ }
```

Argumentos

- n, número de observaciones.
- a un número positivo tal que la variable Normal es truncada en el intervalo (a, ∞).

B.2.9. Generador de observaciones Normales Mixtas.

Descripción

`G.MixNorm` aplica el método de la Composición para generar variables aleatorias normales mixtas, escogiendo a través de la ponderación una de las componentes para simularla.

Código

```
> G.MixNorm<-function(n,P,Mu,Sig){
+   M<-numeric(n)
+   mix<-length(P)
+   if(sum(P)!=1)stop("La ponderacion debe sumar uno")
+   if(length(Mu)!=mix | length(Sig)!=mix){
+     stop("Los parametros estan incompletos")
+   }
+   for(i in 1:n){
+     Y<-rTInv(1,P)+1
+     M[i]<-rnorm(1,Mu[Y],Sig[Y])
+   }
+   return(M)
+ }
```

Argumentos

- n, número de observaciones a simular.
- P, vector que representa la ponderación para cada i-ésima densidad.
- Mu, vector que representa la media de la i-ésima densidad.
- Sig, vector que representa la desviación estándar de la i-ésima densidad.

B.3. Generación de Vectores Aleatorios

B.3.1. Generador de observaciones Multinomiales.

Descripción

`G.MNom` aplica el método de Transformada Inversa similar a la utilizable para la distribución binomial para simular vectores de variables aleatorias Multinomiales.

Código

```
> G.MNom<-function(M,D,n=1){
+   if(sum(D)!=1)stop("El vector de densidades debe sumar 1")
+   if(length(D)>M)stop("El numero de objetos debe ser mayor
+                        o igual que el numero de  posiciones")
+   muestra<-matrix(0,n,length(D))
+   for(i in 1:n){
+     for(j in 1:M){
+       F<-0
+       c<-1
```

```
+       U<-runif(1)
+       while(F<U){
+         F<-F+D[c]
+         if(F<U){c<-c+1}
+       }
+       muestra[i,c]<-muestra[i,c]+1
+     }
+   }
+   return(muestra)
+ }
```

Argumentos

- n, número de observaciones (vectores) a simular (por default n=1).
- D, vector de densidades con la probabilidad de asignar un objeto en cada casilla.
- M, cantidad de objetos a colocar en las casillas bajo D.

B.3.2. Cópula de Cuadras-Auge.

Descripción

`Cuadras.Auge` utiliza el método de derivar directamente la cópula para obtener la distribución condicional a una variable, y el método de la Transformada Inversa para simular vectores provenientes de la cópula Cuadras-Auge.

Código

```
> Cuadras.Auge<-function(n,a){
+   x<-1:(2*n)
+   M=matrix(x,nrow=2,ncol=n)
+
+   for(i in 1:n){
+
+   v<-runif(1)
+   z<-runif(1)
+   if(z<(1-a)*v^(1-a)) {
+     u<-z/(1-a)*v^a
+   } else {
+     if(z<v^(1-a)) {
+       u<-v
+     } else {
+       u<-z^(1/(1-a))
+     }
+   }
```

```
+   M[1,i]=u
+   M[2,i]=v
+   }
+   return(M)
+ }
```

Argumentos

- n, número de observaciones (vectores) a simular.
- a, el par'ametro de dependencia de la cópula.

B.3.3. Tiempo de llegada (Laberinto).

Descripción

`Lab.TLlegada`, para el problema del Laberinto del ejemplo A.2.1, realiza las trayectorias en las que se regresa al estado de partida para realizar el promedio del tiempo de regreso.

Código

```
> Lab.TLegada<-function(n,l,Lab){
+ T<-numeric(n)
+ for(i in 1:n){
+   visita<-0
+   pasos<-0
+   e<-l
+   while(visita==0){
+     #Simulamos un paso sobre la cadena partiendo de e
+     e<-rmarkovchain(1,Lab,e)
+     pasos<-pasos+1
+     if(e==l){visita<-1;T[i]<-pasos}
+   }
+ }
+ sum(T)/n
+ }
```

Argumentos

- n, número de trayectorias a simular.
- l, estado inicial de las trayectorias, debe ser el nombre del estado.
- Lab, es el objeto `markovchain` de la cadena del Laberinto, que debe estar declarado antes.

B.3.4. Probabilidad de transición (Laberinto).

Descripción

`Lab.Pijk`, para el problema del Laberinto del ejemplo A.2.1, realiza las trayectorias para encontrar los casos en que se llega a j desde i en k pasos.

Código

```
> Lab.Pijk<-function(i,j,k,n,Lab){
+ exitos<-numeric(n)
+ for(l in 1:n){
+   pasos<-rmarkovchain(k,Lab,i)
+   if(pasos[k]==j){exitos[l]<-1}
+ }
+ sum(exitos)/n
+ }
```

Argumentos

- i, nombre del estado inicial.
- j, nombre del estado al que se busca llegar
- k, número de transiciones.
- n, número de trayectorias.
- Lab, es el objeto `markovchain` de la cadena del Laberinto, que debe estar declarado antes.

B.3.5. Recurrencias antes de llegar (Laberinto).

Descripción

`Lab.jpori`, para el problema del Laberinto del ejemplo A.2.1, realiza trayectorias en las que se llega a j, buscando registrar las veces que primero se pasó por i, buscando encontrar el promedio de tales recurrencias.

Código

```
> Lab.jpori<-function(i,j,n,Lab){
+   reg_i<-numeric(n)
+   for(k in 1:n){
+   e<-i
+   pasos<-0
```

```
+   llegada_j<-0
+   while(llegada_j==0){
+     e<-rmarkovchain(1,Lab,e)
+     pasos<-pasos+1
+     if(e==i){reg_i[k]<-reg_i[k]+1}
+     if(e==j){llegada_j<-1}
+   }
+ }
+   return(sum(reg_i)/n)
+ }
```

Argumentos

- i, nombre del estado inicial.
- j, nombre del estado al que se busca llegar.
- n, número de trayectorias.
- Lab, es el objeto `markovchain` de la cadena del Laberinto, que debe estar declarado antes.

B.3.6. Cadena del problema de la lluvia.

Descripción

`Par.markov`, para el problema de la lluvia del ejemplo 4.1.3, genera el objeto `markovchain` para cualquier número de paraguas y probabilidad de lluvia.

Código

```
> Par.markov<-function(r,p){
+   estados<-0:r
+   M<-matrix(0,r+1,r+1)
+   M[1,r+1]<-1
+   for(i in 2:(r+1)){
+     M[i,r-i+3]<-p
+     M[i,r-i+2]<-1-p
+   }
+   return(new("markovchain",states=as.character(estados),
+              transitionMatrix=M))
+ }
```

Argumentos

- r, número de paraguas.

- p, probabilidad de lluvia en la ma?ana y tarde.

B.3.7. Cadena de Ehrenfest.

Descripción

`Ehr.markov`, para la Cadena de Ehrenfest del ejemplo A.2.5, genera el objeto `markovchain` para cualquier número de bolas.

Código

```
> Ehr.markov<-function(N){
+   library(markovchain)
+   P<-matrix(0,N+1,N+1)
+   P[1,2]<-1
+   P[N+1,N]<-1
+   for(i in 2:N){
+     P[i,i-1]<-(i-1)/N
+     P[i,i+1]<-(N-i+1)/N
+   }
+   estados<-0:N
+   cad<-new("markovchain",transitionMatrix=P,
+            states=as.character(estados))
+   return(cad)
+ }
```

Argumentos

- N, número de bolas.

B.3.8. Trayectoria del Proceso Poisson.

Descripción

`PPoisson.S`, implementa el algoritmo 33 para simular un Proceso de Poisson a un tiempo t.

Código

```
> PPoisson.S<-function(lambda,T){
+     t<-0
+     P<-0
+     while(t<T){
```

```
+       t<-t-log(runif(1))/lambda
+       if(t<T){P<-P+1}
+     }
+   return(P)
+ }
```

Argumentos

- lambda, es un real positivo conocido como la intensidad del Proceso Poisson.
- T, una real positivo que determina el tiempo del Proceso Poisson.

B.3.9. Probabilidad para el Processo Poisson.

Descripción

`PPoisson.P`, implementa el algoritmo 33 para simular trayectorias a un tiempo límite de manera que la frecuencia de los casos congruentes a los parámetros estiman la probabilidad del evento.

Código

```
> PPoisson.P<-function(lambda,T,x,n){
+   exitos<-numeric(n)
+   for(i in 1:n){
+     t<-0
+     P<-0
+     while(t<T){
+       t<-t-log(runif(1))/lambda
+       if(t<T){P<-P+1}
+     }
+     if(P==x){exitos[i]<-1}
+   }
+   return(sum(exitos)/n)
+ }
```

Argumentos

- lambda, es un real positivo conocido como la intensidad del Proceso Poisson.
- T, es un real positivo que determina el tiempo del Proceso Poisson.
- x, es un entero no negativo representando el valor para el cual se busca la probabilidad de que la variable tome.
- n, es el número de trayectorias a simular en la estimación.

B.3.10. Esperanza del Proceso Poisson.

Descripción

`PPoisson.E`, implementa el algoritmo 33 para simular trayectorias a un tiempo límite de manera que el promedio estime el valor esperado del Proceso Poisson a un tiempo t.

Código

```
> PPoisson.E<-function(lambda,T,n){
+   muestra<-numeric(n)
+   for(i in 1:n){
+     t<-0
+     P<-0
+     while(t<T){
+       t<-t-log(runif(1))/lambda
+       if(t<T){P<-P+1}
+     }
+     muestra[i]<-P
+   }
+   return(sum(muestra)/n)
+ }
```

Argumentos

- lambda, es un real positivo conocido como la intensidad del Proceso Poisson.
- T, es un real positivo que determina el tiempo del Proceso Poisson.
- n, es el número de trayectorias a simular en la estimación.

B.3.11. Tiempo de Espera del Processo Poisson.

Descripción

`PPoisson.TE`, implementa el algoritmo 33 para simular trayectorias a un tiempo límite rescatando sólo el tiempo de espera para el llegar el valor x de manera que el promedio estime el valor esperado de S_x.

Código

```
> PPoisson.TE<-function(lambda,x,n){
+   esperas<-numeric(n)
+   for(i in 1:n){
+     t<-0
+     P<-0
```

```
+     while(P<x){
+       t<-t-log(runif(1))/lambda
+       P<-P+1
+     }
+     esperas[i]<-t
+   }
+   return(sum(esperas)/n)
+ }
```

Argumentos

- lambda, es un real positivo conocido como la intensidad del Proceso Poisson.
- x, es un natural que representa el valor del Proceso Poisson a alcanzar.
- n, es el número de trayectorias a simular en la estimación.

B.3.12. Processo Poisson no Homogéneo

Descripción

`PPoissonNH.S`, implementa el algoritmo 34 para simular un Proceso Poisson no Homogéneo con una función de intensidad $\lambda(t)$ y λ adecuadas, obteniendo los tiempos de salto.

Código

```
> PPoissonNH.S<-function(T,inten,lambda){
+   Inten<-function(t)eval(inten)
+   t<-0
+   P<-0
+   s<-numeric()
+   while(t<T){
+     t<-t-log(runif(1))/lambda
+     U<-runif(1)
+     if(U<Inten(t)/lambda){P<-P+1;s<-c(s,t)}
+   }
+   return(P)
+ }
```

Argumentos

- T, un real positivo límite de tiempo para el proceso.
- inten, es un objeto `expression` que representa la función de intensidad del Proceso Poisson no Homogéneo, debe estar escrita para la variable t y estar definida en $\mathbb{R}^+ \cup \{0\} \to \mathbb{R}^+ \cup \{0\}$.

- lambda, es un real positivo que necesariamente sea mayor o igual que $\lambda(t)$ para $t \leq T$.

B.3.13. Processo Poisson Compuesto

Descripción

`PPoiCE.S`, implementa el algoritmo **??** para simular un Proceso Poisson Compuesto con saltos con distribución exponencial.

Código

```
> PPoiCE.S<-function(lppc,lexp,T){
+   t<-0
+   P<-0
+   while(t<T){
+     t<-t-log(runif(1))/lppc
+     if(t<T){P<-P-log(runif(1)/lexp)}
+   }
+   return(P)
+ }
```

Argumentos

- T, un real positivo límite de tiempo para el proceso.
- lppc, un real positivo representado la frecuencia de los saltos.
- lexp, es un real positivo parámetro de los saltos con distribución exponencial.

B.3.14. Processo de Saltos de Markov

Descripción

`PSM.S`, implementa el algoritmo 38 para simular un Proceso de Saltos de Markov, obteniendo tiempos de salto y estados, a partir de una matriz de intensidades y una frontera de tiempo.

Código

```
> PSM.S<-function(T,Q,D){
+   if(sum(D)!=1)stop("El vector de densidades debe sumar 1")
+   N<-nrow(Q)
+   t<-0
+   tiempos<-numeric(1)
```

```
+   Sal<-numeric(1)
+   Sal[1]<-rTInv(1,D)+1
+   i<-Sal[1]
+   tiempos[1]<-t
+   while(t<T){
+     t<-t+log(runif(1))/Q[i,i]
+     if(t<T){
+       tiempos<-c(tiempos,t)
+       U<-runif(1)
+       F<-0
+       j<-0
+       while(F<U){
+         j<-j+1
+         if(j!=i){F<-F-Q[i,j]/Q[i,i]}
+       }
+       i<-j
+       Sal<-c(Sal,i)
+     }
+   }
+   return(list(Tiempos=tiempos,Estados=Sal))
+ }
```

Argumentos

- T, un real positivo límite de tiempo para el proceso.
- Q, una matriz con valores congruentes con la matriz de intensidades de un PSM (reales negativos en la diagonal y la suma de las demás entradas del renglón igual al valor positivo del de la diagonal).
- D, vector de la distribución inicial, las entradas deben sumar uno.

B.3.15. Estimación de la función Gamma por Monte-Carlo

Descripción

`EMC.GAMMA`, implementa la estimación del valor de la función Gamma como la media muestral, de la función adecuada, de variables exponenciales. También incluye la varianza y desviación estándar del estimador

Código

```
> EMC.GAMMA<-function(n,alfa){
+  U<-runif(n)
+  muestra<-(-log(U))
+  suma1<-0
+  suma2<-0
```

```
+  for(i in 1:n){
+   suma1<-suma1+muestra[i]^(alfa-1)
+   suma2<-suma2+muestra[i]^((alfa-1)*2)
+  }
+  Esp<-suma1/n
+  Var<-((suma2/n)-Esp^2)/n
+  de<-sqrt(Var)
+  return(c(Esp,Var,de))
+ }
```

- n, tama?o de la muestra sobre la cual se hace la estimación.
- alfa, el argumento de la función Gamma para estimar.

B.3.16. Estimación por Máxima Verosimilitud en PSM

Descripción

`PSM.EMV`, implementa la extracción de las estadísticas suficientes de la trayectoria observada de un Proceso de Saltos de Markov para obtener una estimación del Generador Infinitesimal.

Código

```
> PSM.EMV<-function(E,TR){
+   J<-length(TR$Estados)
+   TPer<-numeric(E)
+   N<-matrix(0,E,E)
+   Q<-matrix(0,E,E)
+   for(i in 2:J){
+ TPer[TR$Estados[i-1]]<-TPer[TR$Estados[i-1]]+
+ TR$Tiempos[i]-TR$Tiempos[i-1]
+ N[TR$Estados[i-1],TR$Estados[i]]<-N[TR$Estados[i-1],TR$Estados[i]]+1
+   }
+   for(i in 1:E){
+     for(j in 1:E){
+       if(i!=j){
+         Q[i,j]<-N[i,j]/TPer[i]
+       }
+     }
+     Q[i,i]<--sum(Q[i,])
+   }
+   return(Q)
+ }
```

Argumentos

- E, un entero representando a la cardinalidad del espacio de estados E.
- TR, un objeto `list` que represente observaciones de la trayectoria que requiere estar conformado por tiempos de salto en un objeto llamado Tiempos y otro de transiciones llamado Estados (por simplicidad las entradas deben ser numéricas, i.e. necesariamente los estados son 1,2,...,E).

B.3.17. Inferencia directa sobre el Proceso de Ornstein-Uhlenbeck

Descripción

`Inf.OU.Dir`, implementa la forma cerrada de los estimadores de un Proceso de Ornstein-Uhlenbeck en el caso de incrementos de tiempo constantes.

Código

```
> Inf.OU.Dir<-function(X,Del){
+   n<-length(X)
+   alfa<--log(sum(X[1:(n-1)]*X[2:n])/sum(X[1:(n-1)]^2))/Del
+   sigma<-2*alfa*sum(X[2:n]-X[1:(n-1)]*
+           exp(-Del*alfa))/((n-1)*(1-exp(-2*Del*alfa)))
+   return(c(alfa,sigma))
+ }
```

Argumentos

- X, un vector con la trayectoria observada.
- Δ, el incremento de tiempo constante de cada observación.

B.3.18. Inferencia para Proceso de Ornstein-Uhlenbeck usando `mle()`

Descripción

`Inf.OU.Mle`, implementa una rutina que utilice el valor negativo de la log-verosimilitud de la muestra para parámetros dados, y llama a la función `mle` para hacer la maximización.

Código

```
> Inf.OU.Mle<-function(X,Del){
+   OU.lik<-function(a,s){
+     n<-length(X)
+ -sum(dcOU(x=X[2:n],Dt=Del,x0=X[1:(n-1)],theta=c(0,a,s),log=TRUE))
+   }
+   est<-mle(OU.lik,start=list(a=.1,s=.1),method="L-BFGS-B",
+            lower=c(.00001,.00001,.00001),upper=c(10,10,10))
```

```
+   return(as.numeric(coef(est)))
+ }
```

Argumentos

- X, un vector con la trayectoria observada.
- Δ, el incremento de tiempo constante de cada observación.

B.3.19. Inferencia para Proceso de Ornstein-Uhlenbeck usando `mle` bajo el método de Euler

Descripción

`Inf.OU.Euler`, implementa una rutina que utilice el valor negativo de la log-verosimilitud de la muestra para parámetros dados suponiendo las diferencias como variables aleatorias gaussianas, y llama a la función `mle` para hacer la maximización.

Código

```
> Inf.OU.Euler<-function(X,Del){
+   n<-length(X)
+   Euler.lik<-function(alfa){
+     -sum((X[2:n]-X[1:(n-1)])*(-alfa*X[1:(n-1)])-Del*.5*(alfa*X[1:(n-1)])^2)
+   }
+   est<-mle(Euler.lik,start=list(alfa=1),method="L-BFGS-B",
+            lower=c(.00001,.00001,.00001),upper=c(10,10,10))
+   AL<-as.numeric(coef(est))
+   SIG<-sqrt(sum((X[2:n] - X[1:(n-1)])^2/(Del*(n-1))))
+   return(c(AL,SIG))
+ }
```

Argumentos

- X, un vector con la trayectoria observada.
- Δ, un vector con los tiempos transición entre observaciones, que deben ser cortos para cumplir los supuestos de normalidad.

Bibliografía

[1] Buffon, Georges Louis and Comte de, LeClerk (1744). L'Histoire naturelle, générale et particulière Buffon.

[2] Caballero, Rivero, Uribe y Velarde. Cadenas de Markov. Un enfoque elemental. Number 29 in Textos, Nivel Medio. SMM, 2004.)

[3] Conover, W. J. Practical Nonparametric Statistics. Wiley and Sons, second edition, 1980.

[4] Durrent, R. (1999) Essential of Stochastic Processes. Springer.

[5] Eckhardt, R.(1987). Stan Ulam, John von Neumann, andthe Monte Carlo method. Los Alamos Sci.15Special Issue pp. 131-137.

[6] Halton, J. (1970) A retrospective and prospective survey of Monte Carlo method. SIAM Rev. 12, page 1-63.

[7] Harvey, C. R. . Time-varying conditional covariances in tests of asset pricing models. Journal of Financial Economics, 24, page 289-317., 1989.

[8] Hull T.E.,Dobell A.R. Random Number Generators. SIAM Rev. Vol 4. No. 3, page 230-254, 1962.

[9] Fishman, G. (2006).A First Course in Monte Carlo. Belmont, CA : Thomson Brooks.

[10] Gilks, W.R., Richardson, S. y Spiegelhalter, D.J. (1996). Markov Chain Monte Carlo in Practice. Chapman and Hall.

[11] Iacus, Stefano M. Simulation and Inference for Stochastic Differential Equations. Springer, 2008.

[12] Karlin, Samuel y Taylor (1975). A First Course in Stochastics Processes. Academia Press, New York-London, second edition.

[13] Kinderman, A.J., Monahan, J.F. (1977). Computer generation of random variables using the ratio of uniform deviates. ACM Trans. Math. Software 3, pp. 257-260.

[14] Laguna, M. and Marklund, J. (2005). Business process modeling, simulation and design. USA: Prentice Hall.

[15] L'Ecuyer, Pierre Efficient and Portable Combined Number Generators. Research Contributions, Montreal, Canada, 2010. http://www.iro.umontreal.ca/ lecuyer/myftp/papers/cacm88.pdf.

[16] Lee SCK y Lin XS (2010). Modeling and Evaluating Insurance Losses Via Mixtures of Erlang Distributions.N Am Actuar J 14(1). pp. 107-130

[17] Lehmer. D.H. (1951). Mathematical methods in large-scale computing units. Annu. Comput. Lab. Harvard Univ. 26. pp. 141-146.

[18] Lewis, P.A., Goodman, A.S., and Miller, J.M. (1969). A pseudo-random number generator for the Syslem/360. IBM Syst. J 8, 2. pp. 136-146.

[19] Marsaglia, George, (1964). Generating a variable from the tail of the normal distribution, Technometrics , 6, pp. 101-102.

[20] Marshall, A.W. and Olkin, I. (1988) Families of Multivariate Distributions. Journal of the American Statistical Association, 83, pp. 834-841.

[21] Park S. K. and Miller K. W. (1988). Random Number Gener-ators: Good Ones Are Hard to Find, Communicationsof the ACM.

[22] Ripley, B. D. (1987) Stochastic Simulation. Wiley.

[23] Ross, S. M. y Palmas Velasco, O. A. (1999). Simulation. México: Prentice- Hall.

[24] Ross, S.M. (1996) Stochastic Processes. John Wiley.

[25] Rubinstein y Kroese Simulation and the Monte Carlo Method. Wiley Series in Probability and Statistics, second edition, 2007.

[26] Shannon, R.E. (1975). Systems Simulation. The Art and Science, Prentice-Hall.

[27] Shubik, M. (1960). Simulation of the Industry and the Pirm. Amer Eco Review, L, 5. p. 909.

[28] West Churchman C (1963). On analysis of the concept of Simulation . Symposium on Simulation Models, de Hoggatt y Balderston, Cincinnati.

[29] Stephens, M. A. (1974). EDF Statistics for Goodness of Fit and Some Comparisons, Journal of the American Statistical Association, 69, pp. 730-737.

[30] Wichmann. B.A., and Hill. I.D. (1982). An efficient and portable pseudo- random number generator. Appl. Stat. 31. pp. 188-190.