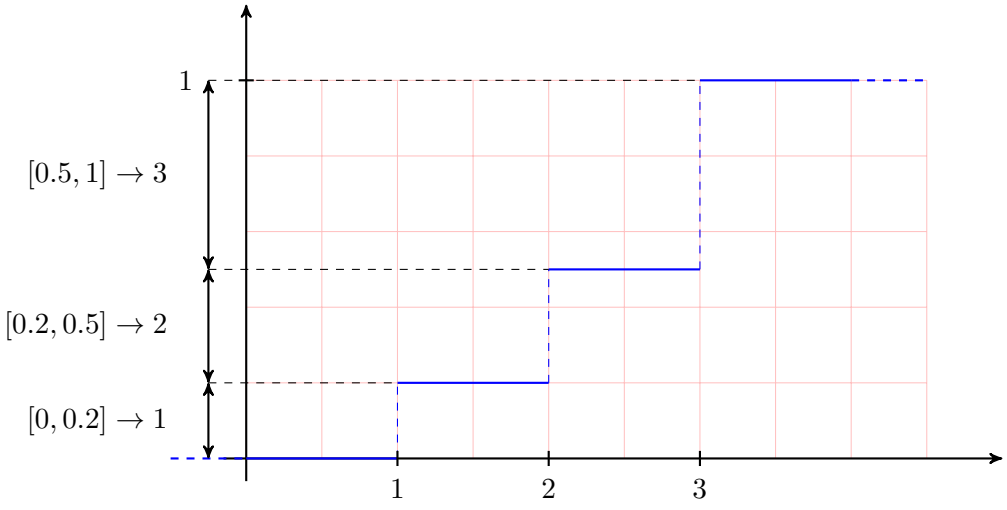
Aureli Alabert

Simulació

teoria i exercicis



1

Introducció

1.1. Què és la simulació

En una primera aproximació, la *simulació* és una tècnica per calcular probabilitats complicades.

Exemple 1.1.1. La cua del supermercat:

Els usuaris arriben a la cua en moments aleatoris. El temps que passa l'usuari a la caixa també és aleatori. Què podem dir del temps que estarà un usuari qualsevol a la cua (*delay*)? □

Les cues són situacions típiques en què calcular probabilitats és difícil.

Exemple 1.1.2. Centre sanitari d'urgències:

Pot haver un sistema de prioritats (no necessàriament se serveix en l'ordre en què s'arriba), potser s'ha de passar per diferents cues abans de marxar (recepció, avaluació, visita mèdica, cures,...). Quin és el temps total que passa un usuari en el sistema? Quants usuaris poden arribar a estar en espera en cada etapa? □

Tenim dues possibilitats:

1. Observar directament en el sistema real les quantitats que ens interessen (usualment, llarg i car).
2. Simular amb ordinador el sistema real, i programar el càlcul de les quantitats que interessen (usualment, més curt i barat).

Si el sistema real no existeix encara (només és un projecte), o volem modificar un sistema existent sense tocar-lo (*what if...?*), només la segona opció és possible.

La simulació ens pot servir per saber com està anant un sistema per interès purament descriptiu, o bé per detectar mals funcionaments i oportunitats de millora, o bé per comparar sistemes alternatius i optimitzar.

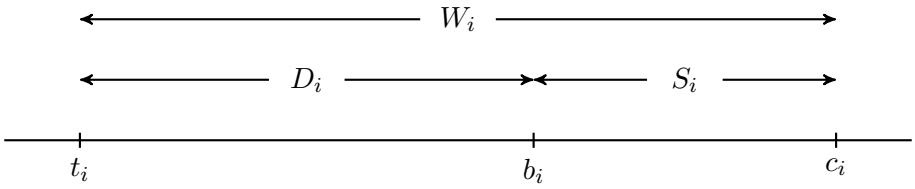
La simulació necessita dades (*input data*). Els usuaris que arriben "a l'atzar" a un centre sanitari, amb quin ritme ho fan? És uniforme en el temps aquest ritme o hi ha hores punta? Venen d'un en un o en grup? Si han d'esperar molt, quin tant per cent renuncia a quedar-se?

La simulació parteix d'aquestes dades d'entrada, presumiblement senzilles de modelitzar, sobre elements petits del sistema, i produeix el resultat del sistema complet (*output*).

1.2. Exemple: Cua simple

Les cues són a la vegada situacions reals que volem modelitzar, i models d'altres situacions que es poden pensar com a cues, encara que no ho siguin. Anem a veure una cua senzilla (tipus caixa de supermercat):

Una cua és un sistema que evoluciona en el temps. Suposarem que el temps comença a l'instant $t_0 = 0$. Suposarem que els clients i arriben d'un en un, en instants t_i. Denotarem per $A_i = t_i - t_{i-1}$ el temps entre l'arribada del client[1] $(i-1)$-èsim i el client i-èsim. Sigui S_i el temps que la caixera[2] passa servint al i-èsim client. Sigui D_i el *delay* del client i-èsim (temps que passa en cua abans de començar a ser servit). Si anomenem c_i l'instant de temps en què l'usuari completa el procés i se'n va, tenim $c_i = t_i + D_i + S_i$.



A la figura, hem escrit a més el temps total que triga un client a completar el procés (*wait*, $W_i := D_i + S_i$), que és el delay més el temps de servei, i l'instant b_i en què comença a ser servit.

> [Mireu la Figura 1.2, pàgina 9 de Law, que representa en l'eix temporal els esdeveniments que van succeint a la cua: A l'instant t_1 arriba el primer client, i com que no hi ha ningú a la cua, comença a ser servit immediatament. Quan arriba el segon client, el primer encara està a la caixa, i per tant aquest segon client s'ha d'esperar. Patirà un delay $c_1 - t_2$. El tercer client també tindrà un delay, perquè arriba abans de la compleció del servei del segon client. I així successivament. Els instants e_j de la figura són aquells en què "succeeix alguna cosa". Es pot pensar que entre instants e_j el sistema està aturat: no hi ha cap canvi en *l'estat del sistema*.]

Podem dir alguna cosa interessant sobre aquesta cua, sense més dades? Imagineu per un moment que la quantitat de arribades a la cua per unitat de temps (*arrival rate*) fos constant igual a λ. Si a la sortida preguntem a cada client quant temps ha estat en el sistema (cua + caixa), podem estimar el temps mitjà $\mathrm{E}[W]$ de tots els usuaris. I si N és la quantitat d'usuaris en el sistema en un moment donat, podem dir que $\mathrm{E}[N] = \lambda\,\mathrm{E}[W]$. Aquesta fórmula ve a ser anàloga a "espai = velocitat × temps".

Per una banda, aquesta fórmula és molt general: No hem usat res sobre el temps de servei, i també val si λ no és una constant, sinó el ritme mitjà d'arribades. Però al mateix temps, hi ha moltes hipòtesis implícites.

[1] masculí genèric
[2] femení genèric

Exercici 1.2.1. Una escola de primària té sis cursos (de primer fins a sisè). Cada any entren a primer 25 nous estudiants, que van progressant cada any pels cursos successius fins a completar el sisè curs. Quina és la quantitat total d'estudiants a l'escola?

Naturalment, es pot pensar d'una manera estàtica i contestar simplement "6 cursos $\times$ 25 estudiants/curs = 150 estudiants". Però intenteu pensar-ho com una cua dinàmica i amb la fórmula $\mathrm{E}[N] = \lambda\,\mathrm{E}[W]$. □

L'Exercici 1.2.1 il·lustra dues coses. En primer lloc, es poden pensar com a cues situacions que no ho semblen. Les cues són per tant, apart de situacions reals interessants, un model útil per a altres problemes.

En segon lloc, ens permet pensar en la gran quantitat d'hipòtesis que hem usat, i que simplifiquen la situació real: I si hi ha estudiants que arriben o marxen en algun grau intermedi? I si hi ha estudiants que repeteixen o se salten un curs? I si la quantitat d'estudiants que entren a primer no és sempre 25, sinó una quantitat aleatòria? I si la quantitat d'estudiants que entren no és constant en el temps, sinó que està lligada a la natalitat, la immigració/emigració, la situació econòmica, etc?

Tornant a la cua del supermercat, és clar que el primer client que arriba no té delay, perquè no troba cua. I segurament hi ha altres clients que fan molta cua. Per tant, les variable aleatòries W_i no tenen totes la mateixa llei. I tampoc són independents, perquè si el client i-èsim ha hagut de esperar molta estona, segurament el $(i+1)$-èsim també en farà. És lògic aleshores preguntar-se quin és el sentit de l'esperança $\mathrm{E}[W]$.

1.3. Input i output

Mirem primer on volem arribar (output), i després mirarem quines dades necessitem (input). Seguim en el cas de la cua simple tipus caixa de supermercat. Suposem que el sistema funciona durant un interval de temps acotat $[0, T]$, per exemple $T = 12$ hores. Ens agradaria saber:

- El temps D que passen a la cua (delay) els clients.
- El temps total W (wait) que els clients passen en tot el sistema (cua + servei).
- La longitud de la cua i/o quantitat d'usuaris N en el sistema complet.
- Quina és la proporció de clients que no han de fer cua?
- Quina és la proporció de temps que no hi ha ningú a la caixa?

i potser altres coses.

Ara bé, els delays D_i no són independents idènticament distribuïts. No podem parlar de la "llei" del delay. Però sí que té sentit calcular

$$d = \frac{1}{n}\sum_{i=1}^{n} D_i$$

on n és la quantitat total de clients atesos. Aquest és el "delay mitjà" dels clients un dia determinat. Al dia següent, potser observem un altre delay mitjà diferent. Si s'han mantingut les mateixes condicions els dos dies, les dues variables d sí que seran independents i idènticament distribuïdes. D'aquesta manera podem obtenir una mostra de variables en el sentit tradicional estadístic, i aplicar qualsevol eina d'inferència que vulguem.

El mateix es pot aplicar al wait W, o a questions sobre el temps de servei S: Quina és la longitud dels intervals de temps en què la caixera no està atenent ningú? O al contrari, ocupada sense parar?

Per a la quantitat d'usuaris a la cua o en el sistema complet, observem que aquest és un valor que depèn de l'instant de temps en què el mirem. Haurem de fer una "mitjana en el temps". Anomenarem $Q(t)$ la quantitat de clients a la cua a l'instant t.

> [Mireu la Figura 1.5, pàgina 15 de Law. La gràfica representa la quantitat de clients a la cua en cada instant de temps. Naturalment, és una funció constant a trossos, que només pren valors naturals. A l'instant e_1 arriba el primer client, però entra directament a la caixa, i per tant la cua segueix buida. Als instants e_2 i e_3 arriben dos clients més abans que el primer client hagi acabat. Per tant, la cua augmenta a 1 i després a 2. Quan marxa el primer client torna a baixar a 1.
>
> Seguiu la resta de la gràfica, imaginant que està passant quan succeeix cada esdeveniment e_j. Fixeu-vos que hi ha un interval pel mig en què no hi ha ningú en tot el sistema, i altres en què no hi ha ningú a la cua, però sí un client a la caixa.]

La mitjana en el temps és

$$q = \frac{1}{T} \int_0^T Q(t)\, dt$$

o sigui la suma de les àrees dels rectangles de la figura, dividida per la longitud de l'interval de temps considerat.

De la mateixa manera podem pensar en la proporció de temps que la caixera ha estat ocupada o ociosa. En cada instant de temps $B(t) = 1$ (*busy*, ocupada) o $B(t) = 0$ (*idle*, ociosa). Tenim la mitjana en el temps

$$b = \frac{1}{T} \int_0^T B(t)\, dt$$

> [Vegeu la Figura 1.6, pàgina 17 de Law, que té correspondència amb la de la pàgina 15. Els intervals de temps en què la caixa ha estat ocupada tenen la funció $B(t)$ igual a 1, i els intervals ociosos igual a zero. Si hi haguessin dues caixes alimentant-se de la mateixa cua, la funció podria pujar fins a 2, i el resultat de la integral es podria interpretar com l'ocupació mitjana (*utilization*) del servei, format per dues caixes.]

Poden haver-hi altres outputs d'interès. Per exemple, potser ens interessi la longitud màxima a què arriba la cua, en relació a preveure l'espai disponible per a què la cua es pugui formar físicament. En tal cas, ens interessa la variable aleatòria

$$m = \max_{[0,T]} Q(t)$$

Observant en dies successius, tindrem una mostra 'iid' de la variable m i podem fer inferència estadística sobre ella. O, per exemple, ens pot interessar la probabilitat d'un cert esdeveniment, com ara si la cua excedeix o no una longitud fixada, o si els períodes d'ocupació contínua de la caixa excedeixen un cert límit.

Com a observació marginal, pot ser que calgui servir els clients que falten quan arriba el temps límit T. O, en alguns casos, la quantitat d'usuaris a servir és fixat, sense límit de temps. Tot això són variants que no posen gaire més dificultat.

Naturalment, la gràcia de la simulació és no haver de fer aquestes observacions diàries de les quantitats que ens interessen, sinó automatitzar-ho tot amb l'ordinador. Per tal que això sigui possible, calen unes dades d'entrada, que bàsicament tenen a veure amb el ritme d'arribades dels usuaris, i el temps que passen en el servei (caixa).

Una hipòtesi habitual és suposar que la quantitat d'usuaris que arriben per unitat de temps segueix una llei de Poisson, amb un cert paràmetre λ, i que el temps de servei segueix una llei Exponencial de paràmetre μ. Els paràmetres concrets λ i μ sí que caldrà deduir-los observant el sistema real, o a partir del coneixement previ de la situació que s'està estudiant.

En aquesta situació tan concreta, i en algunes altres, hi ha càlculs i estudis qualitatius que es poden fer a mà, analíticament. Però no són fàcils, i si compliquem una mica el sistema ja no se sap fer res. Molts resultats teòrics són, a més, límits "in the long run", quan el temps tendeix a infinit, i és delicat saber quan són aplicables i quan no. En canvi, la simulació es pot aplicar de manera més o menys universal. El càlcul exacte de probabilitats, esperances i lleis queda substituït per l'estudi estadístic de la sortida de la simulació.

De tota manera, la hipòtesi d'arribades Poissonianes no és només una conveniència matemàtica. Acabem aquest apartat veient per què és una situació important. Suposem que:

1. Els clients arriben a la cua d'un en un.
2. La probabilitat p que es produeixi una arribada en un interval de temps de longitud h (petita) és proporcional a h. O sigui,

   $$p = \lambda h$$

   per una certa constant de proporcionalitat λ.

   En altres paraules, la probabilitat es duplica si considerem un interval el doble de gran. Aquesta suposició només és vàlida per intervals petits, perquè la probabilitat no pot ser mai més gran que 1. No importa què vulgui dir exactament "petit", perquè després farem un límit quan $h \to 0$.[3]
3. La quantitat d'arribades en intervals de temps disjunts són variables independents. És a dir, no influeix el que hagi passat en un interval de temps sobre el que passi en un interval de temps posterior.

Aquestes hipòtesis modelen raonablement bé la idea que "els usuaris arriben completament a l'atzar".

Fixem un interval de temps $[0, t]$ i el dividim en n subintervals de longitud $h = \frac{t}{n}$. Per n prou gran, en cada subinterval arribarà com a molt una persona (la probabilitat que n'arribin dues o més és infinitesimal i anirà cap a zero quan prenguem límits). Per tant, en $[0, t]$ arribaran com a màxim n usuaris, i la quantitat total d'arribades segueix una llei binomial de paràmetres n i p:

$$\begin{aligned} P\{k \text{ arribades en } [0,t]\} &= \binom{n}{k} \cdot p^k \cdot (1-p)^k \\ &= \binom{n}{k} \cdot \left(\lambda \frac{t}{n}\right)^k \cdot \left(1 - \lambda \frac{t}{n}\right)^{n-k} \\ &= \frac{n!}{k!\,(n-k)!} \cdot \frac{1}{n^k} \cdot (\lambda t)^k \cdot \left(1 - \lambda \frac{t}{n}\right)^n \cdot \left(1 - \lambda \frac{t}{n}\right)^{-k} \end{aligned}$$

Això és una aproximació. És raonable pensar que si fem $n \to \infty$ (equivalentment, la longitud h dels intervals molt petita) obtindrem un bon model per al problema. No és difícil calcular els límits

$$\begin{aligned} \lim_{n\to\infty} \frac{n!}{(n-k)!n^k} &= 1 \\ \lim_{n\to\infty} \left(1 - \lambda \frac{t}{n}\right)^n &= e^{-\lambda t} \\ \lim_{n\to\infty} \left(1 - \lambda \frac{t}{n}\right)^{-k} &= 1 \end{aligned}$$

i per tant obtenim

$$P\{k \text{ arribades en } [0,t]\} = e^{-\lambda t}\, \frac{(\lambda t)^k}{k!}$$

[3] Matemàticament més precís, $p = \lambda h + o(h)$, on $\lim_{h\to 0} \frac{o(h)}{h} = 0$.

i, per unitat de temps, serà

$$P\{k \text{ arribades per unitat de temps}\} = e^{-\lambda}\,\frac{\lambda^k}{k!}$$

Es tracta de la llei de Poisson de paràmetre λ. En aquest context, λ s'anomena *arrival rate*, i és precisament l'esperança d'aquesta llei de probabilitat. Com més freqüent és l'arribada de gent a la cua, més gran és λ.

Exercici 1.3.1. Demostreu els límits anteriors. □

Exercici 1.3.2. Calculeu la probabilitat que la primera arribada a la cua es produeixi més tard que un temps t.
Com que les arribades en intervals disjunts són independents, aquesta probabilitat coincideix amb la probabilitat que el temps entre dues arribades consecutives qualssevol sigui més gran que t. Es dedueix que el temps entre dues arribades consecutives qualssevol segueix una llei exponencial de paràmetre λ. □

Segons l'Exercici 1.3.2, sota les hipòtesis que hem enumerat, l'esperança del temps entre arribades és $1/\lambda$; o sigui, és més petita com més gran és l'*arrival rate*. (La llei de Poisson ens parla de nombre d'arribades per unitat de temps, i la llei exponencial de temps per arribada; lògicament una quantitat és inversa de l'altre.)

La Teoria de Cues és la branca matemàtica que intenta trobar fórmules analítiques (exactes i aproximades) per a situacions d'aquesta mena. En aquest curs no explorarem aquesta teoria.[4]

1.4. Exemple: Inventari

Una empresa que ven un únic producte ha de decidir quantes unitats del producte ha de tenir emmagatzemades durant els propers n mesos (n fixat) per satisfer la demanda que espera tenir. Els temps entre comandes se sap que són variables independents idènticament distribuïdes amb llei exponencial de mitjana 0.1 mesos (en altres paraules, arriben de mitjana 10 comandes al mes, seguint una llei de Poisson).

La mida de les comandes és de una unitat amb probabilitat $\frac{1}{6}$, dues amb probabilitat $\frac{1}{3}$, tres amb probabilitat $\frac{1}{3}$, i quatre amb probabilitat $\frac{1}{6}$.

Al principi de cada mes, l'empresa revisa l'inventari i decideix quina quantitat demanar al seu proveïdor. Si demana z unitats, el cost és $k + 3z$ €, on $k = 32$ si $z > 0$, i $k = 0$ si $z = 0$ (k és un *cost fix* de la comanda). Quan fa la petició al proveïdor, la mercaderia tarda entre 0.5 i 1 mes a arribar, amb distribució uniforme (aquest temps s'anomena *lead time*).

S'utilitza l'estratègia de reposició següent:

$$z = \begin{cases} S - I, & \text{si } I < s \\ 0, & \text{si } I \geq s \end{cases}$$

on I és el nivell d'inventari a principi de mes i s i S són quantitats a decidir.

Quan hi ha una comanda, se satisfà immediatament si hi ha suficient inventari. Si no n'hi ha, se satisfà la quantitat que es pot, i la resta queda en *backlog*, i se satisfarà primer que res quan hi hagi mercaderia.

[4] Si mai hi esteu interessats, recomano el llibre Gross-Shortie-Thompson-Harris, *Fundamentals of Queueing Theory* (5th ed, 2018), que es pot baixar en PDF des de la UAB.

[Vegeu la Figura 1.8, pàgina 49 de Law, d'on està copiat aquest exemple. El nivell d'inventari $I(t)$ evoluciona en el temps segons una funció constant a trossos (línia sòlida) i pot ser negatiu. La funció $I^+(t)$ (línia puntejada) és la part positiva de l'inventari, i $I^-(t)$ (línia discontínua) la part negativa.]

Apart del cost de demanar mercaderia al proveïdor, hi ha el cost de magatzematge (*holding cost*), $h = 1€$ per unitat i mes, i el cost d'escassetat o trencament d'inventari (*shortage cost*, que inclou la possible pèrdua de la venda, i potser del client!), de $b = 5€$ per unitat i mes de backlog.

La mitjana mensual de l'inventari després de n mesos és

$$\bar{I}^+ = \frac{1}{n}\int_0^n I^+(t)\,dt$$

d'on tindrem un holding cost mitjà de $h\bar{I}^+$.

La mitjana mensual de backlog serà

$$\bar{I}^- = \frac{1}{n}\int_0^n I^-(t)\,dt$$

amb un shortage cost de $b\bar{I}^-$.

Ens queda un problema de minimitzar el cost total mitjà en què s'incorre (encarregar mercaderia, holding i shortage), respecte les dues variables s i S. Per resoldre'l, necessitarem avaluar la funcióre cost total en punts (s, S). I per avaluar-la, simularem en un ordinador el desenvolupament dels n mesos una certa quantitat de vegades, per obtenir una mostra de valors que ens permeti estimar la mitjana del cost per a cada estratègia (s, S) que avaluem.

1.5. Definicions i conceptes

En els apartats anteriors hem utilitzat certa nomenclatura sense donar una definició precisa. No ens cal. És tracta només d'acostumar-se al significat de les paraules en el nostre context. Simplement insistirem aquí en dues paraules molt comunes, seguint més o menys el llibre de Law:

- Un *sistema* és una col·lecció de *entitats*, que actuen i interactuen amb alguna finalitat. Aquestes entitats poden ser usuaris (*customers, users, transactions, jobs, items*... depenent del context) o servidors (*servers, tellers, machines,* ...).
- L'*estat* del sistema és la col·lecció de variables necessàries per descriure el sistema en un instant determinat. Per exemple, quantitat d'usuaris a cada cua, l'instant de la seva arribada, el nombre de servidors ocupats, etc.

En el cas de l'estudi per simulació, el seguiment de l'evolució de l'estat del sistema ens permet estudiar estadísticament el *output* d'un sistema complex, a partir d'uns *inputs* de components senzills de sistema.

[A la Figura 1.1, pàgina 4 de Law, hi ha un esquema que no està malament descrivint les diferents maneres d'estudiar un sistema. En aquesta pàgina i les següents tenim una descripció de l'esquema i una classificació dels models de simulació. Pel que fa a aquesta part del curs, ens interessen els models *dinàmics*, *estocàstics*, i *discrets*. Si podeu, feu una lectura ràpida d'aquestes pàgines.]

2

Generació de variables aleatòries

Suposem que tenim una manera de fer que l'ordinador ens doni uns valors aleatoris seguint la llei uniforme a l'interval $(0,1)$. Ja veurem més endavant com és això possible i quines dificultats presenta. De moment suposem-ho cert: hi ha un mecanisme, sigui de hardware o de software, que és capaç de generar autèntics valors d'una variable $\text{Unif}(0,1)$, independents entre sí, i tants com vulguem.

Anem a veure mètodes per generar valors de variables aleatòries amb altres lleis, a partir d'aquests valors uniformes. Algunes d'elles estan ja programades en biblioteques de funcions. Per exemple, el paquet `stats` del R, que es carrega automàticament quan obrim el Rstudio, conté funcions per generar valors de moltes lleis estàndard. Però és convenient conèixer els principis generals d'aquestes mètodes.

La utilitat en la simulació serà naturalment poder alimentar un model de simulació amb els inputs convenients. Per exemple, suposem que en una cua simple hem determinat que el temps entre arribades segueix una llei exponencial de mitjana 5 minuts i el temps de servei és una normal de mitjana tres minuts, condicionada a ser més gran que 1. Com generem valors d'aquestes lleis?

2.1. Inversió

Hi ha una propietat fonamental per generar valors independents seguint qualsevol llei a partir de valors independents de $\text{Unif}(0,1)$:

Si F és una funció de distribució i $U \sim \text{Unif}(0,1)$, aleshores $F^{-1}(U) \sim F$

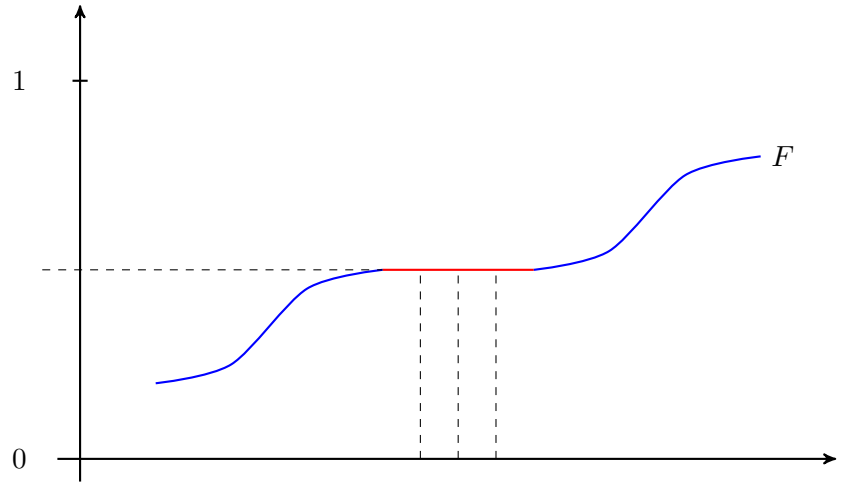
[La idea intuïtiva la podeu veure en la Figura 8.3, pàgina 432 de Law. A la part d'abaix tenim una densitat de probabilitat, a la de dalt la seva funció de distribució. Imagineu que a l'eix vertical anem generant valors d'una llei uniforme en $(0,1)$ i busquem la seva antiimatge per F (o sigui, la seva imatge per la inversa F^{-1}). Allà on la funció de distribució té més pendent (equivalentment, la densitat és més alta) s'acumularan més antiimatges.]

La idea de la demostració (després la formularem més rigorosament) és: Si $X := F^{-1}(U)$, aleshores

$$P\{X \le x\} = P\{F^{-1}(U) \le x\} = P\{U \le F(x)\} = F(x) \ ,$$

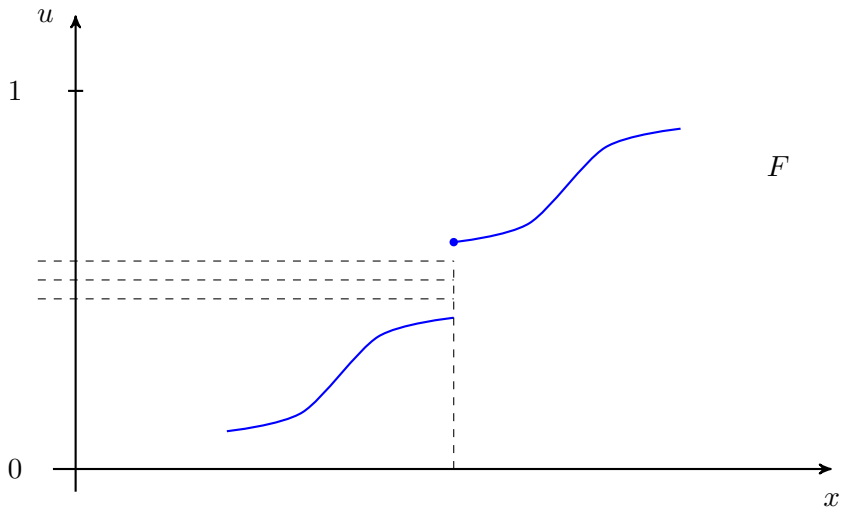
la segona igualtat perquè F és creixent, i la última perquè U és una uniforme en $(0,1)$.

Una dificultat amb tot això és que F^{-1} no té perquè existir. Per dos motius: F pot tenir un interval constant,



i per tant hi ha algun punt amb moltes antiimatges. Tal com està el dibuix, en teoria no hauria de passar, perquè la probabilitat d'obtenir aquest valor exacte en una uniforme és zero. Però com sempre treballem en precisió finita, a la pràctica és possible.

El segon motiu és que F tingui un salt. Qualsevol distribució discreta, o amb una part discreta, té salts a la funció de distribució. Aleshores hi haurà molts valors uniformes que tenen la mateixa antiimatge.



Per fer-ho bé hem de definir la inversa d'una altra manera:

$$F^{-1}(u) := \min\{x : \ F(x) \geq u\} \tag{2.1}$$

tota funció de distribució té inversa en aquest sentit (alguns l'anomenen *pseudoinversa*). El valor x és exactament la *quantila* de probabilitat u tal com la defineix el R a les funcions `q___()`.

Teorema 2.1.1. *Sigui F una funció de distribució qualsevol. Sigui F^{-1} la seva inversa segons la definició de (2.1). Aleshores:*

1. $F^{-1}(u) \leq x \Leftrightarrow u \leq F(x)$, *per a tots* $u \in (0,1)$, $x \in \mathbb{R}$.
2. $U \sim \text{Unif}(0,1) \Rightarrow F^{-1}(U) \sim F$
3. *Si F continua,* $X \sim F \Rightarrow F(X) \sim \text{Unif}(0,1)$

Demostració.

1. $\Rightarrow$) $x \geq \min\{y:\ F(y) \geq u\} =: x_0 \Rightarrow F(x) \geq F(x_0) \geq u$

   $\Leftarrow$) Fem el raonament contrarecíproc: $x < \min\{y:\ F(y) \geq u\} \Rightarrow F(x) < u$

2. Considerem la variable aleatòria $X := F^{-1}(U)$. La seva funció de distribució és:

$$P\{X \leq x\} = P\{F^{-1}(U) \leq x\} = P\{U \leq F(x)\} = F(x)\ ,$$

   la segona igualtat per la part 1, i la tercera perquè $U \sim \text{Unif}(0,1)$.

3. Aquesta propietat sembla equivalent a la 2, però falla si F no és contínua. Efectivament, hi haurà un interval de valors de $(0,1)$ que no estaran en la imatge de F (vegeu l'últim dibuix).

   Vegem que si F és contínua aleshores és cert l'enunciat: Per cada $u \in (0,1)$,

$$P\{F(X) \geq u\} = P\{X \geq F^{-1}(u)\} = P\{X > F^{-1}(u)\} = 1 - F(F^{-1}(u))$$

   la primera igualtat per la part 1, i la segona perquè F correspon a una llei contínua. Hem de veure que $u = F(F^{-1}(u))$ i tindrem que $P\{F(X) < u\} = u$; per tant, $F(X)$ segueix la llei uniforme.

   $\leq$) $F^{-1}(u) \leq F^{-1}(u) \Rightarrow u \leq F(F^{-1}(u))$, per la part 1.

   $\geq$) Sigui $\{y_n\}_n$ una successió tal que $y_n < F^{-1}(u)$ i $y_n \nearrow F^{-1}(u)$. Ara,

$$F^{-1}(u) > y_n \Rightarrow u > F(y_n) \Rightarrow u \geq \lim_{n\to\infty} F(y_n) = F(F^{-1}(u))$$

   la primera implicació per la part 1, i la última igualtat perquè F és contínua.

□

Exemple 2.1.2. La llei exponencial de paràmetre λ té funció de distribució $F(x) = 1 - e^{-\lambda x}$, si $x \geq 0$, i $F(x) = 0$, si $x < 0$. Invertim F: Si $u \in (0,1)$,

$$\begin{aligned} 1 - e^{-\lambda x} &= u \\ e^{-\lambda x} &= 1 - u \\ -\lambda x &= \log(1-u) \\ x &= \frac{-1}{\lambda}\log(1-u) \end{aligned}$$

Per tant: Si sabem generar valors u uniformes independents, i a cadascun li apliquem aquesta transformació, obtenim una mostra de valors d'una exponencial.

Hi ha un truc que permet estalviar una operació: Si U una variable $\text{Unif}(0,1)$, aleshores $1-U$ també és $\text{Unif}(0,1)$. Per tant, es poden generar exponencials fent directament $x = \frac{-1}{\lambda}\log(u)$. □

Exercici 2.1.3. La llei de Weibull de paràmetres α i β té funció de distribució $F(x) = 1 - \exp\{-(x/\beta)^\alpha\}$, si $x > 0$, i $F(x) = 0$, si $x < 0$. Obtingueu F^{-1}. □

Exercici 2.1.4. Considereu la llei amb funció de densitat $f(x) = (x + \frac{1}{2}) \cdot \mathbf{1}_{[0,1]}(x)$. Dibuixeu-la, calculeu la funció de distribució i invertiu-la. □

Suposem ara que volem generar valors d'una variable discreta, que pren valors $x_1, \ldots, x_n$ amb unes certes probabilitats $p_1, \ldots, p_n$. O sigui,

$$P\{X = x_j\} = p_j\ , \quad j = 1, \ldots, n$$

La funció de distribució F és una funció constant a trossos, i en principi no té dificultat calcular la inversa F^{-1} en el sentit (2.1).

[Mireu la Figura 8.4, pàgina 433 de Law, on hi ha representada una tal funció de distribució d'una variable que pren sis valors. La probabilitat de cada valor és la magnitud del salt corresponent de F.]

Després de generar el valor u, només cal determinar en quin salt de F cau. Un algorisme natural seria: Si $u \leq p_1$, aleshores x_1; si no, si $u \leq p_1 + p_2$, aleshores x_2; si no, si $u \leq p_1 + p_2 + p_3$, aleshores x_3; etc.

Exemple 2.1.5. Suposem que volem generar valors d'una variable aleatòria que pren valors $1, 2, 3, 4$ amb probabilitat $0.20, 0.15, 0.25, 0.40$ respectivament. Podem fer

```
if (u <= 0.20) {
  x <- 1
} else if (u <= 0.35) {
  x <- 2
} else if (u <= 0.60) {
  x <- 3
} else {
  x <- 4
}
```

Si s'han de generar molts valors i l'eficiència és important, cal considerar ordenar els valors de més a menys probable:

```
if (u <= 0.40) {
  x <- 4
} else if (u <= 0.65) {
  x <- 3
} else if (u <= 0.85) {
  x <- 1
} else {
  x <- 2
}
```

Ordenar els valors també consumeix temps, encara que només calgui fer-ho un cop. Per això ordenar només val la pena si s'han de generar molts valors. □

Exercici 2.1.6. Comproveu que la segona versió de l'exemple és més eficient computacionalment, calculant l'esperança de la quantitat de comparacions que cal fer en cada cas. □

Està clar que aquest mètode es pot aplicar també a variables discretes que prenen una quantitat infinita de valors, com ara la Poisson o la Geomètrica.

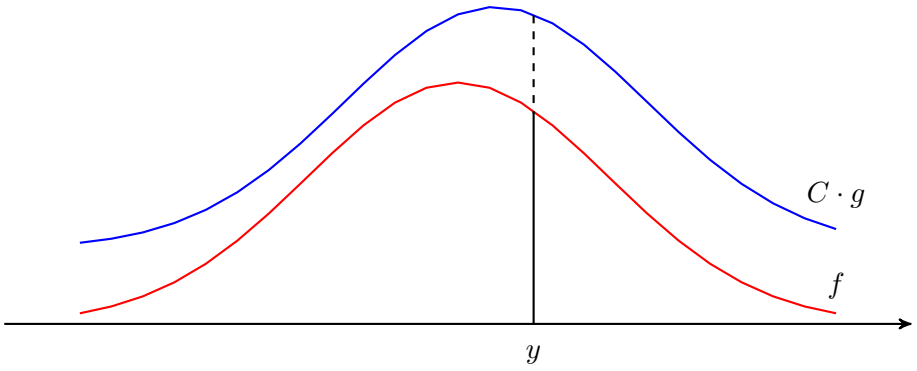
El principal desavantatge del mètode d'inversió és que de vegades no es pot calcular exactament la inversa de F. Per exemple, $f(x) = 20x(1-x)^3\mathbf{1}_{[0,1]}(x)$ és la densitat de la llei Beta de paràmetres (2,4). La funció de distribució serà un polinomi de grau 5; sabem que no existeixen fórmules per invertir-lo. Un altre cas important és el de la llei normal; sabem que no hi ha manera d'aïllar x en l'expressió

$$\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-y^2/2}\, dy = u$$

2.2. Acceptació/Rebuig

Una altra idea general per a generar valors de variables aleatòries és el mètode d'acceptació/rebuig. Suposem que volem generar valors d'una variable X que té densitat $f(x)$. Suposem que sabem generar fàcilment valors d'una altra densitat diferent $g(x)$ i que existeix una constant C tal que

$$f(x) \leq C \cdot g(x)\ , \quad \text{per a tot } x$$



Generem un valor y seguint la llei de densitat g, i l'acceptem com a valor generat per f amb una probabilitat igual a la proporció del segment fins a $C \cdot g(y)$ que queda per sota de f. Si no l'acceptem, el rebutgem i generem un altre y.

L'algorisme consisteix doncs en

1. Generar $Y \sim g$.
2. Generar $U \sim \text{Unif}(0,1)$.
3. Si $U \leq \dfrac{f(Y)}{Cg(Y)}$ retornar $X = Y$; si no, tornar al pas 1.

Demostrarem rigorosament que l'algorisme es correcte. Però la demostració no ajuda massa a entendre el perquè. Mirem primer un exemple.

Exemple 2.2.1.

[Referiu-vos a l'Exemple 8.6, pàgina 442 de Law, i les figures 8.8 i següents.]

Considereu la llei Beta de paràmetres (4,3), que té per densitat $f(x) = 60x^3(1-x)^2\mathbf{1}_{[0,1]}(x)$, dibuixada a la Figura 8.8. La funció constant $r(x)$ de la Figura és la densitat de $\text{Unif}(0,1)$. Si la multipliquem per una certa constant C aconseguirem que vagi per sobre de la densitat f. Si ajustem bé per sobre (en principi no caldria) obtenim la $t(x)$ de la Figura. (En la nostra notació, $r(x) = g(x)$ i $t(x) = Cg(x)$.)

Tornant a l'algorisme, hem de generar valors de $Y \sim \text{Unif}(0,1)$. A la Figura 8.9 estan marcats amb creuetes sobre la recta $t(x)$ els valors de Y generats en una execució particular. Alguns d'aquests valors s'accepten, i estan marcats amb creuetes sobre l'eix horitzontal; els altres s'han rebutjat. La majoria dels rebutjats ho són prop del 0 i el 1, on $t(x)$ és molt més gran que f. En canvi, allà on f és més gran, la majoria s'accepten. El resultat s'adapta a la densitat f, que és el que es pretén.

A la Figura 8.10 tenim una altra manera de veure-ho. Amb cada variable Y es genera una variable U, de manera que tenim un punt de coordenades $(Y, U \cdot t(Y))$. Els punts per sota de la gràfica de f (O sigui, tals que $U \cdot t(Y) \leq f(Y)$), ens donen valors que acceptarem; els que quedin per sobre els rebutjarem. □

Teorema 2.2.2. *Seguint l'algorisme anterior, la llei dels X retornats té densitat f.*

Demostració. Volem veure que per a qualsevol interval B,

$$P\{X \in B\} = \int_B f(y)\, dy$$

(Si les lleis són discretes, en lloc d'integrals hem d'escriure sumes, $P\{X \in B\} = \sum_{\{y \in B\}} f(y)$, però el raonament és exactament el mateix.)

La probabilitat que $X \in B$ és la mateixa que $Y \in B$, condicionat a què hem acceptat Y. Per tant, per definició de probabilitat condicionada,

$$P\{X \in B\} = P\Big\{Y \in B \Big/ U \leq \tfrac{f(Y)}{Cg(Y)}\Big\} = \frac{P\big\{Y \in B,\ U \leq \frac{f(Y)}{Cg(Y)}\big\}}{P\big\{U \leq \frac{f(Y)}{Cg(Y)}\big\}}$$

Calculem a banda numerador i denominador. Tots dos involucren dues variables aleatòries, Y i U, i per tant necessitarem una integral doble respecte la densitat conjunta. Com que Y i U són independents, la conjunta no és més que el producte de densitats. La de la uniforme dóna lloc simplement a una integral ordinària sobre $[0,1]$; la de Y serà una integral respecte $g(y)\,dy$, sobre $\mathbb{R}$. En definitiva, per al denominador,

$$P\Big\{U \leq \frac{f(Y)}{Cg(Y)}\Big\} = \int_{-\infty}^{\infty}\int_0^1 \mathbf{1}_{\{u \leq \frac{f(y)}{Cg(y)}\}}\,du\ g(y)\,dy = \int_{-\infty}^{\infty} \frac{f(y)}{Cg(y)}\ g(y)\,dy = \frac{1}{C}$$

Per al numerador,

$$P\Big\{Y \in B,\ U \leq \frac{f(Y)}{Cg(Y)}\Big\} = \int_{-\infty}^{\infty}\int_0^1 \mathbf{1}_{\{u \leq \frac{f(y)}{Cg(y)}\}}\,du\ \mathbf{1}_B(y)g(y)\,dy = \int_B \frac{1}{C}\ f(y)\,dy$$

El quocient és per tant $\int_B f(y)\,dy$, com volíem demostrar. □

En el càlcul del denominador hem vist que la probabilitat d'acceptar el valor de Y és $\frac{1}{C}$. La constant C sempre és més gran que 1 (penseu per què). En el cas de l'Exemple 2.2.1, $\frac{1}{C} = \frac{1}{2.0736} \approx 0.4823$. És a dir, que de l'ordre d'un 52% de parelles (Y,U) es rebutgen. Naturalment, convé que C sigui el més propera possible a 1.

Per això, a l'Exemple 8.6 de Law es calcula el màxim de f i s'agafa C igual a aquest màxim. El mètode funcionaria igual agafant una C més gran, però es rebutjaria més vegades.

Exercici 2.2.3. A l'Exemple 8.6, pàgina 442 de Law, es diu que el màxim de f es pot calcular derivant i igualant a zero: $f'(x) = 0$. Comproveu que és cert. Comproveu que per a qualsevol densitat Beta, aquest càlcul és molt fàcil. □

No sempre es pot agafar com a g una densitat uniforme. Només és possible quan la densitat f és una funció acotada i té el suport acotat.

Exemple 2.2.4. Hem vist que la densitat normal no és bona per aplicar el mètode d'inversió exactament. Tampoc podem aplicar el mètode d'acceptació/rebuig majorant-la per un múltiple d'una uniforme. Veurem en una pràctica que sí es pot fer amb un múltiple d'una llei molt fàcil de generar, l'exponencial, amb un petit truc previ. □

2.3. Altres mètodes generals

Suposem que la funció de distribució F de la llei que volem generar es pot expressar com una combinació convexa d'altres funcions de distribució:

$$F(x) = \sum_j p_j F_j(x)\ , \quad \text{amb } p_j \geq 0,\ \sum_j p_j = 1 \tag{2.2}$$

La suma pot tenir una quantitat finita o infinita de termes. Aquesta situació es dóna quan la llei de la variable depèn del resultat d'una altra variable discreta J, de forma que p_j és la probabilitat que $J = j$, i en tal cas X segueix la llei de F_j.

L'algorisme obvi és:

1. Generar J amb la llei $P\{J = j\} = p_j$.
2. Generar $X \sim F_j$.

Exemple 2.3.1.

[Referiu-vos a la Figura 8.6, pàgina 439 de Law.]

La llei de Laplace de paràmetres $(0, 1)$ té funció de densitat

$$f(x) = \tfrac{1}{2}e^{-|x|} = \tfrac{1}{2}e^{x}\mathbf{1}_{(-\infty,0)}(x) + \tfrac{1}{2}e^{-x}\mathbf{1}_{(0,+\infty)}(x)$$

La gràfica és la de dues densitats exponencials posades esquena contra esquena, rebaixades convenientment perquè l'àrea total sigui 1. A partir de valors uniformes, i simplificant operacions al màxim, l'algorisme es pot expressar:

1. Generar $U_1, U_2 \sim \text{Unif}(0, 1)$ independents.
2. Si $U_1 \leq 0.5$, retornar $X = \log(U_2)$; si no, retornar $X = -\log(U_2)$.

□

Exercici 2.3.2. Formuleu el mètode d'inversió per a la llei de Laplace de l'Exemple 2.3.1. Des del punt de vista de l'eficiència computacional, quin mètode sembla millor? □

Exemple 2.3.3. Considerem altre cop la densitat de l'Exercici 2.1.4. Es pot expressar

$$f(x) = (x + \tfrac{1}{2}) \cdot \mathbf{1}_{[0,1]}(x) = \tfrac{1}{2} \cdot \mathbf{1}_{[0,1]}(x) + \tfrac{1}{2}(2x) \cdot \mathbf{1}_{[0,1]}(x)$$

Per tant, té la forma de (2.2), amb $F_1 \sim \text{Unif}(0, 1)$ i $F_2(x) = x^2$, $x \in (0, 1)$. La inversió de F_2 implicaria generar una uniforme U_2 i fer-ne l'arrel quadrada. Però F_2 és la funció de distribució del màxim de dues uniformes (comproveu-ho). Usant aquest fet, tenim finalment l'algorisme:

1. Generar $U_1 \sim \text{Unif}(0, 1)$.
2. Si $U_1 \leq 0.5$, generar $U_2 \sim \text{Unif}(0, 1)$ i retornar $X = U_2$.
3. Si $U_1 > 0.5$, generar $U_2, U_3 \sim \text{Unif}\,(0, 1)$ independents, i retornar $X = \max\{U_2, U_3\}$.

Comparant aquest algorisme amb el d'inversió de l'Exercici 2.1.4, aquell resulta ser més eficient, malgrat haver de calcular una arrel quadrada. Comproveu-ho amb el paquet `microbenchmark` que veurem a pràctiques. □

Si la variable que volem generar és la suma de n variables que ja sabem generar, podem fer-ho així.

Exemple 2.3.4. La llei Gamma de paràmetres (n, λ), amb $n \in \mathbb{N}$, també anomenada n-Erlang(λ), és la de la suma de n exponencials de paràmetre λ/n, que són fàcils de generar per inversió. Per tant, podem fer:

1. Generar $Y_1, \ldots, Y_n \sim \text{Exp}(\lambda/n)$, independents.
2. Retornar $X = Y_1 + \cdots + Y_n$

En aquest cas, a més, podem estalviar càlculs usant les propietats de la funció logaritme:

$$X = \sum_{k=1}^{n} Y_k = \sum_{k=1}^{n} \frac{-n}{\lambda} \log U_k = \frac{-n}{\lambda} \log\Big(\prod_{k=1}^{n} U_k\Big)$$

Per tant, podem fer

1. Generar $U_1, \ldots, U_n \sim \text{Unif}(0, 1)$, independents.
2. Retornar $X = \dfrac{-n}{\lambda} \log\Big(\displaystyle\prod_{k=1}^{n} U_k\Big)$

□

De vegades la pròpia definició d'una llei es fa a partir d'operacions amb variables aleatòries d'altres lleis. Llavors hi ha una manera natural (encara que pot no ser l'òptima) per generar aquesta llei.

Exemple 2.3.5. Si suposem que sabem generar variables normals $Z_k \sim N(0,1)$ independents, aleshores podem generar variables khi-quadrat $X \sim \chi^2(m)$, fent

$$X = Z_1^2 + \cdots + Z_m^2$$

□

Exemple 2.3.6. Sabent generar $Y \sim \chi^2(m)$ i $Z \sim N(0,1)$ independents, podem generar variables t-Student $X \sim t(m)$, fent

$$X = \frac{Z}{\sqrt{Y/m}}$$

□

Exemple 2.3.7. Sabent generar $Y_1 \sim \chi^2(m_1)$ i $Y_2 \sim \chi^2(m_2)$ independents, podem generar variables F-Fisher $X \sim F(m_1, m_2)$, fent

$$X = \frac{Y_1/m_1}{Y_2/m_2}$$

□

Exemple 2.3.8. Sabent generar $Y_1 \sim \text{Gamma}(\alpha, 1)$ i $Y_2 \sim \text{Gamma}(\beta, 1)$ independents, podem generar variables $X \sim \text{Beta}(\alpha, \beta)$, fent

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

□

Exemple 2.3.9. Sabent generar $Z \sim N(0,1)$, podem generar variables logNormals $X \sim \text{LogN}(0,1)$, fent

$$X = e^Z$$

□

Exercici 2.3.10. Com podem generar $X \sim N(\mu, \sigma^2)$ si sabem generar $Z \sim N(0,1)$? □

Exercici 2.3.11. Com podem generar $X \sim \text{Unif}(a,b)$ si sabem generar $U \sim \text{Unif}(0,1)$? □

2.4. Mètodes particulars

Com ho fa en concret el R per generar les lleis més comunes? El paquet `stats` conté les funcions[1]

`rbeta()`	`rbinom()`	`rcauchy()`	`rchisq()`	`rexp()`
`rf()`	`rgamma()`	`rgeom()`	`rhyper()`	`rlnorm()`
`rlogis()`	`rnbinom()`	`rnorm()`	`rpois()`	`rt()`
`runif()`	`rweibull()`	`sample()`		

entre d'altres, que generen lleis habitualment utilitzades en Estadística i modelització de fenòmens aleatoris.

Totes aquestes funcions usen les idees bàsiques que hem vist, però buscant la màxima eficiència, en termes de temps i de vegades de quantitat de memòria utilitzada.

[1] Els objectes que conté el paquet `stats` es poden veure fent `ls("package:stats")`. El mateix funciona per qualsevol paquet de R.

2.4.1 Lleis contínues

El R usa inversió en les lleis exponencial, Weibull, Cauchy (amb funció de distribució $F(x) = \frac{1}{\pi}\arctan(\frac{x-x_0}{\gamma})+\frac{1}{2}$), i logística (funció de distribució $F(x) = (1+e^{-(x-x_0)/s})^{-1}$), totes fàcilment invertibles.

Per la normal i la t-Student també fa inversió, usant una molt bona aproximació de la inversa de la funció de distribució. En el cas de la normal, almenys quinze dígits significatius es poden considerar bons en el càlcul de les quantiles. En el cas de la t-Student, almenys sis dígits són bons. Per tant, a tots els efectes pràctiques, és suficient. De tota manera, hi ha mètodes exactes per generar normals, com veurem en una pràctica, i a partir d'ella la t-Student; per tant, sempre podem programar-nos un mètode exacte si volem.

Les lleis Beta i Gamma es generen mitjançant el mètode d'acceptació/rebuig amb funcions majorants sofisticades per tal de perdre la menor quantitat possible de uniformes. Per la khi-quadrat, s'usa que $\chi^2(m)$ és el mateix que Gamma$(\frac{m}{2}, 2)$.

La logNormal la fa per construcció tal com hem vist a l'Apartat 2.3. La llei de Fisher està força mal documentada i no està clar què fa.

2.4.2 Lleis discretes

Totes les lleis discretes es podrien generar per inversió, com hem vist, però hi ha molts trucs per buscar la màxima eficiència.

- Uniforme discreta: En el cas uniforme $P\{X = x_j\} = \frac{1}{n}$, $j = 1, \ldots, n$, és immediat trobar en quin salt de la funció de distribució tenim l'antiimatge de U: Si $\frac{j-1}{n} < U \leq \frac{j}{n}$, la antiimatge és x_j. I l'índex j s'obté per tant arrodonint a l'enter superior el valor nU.

  La funció del R `sample(x= , size= , replace=TRUE)` genera una mostra de mida `size` del vector de valors `x`.

- Discreta finita general: La funció `sample(x= , size= , replace=TRUE, prob= )` genera una mostra de mida `size` del vector de valors `x` amb el corresponent vector de probabilitats proporcional a `prob`. Quan `x` té pocs valors usa inversió; amb més de 200 valors s'usa el *mètode del àlies* (vegeu Law pàg 466-468). Consisteix en representar la probabilitat P com a

  $$P = \frac{1}{n-1}\sum_{i=1}^{n-1} Q_i$$

  on Q_i són probabilitats que només donen pes a dos valors de `x`. Aquesta representació es pot aconseguir de manera sistemàtica, i després amb dues uniformes es pot generar el valor desitjat: una d'elles per seleccionar la Q_i i l'altra per determinar quin dels seus dos possibles valors és el resultat.

  Amb `size=length(x)`, `replace=FALSE`, i especificant `prob` com uniforme (tots tres són els valors per omissió), el R genera una permutació aleatòria dels elements de `x`. La manera eficient de generar-la és intercanviar una posició a l'atzar (uniformement) amb l'última de la llista. Aquest últim element queda fixat. Fer el mateix amb la llista que queda; això fixa el penúltim element de la llista final. I així successivament fins el primer element. Es pot demostrar que totes les permutacions tenen la mateixa probabilitat de sortir.

- Binomial i Poisson: S'utilitza acceptació/rebuig amb una majorant sofisticada. En el cas Poisson amb paràmetre petit ($\lambda < 10$?), segurament fa inversió.

- Geomètrica: Recordeu que la definició que nosaltres acostumem a fer de la llei Geomètrica de paràmetre p és: 'Quantitat d'experiments que calen per observar per primer cop un esdeveniment de probabilitat p'. En canvi les funcions `_geom()` del R pensen en la 'quantitat de d'experiments en què no s'ha observat l'esdeveniment de probabilitat p abans del

primer cop que s'observa'. Si X segueix la primera definició i Y segueix la segona, tenim que $X = Y + 1$. Per tant, si usem `rgeom()` per generar valors, sempre hem de sumar 1 al resultat per ser conformes a la nostra definició habitual.

Seguint la nostra definició, el mètode que usa el R és un mètode d'inversió a través de l'Exponencial: Si $X \sim \text{Geom}(p)$, aleshores la seva funció de distribució

$$P\{X \leq n\} = 1 - (1-p)^n = 1 - e^{\log(1-p)n}$$

coincideix amb la funció de distribució de la $\text{Exp}(-\log(1-p))$ en el punts enters. Per tant, podem usar inversió a partir d'una uniforme U, fent $\frac{\log(U)}{\log(1-p)}$ (naturalment, el $\log(1-p)$ només cal calcular-lo una vegada) i després arrodonir cap amunt: Si l'Exponencial dóna una valor $0 < x \leq 1$, és $X = 1$; si $1 < x \leq 2$, llavors $X = 2$; etc.

El R simplement fa l'arrodoniment cap avall per ser conforme a la seva definició de $Y \sim \text{Geom}(p)$.

- Binomial negativa: Usa que la llei d'una binomial negativa de paràmetres (n, p) coincideix amb la llei d'una variable $Y \sim \text{Pois}(\Lambda)$ on $\Lambda \sim \text{Gamma}(n, \frac{1-p}{p})$. Per tant es pot generar a partir de generar una Gamma i després una Poisson.

3

Generació de vectors aleatoris i processos estocàstics

3.1. Vectors aleatoris

Generar vectors aleatoris amb components independents es redueix a generar el nombre necessari de variables independents entre sí. El cas interessant és quan el vector ha de tenir una llei conjunta que no factoritza.

La manera natural de generar un vector aleatori n-dimensional $(X_1, \ldots, X_n)$ és a través de les lleis condicionades: Podem generar X_1 amb la seva llei marginal, obtenint un valor x_1; després generar X_2 mitjançant la seva llei condicionada a $X_1 = x_1$, obtenint x_2; després X_3 amb la seva llei condicionada a $X_1 = x_1$ i $X_2 = x_2$; i així successivament fins a X_n.

Exemple 3.1.1. La llei multinomial de paràmetres $(m; p_1, \ldots, p_n)$ és la del vector aleatori $X = (X_1, \ldots, X_n)$ que compta quantes vegades s'han obtingut uns certs esdeveniments $A_1, \ldots, A_n$ en m experiments idèntics, quan en cada experiment la probabilitat d'obtenir A_k és p_k. El cas $n = 2$ és equivalent a la binomial, on només es compta un dels dos esdeveniments (x_1), i l'altre es dedueix de $x_2 = m - x_1$. En general, també un dels elements del vector $(x_1, \ldots, x_n)$ queda determinat pels altres $n - 1$.

La llei conjunta ve determinada per la fórmula

$$P\big\{X = (x_1, \ldots, x_n)\big\} = \begin{cases} \dfrac{m!}{x_1! \cdots x_n!} \displaystyle\prod_{k=1}^{n} p_k^{x_k} \ , & \text{si } x_1 + \cdots + x_n = m \\ 0 \ , & \text{en cas contrari} \end{cases}$$

però el que és útil és que la llei marginal de la primera component X_1 és binomial de paràmetres (m, p_1); condicionat a $X_1 = x_1$, la llei de X_2 és binomial de paràmetres $\big(m - x_1, p_2/(1 - p_1)\big)$; condicionat a $X_1 = x_1, X_2 = x_2$, la llei de X_3 és binomial de paràmetres $\big(m - x_1 - x_2, p_3/(1 - p_1 - p_2)\big)$; etc. La funció `rmultinom()` del R genera vectors multinomials d'aquesta manera.

Una altra manera de generar multinomials és directament de la seva definició: Generem m variables discretes amb vector de probabilitats $(p_1, \ldots, p_n)$, i sumem quantes vegades surt cada esdeveniment $A_1, \ldots, A_n$. □

Exemple 3.1.2. Per a variables normals multidimensionals, en principi també es pot aplicar el mètode general, perquè totes les marginals i condicionades són també lleis normals. Però és més directe usar que, si $Z_1, \ldots, Z_n$ són normals unidimensionals estàndard independents, i fem

$$X = AZ + \mu \ ,$$

on $Z = (Z_1, \ldots, Z_n)$ (vector columna), A és una matriu $n \times n$, i μ és un vector numèric n-dimensional, aleshores X és una normal multidimensional de vector de mitjanes μ i matriu de covariàncies $\Sigma = AA^\top$.

Per tant, si el que tenim donats són μ i Σ, només hem de trobar la factorització de Σ a través de A i la seva transposada. Aquesta factorització és única i s'anomena *descomposició de Choleski*. El R té la funció `chol()` en el paquet `base`, que retorna una matriu triangular superior que correspon a $A^\top$.

La funció `mvrnorm(n= , mu= , Sigma= )` del paquet `MASS` genera normals multidimensionals fent essencialment això. Aquí, `n` és la quantitat de vectors a generar, `mu` és el vector de mitjanes i `Sigma` és la matriu de covariàncies. □

Exercici 3.1.3. Comproveu que si Z_1 i Z_2 són variables aleatòries $N(0,1)$, aleshores

$$(Z_1, \tfrac{1}{2}Z_1 + \tfrac{\sqrt{3}}{2}Z_2)$$

és un vector normal amb vector de mitjanes $\mu = (0,0)$ i matriu de covariàncies

$$\Sigma = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix}$$

Comproveu que en efecte la funció `chol()` aplicada a la matriu Σ dóna el que ha de donar. □

Un altre mètode general aplicable a vectors és el mètode d'acceptació/rebuig aplicat directament a la funció de densitat multidimensional, sempre i quan es pugui trobar la densitat auxiliar g de la qual sigui fàcil generar vectors aleatoris. Això sol ser més difícil en lleis multidimensionals, i l'eficiència (proporció d'acceptacions) també es ressent quan la dimensió augmenta.

De vegades es veu clar que determinades variables que intervenen en una simulació no són independents, però tampoc no sabem especificar del tot la seva distribució conjunta. De fet, freqüentment la llei de la qual volem generar valors és un model d'una situació de la qual només tenim observacions. De les observacions podem deduir lleis marginals i correlacions, però no gaire cosa més. En tal cas, voldríem poder generar vectors aleatoris a partir només d'aquesta informació.

Com hem vist, podem generar lleis normals multidimensionals $Z = (Z_1, \ldots, Z_n)$ amb marginals estàndard (és a dir, $Z_i \sim N(0,1)$) i amb unes correlacions concretes ρ_{ij} entre Z_i i Z_j. Hi ha un mètode anomenat NORTA ('NORmal To Anything'), que transforma aquestes normals en el vector desitjat:

Si Φ és la funció de distribució d'una normal unidimensional estàndard i F_i és la funció de distribució que volem per a la component i-èsima, aleshores $F_i^{-1}(\Phi(Z_i))$ té distribució F_i, gràcies al Teorema 2.1.1, punts 2 i 3. La part difícil és calcular com han de ser les correlacions entre les normals Z_i per tal que després d'aplicar Φ i F_i^1 quedin les correlacions desitjades ρ_{ij} en el vector final. Tot això ho tenim ja programat a la funció `rnorta()` del paquet `SimCorMultRes` del R.

3.2. Processos estocàstics

Els *processos estocàstics* modelen l'evolució en el temps d'una magnitud aleatòria. A *temps discret*, és equivalent a pensar en una successió de variables aleatòries $\{X_n\}_{n\in\mathbb{N}}$. A *temps continu*, és una família $\{X_t\}_{t\geq 0}$. Les variables no tenen perquè ser independents entre sí ni tenir la mateixa distribució.

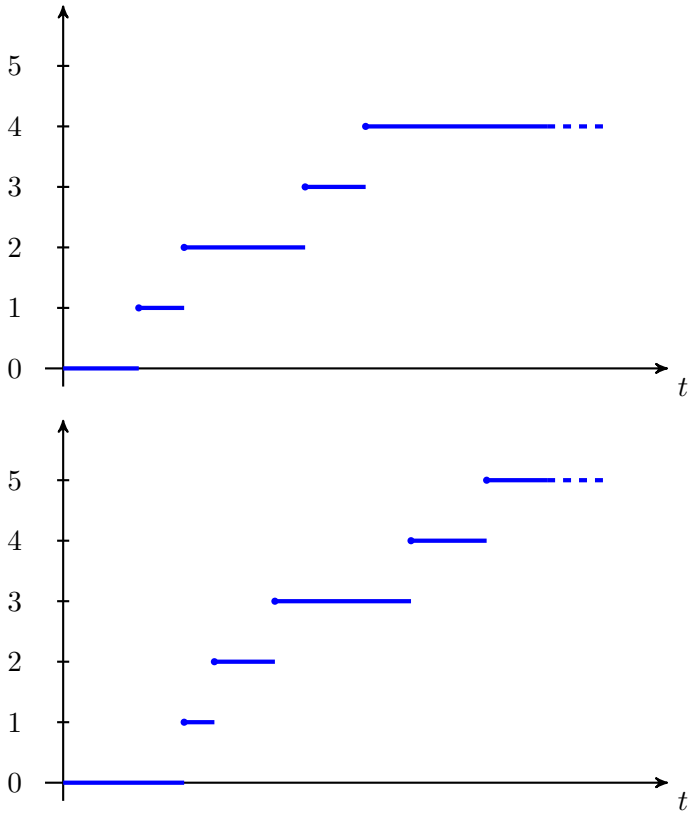
Ens interessen aquí els processos a temps continu. Sempre podem pensar que totes les variables del procés estan definides sobre el mateix espai de probabilitat base Ω, i que per a cada element

de $\omega \in \Omega$, tenim una possible *trajectòria* del procés. Un trajectòria és doncs una funció concreta $X(\omega)\colon \mathbb{R}^+ \to \mathbb{R}$.

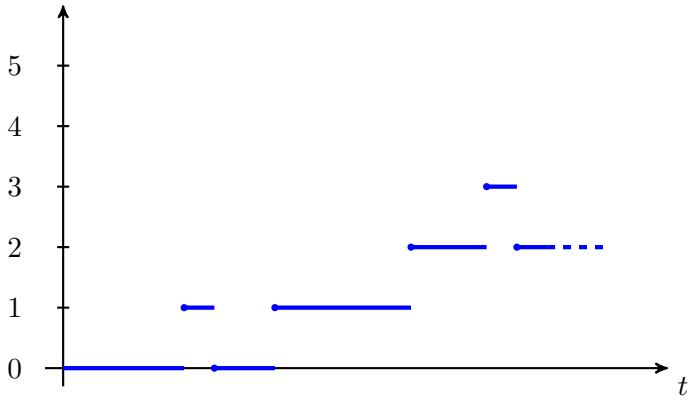
Dins dels processos a temps continu, n'hi ha que tenen trajectòries que són funcions continues, i n'hi ha que tenen trajectòries amb constants a trossos, però amb salts. Els primers típicament modelen fenòmens físics (moviments de partícules, fluctuacions de temperatures, ...). Ens interessen aquí els segons.

3.2.1 Arribades i longituds de cues

Dins els processos a temps continu, ens interessen aquells que tenen trajectòries constants a trossos. Per exemple, el procés que compta les arribades a una cua, té trajectòries creixents, amb salts cada cop que hi ha una arribada. Si les arribades són individuals, els salts sempre tenen magnitud 1. Dues possibles trajectòries:



El procés que diu la quantitat de gent que hi ha a la cua a cada instant té també salts decreixents, com ara



I si poden haver-hi arribades o serveis en grup, el procés pot tenir salts de magnitud més gran que 1.

Quan la quantitat d'arribades per unitat de temps segueix una llei de Poisson, és fàcil generar el procés d'arribades, que s'anomena precisament *procés de Poisson*. Sabem que els temps entre dues arribades successives són variables independents amb llei exponencial del mateix paràmetre que la Poisson. Per tant,

$$t_k = t_{k-1} - \frac{1}{\lambda} \log U_k$$

va donant els temps de l'arribada k-èsima a partir d'uniformes independents U_k, i els temps de les arribades anteriors, començant en $t_0 = 0$.

Si cal combinar-lo amb un servei que disminueix gent de la cua, s'ha d'anar generant la variable que dóna el temps de servei de cada usuari i anar combinant els esdeveniments per simular una trajectòria. Hem vist com fer-ho a la primera pràctica.

3.2.2 Arribades en grup

Si poden haver-hi arribades en grup, i se suposa la llei de la magnitud del grup, no té cap dificultat simular el nombre total d'usuaris que arriben. Per exemple, si la quantitat de vehicles passant per un cert punt d'una autopista segueix un procés de Poisson d'intensitat λ, i tenim la probabilitat que hi hagi n ocupants a cada vehicle (potser depenent del temps), podem simular el procés del nombre total de persones que passen per aquell punt, simulant primer els t_i amb els procediments anteriors i després el valor de n corresponent (potser depenent de t_i).

3.2.3 Arribades no estacionàries

Modelar les arribades com un procés de Poisson amb paràmetre λ pressuposa que la intensitat de trànsit es manté constant en el temps. Molts cops això no és realista. Possiblement en un supermercat hi ha hores en què hi ha molta gent i hores en què n'hi ha poca. A més, el temps total que està funcionant el sistema és un dia, i per tant finit.

En casos en què el temps total no és finit perquè és un sistema que funciona constantment, típicament es pot considerar cíclic, és a dir, que les mateixes condicions es repeteixen cíclicament, i per tant, fent moltes rèpliques del mateix cicle s'obtenen conclusions generals. Un exemple d'aquesta situació és el volum de trànsit en una autopista, o sigui, la quantitat de vehicles per unitat de temps que passen per un punt.

> [Mireu la Figura 8.15, pàgina 477 de Law, que podem pensar que representa la intensitat de trànsit d'una autopista al llarg d'un dia. S'aprecia que hi ha hores-punta cap a les 8 del matí i cap a les 6 de la tarda; una hora-vall cap a migdia, i un període de molt poc trànsit a la nit.]

En la situació d'aquest exemple, podem pensar que tenim un procés de Poisson amb un paràmetre que varia en el temps $\lambda(t)$. Això s'anomena *procés de Poisson no-estacionari*[1]. Fem una definició precisa: Un procés de Poisson d'intensitat $\lambda\colon \mathbb{R}^+ \to \mathbb{R}$ és un procés estocàstic $\{N(t)\}_{t\geq 0}$ tal que:

1. $N(0) = 0$.
2. $N(b) - N(a)$ és independent de $N(b') - N(a')$ si (a, b) i (a', b') són intervals de temps disjunts.

[1] De vegades es diu 'no-homogeni'.

3. Per a tot temps t, i $h > 0$,
   - $P\{N(t+h) - N(t) = 1\} = \lambda(t)h + o(h)$
   - $P\{N(t+h) - N(t) = 0\} = 1 - \lambda(t)h + o(h)$
   - $P\{N(t+h) - N(t) \geq 2\} = o(h)$

D'aquí es dedueix que la llei del nombre d'arribades en un interval $[t, t+h]$ és

$$N(t+h) - N(t) \sim \text{Pois}\Big(\int_t^{t+h} \lambda(s)\,ds\Big)$$

Quan $\lambda(t)$ és una funció constant tenim la situació del procés de Poisson estacionari, que és el que havíem vist a l'apartat 1.3 al modelitzar la cua simple.

Calculem el temps entre arribades en el cas no-estacionari: Si $W := T_k - T_{k-1}$ és el temps fins a l'arribada k-èsima després de l'arribada $k-1$,

$$P\{W > t\} = P\{\text{zero arribades en } [t_{k-1}, t]\} = P\{N(t) - N(t_{k-1}) = 0\} = e^{-\int_{t_{k-1}}^{t} \lambda(s)\,ds}$$

i per tant la seva funció de distribució és

$$F_W(t) = 1 - e^{-\int_{t_{k-1}}^{t} \lambda(s)\,ds}$$

Com podem generar trajectòries d'aquest procés?: Una possibilitat és invertir la funció de distribució que acabem de calcular. Suposem que hem generat els temps fins l'arribada $k-1$: $0 = t_0 < t_1 < \cdots < t_{k-1}$. Denotem $\Lambda(t) := \int_0^t \lambda(t)\,dt$ i plantegem

$$\begin{aligned}
1 - e^{-\big(\Lambda(t) - \Lambda(t_{k-1})\big)} &= u \\
e^{-\big(\Lambda(t) - \Lambda(t_{k-1})\big)} &= 1 - u \\
-\big(\Lambda(t) - \Lambda(t_{k-1})\big) &= \log(1-u) \\
\Lambda(t) &= \Lambda(t_{k-1}) - \log(1-u) \\
t &= \Lambda^{-1}\big(\Lambda(t_{k-1}) - \log(1-u)\big)
\end{aligned}$$

Veiem que a partir de valors uniformes u podem generar els successius temps t de les arribades. Podem observar com, en aquest cas, a diferència del temps de Poisson estacionari, la llei del temps entre arribades depèn de quan s'han produït les arribades anteriors.

Exercici 3.2.1. Considerem un procés de Poisson no estacionari a l'interval de temps $T = [0, 12]$, amb intensitat $\lambda(t) = \mathbf{1}_{[0,4)}(t) + 2 \cdot \mathbf{1}_{[4,8)}(t) + \mathbf{1}_{[8,12)}(t)$. Calculeu les funcions Λ i la seva inversa Λ^{-1} que intervenen a la fórmula anterior. □

Una altra possibilitat és un mètode similar al d'acceptació/rebuig. Sigui λ^* una cota superior de $\lambda(t)$, com en el dibuix de l'esmentada Figura 8.15 del llibre de Law. L'algorisme consisteix en generar punts del procés de Poisson homogeni de paràmetre λ^* i acceptar els punts t_k generats amb probabilitat $\lambda(t_k)/\lambda^*$. Allà on $\lambda(t)$ és petit, la probabilitat d'acceptar el punt és petita; on $\lambda(t)$ s'acosta a la cota λ^*, la probabilitat d'acceptar és gran.

Es pot demostrar que aquest procediment de supressió de punts (anomenat *thinning*) dóna lloc a un procés amb les propietats de la definició.

4

Simulació d'Esdeveniment Discrets

4.1. Introducció a la DES

El concepte de *Discrete Event Simulation* (DES) es refereix a la modelització i simulació d'un sistema que evoluciona en el temps i en què les variables d'interès en el model, que constitueixen *l'estat del sistema*, només canvien en instants separats.[1]

El cas de la cua de supermercat és un exemple d'aquest tipus de model. Els esdeveniments que marquen un canvi en l'estat del sistema són les arribades de clients i la finalització del servei a un client. Aquests esdeveniments poden canviar la longitud de la cua, i l'estat de ocupat o ociós de la caixa. També podem considerar que el moment d'arribada de cada client a la cua forma part de l'estat del sistema, en el sentit que ho necessitarem per calcular els delays.

Recordeu la Figura 1.2, pàgina 9 de Law. L'arribada d'un client fa que el servei canviï d'ociós a ocupat, o bé que la longitud de la cua s'incrementi en una unitat. La compleció d'un servei fa que el servei canviï d'ocupat a ociós, o bé que la cua es decrementi en una unitat.

Mireu la Figura 1.3, pàgina 10 de Law, reproduïda a la pàgina següent. És l'esquema general d'un programa que fa simulació d'esdeveniments discrets. El nucli central és el bucle

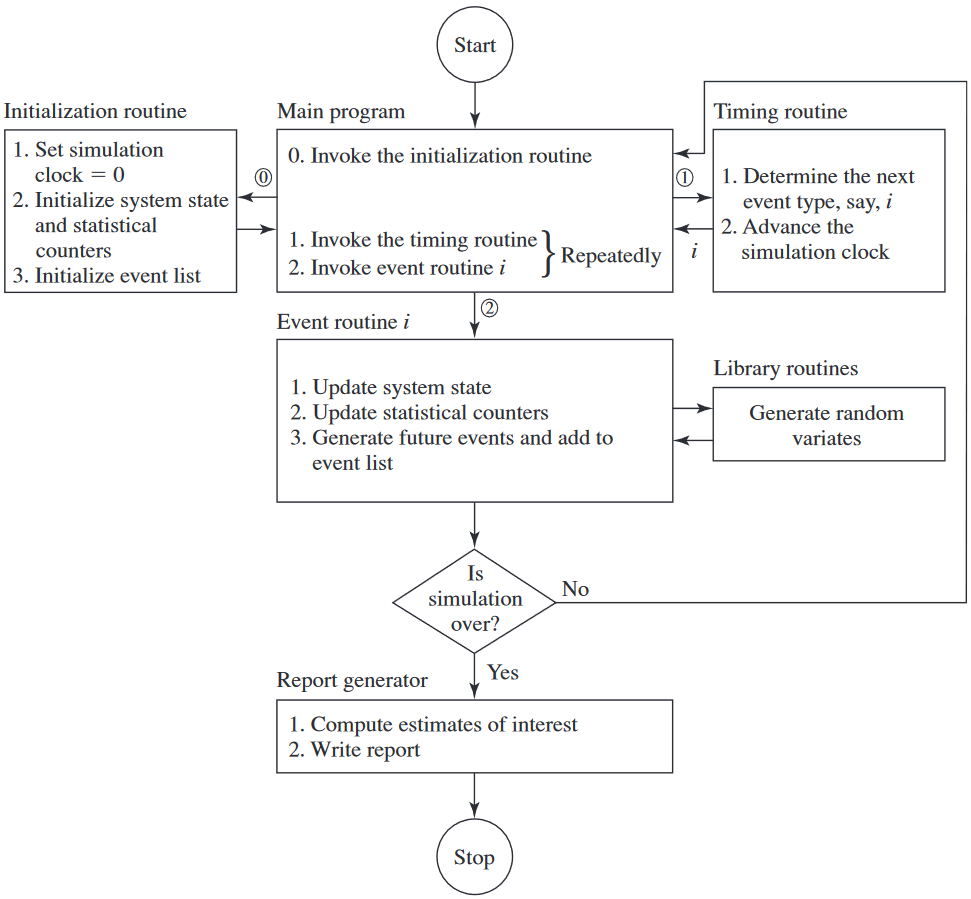
```
1. Invoke the timing routine
2. Invoke event routine i
```

Aquest algorisme va llegint i escrivint sobre una llista (implementada d'alguna manera convenient) anomenada *Future Event List* (FEL), que conté, ordenats cronològicament, els propers esdeveniments que succeiran.

La timing routine llegeix quin és el següent esdeveniment de la llista i determina el seu tipus. I també actualitza el *rellotge de simulació* a l'instant en què succeeix aquest següent esdeveniment. El rellotge no es més que una variable que conté el moment en què estem.

Seguidament la rutina corresponent al tipus d'esdeveniment actualitza l'estat del sistema, actualitza els *comptadors estadístics* (que serviran després per analitzar els resultats), i determina algun esdeveniment que succeirà més endavant i l'insereix en el lloc convenient de la FEL.

[1] En llenguatge matemàtic rigorós, el conjunt d'instants en què es produeix algun canvi és un subconjunt discret.



Exemple 4.1.1.

[Seguim l'Apartat 1.4.2, pàgina 18 de Law, i la Figura 1.7 (pàgines 19 a 23). Alternativament, podeu seguir les taules que hi ha al final d'aquest apartat.

Aquest exemple simula una cua simple. Se suposa que hi haurà arribades d'usuaris amb uns temps entre arribades de $0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, \ldots$, i uns temps de servei $2.0, 0.7, 0.2, 1.1, 3.7, 0.6, \ldots$. *Però això ara mateix no ho sabem!* Aquests números són els mateixos de les Figures 1.5 i 1.6, que podeu anar mirant de tant en tant.

Mireu la Figura 1.7(a): L'estat del sistema està representat per les variables 'Server status' (que pot ser 0 o 1 segons estigui ociós o ocupat), 'Number in queue' (longitud de la cua), i dues més auxiliars per fer els càlculs estadístics. Els comptadors estadístics van mantenint totes les variables que ens interessaran com a output; en aquest cas, el nombre d'usuaris que han sofert delay, la suma de tots els delays (per calcular després la mitjana), i l'àrea sota les funcions $Q(t)$ i $B(t)$ (vegeu Apartat 1.3) per calcular les corresponents mitjanes.

La Future Event List està representada aquí per dos registres anomenats A (per 'arrival') i D (per 'departure'). Però atenció: Això és una simplificació, perquè aquest exemple no necessita més. En general, cal mantenir una llista d'una longitud variable, on cada entrada a la llista és una parella `(temps, tipus)`, indicant el tipus d'esdeveniment i l'instant en què succeirà. I a més, cal mantenir la llista ordenada en el `temps`.

Al principi de la simulació, només un tipus d'esdeveniment és possible: Una arribada. Per tant, s'ha de pensar potser en una inicialització independent del bucle principal. A l'exemple, se suposa que una rutina d'inicialització (vegeu l'esquema de la Figura 1.3) ha fet aquesta feina i ha col·locat a la FEL un esdeveniment d'arribada a l'instant 0.4. Estem justament en la situació de la Figura (a). Podeu pensar que el `D=`∞ simplement no hi és.

Un cop feta la inicialització, la timing routine agafa el primer esdeveniment de la FEL (en aquest moment, l'únic que hi ha), avança el rellotge al temps indicat, determina que es tracta d'una arribada, i l'esborra de la FEL. El control passa a la rutina que tracta l'esdeveniment arribada.

La rutina d'arribades comprova que el servei està ociós, i el canvia a 'ocupat'. Genera dos esdeveniments més: Quan serà la propera arribada ($0.4 + 1.2 = 1.6$), i quan marxarà l'usuari que acaba d'arribar ($0.4+2.0 = 2.4$). Posa aquest dos esdeveniments a la FEL. A més, determina que el segon usuari haurà d'esperar en cua (perquè $1.6 < 2.4$), i per tant incrementa la variable 'Number delayed'. Estem en la situació de la figura (b).

La rutina d'arribades també ha de actualitzar les variables 'Àrea', però ara mateix, a l'instant $t = 0.4$, segueixen essent zero: No hi ha ningú a la cua i el servei ha estat desocupat fins ara. El control torna a la timing routine, que agafa el primer esdeveniment de la FEL, etc, igual que abans.

De nou el primer esdeveniment de la FEL és una arribada: Es generarà la següent arribada ($1.6 + 0.5 = 2.1$), però no quan marxarà el que acaba d'arribar, perquè encara no ha començat el seu servei. La mida de la cua en aquest moment ($t = 1.6$) segueix essent zero, i l'àrea sota $B(t)$ és $1 \times (1.6 - 0.4) = 1.2$. Estem en la situació de la figura (c).

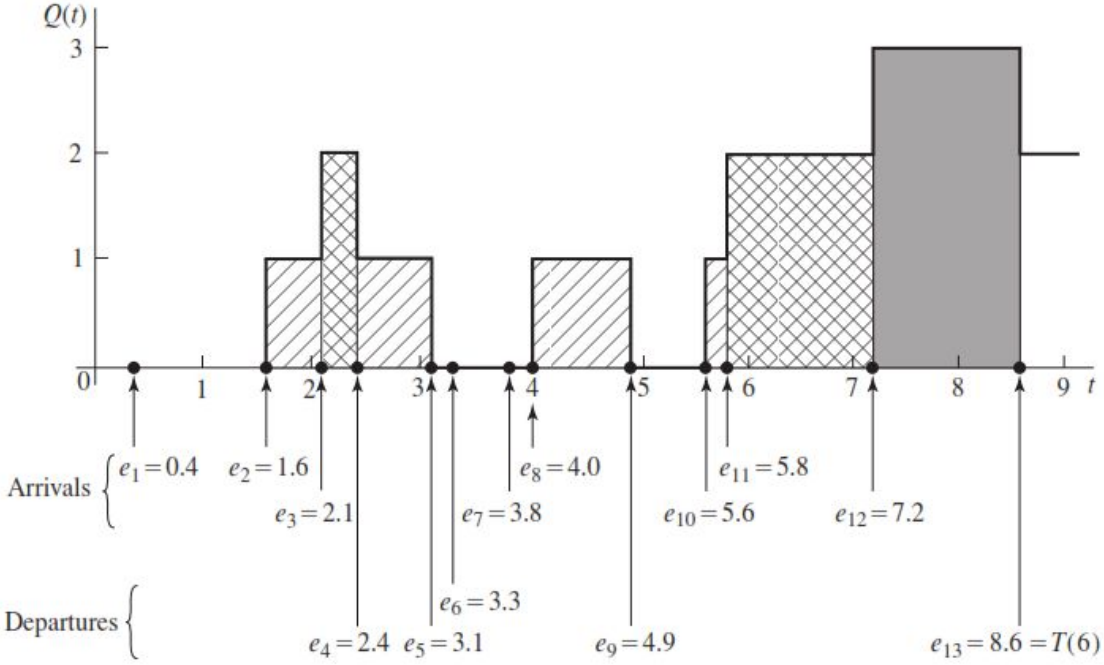
Intenteu seguir les figures (d)–(n) per comprovar que aneu entenent l'evolució de l'algorisme. En particular, a la figura (e) es produeix la marxa del primer usuari ($t = 2.4$), i s'invoca la rutina de partida: Com que hi ha gent a la cua, es manté el 'Server status' a 1; si no n'hi hagués, canviaria a 0. El 'Number in queue' baixa en una unitat. Es genera el temps de servei del segon usuari i per tant es pot posar a la FEL l'esdeveniment de la seva sortida ($2.4 + 0.7 = 3.1$). S'actualitza el delay total perquè ja tenim el que ha sofert el segon usuari ($2.4 - 1.6 = 0.8$). L'àrea sota la longitud de la cua s'actualitza afegint 2 usuaris multiplicats per la diferència $2.4 - 2.1 = 0.3$ entre el temps actual i l'anterior actualització; anàlogament, s'afegeix 0.3 a l'àrea sota l'ocupació del servei.]

□

En el llibre de Law teniu diagrames de flux per a les rutines d'arribada i partida (Figures 1.8 i 1.9), i un codi complet en C. Nosaltres implementarem aquesta simulació en R, en una pràctica, sense eines específiques. En particular, la FEL s'implementarà com un dataframe. En pur C es pot fer com una llista enllaçada.

Clock	FEL	System State	Statistical Counters
0	0.4 A	Server Occupation: 0 Queue length: 0	Delayed: 0 Total Delay: 0 Area under $Q(t)$: 0 Area under $B(t)$: 0
0.4	1.6 A 2.4 D	Server Occupation: 1 Queue length: 0	Delayed: 0 Total Delay: 0 Area under $Q(t)$: 0 Area under $B(t)$: 0
1.6	2.1 A 2.4 D	Server Occupation: 1 Queue length: 1	Delayed: 1 Total Delay: 0 Area under $Q(t)$: 0 Area under $B(t)$: 1.2
2.1	2.4 D 3.8 A	Server Occupation: 1 Queue length: 2	Delayed: 2 Total Delay: 0 Area under $Q(t)$: 0.5 Area under $B(t)$: 1.7
2.4	3.1 D 3.8 A	Server Occupation: 1 Queue length: 1	Delayed: 2 Total Delay: 0.8 Area under $Q(t)$: 1.1 Area under $B(t)$: 2.0
3.1	3.3 D 3.8 A	Server Occupation: 1 Queue length: 0	Delayed: 2 Total Delay: 1.8 Area under $Q(t)$: 1.8 Area under $B(t)$: 2.7

3.3	3.8 A	Server Occupation:	0	Delayed:	2
		Queue length:	0	Total Delay:	1.8
				Area under $Q(t)$:	1.8
				Area under $B(t)$:	2.9
3.8	4.0 A	Server Occupation:	1	Delayed:	2
	4.9 D	Queue length:	0	Total Delay:	1.8
				Area under $Q(t)$:	1.8
				Area under $B(t)$:	2.9
4.0	4.9 D	Server Occupation:	1	Delayed:	3
	5.6 A	Queue length:	1	Total Delay:	1.8
				Area under $Q(t)$:	1.8
				Area under $B(t)$:	3.1
4.9	5.6 A	Server Occupation:	1	Delayed:	3
	8.6 D	Queue length:	0	Total Delay:	2.7
				Area under $Q(t)$:	2.7
				Area under $B(t)$:	4.0
5.6	5.8 A	Server Occupation:	1	Delayed:	4
	8.6 D	Queue length:	1	Total Delay:	2.7
				Area under $Q(t)$:	2.7
				Area under $B(t)$:	4.7
5.8	7.2 A	Server Occupation:	1	Delayed:	5
	8.6 D	Queue length:	2	Total Delay:	2.7
				Area under $Q(t)$:	2.9
				Area under $B(t)$:	4.9
7.2	8.6 D	Server Occupation:	1	Delayed:	6
	9.1 A	Queue length:	3	Total Delay:	2.7
				Area under $Q(t)$:	5.7
				Area under $B(t)$:	6.3
8.6	9.1 A	Server Occupation:	1	Delayed:	6
	9.2 D	Queue length:	2	Total Delay:	5.7
				Area under $Q(t)$:	9.9
				Area under $B(t)$:	7.7



4.2. Pipelines en R: El paquet magrittr

Practicarem simulacions una mica més complicades amb un paquet de R anomenat `simmer`. La sintaxi de les instruccions en `simmer` requereix entendre i carregar prèviament el paquet `magrittr`.

Altres paquets de R requereixen `magrittr`, i el podem també utilitzar en els nostres propis programes. Per tant, és interessant en sí mateix.

Un cop instal·lat i carregat el paquet,[2] feu `vignette("magrittr")` i veureu la introducció oficial. Aquí farem un resum ràpid.

En la majoria de llenguatges de programació, la composició de funcions utilitza la notació matemàtica habitual: `h(g(f(x)))`, de manera que la funció que s'aplica la última és la que es llegeix primer, i queda tot en un ordre que no sembla el natural. Ho podem evitar guardant resultats intermedis, com ara `y=f(x); z=g(y); h(z)`, que a vegades pot ser pesat. El paquet `magrittr` introdueix l'operador "pipeline" `%>%`, que permet fer

```
x %>% f %>% g %>% h
```

o, més llegible,

```
x %>%
  f %>%
  g %>%
  h
```

Si la funció té altres arguments, com ara `f(x,y,z,t)`, podem fer `x %>% f(y,z,t)`, i si no es tracta del primer argument, podem fer `z %>% f(x,y,.,t)`. El punt marca el lloc on va l'argument.

Exemple 4.2.1. (Operador *pipe* `%>%`.) Volem generar 10 000 valors de la llei $N(0,1)$, posar-los en una matriu de 100 columnes, sumar les columnes, i calcular i imprimir la desviació tipus de les sumes. Amb variables intermèdies,

```
x <- rnorm(mean = 0, sd = 1, n = 10000)
y <- matrix(x, ncol = 100)
z <- colSums(y)
s <- sd(z)
print(s)
```

En una sola instrucció, amb la notació funcional habitual,

```
print(sd(colSums(matrix(rnorm(mean = 0, sd = 1, n = 10000), 100))))
```

En una sola instrucció, amb pipelines,

```
rnorm(mean = 0, sd = 1, n = 10000) %>%
  matrix(ncol = 100) %>%
  colSums() %>%
  sd() %>%
  print()
```

□

Exemple 4.2.2. (Operador *tee pipe* `%T>%`.) Anàlogament a l'operador "tee pipe" de bash, podem fer que la mateixa sortida vagi a dues funcions diferents. Això és necessari quan hi ha funcions que no retornen cap valor.

Suposem que volem fer un histograma de les sumes de columnes de l'exemple anterior, i després continuar els càlculs. Haurem de fer `hist()` abans de `sd()`, però això no funcionarà perquè

[2] És possible que tingueu ja instal·lat el paquet `tidyverse`; en tal cas, `magrittr` ja està també instal·lat. I carregant `tidyverse`, es carrega també `magrittr`.

`hist()` no retorna res (proveu-ho) i no es pot continuar el pipeline. El que ens interessa és usar el mateix valor que ha entrat a `hist()`. Això s'aconsegueix amb `%T>%`.

```
rnorm(mean = 0, sd = 1, n = 10000) %>%
  matrix(ncol = 100) %>%
  colSums() %T>%
  hist() %>%
  sd() %>%
  print()
```

□

Exemple 4.2.3. (Operador *exposition pipe* `$`.) Per tal de poder usar els noms interns d'un objecte, s'utilitza `%$%`. Amb les dades del dataset `iris`, calculem la correlació entre les longituds dels sèpals i els pètals de l'espècie *iris virginica*.

```
iris %>%
  subset(Species == "virginica") %$%
  cor(Sepal.Length, Petal.Length) %>%
  print()
```

□

Exemple 4.2.4. (Operador *assignment pipe* `%<>%`.) Per modificar l'objecte sobre el que estem fent càlculs amb els resultats d'aquests càlculs, podem fer

```
x <- 1:100
x <- x %>%
  log() %>%
  print()
```

o bé usar l'operador `%<>%`:

```
x <- 1:100
x %<>%
  log() %>%
  print()
```

Les dues coses sou prou intel·ligibles.[3] □

Exemple 4.2.5. Per facilitar operacions aritmètiques, tenim àlies com ara `multiply_by()` i `add()`. Per exemple, generem normals de mitjana 2 i desviació tipus 5 "manualment", a partir de normals estàndard:

```
rnorm(mean = 0, sd = 1, n = 10000) %>%
  multiply_by(5) %>%
  add(2) %>%
  {
    cat("Mean:", mean(.), "\n")
    cat("Variance:", var(.), "\n")
  }
```

Observeu l'ús de les claus `{ }` per agrupar. També es pot escriure

```
rnorm(mean = 0, sd = 1, n = 10000) %>%
  `*`(5) %>%
  `+`(2) %>%
  {
    cat("Mean:", mean(.), "\n")
```

[3] Marginalment, observeu que la funció `print()` sí retorna un valor: el seu mateix argument.

```
  cat("Variance:", var(.), "\n")
}
```

Recordeu que, encara que és una sola instrucció i es pot escriure de qualsevol manera, és aconsellable per claredat escriure-la en diverses línies, acabades amb l'operador, i amb indentació. □

4.3. El paquet simmer

El paquet simmer facilita la formulació de models de simulació i l'estudi de resultats. Utilitza la sintaxi magrittr i queda bastant agradable de llegir i escriure. Hi ha algun altre paquet dedicat a simulació en R, i també les alternatives SimPy i SimJulia per Python i Julia respectivament.

Llegiu la introducció dels autors fent vignette(package="simmer") o en https://r-simmer.org/articles/simmer-01-introduction.html. Podeu saltar la part que parla de la paral·lelització amb mcapply, que no usarem. Alternativament, hi ha un vídeo a YouTube que fa una introducció a la DES usant aquest mateix exemple de la vinyeta: https://www.youtube.com/watch?v=Qe1NvHJcmZs. Interessa sobretot a partir del minut 4:20. La durada total és de 12:38.

A classe farem una demostració sobre aquest exemple, que tindreu també escrit a la Pràctica 4.

5

Anàlisi de resultats

Fem simulació pel mateix motiu que recolliríem una mostra real: per fer inferència estadística i prendre decisions. Volem estimar un paràmetre que ens interessa, o aproximar la llei d'una variable aleatòria amb un histograma. O també comparar resultats sota escenaris diversos.

Bona part del que es pot dir sobre el *output* d'una simulació és el mateix que es pot dir d'una mostra agafada de la realitat. Però hi ha algunes subtileses.

Per poder utilitzar la teoria clàssica de la inferència estadística necessitem una mostra de variables independents idènticament distribuïdes (IID), com ja vam discutir a l'Apartat 1.3.

Típicament, durant una simulació es recullen molts valors d'una determinada magnitud (per exemple, el delay dels usuaris en una cua, o el retard en servir les comandes en el problema de l'inventari). Però aquests valors no són IID. En canvi si fem diverses *rèpliques* de la simulació, sí podem obtenir una mostra IID: Quin és el delay del primer usuari que pateix delay? Quina és la mitjana dels delays? Quina és la suma de tots els retards en servir comandes? Fent diverses rèpliques idèntiques de la mateixa simulació, aquests valors, obtinguts en les mateixes condicions, sí són IID.

En general, suposarem que la nostra simulació té un final natural. Per exemple, si estem estudiant el funcionament de les cues en un supermercat, hi ha un final natural, que és clarament l'hora de tancar. Pot ser que l'hora final de la simulació sigui aleatòria, però és un final. Fins i tot si parlem d'un supermercat 24/7, segur que té cicles, i es pot suposar un final, després del qual tot torna a començar en les mateixes condicions. Veurem alguna cosa sobre el cas "sense final" a l'Apartat 5.2.

5.1. Anàlisi estadística

5.1.1 Estimació de l'esperança

Suposem que tenim n rèpliques d'una simulació, i per tant hem observat una mostra $X_1, \ldots, X_n$ de variables IID d'una certa magnitud aleatòria X, i que en interessa el paràmetre esperança $\theta = \mathrm{E}[X]$, que és la situació habitual.

Sabem que la variable aleatòria mitjana mostral

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

estima θ sense biaix, és a dir, $\mathrm{E}\left[\bar{X}\right] = \theta$. L'error quadràtic

$$\mathrm{E}\left[\left(\bar{X} - \theta\right)^2\right] = \mathrm{Var}\left[\bar{X}\right] = \frac{\sigma^2}{n} \ ,$$

on $\sigma^2 = \text{Var}\big[X\big]$, es pot estimar usant que la variància mostral

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

és un estimador sense biaix de $\text{Var}\big[X\big]$.

A partir d'aquí, usant el Teorema Central del Límit,

$$\frac{\bar{X}-\theta}{\sigma/\sqrt{n}} \underset{n\to\infty}{\longrightarrow} N(0,1)\ ,$$

es construeix un interval de confiança asimptòtic[1]

$$\Big[\bar{X} - \frac{1.96S}{\sqrt{n}}, \bar{X} + \frac{1.96S}{\sqrt{n}}\Big]$$

que podem escriure més breument

$$\bar{X} \pm \frac{1.96S}{\sqrt{n}}$$

En principi no tenim idea de quina és la llei de la mostra, i per tant, per poder deduir aquest interval de confiança, hem de suposar que la mostra és gran, i recolzar-nos en el Teorema Central del Límit. La mida de la mostra no sol ser problema en simulació, a diferència de quan cal utilitzar mostres reals, en què el cost d'obtenir observacions pot ser elevat.

Si es vol un interval $\bar{X} \pm \varepsilon$, per un cert ε donat, es pot trobar la mida de la mostra necessària fent

$$\frac{1.96S}{\sqrt{n}} \leq \varepsilon \ \Rightarrow\ n \geq \Big(\frac{1.96S}{\varepsilon}\Big)^2 \tag{5.1}$$

Aquí cal anar amb compte perquè, degut a la variabilitat de S, la condició es podria complir per casualitat amb n molt petites, quan encara S no és una bona aproximació de la desviació tipus. És convenient treballar amb mostres d'almenys $n \geq 100$, o més gran per sistemes complexos que puguin donar lloc a una llei de X molt dispersa.

Si volem anar simulant rèpliques i comprovant la condició es pot actualitzar la variància mostral iterativament: Si $\bar{X}_n$ i S_n^2 són els estadístics de la mostra de mida n, es té

$$S_{n+1}^2 = \frac{n-1}{n}S_n^2 + (n+1)(\bar{X}_{n+1} - \bar{X}_n)^2$$

on $\bar{X}_{n+1}$ també es pot actualitzar iterativament, amb

$$\bar{X}_{n+1} = \frac{1}{n+1}(n\bar{X}_n + X_{n+1})$$

Els intervals $\bar{X} \pm \varepsilon$ acoten l'*error absolut*. Escrit rigorosament,

$$P\{|\bar{X} - \theta| \leq \varepsilon\} = 0.95$$

De vegades podem estar interessats en l'*error relatiu*:

$$P\Big\{\frac{|\bar{X}-\theta|}{|\theta|} \leq \varepsilon\Big\} = 0.95$$

[1] Si suposem un nivell de confiança del 95%. En general, per una confiança de $1-\alpha$, tindríem en lloc del 1.96 la quantila $1-\frac{\alpha}{2}$ de la llei normal. En R, `qnorm(mean=0, sd=1, p=1-alpha/2)`.

Quina seria a mida de la mostra adequada? Posem $\eta = \frac{\varepsilon}{1+\varepsilon}$, i busquem la mida de mostra n que ens doni

$$\frac{1.96S}{\sqrt{n}} \leq \eta \cdot |\bar{X}|$$

Tindrem un error absolut de $\eta \cdot |\bar{X}|$, o sigui

$$P\{|\bar{X} - \theta| \leq \eta|\bar{X}|\} = 0.95$$

Per la desigualtat triangular, aquesta probabilitat és més petita o igual que

$$\begin{aligned} &P\big\{|\bar{X} - \theta| \leq \eta|\bar{X} - \theta| + \eta|\theta|\big\} = \\ &P\Big\{\frac{|\bar{X} - \theta|}{|\theta|} \leq \frac{\eta}{1-\eta}\Big\} = \\ &P\Big\{\frac{|\bar{X} - \theta|}{|\theta|} \leq \varepsilon\Big\} \end{aligned}$$

i ja hem obtingut el que volíem.

5.1.2 Estimació de probabilitats

Si el que estem fent és estimant la probabilitat d'un cert esdeveniment, és com si estiguéssim estimant el paràmetre p d'una llei de Bernoulli; la de la variable X que dóna 1 si s'ha produït l'esdeveniment i 0 en cas contrari.

La mitjana mostral $\bar{X}$ és la proporció d'elements de la mostra original que presenten aquest esdeveniment.[2] Podem anar sobre segur posant $S = \frac{1}{2}$, que és una cota superior de la desviació tipus d'una Bernoulli. Per tant, l'interval de confiança conservador és

$$\bar{X} \pm \frac{1.96}{2\sqrt{n}}$$

i la mida de la mostra necessària es calcularia igual que abans.

Exemple 5.1.1. Potser més interessant que l'esperança de la longitud d'una cua, i que el seu màxim, és la probabilitat que superi una certa quantitat d'usuaris, per exemple de cara a saber quina proporció de dies la cua superarà l'espai físic assignat i caldrà prendre accions.

Si N és la longitud de la cua i volem estimar un interval de confiança pel valor $p = P\{N \geq 20\}$, amb un cert error absolut màxim ε, determinarem la mida de la mostra, és a dir, la quantitat de rèpliques de la simulació, amb

$$\frac{1.96}{2\sqrt{n}} \leq \varepsilon \;\Rightarrow\; n \geq \Big(\frac{1.96}{2\varepsilon}\Big)^2$$

□

Exercici 5.1.2. L'estimació de quantiles com a l'exemple 5.1.1 es fa freqüentment per esdeveniments rars, de probabilitat petita. En tal cas no és molt adequat treballar amb l'error absolut. Per exemple, si obtenim un interval 0.04 ± 0.06 no estem donant molta informació. En tal cas és millor imposar un error relatiu màxim.

Amb un nivell de confiança del 95%, determineu la mida de la mostra per obtenir un error relatiu de $\varepsilon = 0.25$, si va sortint una mitjana (proporció) prop de $\bar{x} = 0.04$. □

[2] Se sol denotar $\hat{p}$ en lloc de $\bar{X}$

5.2. Steady-state vs transient-state

El resultat d'una simulació depèn de les condicions inicials. No és el mateix començar amb una cua buida que amb una cua on ja hi ha usuaris esperant en el moment que comença el servei. Pot ser que al cap d'un temps les condicions inicials ja no es notin, i el sistema hagi entrat en un *estat estacionari* (*steady-state*). Mentre no s'arriba a aquesta situació, es diu que estem en *estat transitori* (*transient-state*).

Aquesta distinció només és important per sistemes que han de funcionar molt temps, i on només ens interessa què passarà "a la llarga", quan les condicions inicials "d'escalfament del sistema" ja no influeixin. Les corresponent simulacions s'hauran de fer per un temps molt llarg, per tal que no depenguin de les condicions inicials. Es parla de *non-terminating simulations* versus *terminating simulations*, o de *infinite horizon* versus *finite horizon*.

En aquest curs de simulació estem posant èmfasi en les *terminating* simulations, però anem a veure una mica què es pot fer en la situació *non-terminating*.

Amb les non-terminating simulations es plantegen dos problemes:

- Com se sap quan acaba l'estat transitori i comença l'estacionari? Això no està ben definit. Sempre hi haurà una petita influència de les condicions inicials. Però una manera d'eliminar part de la influència és no considerar el que hagi succeït en un interval inicial de temps i recollir dades només a partir d'un cert moment. Per fer això, tenir una idea gràfica de quan comença a estabilitzar-se la quantitat que ens interessa pot anar bé.
- Com fer moltes rèpliques de una simulació que ha de ser molt llarga? És pot fer una sola rèplica i fer alguna cosa útil amb ella? Això últim és el que se sol fer. Si volem estudiar el steady-state es preferible fer una simulació molt llarga i només haver de patir una vegada les condicions inicials. Veurem el *mètode de batch means* per a aquesta situació, que és el més senzill i més aplicat.

Exemple 5.2.1.

[Mireu la Figura 9.2, pàgina 492 de Law. Aquesta figura representa la situació següent: Hi ha un cua en què el temps entre arribades és exponencial de mitjana 1 i el temps de servei és també exponencial de mitjana 9/10. La gràfica representa l'esperança del delay de l'usuari i-èsim, per diferents valors inicials d'usuaris a la cua a l'instant zero. Aquestes esperances es poden determinar analíticament.

La situació canvia molt depenent de la condició inicial, però a la llarga l'esperança tendeix a 8.1. En estat estacionari, tots els usuaris tenen una esperança de delay de 8.1.]

De fet, la llei sencera d'aquest delay "a la llarga" D es coneix explícitament: Si el temps entre arribades segueix una Exponencial de paràmetre λ, i el temps de servei segueix una Exponencial de paràmetre $\nu > \lambda$, aleshores

$$P\{D \leq x\} = (1 - \tfrac{\lambda}{\nu}) + \tfrac{\lambda}{\nu}(1 - e^{-(\nu-\lambda)x}) \ , \quad x \geq 0 \tag{5.2}$$

Podeu comprovar (exercici) que amb els números de l'exemple, efectivament $\mathrm{E}[D] = 8.1$. □

El mètode de batch means (mitjanes per blocs) consisteix en fer una sola simulació molt llarga, dividir les observacions en blocs, i tractar cada bloc com si fossin rèpliques independents. El primer bloc, opcionalment, es pot descartar per reduir l'impacte de les condicions inicials. En detall, suposant que el que observem són delays:

- Fixem primer una mida mostral, o sigui, la quantitat n d'observacions que farem (per exemple, els delays D_i de n usuaris).
- Decidim un *batch size* b. Després veiem com.

- Fem la simulació fins a obtenir els n delays $D_1, \ldots, D_n$ i dividim la successió en $k = n/b$ *batches*:

$$\underbrace{D_1, \ldots, D_b}_{\text{batch } 1}, \underbrace{D_{b+1}, \ldots, D_{2b}}_{\text{batch } 2}, \underbrace{D_{2b+1}, \ldots, D_{3b}}_{\text{batch } 3}, \ \ldots \ \ldots \underbrace{D_{(k-1)b+1}, \ldots, D_{kb}}_{\text{batch } k},$$

- Calculem la mitjana de cada batch:

$$\bar{D}_j = \frac{1}{b} \sum_{i=1}^{b} D_{(j-1)b+i} \ , \quad j = 1, \ldots, k$$

- Fem estadística habitual amb aquesta mostra:

$$\bar{D} = \frac{1}{k} \sum_{j=1}^{k} \bar{D}_j$$

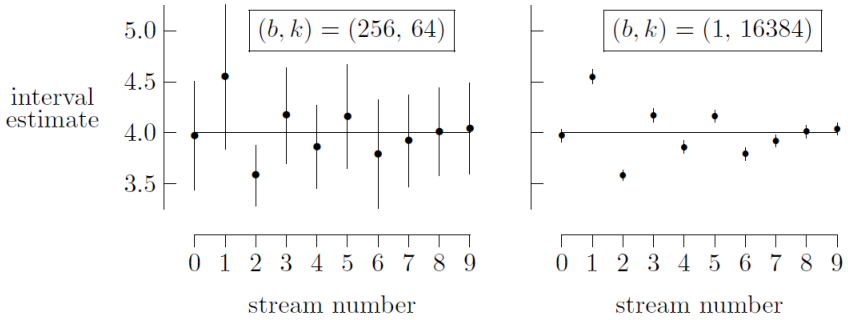
$$S^2 = \frac{1}{k-1} \sum_{j=1}^{k} \left(\bar{D}_j - \bar{D} \right)^2$$

  i els intervals de confiança asimptòtics corresponents.

Estrictament parlant, les batch means $\bar{D}_j$ no són independents entre sí, però tenen menys correlació que dues observacions consecutives directes D_i, D_{i+1}. Com més gran és el batch size, menys correlació hi haurà entre les diferents $\bar{D}_j$. Això es pot millorar encara deixant un espai sense recollir observacions entre els diferents batches (*spaced batch means*).

Com combinar les quantitats b i k? Si agafem batches molt grans, tindrem una mostra final petita; si els agafem massa petits, hi haurà massa correlació entre ells. Idealment, cal estudiar la correlació entre elements de la mostra que estan a una certa distància per veure fins on es nota la correlació, i agafar els batches prou grans perquè la correlació tingui poca influència. Alternativament, es pot experimentar amb unes quantes rèpliques curtes i calcular per a totes elles intervals de confiança per a diferents b i k. Si b es prou gran, els intervals se solaparan força; si és massa petit, tendiran a no solapar-se.

A la figura següent[3] veiem el resultat d'un experiment sobre una cua simple amb temps entre arribades exponencial de paràmetre 1, i temps de servei exponencial de paràmetre 1.25. Es pot demostrar analíticament que quan $t \to \infty$ la mitjana del temps en el sistema (delay + temps de servei) de tots els usuaris tendeix a 4, aplicant altre cop la fórmula (5.2).



S'han fet 10 rèpliques amb una mida mostral de $n = 16384$ usuaris. A l'esquerra hi ha els 10 intervals de confiança obtinguts, al 95% de confiança teòric, agafant 64 blocs de 256 elements cadascun; els intervals són una mica amplis però atrapen el valor verdader del paràmetre nou d'ells. A la dreta, s'ha agafat un sol bloc amb tots els valors mostrals (atenció: ha de dir $(b, k) = (16384, 1)$); els intervals són molt precisos, però només tres atrapen el valor verdader.

[3] Del llibre Leemis–Park, *Discrete-Event Simulation. A First Course*, 1994-2004.

Naturalment, en un cas real no sabem quin és exactament el verdader valor de paràmetre. El que hem de buscar és que els intervals de confiança de les diferents rèpliques se solapin apreciablement.

5.3. Reducció de la variància

L'estimació natural de $\theta = \mathrm{E}[X]$ per intervals de confiança sabem que és

$$\bar{X} \pm \frac{1.96S}{\sqrt{n}}$$

essent $\bar{X}$ i S la mitjana i la desviació tipus mostral respectivament (i sempre suposant que volem un 95% de confiança, i que tenim una mostra prou gran com perquè el Teorema Central del Límit tingui efecte). Està clar que fixada la mida de la mostra n, si tenim un altre mètode de simulació que ens dona una desviació tipus més petita, la longitud de l'interval de confiança es redueix proporcionalment a la reducció de S.

Exemple 5.3.1. Si tenim un mètode senzill amb una certa S_1, i un mètode més sofisticat que ens dóna $S_2 = 0.75S_1$, l'error es redueix en la mateixa proporció que la desviació tipus S:

$$\frac{1.96S_2}{\sqrt{n}} = \frac{1.96S_1}{\sqrt{n}} \cdot 0.75$$

Suposant que el temps de càlcul pel mètode senzill i pel sofisticat seguin comparables, com es redueix la mida de la mostra necessària per aconseguir la mateixa precisió?

$$\frac{S_1}{\sqrt{n_1}} = \frac{S_2}{\sqrt{n_2}} \;\Rightarrow\; \frac{S_1}{\sqrt{n_1}} = \frac{0.75S_1}{\sqrt{n_2}} \;\Rightarrow\; \frac{n_2}{n_1} = 0.75^2 = 0.5625$$

La mida de la mostra es redueix quasi a la meitat. Això és independent del nivell de confiança. □

Un punt important és determinar si val la pena usar un mètode sofisticat (esforç intel·lectual, temps de feina, possibles errors de programació, més temps de càlcul) per aconseguir una reducció en la mida de la mostra. En el mostreig de dades reals, on és costós tenir una mostra gran, sí tenen molta importància els mètodes de mostreig que redueixin la variància. En el cas de la simulació, depèn de la situació concreta.

Veurem dos mètodes de reducció de la variància: el *mostreig estratificat*, i els *nombres aleatoris comuns*. S'apliquen a situacions diferents.

5.3.1 Mostreig estratificat

És absolutament vital en el mostreig estadístic amb dades reals. Particularment, les enquestes a persones utilitzen aquesta metodologia tot el possible.

Suposem que volem estimar $\theta = \mathrm{E}[X]$, i sigui Y una variable discreta, que pren valors $y_1, \ldots, y_k$ amb probabilitats conegudes $p_1, \ldots, p_k$. Suposem també que sabem simular $\mathcal{L}(X \,|\, Y = y_i)$, la llei condicionada de X a cadascun dels valors $Y = y_i$.

Sabem que l'esperança de X es pot calcular precondicionant a cadascun dels valors de Y i sumant:

$$\mathrm{E}[X] = \sum_{i=1}^{k} \mathrm{E}\left[X \,|\, Y = y_i\right] \cdot p_i \tag{5.3}$$

En lloc de simular n rèpliques de X, fem $n \cdot p_i$ rèpliques de $X_i \sim \mathcal{L}(X \big/ Y = y_i)$. I definim

$$\tilde{X} := \sum_{i=1}^{k} \bar{X}_i \cdot p_i$$

Es comprova immediatament usant (5.3) que $\tilde{X}$ és un estimador sense biaix de θ. Calculem la seva variància:

$$\mathrm{Var}[\tilde{X}] = \sum_{i=1}^{k} \mathrm{Var}[\bar{X}_i] \cdot p_i^2 = \sum_{i=1}^{k} \frac{\mathrm{Var}[X_i]}{n \cdot p_i} \cdot p_i^2 = \frac{1}{n} \sum_{i=1}^{k} \mathrm{Var}[X_i] \cdot p_i \tag{5.4}$$

Utilitzarem ara la *fórmula de la variància total*, que es dedueix així:

$$\begin{aligned}
\mathrm{Var}[X] &= \mathrm{E}[X^2] - \mathrm{E}[X]^2 \\
&= \sum_{i=1}^{k} \mathrm{E}\left[X^2 \big/ Y = y_i\right] \cdot p_i - \Big(\sum_{k=1}^{k} \mathrm{E}\left[X \big/ Y = y_i\right] \cdot p_i \Big)^2 \\
&= \sum_{i=1}^{k} \mathrm{Var}\left[X \big/ Y = y_i\right] \cdot p_i + \sum_{i=1}^{k} \mathrm{E}\left[X \big/ Y = y_i\right]^2 \cdot p_i \\
&\qquad\qquad - \Big(\sum_{k=1}^{k} \mathrm{E}\left[X \big/ Y = y_i\right] \cdot p_i \Big)^2 \\
&= \sum_{i=1}^{k} \mathrm{Var}\left[X_i\right] \cdot p_i + \sum_{i=1}^{k} \mathrm{E}\left[X_i\right]^2 \cdot p_i - \Big(\sum_{k=1}^{k} \mathrm{E}\left[X_i\right] \cdot p_i \Big)^2
\end{aligned}$$

Els dos últims termes junts són la variància de la llei discreta que pren els valors $\mathrm{E}[X_i]$ amb probabilitats p_i, i per tant sumen una quantitat $V_i \geq 0$.

Substituïm el primer sumand a l'última expressió de (5.4) i obtenim

$$\mathrm{Var}[\tilde{X}] = \frac{1}{n}\big(\mathrm{Var}[X] - V_i \big) \leq \frac{1}{n} \mathrm{Var}[X] = \mathrm{Var}[\bar{X}]$$

Conclusió: L'estimador estratificat $\tilde{X}$ té menys variància que l'estimador natural $\bar{X}$.

Com podem estimar $\mathrm{Var}[\tilde{X}]$? Si S_i^2 és la variància mostral de les $n \cdot p_i$ rèpliques de X_i, aleshores

$$\frac{1}{n} \sum_{i=1}^{k} S_i^2 \cdot p_i$$

és un estimador sense biaix de $\frac{1}{n} \sum_{i=1}^{k} \mathrm{Var}[X_i] \cdot p_i$, que és igual a $\mathrm{Var}[\tilde{X}]$, per (5.4).

Finalment, aplicant el Teorema Central del Límit com abans, tenim

$$\frac{\tilde{X} - \theta}{\sqrt{\frac{1}{n} \sum_{i=1}^{k} S_i^2 \cdot p_i}} \xrightarrow[n \to \infty]{} N(0,1)$$

i podem construir un interval de confiança asimptòtic de igual manera que quan en el denominador tenim $\frac{S}{\sqrt{n}}$:

$$\tilde{X} \pm \frac{1.96 \sqrt{\sum_{i=1}^{k} S_i^2 \cdot p_i}}{\sqrt{n}}$$

Exemple 5.3.2. Podem usar el mostreig estratificat quan podem fer grups o "estrats" (de persones, o del que sigui que estem tractant) amb unes característiques similars que es reflectiran en la magnitud que estem estimant. Per exemple, suposem que en un centre sanitari hi arriben infants, adults, i adults grans, i tots entren a la mateixa cua. Però sabem que no tots requereixen el mateix temps d'atenció per part dels metges.

Al fer la simulació podem assignar aquests temps diferents segons el tipus de persona. Per una banda, podrem tenir estimacions diferents del temps en el sistema per a cada tipus per separat, que pot ser útil en sí mateix. Per altra banda, si volem estimar el temps d'una persona genèrica, ho podem fer usant l'interval de confiança que acabem de veure, amb la proporció p_i de cada tipus de persona dins de la població general. □

El que acabem de fer s'anomena *proportional allocation*, perquè hem repartit proporcionalment la mida mostral n segons la probabilitat de cada estrat: $np_1, \dots, np_k$. Aquest mètode redueix la variància, i ho fa millor com més homogeni sigui cada estrat, perquè la variància interna $\mathrm{Var}[X_i]$ serà més petita, i com més separades entre sí estiguin les esperances $\mathrm{E}[X_i]$.

Si tenim algun coneixement, o podem estimar prèviament aquestes variàncies internes de cada estrat, podem fer-ho encara millor que amb la proportional allocation. Si assignem mides de mostra $n_1, \dots, n_k$ a cada estrat, de moment indeterminades, amb $n_1 + \cdots + n_k = n$ la mida de mostra total, i definim, semblant al que hem fet abans,

$$\hat{X} := \sum_{i=1}^{k} \bar{X}_i p_i \ ,$$

ara tindrem

$$\mathrm{Var}[\hat{X}] = \sum_{i=1}^{k} \mathrm{Var}[\bar{X}_i] \cdot p_i^2 = \sum_{i=1}^{k} \frac{\mathrm{Var}[X_i] \cdot p_i^2}{n_i} \tag{5.5}$$

El que voldríem és trobar els n_i, enters positius i que sumin n, que fan (5.5) el més petita possible. Deixant de banda l'exigència que els n_i siguin enters, el resultat seria

$$n_i = n \cdot \frac{\sigma[X_i] \cdot p_i}{\sum_{j=1}^{k} \sigma[X_j] \cdot p_j} \tag{5.6}$$

on hem denotat $\sigma[X_i] := \sqrt{\mathrm{Var}[X_i]}$.

Exercici 5.3.3. Comproveu que si totes les variàncies $\mathrm{Var}[X_i]$ fossin iguals, la fórmula (5.6) es redueix a la proportional allocation $n_i = np_i$. Si arrodonim aquests valors a enter obtenim una bona aproximació del valor òptim que busquem. □

Exercici 5.3.4. Comproveu que, fent servir S_i^2 com estimador de $\mathrm{Var}[X_i]$, com sempre, l'assignació (5.6) dóna lloc a l'interval de confiança

$$\hat{X} \pm \frac{1.96 \sum_{i=1}^{k} S_i \cdot p_i}{\sqrt{n}}$$

□

5.3.2 Nombres aleatoris comuns

Suposem que volem comparar dues configuracions d'un sistema, o dos paràmetres numèrics. Per exemple, volem veure la diferència entre usar una sola cua amb dos servidors o dues cues separades, pel que fa al temps d'espera dels usuaris. O volem saber què passaria amb l'ocupació del servei si el ritme d'arribades augmentés en un 10%.

L'eina estadística apropiada és l'interval de confiança per a la diferència d'esperances: Si $\theta_1 = \mathrm{E}[X_1]$ i $\theta_2 = \mathrm{E}[X_2]$ són les esperances de les dues configuracions, volem un interval de confiança per $\theta_1 - \theta_2$. Ens preguntem si és millor fer dues simulacions independents, obtenint dues mostres independents, o correlacionar-les d'alguna manera.

En Inferència Estadística s'estudien dues situacions: Mostres independents i dades aparellades. En el cas de mostres independents, es té que, per mides de mostra grans n_1 i n_2,

$$\frac{\bar{X}_1 - \bar{X}_2 - (\theta_1 - \theta_2)}{\sqrt{\frac{\mathrm{Var}[X_1]}{n_1} + \frac{\mathrm{Var}[X_2]}{n_2}}} \sim N(0,1)$$

aproximadament, d'on es dedueix l'interval de confiança, amb S_1^2 i S_2^2 les corresponents variàncies mostrals,

$$\bar{X}_1 - \bar{X}_2 \pm 1.96\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

El cas de dades aparellades apareix quan un element d'una mostra està lligat conceptualment a un element de l'altra mostra. Les mides de les mostres són iguals i es té, per mostres grans, que

$$\frac{\bar{X}_1 - \bar{X}_2 - (\theta_1 - \theta_2)}{\sqrt{\frac{\mathrm{Var}[X_1 - X_2]}{n}}} \sim N(0,1)\ ,$$

i surt l'interval de confiança aproximat

$$\bar{X}_1 - \bar{X}_2 \pm \frac{1.96 S_{X_1 - X_2}}{\sqrt{n}}$$

La relació entre les variàncies de X_1, X_2, i $X_1 - X_2$ ve donada per la fórmula

$$\mathrm{Var}[X_1 - X_2] = \mathrm{Var}[X_1] + \mathrm{Var}[X_2] - 2\,\mathrm{Cov}\,[X_1, X_2]$$

Per tant, si la covariància entre les dades aparellades és positiva, la variància es redueix.

Tornant a la situació de simulació, com podem aconseguir aparellar les dades de simulacions de dues configuracions diferents d'un sistema? La base és usar en les dues simulacions els mateixos nombres aleatoris Unif$(0,1)$ que s'ocupen de generar les diverses variables aleatòries input del sistema.

Exemple 5.3.5. Tenim una cua simple amb un temps entre arribades i temps de servei exponencials amb mitjanes 1 i 1.20 usuaris per minut respectivament respectivament, i estem interessats en els períodes d'ocupació contínua del servei (des que arriba un usuari estant el servei ociós, fins que marxa un usuari deixant-lo altre cop ociós). Preveiem que hi haurà un augment del ritme d'arribades de 1 a 1.10 per minut i estem considerant intentar canviar el ritme de servei a 1.25 usuaris per minut. Volem estimar la diferència en les mitjanes dels temps d'ocupació contínua amb una i altra configuració.[4]

Podem generar primer tots els temps entre arribades a partir d'una successió $U_1, \ldots, U_n$ d'uniformes, mitjançant $A_{1,i} = -\log(1 - U_i)$ i els temps de servei mitjançant una altra successió d'uniformes $V_1, \ldots, V_n$, amb $S_{1,i} = -\frac{1}{1.20}\log(1 - V_i)$.

Després repetim amb els paràmetres del segon escenari, $A_{2,i} = -\frac{1}{1.10}\log(1 - U_i)$ i $S_{2,i} = \frac{1}{1.25}\log(1 - V_i)$.

Els outputs X_1 i X_2 que s'obtinguin de les dues simulacions són dades aparellades perquè provenen de les mateixes successions d'uniformes. Naturalment, caldrà fer moltes rèpliques amb

[4] Els valors teòrics són 5 i 20/3 respectivament en estat estacionari.

successions d'uniformes independents en cada rèplica, però compartides entre els dos escenaris. □

Veurem en el capítol següent com fer per reutilitzar una successió d'uniformes en dues simulacions diferents.

Hi ha algunes consideracions importants a fer:

- Per a què això funcioni bé, la correlació que s'indueix entre les dades aparellades finals ha de ser positiva: Valors grans de $X_{1,i}$ (mostra 1) s'han de correspondre amb valors grans de $X_{2,i}$ (el corresponent de la mostra 2), i valors petits amb valors petits.
- Aquesta correlació positiva serà possible si valors petits o grans de la uniforme també donen lloc a valors petits o grans del output que interessa. En l'exemple anterior, valors petits de U_i donen lloc a valors petits de l'exponencial i per tant a valors petits del temps entre arribades, tant si el paràmetre és 1 com si és 1.10. Podríem haver posat $\log U_i$ en lloc de $\log(1-U_i)$ (valors petits de la uniforme donaran valors grans del temps entre arribades), *sempre i quan fem servir la mateixa fórmula per als dos escenaris.*
- El mètode de inversió, sense trucs addicionals, assegura la correlació positiva entre les uniformes i la variable aleatòria de input que es genera.
- El mètode d'acceptació/rebuig és problemàtic perquè, si no es prenen precaucions, destruirà la sincronització entre les variables input generades en un escenari i les generades en l'altre.
- En general, s'ha de procurar que una uniforme usada en un escenari s'utilitzi exactament per a la mateixa cosa en l'altre escenari. Si els dos escenaris responen a configuracions essencialment diferents (diferent nombre de servidors, per exemple), segurament hi haurà complicacions de programació, i per tant feina addicional. Recordem que només estem intentant reduir la variància; cal valorar si val la pena, o augmentant la mida de la mostra en escenaris independents ja seria suficient.

6

Dades d'entrada i generació d'uniformes

6.1. Dades d'entrada

La simulació necessita *input data*, com vam discutir a l'Apartat 1.3. En una cua simple, per exemple, cal donar la llei de la quantitat d'arribades per unitat de temps (equivalentment, del temps entre arribades) i la llei del temps de servei (equivalentment, de la quantitat d'usuaris servits per unitat de temps). A partir d'aquestes dades d'entrada, es poden estudiar molts outputs diferents via simulació.

Determinar les lleis de les dades d'entrada és una qüestió de *modelització*. Hem de conèixer bé les propietats de les lleis més habituals per decidir quina s'adapta més a la nostra situació. Les lleis de probabilitat "amb nom" venen agrupades en famílies, que depenen d'un o més paràmetres. Per exemple, la llei exponencial no és una sola llei, sinó una família que depèn d'un paràmetre; la llei normal és una família que depèn de dos paràmetres.

Per determinar uns paràmetres concrets hem de fer necessàriament observacions de la realitat i estimació. Si no es pot observar la realitat (potser perquè encara no existeix), la simulació servirà per fer hipòtesis "què passaria si..." o per comparar sistemes amb diferents paràmetres, com hem vist a l'Apartat 5.3.

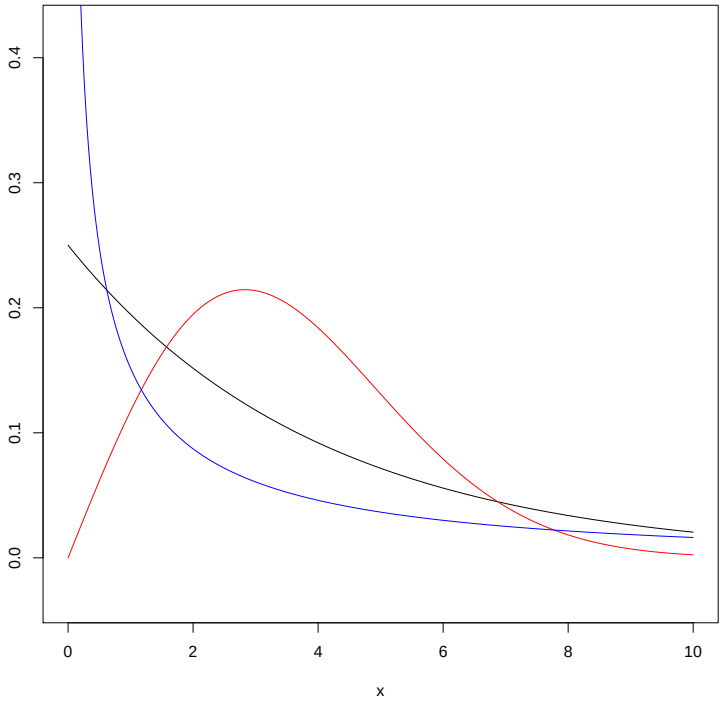
Modelitzar és un art que requereix pràctica. Veurem algunes idees.

6.1.1 Modelitzar el temps

En una cua simple en què les arribades es produeixen independentment les unes de les altres, la llei de Poisson és adequada, pels raonaments que vam fer a l'Apartat 1.3. Però possiblement amb un paràmetre depenent del temps $\lambda(t)$, donant lloc a un procés de Poisson no-estacionari (Apartat 3.2.3).

El temps de servei en una cua simple es modela típicament amb una llei Exponencial, si no hi ha altra informació. Però no sempre és el model més adequat. La llei Exponencial dóna lloc a resultats teòrics molt bonics per a l'estat estacionari, però cal pensar bé si escau en cada cas concret.

Si sabem que un usuari porta ja 10 minuts essent servit a la caixa del supermercat, la propietat de falta de memòria de la llei exponencial ens diu que el temps que li queda segueix la mateixa llei Exponencial, i per tant és com si acabés de començar a ser servit. Si creiem que això no reflecteix la realitat, aleshores una llei de Weibull pot ser millor, perquè introdueix un altre paràmetre "shape" que deforma la llei exponencial.



$\text{Exp}(\lambda): \ \lambda e^{-\lambda x}$

$\text{Weibull}(a,b):$

$\frac{a}{b}(\frac{x}{b})^{a-1}e^{-(\frac{x}{b})^a}$

En aquesta figura tenim, en negre, la densitat de l'Exponencial de paràmetre $\lambda = 0.25$; en blau, la densitat de la Weibull de paràmetres $b = 4$ (fa el paper de $\lambda = 0.25$ de l'Exponencial) i $a = 0.5$; en vermell, la Weibull de paràmetres $b = 4$ i $a = 2$. En R:

```
dexp(x, rate=0.25) = dweibull(x, shape=1, scale=4)
dweibull(x, shape=0.5, scale = 4)
dweibull(x, shape=2, scale = 4)
```

Veiem que un paràmetre shape més petit que 1 farà que s'acumulin valors de x més petits que per l'Exponencial; en canvi shapes més grans que 1 produiran valors una mica més grans. Però també, si mirem cap a l'altre extrem (les "cues" de les densitats), veiem que shapes més grans que 1 van cap a zero molt més ràpidament; i shapes més petits que 1 van més lents[1].

Un cop decidim utilitzar la llei de Weibull per al temps de servei, caldrà fer observacions per ajustar els paràmetres a i b el millor possible.

Hi ha moltes lleis que tenen una densitat semblant a la Weibull. Per exemple, les lleis Gamma, que també generalitzen l'Exponencial ($\text{Gamma}(1,\beta) = \text{Exp}(\frac{1}{\beta})$), i els paràmetres també s'anomenen "shape" el primer i "scale" el segon). A la família Gamma pertany també la subfamília khi-quadrat: $\chi^2(m) = \text{Gamma}(\frac{m}{2}, 2)$. La LogNormal també té una forma similar a la Gamma.

La llei Beta pot imitar la mateixa forma, però amb suport en $[0,1]$, per tant sense cua. Evidentment, la podem estirar i desplaçar perquè faci la mateixa forma en un interval qualsevol $[a,b]$.

Exercici 6.1.1. Com generaríeu una variable amb la forma d'una llei Beta, però amb suport en un interval $[a,b]$ qualsevol, suposant que sabem generar una llei Beta ordinària? □

El problema de modelització més important amb totes aquestes lleis que s'assemblen és "el pes de les cues" de la densitat. Una cua "pesada", que tendeix cap a zero lentament, produirà de tant en tant valors molt més alts que una cua "lleugera". Això pot tenir molta incidència en els resultats de la simulació.

[1] Encara que no es veu a la figura, la línia blava acaba creuant la negra.

Exercici 6.1.2. Dibuixeu juntes les densitats de dues lleis Weibull, amb `scale=4`, i `shape=1` (= Exp(0.25)) i `shape=1.1`, entre 0 i 10. Quina cua és més pesada?

La diferència no sembla gaire important. Però si generem una mostra de 100 valors de cadascuna de les distribucions i les ordenem, apreciarem la diferència en els valors alts. Una d'elles produeix més valors alts que l'altre. Pot ser que fins i tot hi hagi algun valor molt extrem.

```
print(sort(rweibull(n=100, shape=1, scale=4)))
print(sort(rweibull(n=100, shape=1.1, scale=4)))
```

També podem observar que amb els valors petits va a l'inrevés. Fent la diferència dels dos vectors ordenats obtinguts podeu apreciar millor aquests aspectes. □

6.1.2 La llei Normal i lleis truncades

La família Normal modelitza aquelles magnituds que són la suma de la contribució de molts petits factors. En simulació, la llei Normal no té tanta aplicació per modelitzar dades d'entrada. La sortida segurament tampoc serà una llei normal, però gràcies al Teorema Central del Límit, si la mostra és molt gran, la variable $\frac{\bar{X}-\mathrm{E}[X]}{S/\sqrt{n}}$ s'assembla molt a una normal, i això és el que permet fer intervals de confiança asimptòtics per a $\mathrm{E}[X]$.

Si estem tractant amb quantitats positives com ara el temps entre esdeveniments, la llei Normal en principi és inadequada, perquè eventualment podria produir valors negatius. A l'exemple del tutorial de `simmer` s'utilitzen llei normals per modelar temps de servei i temps entre arribades. Amb una mitjana 15 i desviació tipus 1 la probabilitat de prendre un valor negatiu (feu `pnorm(mean=15, sd=1, q=0)`) és ridículament petita. Però no és bona idea confiar en això. A més, si `simmer` veu un valor negatiu de temps, atura la simulació.

Una possibilitat per usar lleis normals com a dades d'entrada que han de ser positives (o més grans que un cert valor) és *truncar* la distribució. És molt fàcil calcular la funció de distribució d'una llei truncada, tant si es tracta de la Normal com de qualsevol altre: Suposem que volem generar variables X que segueixin la llei de $Y \sim F$ condicionada a què $Y > a$. Si G és la funció de distribució que busquem,

$$G(x) = P\{X \le x\} = P\big\{Y \le x \big/ Y > a\big\} = \frac{P\{a < Y \le x\}}{P\{Y > a\}} = \frac{F(x) - F(a)}{1 - F(a)} \qquad (x > a)$$

A partir d'aquí, si podem aplicar el mètode d'inversió a F, podem fer

$$G(x) = u \;\Rightarrow\; \frac{F(x) - F(a)}{1 - F(a)} = u \;\Rightarrow\; x = F^{-1}\big(u + (1-u)F(a)\big)$$

I si podem aplicar acceptació/rebuig a la densitat $f(x) = F'(x)$, aleshores derivant $G(x)$ tenim la densitat

$$g(x) = \frac{1}{1 - F(a)} f(x)$$

que només difereix de f en una constant, i també podrem aplicar acceptació/rebuig a h.

Exercici 6.1.3. Sposem que Sabem que un cert temps de servei segueix una llei Weibull de paràmetres a, b, però que hi ha un temps mínim de 2 minuts. Suposem que ho volem modelar com la mateixa Weibull condicionada a ser més gran que 2. Trobeu explícitament la funció de distribució d'aquesta llei truncada (recordeu que la Weibull té funció de distribució $F(x) = 1 - \exp\{-(x/b)^a\}$, $x \ge 0$). Invertiu-la explícitament.

De vegades a la llei Weibull s'hi incorpora un tercer paràmetre c, que la desplaça cap a la dreta c unitats. La funció de distribució és $H(x) = 1 - \exp\{-((x-c)/b)^a\}$, $x \ge c$. Comproveu que aquesta llei desplaçada és diferent de la llei truncada anterior (podeu posar $c = 2$ o fer-ho en general), excepte en un cas. Quin? □

Exercici 6.1.4. Trobeu la funció de distribució G de la llei que resulta de truncar (condicionar) la llei de $Y \sim F$ a $a \leq Y \leq b$. □

[El capítol 6 del llibre de Law està dedicat a la modelització de dades d'entrada i és molt complet. En particular, hi ha gràfics i propietats de moltes lleis de probabilitat. Tot aquest material es pot trobar fàcilment a internet, però aquí està convenientment agrupat.]

6.1.3 Modelitzar lleis discretes

Les lleis discretes que apareixen a la pràctica solen venir ben determinades per la situació física que han de modelar. Per exemple, si creiem que certs esdeveniments $A_1, \ldots, A_k$ passen amb probabilitats $p_1, \ldots, p_k$, això exactament és el que hem de generar.

I per altra banda, les lleis discretes clàssiques responen a definicions físiques concretes. Les més habituals:

- La llei Binomial de paràmetres (n, p) és la quantitat de vegades que succeeix un esdeveniment de probabilitat p en n experiments idèntics independents.
- La llei Geomètrica de paràmetre p és la quantitat d'experiments que cal fer fins que aparegui per primer cop un esdeveniment donat de probabilitat p.
- La llei Binomial Negativa de paràmetres (s, p) és la quantitat d'experiments que cal fer fina que aparegui per s-èsima vegada un esdeveniment donat de probabilitat p.

Per tant, no sol ser problema modelitzar correctament una llei discreta.

6.1.4 Ús de dades històriques

Hi ha tres maneres d'usar dades històriques:

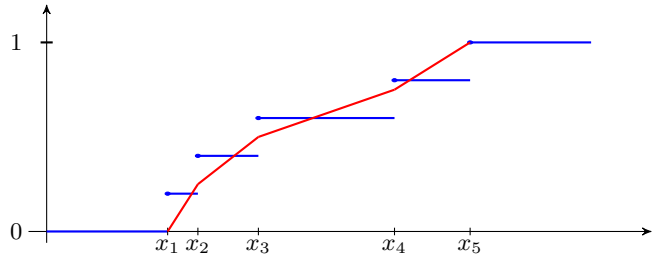
a) Observacions prèvies d'una variable poden servir per ajudar-nos a modelitzar-la amb la família i paràmetres correctes. Aquesta és segurament la situació ideal la majoria de vegades, tot i les dificultats de modelització que hem discutit en el Subapartat 6.1.1.

   Hi ha tests estadístics (*goodness of fit tests*) per decidir si una certa distribució s'ajusta a unes dades o no. El problema és que aquests tests serveixen per descartar lleis que no s'ajusten, però no per assegurar que una llei s'ajusta bé. Moltes lleis diferents es poden ajustar força bé a unes dades, però les cues són un problema. Típicament tenim poques dades de les cues i pot canviar molt el resultat d'una simulació si no modelem bé les cues (recordeu l'Exercici 6.1.2).

b) Una altra manera d'usar dades històriques és mitjançant la *funció de distribució empírica*. Tant si es tracta d'una variable discreta (els valors es poden repetir) com continua (tots els valors històrics són diferents), podem mostrejar dels valors observats amb la probabilitat igual a la proporció amb què han estat observats (en el cas continu, possiblement de manera equiprobable).

   L'avantatge és que estem usant valors que realment han succeït en el passat. El desavantatge, particularment amb variables contínues, és que no generarem mai valors intermedis que no han succeït abans però que podrien succeir.

   Per atenuar aquest inconvenient, podem fer una aproximació que sigui contínua. Per exemple, a la figura següent tenim una aproximació raonable F (en vermell) a una funció de distribució empírica (en blau). L'aproximació serà més bona com dades hi hagi.



$$F(x) = \frac{1}{n-1}\left(i - 1 + \frac{x - x_i}{x_{i+1} - x_i}\right), \quad \text{si } x_i < x \leq x_{i+1}, \quad i = 1, \ldots, n-1$$

Té l'inconvenient que no es generaran mai valors per sota del mínim o per sobre del màxim de les dades històriques.

c) També es poden usar les dades històriques mateixes, tal qual i en el mateix ordre en què estan registrades. Això permet comparar un sistema en funcionament contra una configuració alternativa. O per validar una simulació comparant el que està sortint amb el que va sortir a la realitat.

L'inconvenient és que normalment no hi ha prou quantitat de dades per fer una simulació consistent.

6.2. Generació de variables uniformes

L'ingredient bàsic per a la simulació és un flux continu de números $u_1, u_2, \ldots$ que es puguin interpretar com observacions d'una successió de variables aleatòries $U_1, U_2, \ldots$ independents totes elles i amb llei $\text{Unif}(0, 1)$.

Totes les implementacions en software són per força deterministes, el que és contradictori amb l'aleatorietat que es vol. Per això també es parla dels $u_1, u_2, \ldots$ com a números *pseudoaleatoris*. La qüestió és que no cal que siguin aleatoris, sinó només que ho semblin; és a dir, que cap procediment estadístic pugui demostrar que no ho són.

Tots els algorismes generadors d'aquests nombres (pseudo)aleatoris són de tipus iteratiu i responen al esquema següent:

- Hi ha un conjunt finit d'*estats* E.
- Els estats evolucionen aplicant iterativament una funció: $e_{n+1} = f(e_n)$.
- Hi ha una funció $g\colon E \to (0, 1)$, que produeix $u_n = g(e_n)$.
- Hi ha un estat inicial e_0, aleatori o determinista, anomenat la *llavor* (*seed*).

Que E sigui finit implica:

- Que només un subconjunt finit de nombres reals en $(0, 1)$ pot ser un dels u_n.
- Que els e_n es repetiran en algun moment i per tant la seqüència de u_n també. El més petit k tal que $e_{n+k} = e_n$ s'anomena el *període* del generador.

El primer punt no és un inconvenient greu. Sempre cal arrodonir a una determinada precisió finita. Si hi ha prou decimals, això no té importància.

El segon punt és l'inconvenient principal dels generadors mal dissenyats. Si el període és curt i en una simulació necessitem més números que el període, començarem a repetir la seqüència i totes les conclusions queden invalidades.

Els primers algorismes generadors de nombres aleatoris (c. 1950) eren *generadors lineals congruencials*. Els algorismes més bons actuals són sofisticacions de la mateixa idea, que és molt fàcil d'entendre:

- El conjunt d'estats són els nombres enters $E = \{1, \ldots, M-1\}$, per un M gran.
- Avancem a l'estat següent mitjançant la fórmula

$$e_{n+1} = Ae_n + C \pmod{M}$$

  per certes constants A i C, i on mod M vol dir el residu de la divisió entera per M.
- Els nombres aleatoris generats són $u_n = \frac{e_n}{M}$.

Està clar que el període més llarg possible és M, perquè hi ha $M-1$ estats. Per tant convé que M sigui gran. Però no està garantit que el període sigui M, sinó que podria ser més curt. Això depèn de com s'escullin A, C i M. Per exemple, per a què el període sigui màxim és necessari que C i M siguin primers entre sí. Però calen encara altres condicions.

Típicament s'agafa M de la forma $M = 2^p$. Això és degut a què el càlcul del residu és molt eficient si dividim per una potència de 2: Per exemple, si $M = 2^{32}$, els estats e_n ens caben en un enter de 4 bytes (32 bits). Al fer $Ae_n + C$ ens pot sortir un número més gran. Quan el posem en un enter de 32 bits es produeix un *integer overflow*, i es perden tots els bits més enllà (a l'esquerra) del bit 32. I el que queda és exactament el residu de dividir per 2^{32}.

Agafant concretament aquest $M = 2^{32}$ i els A i C adequats, podem tenir un període de $2^{32} \approx 4 \times 10^9$, que no està malament. Millor està si agafem 8 bytes (64 bits), que dona un període de $2^{64} \approx 1.6 \times 10^{19}$.

Els generadors bons actuals combinen diferents generadors i barregen entre sí les successions generades. L'algorisme que fa servir el R per omissió es diu *Mersenne–twister* (1998), i té l'allucinant període de $2^{19937} - 1$. Aquest número és impossible d'imaginar[2]. El Mersenne–twister també és el generador que utilitza la funció `random()` del mòdul `random.py` de Python, i és un dels moltíssims que ofereix la biblioteca de GSL del C.

Un període llarg no és l'única bona propietat que ha de tenir un generador de nombres aleatoris. Els nombres aleatoris d'un generador lineal congruencial, quan se'ls agafa agrupats per parelles solapades (u_1, u_2), (u_2, u_3), (u_3, u_4), ... o per trios (u_1, u_2, u_3), (u_2, u_3, u_4), (u_3, u_4, u_5),..., o en general en dimensió d, poden acumular-se al llarg d'uns pocs hiperplans que travessen el cub d-dimensional $[0, 1]^d$.

> [Vegeu la Figura 7.2, pàg 415 de Law. És tracta de les parelles solapades de l'algorisme lineal congruencial amb $A = 18$, $C = 0$, $M = 101$. Apart que el període sigui curt, és obvi que alguna cosa falla en l'aleatorietat esperada. El cas de la Figura 7.3 és més exagerat, posant $A = 2$.
>
> Això són exemples buscats expressament, però la Figura 7.4 correspon al generador RANDU (c. 1960), que va formar part dels compiladors de Fortran durant molts anys (potser fins els 1990's); en particular en el de l'ordinador IBM/360, molt utilitzat en els 1960's i 1970's. Posa $A = 65539$, $C = 0$ i $M = 2^{31}$. Aquí veiem com falla l'aleatorietat quan mirem els nombres de tres en tres.]

El Mersenne–twister no pateix aquest fenòmen fins a la dimensió $d = 624$.

Cal tenir en compte que en compiladors/entorns diferents, el mateix algorisme pot donar resultats diferents, per detalls d'implementació. Si s'ha de reproduir una seqüència de nombres aleatoris, és recomanable fer-ho amb la mateixa implementació de l'algorisme. També és recomanable no usar "la funció estàndard que ve incorporada en el compilador/interpret", sinó una library/package/module ben documentada.

[2] Per fer-nos una idea, si generem un número aleatori cada nanosegon (10^{-9} segons), que és força optimista, i agafant com a unitats l'Edat De l'Univers (EDU), tardaríem $\geq 2^{19848}$ EDUs en esgotar el període. Bé, doncs segueix essent impossible d'imaginar ...

6.2.1 En R

En R podem demanar quins generadors aleatoris s'estan usant per a les uniformes, per a les normals i per a les uniformes discretes, i canviar-los, amb la funció `RNGkind()`:

```
RNGkind()
RNGkind(kind="   ", normal.kind="   ", sample.kind="   ")
```

Podeu fer `help(RNGkind)` per veure les opcions. (El mètode per a uniformes discretes va canviar en la versió 3.6.0 del R, i s'ofereix la possibilitat de tornar a l'anterior per raons de reproducibilitat.)

L'estat del generador es guarda en l'objecte `.Random.seed`. Aquest objecte no està definit fins que no es crida per primera vegada alguna funció que el necessita. Per exemple,

```
print(.Random.seed) # No existeix
runif(n=1)
print(.Random.seed) # Ara sí existeix
```

L'estat del Mersenne–twister és un vector de 626 enters. En realitat el primer enter només codifica quins generadors s'estan usant. La resta els veiem com enters positius i negatius, tot i que en el codi C subjacent són enters sense signe.

Es possible guardar i tornar a carregar l'estat del generador:

```
save(.Random.seed, file="   ", ascii=TRUE) # Guardem l'estat en un fitxer,
rm(.Random.seed) # l'eliminem, i
load(file="   ") # el tornem a carregar
```

D'aquesta manera podríem continuar una seqüència de nombres aleatoris en sessions diferents, o guardar l'estat al principi per poder usar els mateixos nombres aleatoris en una altra simulació. Però el mètode recomanat per a això és usar la funció `set.seed()`:

```
set.seed(seed= ..., kind="   ", normal.kind="   ", sample.kind="   ")
```

El primer argument ha de ser un número entre -2^{31} i 2^{31}, o `NULL`. Si no s'especifiquen els altres, no hi ha canvis respecte els que estan en ús. El valor `NULL` reinicialitza l'objecte `.Random.seed`, combinant el rellotge intern de l'ordinador i el número d'identificació del procés en el sistema operatiu. Això és el que passa el primer cop que cridem una funció que necessita el `.Random.seed` en cada sessió[3], si abans no hem fet `set.seed()`.

No hi ha cap motiu per usar `set.seed()` si no s'ha de reproduir la mateixa successió de valors aleatoris. És erroni començar tot programa amb el típic `set.seed(42)` o similar.

6.2.2 Aleatorietat per hardware/natura

Existeix l'aleatorietat autèntica a la natura? Sí. Els estats quàntics de la matèria són distribucions de probabilitat que no es concreten en un valor fins que no es fa una observació. Això permet construir generadors quàntics de números aleatoris.

Hi ha aparells que proporcionen nombres aleatoris, en diferents formats en base a propietats quàntiques. Per exemple, els aparells que ven `https://www.idquantique.com`, externs, que es connecten al port USB, o interns, connectats a la placa mare, es basen en l'emissió de fotons contra un mirall que pot deixar passar el fotó o reflectir-lo amb probabilitat $\frac{1}{2}$. Per tant, en principi, cada fotó produeix un bit, perfectament aleatori i independent del bit anterior. A partir dels bits es poden construir valors uniformes en $(0, 1)$ amb la precisió que es vulgui, usant la quantitat adequada de bits.

[3]La sessió de R (*workspace*) es pot guardar, i al tornar a arrencar el R es continua la sessió anterior. En particular, l'estat del generador aleatori seria el mateix que vam deixar. Però en general no és recomanable guardar sessions de R.

El problema és que si cal reutilitzar la successió l'hem de guardar en algun lloc. Bàsicament per això, i perquè la generació dels nombres aleatoris pot ser lenta, que no s'utilitzen gaire aquests aparells.

Hi ha alguns llocs web que proporcionen *streams* de nombres aleatoris, en diferents formats, en base a propietats quàntiques i altres. Per exemple: `http://qrng.anu.edu.au` es basa en fluctuacions quàntiques del buit. El problema és el mateix que amb el hardware local, i possiblement encara amb més lentitud.

Exercici 6.2.1. El generador quàntic mencionat té un petit problema. Què passa si no està ben calibrat i la probabilitat de reflectir el fotó no és exactament $\frac{1}{2}$?

Això té solució (i és el que realment fa el generador). Diguem que 0 és deixar passar i 1 és reflectir. El mètode és:

1. Llançar dos fotons.
2. Si surt 01, aleshores declarar 0.
3. Si surt 10, aleshores declarar 1.
4. Si surt 00 o 11, aleshores descartar els dos fotons i tornar al pas 1.

a) Demostreu que declarem 0 o 1 amb la mateixa probabilitat $\frac{1}{2}$, independentment de la probabilitat que un fotó es reflecteixi.

b) Calculeu quants fotons calen, de mitjana, per declarar un resultat vàlid (això sí que dependrà del desequilibri inicial).

□

Observeu que el procediment de l'exercici anterior es pot aplicar igualment bé al llançament d'una moneda sospitosa.