Stochastické simulačné metódy

Radoslav Harman a Samuel Rosa, KAMŠ, FMFI UK

3. októbra 2019

Obsah

1 Úvod

Uvažujme problém analýzy systému, ktorý pozostáva z mnohých náhodných prvkov. Predstavme si napríklad situáciu na dopravnom uzle, v ktorej sa navzájom komplikovaných spôsobom kombinujú okamihy príchodu vozidiel. Je veľmi nepravdepodobné, že by sme takýto systém vedeli popísať analyticky riešiteľnými matematickými rovnicami, bez toho, aby náš model stratil podstatné prvky. Je však obvykle relatívne jednoduché simulovať dynamiku celého systému na počítači a sledovaním tejto simulácie pochopiť ako sa systém správa.[1] Pre takúto simuláciu je kľúčovým 1) vedieť počítačom generovať vhodný typ náhodnosti a 2) štatisticky spracovať dáta, ktoré nám simulácia poskytne. Metódy riešenia týchto dvoch úloh sú náplňou predmetu Stochastické simulačné metódy.

[1] Prípadne sa môžeme obmenami parametrov simulovaného systému snažiť dosiahnuť požadované správanie; takýmito optimalizačnými problémami sa však na tomto predmete nebudem zaoberať.

Okrem simulácie komplexných systémov sa generovanie náhodnosti a štatistického spracovania simulačných dát používa napríklad v oblastiach:

- Simulovanie správania používateľov sociálnej siete alebo klientov firmy;
- Algoritmy stochastického náhodného výberu z veľkej množiny dát;
- Monte-Carlo metódy výpočtu integrálov a riešenia rovníc;
- Testovanie správnosti a citlivosti matematických a štatistických metód;
- Generovanie náhodných prvkov v optimalizačných a iných algoritmoch;
- Kryptografické aplikácie, počítačová grafika, hry, ...

V tomto učebnom texte používame jazyk `R` kvôli svojej jednoduchosti, všeobecnej dostupnosti a rozšírenosti, ako jazyk na presný a testovateľný popis algoritmov. Na náročnejšie reálne simulácie je však vhodné použiť špecializovanú knižnicu pre `R`, alebo úplne iný sofvér, napríklad Python, Matlab, alebo C++.

2 Generovanie realizácií z rovnomerného rozdelenia na intervale

2.1 Základné princípy a veta o inverznej transformácii

Základom pre simulovanie náhodnosti na počítači sú metódy generovania náhodných čísel, pričom pod pojmom "náhodné čísla" rozumieme realizácie nezávislých náhodných premených s *rovnomerným* rozdelením na intervale $[0, 1]$. Dôvodov ústredného postavenia rovnomerného rozdelenia v simulačných metódach je viacero. Po prvé, rovnomernú náhodnosť vieme pomerne jednoducho vyprodukovať algoritmicky. Po druhé, máme k dispozícii množstvo silných testov kvality generátorov realizácií z rovnomerného rozdelenia. A po tretie, je vypracovaných veľa postupov, ako transformovať práve rovnomerné náhodné čísla na realizácie náhodných premenných s *akýmkoľvek* zadaným rozdelením. Nasledovná veta ukazuje, že v princípe stačí použiť transformáciu náhodných čísel kvantilovou funkciou cieľového rozdelenia.[2]

Veta 2.1 (Veta o inverznej transformácii)**.** *Nech $F : \mathbb{R} \to [0, 1]$ je distribučná funkcia a nech G je jej kvantilová funkcia, t.j. $G(u) = \sup\{x \in \mathbb{R} : F(x) \leq u\}$ pre všetky $u \in (0, 1)$. Nech náhodná premenná U má rozdelenie $R(0, 1)$. Potom náhodná premenná $G(U)$ má distribučnú funkciu F.*

Dôkaz. Túto vetu je možné dokázať veľmi jednoducho pre prípad, že F je spojitá funkcia, rastúca na celom $\mathbb{R}$; pozri príklad. Dokážme však túto vetu v úplnej všeobecnosti, t.j. tak, aby sme zahrnuli aj distribučné funkcie diskrétnych náhodných premenných.

[2] Avšak nielen rovnomerné, ale aj ľubovoľné iné spojité rozdelenie je možné transformovať na akékoľvek cieľové rozdelenie. Premyslite si prečo. (Pomôcka: Ak je X náhodná premenná so spojitým rozdelením a rastúcou distribučnou funkciou F, tak $F(X) \sim R(0, 1)$.)

Najprv ukážeme, že pre každé $z \in \mathbb{R}$ a pre každé $u \in (0,1)$ platí $G(u) < z \Leftrightarrow u < F(z)$, alebo ekvivalentne, že pre každé $z \in \mathbb{R}$ a pre každé $u \in (0,1)$ platí $G(u) \geq z \Leftrightarrow F(z) \leq u$.

Označme $M_u = \{z \in \mathbb{R} : F(z) \leq u\}$. Ak $F(z) \leq u$, potom $z \in M_u$ a teda $G(u) = \sup M_u \geq z$. Tým sme ukázali implikáciu "$\Rightarrow$". Opačná implikácia plynie nasledovne: Nech $G(u) = \sup M_u \geq z$. Z definície supréma dostávame, že pre každé $n \in \mathbb{N}$ existuje $z_n \in M_u$, t.j. z_n spĺňajúce $F(z_n) \leq u$, pre ktoré $z - 1/n \leq z_n$. Z neklesajúcosti F máme $F(z - 1/n) \leq F(z_n) \leq u$ a zo spojitosti zľava dostávame $F(z) \leq u$. Tým sme ukázali aj opačnú implikáciu.

Teraz už môžeme vyjadriť distribučnú funkciu náhodnej premennej $G(U)$, kde $U \sim R(0,1)$ v bode $z \in \mathbb{R}$: $F_{G(U)}(z) = P[G(U) < z] = P[U < F(z)] = F(z)$. □

Poznamenajme, že predchádzajúcu vetu vieme efektívne aplikovať len vtedy, ak máme k dispozícii rýchly algoritmus na výpočet funkčných hodnôt kvantilovej funkcie, čo je skôr výnimka ako pravidlo. K metódam založeným na vete o inverznej transformácii sa ešte vrátime v častiach 3.1 a 4.1. Navyše, takzvaná metóda podmienených rozdelení zaručuje, že ak vieme generovať z akýchkoľvek jednorozmerných rozdelení pravdepodobnosti, tak v princípe vieme generovať realizácie náhodných *vektorov* s akýmkoľvek známym rozdelením.

Cieľom tejto časti je stručne popísať generovanie náhodných čísel a metódy štatistického overovania ich kvality. Nebudeme sa však podrobne zaoberať najefektívnejšími (no súčasne veľmi komplikovanými) metódami na generovanie náhodných čísel, ktoré sú implementované v štandardných programovacích jazykoch a nástrojoch na simulácie a analýzu dát. Dôležité pre nás budú základné *princípy*.

2.2 Fyzikálne a algoritmické generátory náhodných čísel

Prvou zaujímavou možnosťou je použiť na generovanie náhodnosti fyzikálne princípy, napríklad zariadenia využívajúce atómový rozpad, atmosferický šum a podobne. Výhodou fyzikálnych generátorov je to, že generujú "pravú" náhodnosť, aspoň pokiaľ akceptujeme principiálnu nepredvídateľnosť kvantovofyzikálnych javov[3]. Avšak praktické nevýhody takýchto zariadení obvykle (alebo možno *zatiaľ*) prevažujú nad ich výhodami a v súčasnosti sú na generovanie realizácií náhodných čísel používané väčšinou algoritmické generátory založené na deterministických rekurenciách.

Algoritmické generátory nám síce neposkytujú pravú náhodnosť[4] a ich výstupom, striktne vzaté, nemôžu byť realizácie nezávislých náhodných premenných s rovnomerným rozdelením. Avšak deterministickú povahu realizácií kvalitných algoritmických generátorov a ich prípadné odchýlky od rovnomernosti nie je možné odhaliť ani pomocou veľmi prísnych štatistických testov. Takéto generátory potom úplne postačujú na simulácie spojené s riešením praktických problémov. Navyše, výhodou algoritmických

[3] Nebudeme sa hlbšie zaoberať otázkou existencie pravej náhodnosti.

[4] Preto sa tieto náhodné čísla zvyknú niekedy nazývať termínom *pseudo*náhodné čísla. Túto terminológiu však nebudeme používať.

generátorov voči fyzikálnym generátorom je vysoká rýchlosť produkovania realizácií na počítači pri nízkych nákladoch, existencia teoretických zdôvodnení ich kvality, stabilita v čase a možnosť reprodukovať tie isté postupnosti[5].

Veľkú triedu algoritmických generátorov náhodných čísel tvoria takzvané rekurentné generátory, s ktorými sa stručne zoznámime.

Definícia 2.1. *Nech* $d, m \in \mathbb{N}$, $\mathcal{M}_m = \{0, ..., m-1\}$. *Všeobecným rekurentným generátorom rádu* d *nazveme zobrazenie* $f : \mathcal{M}_m^d \to \mathcal{M}_m$. *Realizáciou generátora* f *pre počiatočný stav* $(x_1, ..., x_d) \in \mathcal{M}_m^d$ *nazveme postupnosť* $(x_i)_{i=1}^{\infty}$, *kde* $x_i = f(x_{i-1}, ..., x_{i-d})$ *pre všetky* $i > d$.

Ak $(x_i)_{i=1}^{\infty}$ je realizáciou (vhodného) rekurentného generátora $f : \mathcal{M}_m^d \to \mathcal{M}_m$, potom hodnoty $(x_i/m)_{i=1}^{\infty}$ považujeme za postupnosť náhodných čísel. Je zrejmé, že takáto postupnosť je podmienene deterministická[6]; náhodným prvkom v reálnych simuláciách však býva voľba počiatočného stavu (takzvaný *seed*), ktorý môžeme založiť napríklad na systémovom čase[7], ale mohli by sme na to použiť povedzme aj náhodnosť vygenerovanú interakciou ľudského užívateľa s počítačom, alebo iné fyzikálne zdroje náhodnosti. Samozrejme, vlastnosti rekurentného generátora závisia komplikovaným spôsobom na číslach d, m, na funkcii f a tiež na počiatočnom stave. Ukazuje sa, že kvalitné generátory môžu mať pomerne malý rád d, hoci pochopiteľne musia mať veľkú množinu stavov, ktorá je určená číslom m. Najkritickejšia je voľba funkcie f; prevažná väčšina funkcií f zvolených bez veľmi dôkladnej analýzy (aj ak je predpis funkcie f veľmi komplikovaný) vedie ku generátorom s neuspokojivými vlastnosťami.

Uvedomíme si, že množina stavov akéhokoľvek generátora z definície 2.1 je len konečná (má m^d prvkov), takže nech by sme začali s akýmkoľvek počiatočným stavom, po dostatočne veľa iteráciách musíme dôjsť do niektorého stavu v ktorom sme už boli, čím sa generovaná postupnosť "zacyklí".

Definícia 2.2. *Nech* $(x_i)_{i=1}^{\infty}$ *je realizácia rekurentného generátora* f *z definície 2.1 pre počiatočný stav* $(x_1, ..., x_d) \in \mathcal{M}_m^d$. *Nech* n *je najmenšie číslo s vlastnosťou* $x_n = x_k, x_{n+1} = x_{k+1}, \ldots, x_{n+d-1} = x_{k+d-1}$ *pre nejaké* $0 \leq k < n$. *Potom číslo* $n - k$ *nazývame periódou generátora* f *pre počiatočný stav* $(x_1, ..., x_d) \in \mathcal{M}_m^d$.

Nutnou, ale nie postačujúcou podmienkou kvality rekurentného generátora je dostatočne dlhá perióda pre všetky prípustné počiatočné stavy.

[5] To oceníme napríklad pri odlaďovaní simulačných programov. Samozrejme, aj ak pracujeme s fyzikálnym generátorom, tak môžeme reprodukovať tie isté postupnosti, ale musíme si ich priebežne ukladať do pamäte. Pri deterministických generátoroch s stačí pamätať *seed*; viď ďalej.

[6] Tým myslím to, že ak je známy počiatočný stav, tak všetky prvky tejto postupnosti sú nenáhodné.

[7] Alebo môžeme *nechať si ho vygenerovať* zo systémového času na to určeným príkazom. Ak len priamo generujeme bez voľby seed-u, procedúra si obvykle vygeneruje seed automaticky.

2.3 Lineárne kongruenčné generátory prvého rádu

Spomedzi rekurentných generátorov sú najznámejšie[8] lineárne kongruenčné generátory prvého rádu.

Definícia 2.3. *Nech $a, c, m \in \mathbb{N}$. Lineárnym kongruenčným generátorom prvého rádu $LCG(a, c, m)$ nazývame zobrazenie $f : \mathcal{M}_m \to \mathcal{M}_m$ definované $f(x) = ax + c \mod m$. Čísla a, c, m nazývame postupne multiplikátor, inkrement a modulus. Ak $c = 0$, tak tento generátor nazývame multiplikatívny, inak ho nazývame zmiešaný.*

Príklad 2.1. *Generátor $LCG(6, 0, 8)$ je zobrazenie, ktoré môžeme znázorniť nasledovne: $3 \to 2 \to 4 \to 0 \to 0 \to ...$, $7 \to 2 \to 4 \to 0 \to 0 \to ...$, $1 \to 6 \to 4 \to 0 \to 0 \to ...$, $5 \to 6 \to 4 \to 0 \to 0 \to ....$ Tento generátor teda degeneruje do nuly pre všetky počiatočné stavy. Generátor $LCG(7, 0, 11)$ môžeme zapísať takto: $... \to 9 \to 8 \to 1 \to 7 \to 5 \to 2 \to 3 \to 10 \to 4 \to 6 \to 9 \to 8 \to ...$, $... \to 0 \to 0 \to ....$ Tento generátor má periódu 10 pre akýkoľvek nenulový počiatočný stav. Výslednú postupnosť môžeme síce len ťažko použiť ako základ seriózneho generátora náhodných čísel, ale očividne sa jedná o zlepšenie voči generátoru $LCG(6, 0, 8)$.*

Všimnime si, že dĺžka periódy generátora $LCG(a, c, m)$ je maximálne m a dĺžka periódy multiplikatívneho generátora $LCG(a, 0, m)$ je maximálne $m - 1$ pre akýkoľvek počiatočný stav. Na potreby súčasných simulácií sa uvádza minimálna akceptovateľná dĺžka periódy 10^9, v niektorých aplikáciách však nepostačuje ani 10^{15}, pretože súčasné počítače sú už schopné generovať z algoritmických generátorov mnoho miliónov realizácií každú sekundu.

Existuje množstvo matematických tvrdení, ktoré určujú dĺžku periódy pre rôzne generátory $LCG(a, c, m)$ a počiatočné stavy x_1. Uvedieme len niektoré z nich; ich dôkazy využívajú komplikované techniky teórie čísel a stoja mimo rámec tejto základnej učebnice.

Veta 2.2. *Nech $m \geq 3$ je prvočíslo. Potom generátor $LCG(a, 0, m)$ má periódu $m-1$ pre každý počiatočný stav $x_1 \neq 0$ vtedy a len vtedy, keď je m takzvaný prvočíselný modulus pre a, čiže keď platí $a^{m-1} = 1 \mod m$ a $a^{k-1} \neq 1 \mod m$ pre všetky $1 < k < m - 1$.*

Vo všeobecnosti však nie je jednoduché nájsť dvojice čísel a a m z predchádzajúcej vety. Často sa používal napríklad $m = 2^{31} - 1$, čo je prvočíselný modulus[9] pre $a = 16807$ (zodpovedajúci LCG sa niekedy nazýva "minimal standard"), $a = 39373$, $a = 742938285$ a iné multiplikátory.

Okrem prvočíselných modulov sa z dôvodov výpočtovej jednoduchosti používali aj moduly tvaru $m = 2^\beta$, pre ktoré platí napríklad veta:

Veta 2.3. *Nech $\beta \geq 3$ je prirodzené číslo. Generátor $LCG(a, 0, 2^\beta)$ má periódu $2^{\beta-2}$ pre počiatočný stav x_1 vtedy a len vtedy, keď platí $a = \pm 3 \mod 8$ a x_1 je nepárne číslo.*

[8] Aj keď v súčasnosti už nie najpoužívanejšie; aspoň nie v "čistej forme" a určite nie na vážne aplikácie. Ak však potrebujeme veľmi jednoduchý generátor a stačí nám nízka kvalita, ako napríklad pri programovaní jednoduchý hier, LCG môže byť akceptovateľná voľba.

[9] $2^{31} - 1$ je jedno z takzvaných Mersennových prvočísel; viď ďalšia podkapitola.

Tabuľka 1: Konkrétne lineárne kongruenčné generátory. Poznamenávame, že RANDU a ANSI-C sú veľmi nekvalitné generátory a uvádzame ich len z historických dôvodov. Ostatné uvedené generátory môžu byť na nenáročné aplikácie dostatočne kvalitné.

Názov	a	c	m
Goodman, Miller (1969)	16807	0	$2^{31}-1$
Marsaglia (1972)	69069	0	2^{32}
Fishman, Moore (1986)	742938285	0	$2^{31}-1$
L'Ecuyer (1988)	39373	0	$2^{31}-1$
Fishman (1990)	1099087573	0	2^{32}
RANDU	65539	0	2^{31}
ANSI-C	1103515245	12345	2^{31}

Generátor $LCG(a, c, 2^\beta)$ *má periódu* $m = 2^\beta$ *pre každý počiatočný stav vtedy a len vtedy, keď platí* $a = 1 \mod 4$ *a* c *je nepárne.*

Z tejto triedy majú dostatočnú kvalitu (na mierne nenáročnejšie simulácie) napríklad generátory s $m = 2^{32}$, $c = 0$ a $a = 69069$, alebo $a = 1099087573$.[10] Skúsenosť ukazuje, že vo všeobecnosti zmiešané generátory (s nenulovou hodnotou c) negenerujú kvalitnejšie náhodné čísla ako multiplikatívne generátory (s nulovou hodnotou c).

2.4 Mersenne Twister

V súčasnosti sa už obvykle používajú oveľa komplikovanejšie rekurentné generátory než LCG, ktoré produkujú kvalitné náhodné čísla s astronomicky vysokými periódami, ako napríklad takzvané kombinované kongruenčné generátory, moderné verzie generátora "Wichmann-Hill", generátory "multiple-with-carry" a predovšetkým takzvaný "Mersenne twister".

Generátor Mersenne twister bol navrhnutý japonskými matematikmi Makoto Matsumoto a Takuji Nishimura v roku 1997 a odvtedy dosiahol nesmiernu popularitu. V súčasnosti je zrejme najviac používaným generátorom pseudonáhodných čísel, okrem iných pomocou neho generujú pseudonáhodné čísla programy R aj Matlab. Názov generátoru vyplýva z toho, že jeho periódou sú Mersennove prvočísla a že v rekurentnom vzťahu využíva zakrútenie (twist) výsledku.

Definícia 2.4. Mersennovým prvočíslom *nazývame také prvočíslo* p*, ktoré spĺňa* $p = 2^n - 1$ *pre nejaké* $n \in \mathbb{N}$.

Rekurentný predpis generátora Mersenne Twister je najjednoduchšie matematicky vyjadriteľný pre bitové zápisy celých čísel. Bitový zápis nezáporného celého čísla je postupnosť núl a jednotiek vyjadrujúca toto číslo v dvojkovej sústave, pričom jeden prvok postupnosti sa nazýva bit.

[10] Modulus 2^{32} je vhodný na nízkoúrovňovú implementáciu z toho dôvodu, že všetky čísla z príslušného generátora vieme reprezentovať presne 32 bitmi.

Definícia 2.5. Vektorom bitov dĺžky w *je riadkový vektor* $x = \left(x_{w-1}, x_{w-2}, \ldots, x_0\right)$ *spĺňajúci* $x_i \in \{0,1\}$ *pre každé* $i = 0, \ldots, w-1$. *Potom celým číslom definovaným vektorom* x *je číslo* $\sum_{i=0}^{w-1} x_i 2^i$.

Na počítanie s vektormi bitov môžeme používať základné bitové operácie po zložkách (angl. *bitwise operations*). Výhodou takýchto operácií je ich veľká výpočtová rýchlosť.

Definícia 2.6. *Nech* x *a* y *sú vektormi bitov dĺžky* w. *Potom ich* súčtom (bitovým výlučným alebo, bitwise XOR) *rozumieme vektor* $z = x \oplus y$ *po zložkách spĺňajúci* $z_i = x_i + y_i \pmod 2$, bitovým alebo (bitwise OR) *je vektor* $z = x \,\mathrm{OR}\, y$ *spĺňajúci* $z_i = 1$ *ak* $x_i = 1$ *alebo* $y_i = 1$ *a* $z_i = 0$ *inak a* bitovým a súčasne (bitwise AND) *je súčin po zložkách* $z = x \,\mathrm{AND}\, y$ *spĺňajúci* $z_i = x_i y_i$. Bitovým posunom doľava (bitwise shift left) *vektora* x *o* k *prvkov je vektor* $(x \ll k) = \left(x_{w-k-1}, x_{w-k-2}, \ldots, x_0, 0_k^T\right)$ *a* bitovým posunom doprava (bitwise shift right) *vektora* x *o* k *prvkov je vektor* $(x \gg k) = \left(0_k^T, x_{w-1}, x_{w-2}, \ldots, x_k\right)$.

Poznamenajme, že vynásobenie vektora x maticou A sprava je analogické k bežnému násobeniu maticou, akurát počítame modulo 2, t.j. $b = xA$ spĺňa $b_j = \sum_i x_i A_{ij} \pmod 2$, kde A_{ij} sú prvky matice A.

Hlavnými parametrami algoritmu Mersenne twister sú prirodzené číslo w - dĺžka vektorov, prirodzené číslo n - stupeň rekurzie, prirodzené číslo m spĺňajúce $1 \leq m \leq n$ a celé číslo r $(0 \leq r \leq w-1)$ - separujúci bod. Algoritmus generuje vektory bitov dĺžky w a z nich "rovnomerne náhodné" celé čísla na intervale $[0, 2^w - 1]$. Takže predelením každého získaného celého čísla konštantou 2^w dostávame rovnomerne rozdelené (pseudo)náhodné čísla na intervale $[0, 1]$.

Pre dané r a vektor bitov x definujme jeho hornú časť $u(x)$ ako horných $w-r$ bitov, $u(x) = (x_{w-1}, x_{w-2}, \ldots, x_r)$. Dolnú časť $l(x)$ definujeme ako dolných r bitov vektora x, $l(x) = (x_{r-1}, x_{r-2}, \ldots, x_0)$. Spojením dvoch vektorov x, y, s dĺžkami d_1 a d_2 je vektor $x|y = (x_{d_1-1}, \ldots, x_0, y_{d_2-1}, \ldots, y_0)$.

Vektory bitov dĺžky w vytvára generátor Mersenne twister pomocou rekurentného predpisu

$$x^{(k+n)} = x^{(k+m)} \oplus \left(l(x^{(k)})|u(x^{(k+1)})\right)A, \quad k = 0, 1, \ldots, \tag{1}$$

kde A je maticou zakrútenia (*twist matrix*)

$$A = \begin{pmatrix} 0_{w-1} & I_{w-1} \\ a_{w-1} & (a_{w-2}, \ldots, a_0) \end{pmatrix},$$

pričom 0_{w-1} je $w \times 1$ stĺpcový vektor núl, I_{w-1} je matica identity a $a_i \in \{0,1\}$ pre $i = 0, \ldots, w-1$. Táto rekurzia na vstupe vyžaduje prvých n vektorov $x^{(0)}, x^{(1)}, \ldots, x^{(n-1)}$ a z nich už dokáže dopočítať všetky ďalšie $x^{(j)}$. Spojenie vektorov $\left(l(x^{(k)})|u(x^{(k+1)})\right)$ je vlastne nový vektor dĺžky w, ktorý vznikol kombináciou hornej časti k-tej iterácie a dolnej časti $(k+1)$-vej iterácie, $\left(x_{w-1}^{(k)}, \ldots, x_r^{(k)}, x_{r-1}^{(k+1)}, \ldots, x_0^{(k+1)}\right)$. Takýto vektor sa

dá zapísať ako bitové "alebo" vektorov $\left(l(x^{(k)}), 0_r^T\right)$ a $\left(0_{w-r}^T, u(x^{(k+1)})\right)$. Všimnime si, že výraz xA sa dá zapísať pomocou bitových operácií. Označme $a := (a_{w-1}, \ldots, a_0)$, potom

$$xA = \begin{cases} x \gg 1, & x_0 = 0 \\ (x \gg 1) \oplus a, & x_0 = 1. \end{cases}$$

Nové iterácie algoritmu pomocou rekurentného vzťahu (1) sa teda dajú získať z predchádzajúcich iterácií pomocou bitových operácií, čo umožňuje veľkú rýchlosť algoritmu.

Kvôli zvýšeniu rovnomernosti výsledných vektorov, každý vektor $x^{(j)}$ je ešte sprava prenásobený maticou doladenia (*tempering matrix*) T, ktorá podobne ako A spĺňa, že xT sa dá vyjadriť iba pomcou bitových operácií.

Vhodnou voľbou parametrov dokážeme získať postupnosť $x^{(j)}$ s periódou až 2^{nw-r}. Štandardná verzia algoritmu má periódu $2^{624 \cdot 32 - 31} = 2^{19937} - 1$, pričom jej jednotlivé koeficienty sú $w = 32$, $n = 624$, $m = 397$, $r = 31$. Táto verzia generátora Mersenne Twister sa nazýva MT19937.

Prenásobením maticou T sa zabezpečuje zlepšenie teoretickej vlastnosti algoritmu, ktorá je popísaná v nasledujúcej definícii.

Definícia 2.7. *Pre daný vektor bitov x dĺžky w definujme* $\mathrm{trunc}_v(x)$ *ako vektor utvorený z prvých v bitov vektora x pre $1 \leq v \leq w$,*

$$\mathrm{trunc}_v(x) = (x_{w-1}, \ldots, x_{w-v}).$$

Potom hovoríme, že pseudonáhodná postupnosť bitových vektorov $x^{(i)}$ s periódou P je k-rovnomerne rozložená vo v-bitovej presnosti *ak v P zložených vektoroch*

$$\left(\mathrm{trunc}_v(x^{(i)}), \mathrm{trunc}_v(x^{(i+1)}), \ldots, \mathrm{trunc}_v(x^{(i+k-1)})\right), \quad i = 0, 1, \ldots, P-1$$

dĺžky kv, sa každý z $2^{kv} - 1$ nenulových kombinácií bitov opakuje λ-krát pre nejaké $\lambda \in \mathbb{N}$ a nulová kombinácia 0_{kv}^T sa opakuje $(\lambda - 1)$-krát. Poznamenajme, že súčty $i + j$ v horných indexoch $x^{(i+j)}$ sú počítané modulo P.

Vlastnosť k-rovnomerného rozdelenia s presnosťou v má priamočiaru geometrickú interpretáciu. Každý vektor $x^{(i)}$ preveďme na celé číslo $m_i = \sum_j x_j^{(i)} 2^i$ a to predeľme $2^w - 1$, aby sme dostali čísla na $[0, 1]$, definujme teda $u_i = m_i/(2^w - 1)$. Potom z k čísel $u_i, \ldots, u_{i+k-1}$ môžeme utvoriť vektor $\mathbf{u}^{(i)} := (u_i, \ldots, u_{i+k-1})^T$, ktorý sa bude nachádzať v k-rozmernej jednotkovej kocke $[0, 1]^k$. Od kvalitného generátora náhodných čísel sa očakáva, že takéto vektory $\mathbf{u}^{(i)}$ budú na kocke $[0, 1]^k$ rozložené čo najrovnomernejšie.

Ak interval $[0, 1]$ na každej osi rozdelíme na 2^v častí, tak prvých v bitov vektoru x určuje, do ktorej časti prislúchajúci bod u padne. Prvý bit x_1 určí, do ktorej polovice padne u; druhý bit x_2 určí do ktorej štvrtiny v rámci zvolenej polovice padne u; atď. až po v-ty bit, ktorý určuje polohu u v presnosti $1/2^v$. Takýmto rozdelením každej osi na 2^v častí sme "rozkrájali" kocku na 2^{kv} menších kociek. Postupnosť $x^{(i)}$ je k-rovnomerného rozdelenia s presnosťou v ak každá z menších 2^{kv} kociek obsahuje λ bodov $\mathbf{u}^{(i)}$, až na

kocku začínajúcu v počiatku, ktorá obsahuje $\lambda-1$ bodov. Teda čím vyššie v máme, s tým väčšou presnosťou je zaručená istá rovnomernosť, a s vyšším k získavame rovnomernosť vo viac dimenziách.

Štandardný generátor Mersenne twister MT19937 je až 623-rovnomerne rozdelený s presnosťou 32 bitov a má vysoké hodnoty k-rovnomerného rozdelenia pre presnosti $v = 1, 2, \ldots, 31$.

2.5 Testy kvality generátorov náhodných čísel

Kvalitu novonavrhnutého generátora náhodných čísel je potrebné testovať sériou overených štatistických testov. Najznámejšie sady testov generátorov náhodných čísel sú "Diehard" navrhnutý Georgeom Marsagliom a "TestU01" navrhnutý Pierrom L'Ecuyerom. Testovanie kvality generátorov náhodných čísel je veľmi široká problematika, takže sa opäť sústredíme len na všeobecné princípy.

Veľmi všeobecnou triedou testov štatistickej kvality generátora náhodných čísel sú chíkvadrát testy dobrej zhody. Všimnime si, že testy dobrej zhody vo viacrozmerných priestoroch môžu byť súčasne citlivé na rovnomernosť jednotlivých realizácií a aj na ich nezávislosť, pretože náhodné premenné $X_1, \ldots, X_m$ majú rozdelenie $R(0,1)$ a sú nezávislé vtedy a len vtedy, keď náhodný vektor $\mathbf{X} = (X_1, \ldots, X_m)'$ má rovnomerné rozdelenie na kocke $[0,1]^m$.

Veta 2.4. *Nech $B_1, \ldots, B_k$ je rozklad $(0,1]^m$ na borelovské podmnožiny s Lebesguovou mierou $p_1, \ldots, p_k$. Nech $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3 \ldots$ je postupnosť nezávislých náhodných vektorov s rovnomerným rozdelením na $[0,1]^m$. Pre $n \in \mathbb{N}$ a $i = 1, \ldots, k$ definujme náhodnú premennú $N_n^{(i)}$ ako počet tých vektorov spomedzi $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, ktoré padnú do množiny B_i. Potom postupnosť náhodných premenných $(Z_n)_{n=1}^{\infty}$, kde*

$$Z_n = \sum_{i=1}^{k} \frac{(N_n^{(i)} - np_i)^2}{np_i},$$

má asymptoticky rozdelenie χ^2_{k-1}.

Uvedomme si, že pri testovaní generátorov náhodných čísel máme k dispozícii veľmi veľké rozsahy výberov (veľmi veľké n), takže limitný charakter predchádzajúceho tvrdenia je len zanedbateľnou nevýhodou. Vhodnou voľbou rozkladu množiny $(0,1]^m$ vo vete 2.4 môžeme získať množstvo špecifických testov dobrej zhody.

Príklad 2.2. *Najjednoduchší test rovnomernosti náhodných čísel dostaneme tak, že vo vete 2.4 položíme $m = 1$ a pre zvolené $s \in \mathbb{N}$ definujeme $B_i = ((i-1)/s, i/s]$, t.j. $p_i = 1/s$ pre $i = 1, ..., s$. Ak* `x` *reprezentuje vstupný vektor, tak príslušný chíkvadrát test vykonáme pomocou príkazu*[11]*:* `chisq.test(table(ceiling(s*x)), p=rep(1/s, s))`

[11] Pre malú dĺžku testovaného vektora v porovnaní s číslom s sa môže stať, že niektoré intervaly B_i zostanú prázdne a v takom prípade uvedený príkaz zlyhá (okrem toho, že aproximácia chíkvadrát rozdelením môže byť veľmi nepresná).

Iné (menej často používané) viacrozmerné testy dobrej zhody sú napríklad pokrový trojrozmerný a päťrozmerný test, test počtov permutácií rôznych typov podpostupností dĺžky K a podobne.

Dva časté problémy aplikácie testov dobrej zhody, jeden matematický a druhý algoritmický, sú určenie pravdepodobností p_i a vytvorenie efektívneho algoritmu na výpočet hodnôt $N_n^{(i)}$. Napríklad jedna verzia testu "32x32 Binary Rank Test" v sade testov Diehard je založená na nasledovnom princípe: reprezentuj čísla z testovaného generátora pomocou 32 bitového zápisu, z 32 takýchto čísel vytvor binárnu maticu typu 32×32 a vyrátaj jej hodnosť. To opakuj n krát a urči, koľko matíc malo hodnosti 32, 31, 30 a < 29. Ide o veľmi špeciálny chíkvadrát test dobrej zhody v $m = 32$ rozmernom priestore s počtom oblastí $k = 4$, pričom vypočítať hodnoty $p_1, \ldots, p_4$ ani čísla $N_n^{(1)}, \ldots, N_n^{(k)}$ nie je vôbec jednoduché (čísla $p_1, \ldots, p_4$ sa dajú nájsť tabelované).

Často používaný je aj Kolmogorovov-Smirnovov test, ktorého testovacia štatistika sa zakladá na maximálnej odchýlke mezi teoretickou distribučnou funkciou F (v prípade rovnomerného rozdelenia samozrejme $F(x) = x$ pre všetky $x \in (0, 1)$) a empirickou distribučnou funkciou $\hat{F}_n$ skonštruovanou z náhodného výberu $X_1, \ldots, X_n$, t.j. na štatistike

$$D_n = \max_{x \in \mathbb{R}} |F(x) - \hat{F}_n(x)|.$$

Náhodné premenné $\sqrt{n} D_n$ konvergujú pre $n \to \infty$ podľa distribučnej funkcie k známemu asymptotickému rozdeleniu, pomocou ktorého je možné testovať hypotézy. (Toto rozdelenie je však komplikované, a preto ho nebudeme uvádzať.)

Príklad 2.3. *Ak* `x` *reprezentuje výstup z generátora náhodných čísel, tak Kolmogorovov-Smirnovov test dobrej zhody s rovnomerným rozdelením vykonáme v jazyku R pomocou príkazu:* `ks.test(x, punif)`

Ďalšou zaujímavou možnosťou je použiť test seriálnej korelácie založený na nasledovnej vete.

Veta 2.5. *Nech $k \in \mathbb{N}$. Pre každé $n > k$ nech je $X_1, X_2, \ldots, X_n$ náhodný výber z $R(0, 1)$. Nech $r_n^{(k)}$ je výberový korelačný koeficient pre vektory $\mathbf{X}_1 = (X_1, X_2, \ldots, X_{n-k})'$ a $\mathbf{X}_2 = (X_{k+1}, X_{k+2}, \ldots, X_n)'$, t.j.*

$$r_n^{(k)} = \frac{(n-k-1)^{-1} \sum_{i=1}^{n-k} (X_i - \bar{X})(X_{i+k} - \bar{X})}{(n-1)^{-1} \sum_{i=1}^{n} (X_i - \bar{X})^2}$$

Potom náhodná premenná $\sqrt{n} r_n^{(k)}$ má asymptoticky rozdelenie $N(0, 1)$.

Niekedy používame pojem "testy nezávislosti" pre tie testy, ktoré sú citlivé iba na nezávislosť náhodných premenných $X_1, \ldots, X_n$, ale nie sú citlivé na žiadnu rastúcu transformáciu hodnôt. Príkladom takéhoto testu sú všetky testy založené na rozdelení permutácií, alebo aj test seriálnej korelácie. Na druhej strane pod pojmom "testy rovnomernosti" rozumieme testy citlivé na rovnomernosť rozdelenia samotných náhodných premenných, ktoré však neodhalia prípadné závislosti; presnejšie, testy v ktorých má

testovacia štatistika rozdelenie invariantné na akúkoľvek permutáciu vygenerovaných hodnôt. Príkladom je Kolmogorov-Smirnov test, alebo chíkvadrát testy pre $m = 1$.

Náročné súbory testov kvality generátorov náhodných čísel (Diehard, TestU01) obsahujú mnoho ďalších testov. Napríklad Diehard obsahuje 'Birthday spacings test", "Parking lot test", "Runs test", "The craps test" a iné. Inou skupinou testov, silnou, ale špecificky určenou pre generátory LCG, sú takzvané spektrálne testy. Dá sa totiž ukázať, že LCG generátory produkujú náhodné čísla s "mriežkovou štruktúrou".

Príklad 2.4. *Uvažujme nasledovný elementárny LCG v jazyku R:*

```
LCG <- function(a, c, m, x1, n) {
         x <- c(x1, rep(0, n-1))
         for(i in 2:n) x[i] <- (a*x[i-1] + c)%%m
         return(x) }
```

Pomocou tohto programu nájdite periódu niektorých jednoduchých LCG pre všetky vstupné stavy. Presvedčte sa o "mriežkovej štruktúre" výsledkov pre nízke hodnoty m, ak vygenerované náhodné čísla $X_1, X_2, X_3, \ldots$ zobrazíte ako body $(X_1, X_2), (X_2, X_3), \ldots$ v rovine. Overte kvalitu generátora "minimal standard" a tiež generátora "Mersenne twister", ktorý používa program R, pomocou základného chíkvadrát testu dobrej zhody, Kolmogorovho-Smirnovovho testu a pomocou testu seriálnej korelácie.

3 Generovanie realizácií diskrétnych náhodných premenných a vektorov

V tejto kapitole sa sústredíme na generovanie diskrétnych náhodných premenných s dôrazom na všeobecné techniky a princípy, ktoré nám umožnia generovať akékoľvek diskrétne rozdelenie podľa potreby. Menej sa budeme zaoberať technikami využívajúcimi špeciálne vlastnosti najbežnejších diskrétnych rozdelení, ako napríklad binomického, Poissonovho, či negatívne binomického rozdelenia, pretože najefektívnejšie generátory z týchto rozdelení sú často technicky veľmi komplikované a pritom sú obvykle implementované v základých softwarových balíkoch.

Uvažujme diskrétnu náhodnú premennú X, ktorá nadobúda hodnoty $x_1, x_2, ...$ s pravdepodobnosťami $p_1, p_2, ...$ Bez hlbších znalostí používaných techník je v jazyku `R` možné vygenerovať `n` nezávislých realizácií premennej X nadobúdajúcej hodnoty vektora `x` s pravdepodobnosťami definovanými vektorom `p` pomocou príkazu:

```
sample(x=x,size=n,replace=T,prob=p).
```

Zhodu vygenerovaných realizácií `y` s požadovaným rozdelením potom môžeme vykonať pomocou chíkvadrát testu napríklad takto: `chisq.test(x=table(y),p=p)`[12].

Ak chceme generovať z diskrétneho rozdelenia s nekonečne (samozrejme spočítateľne) veľkým oborom hodnôt, alebo chceme funkciu `sample` z dôvodov efektivity obísť,

[12] Tento jednoduchý príkaz funguje korektne, len ak je každá možná hodnota zastúpená aspoň raz.

prípadne používame iný jazyk, v ktorom nie je implementovaná analogická procedúra, máme niekoľko možností. Najprv si uvedomíme, že generovať realizácie náhodnej premennej X popísanej vyššie je prakticky totožné s generovaním realizácií náhodnej premennej X^* takej, že $X = x_{X^*}$, pričom X^* nadobúda hodnoty $1, 2, ...$ s pravdepodobnosťami $p_1, p_2, ...$ V ďalšom teda môžeme bez straty na všeobecnosti predpokladať, že $x_k = k$ pre všetky k.

3.1 Metóda inverznej transformácie pre diskrétne rozdelenia

Prvou možnosťou je použiť vetu o inverznej transformácii. Označeme $s_k = \sum_{i=1}^{k} p_i$ pre $k = 1, 2, ...$ Kvantilová funkcia náhodnej premennej X je

$$G(y) = \begin{cases} 1 & \text{ak } 0 < y < p_1 \\ k & \text{ak } s_{k-1} \leq y < s_k \text{ pre } k = 2, 3, ... \end{cases}$$

a ak $U \sim R(0, 1)$, potom $G(U)$ má požadované rozdelenie. Keď si podrobnejšie všimneme funkciu G, zistíme, že metóda inverznej transformácie je ekvivalentná nasledovnému jednoduchému postupu: interval $(0, 1)$ rozbijeme na intervaly $I_1 = (0, s_1)$, $I_2 = [s_1, s_2)$, $I_3 = [s_2, s_3)$,... . Ak realizácia $u \in (0, 1)$ náhodnej premennej $U \sim R(0, 1)$ padne do intervalu I_k, tak k môžeme považovať za realizáciu náhodnej premennej X. Pre viacero bežných typov rozdelení pravdepodobnosti môžeme hodnoty s_k počítať postupne na základe jednoduchého predpisu pre podiel p_{k+1}/p_k. Napríklad ak $X \sim Bin(n, p)$, tak $p_{k+1} = p_k \frac{n-k}{k+1} \frac{p}{1-p}$ pre $k = 0, ..., n-1$, ak $X \sim Po(\lambda)$, tak $p_{k+1} = p_k \frac{\lambda}{k+1}$ pre $k = 0, 1, 2, ...$ a tak ďalej.

Priamočiare sekvenčné hľadanie intervalu I_k, do ktorého padla realizácia rozdelenia $R(0, 1)$, môže byť veľmi zdĺhavé. Toto prehľadávanie môžeme značne urýchliť, ak si generovanie vhodne "predpripravíme". Všimnime si napríklad, že v metóde inverznej transformácie potrebujeme v strednej hodnote približne $E(X) = \sum_i i p_i$ testov. Táto stredná hodnota však očividne závisí od poradia hodnôt p_i a najmenšia je, ak $p_1 \geq p_2 \geq \cdots$. Možné je teda využiť permutáciu, ktorá hodnoty p_i usporiada od najväčšej po najmenšiu.

3.2 Metóda deliacich bodov

Ďalšou možnosťou je použiť metódu štartovacích, tzv. "deliacich" bodov, ktorej teoretickým základom je nasledovná veta.

Veta 3.1 (Metóda deliacich bodov)**.** *Pre fixné $d \in \mathbb{N}$ a pre $j = 1, ..., d$ definujme deliace body $I_j = \min\{i \in \{1, ..., m\} : s_i > (j-1)/d\}$. Nech $U \sim R(0, 1)$, $D = \lceil d.U \rceil$ a nech $N = \min\{k \geq I_D : s_k > U\}$. Potom $P[N = i] = p_i$ pre každé $i = 1, ..., m$.*

Formálny dôkaz metódy deliacich bodov je technicky zdĺhavý, avšak jeho myšlienka je jednoduchá: I_D určuje index "intervalu", od ktorého môžeme začať prehľadávať, kam padla náhodná premenná U (a tento index vieme na základe predpripraveného vektora hodnôt $I_1, ..., I_k$ určiť veľmi rýchlo).

Všimnime si, že metóda inverznej transformácie je špeciálnym prípadom metódy deliacich bodov (pre $d = 1$). Počet deliacich bodov obvykle veľmi ovplyvňuje rýchlosť generovania. Ukazuje sa, že nie je vhodný ani príliš malý, ani príliš veľký počet deliacich bodov. Predchádzajúce dva postupy; t.j. deliace body a usporiadanie na urýchlenie hľadania správneho intervalu I_k, môžeme s výhodou kombinovať.

V jazyku `R` je možné zapísať metódu deliacich bodov nasledovne:

```
function(n,p,d)
# Generuje n realizacii z rozdelenia p[1],...,p[m]
# d je pocet deliacich bodov
m<-length(p); s<-rep(0,m)
s[1]<-p[1]; for(i in 2:m) s[i]<-s[i-1]+p[i]
I<-rep(0,d); for(j in 1:d) I[j]<-min((1:m)[s>(j-1)/d])
res<-rep(0,n); U<-runif(n)
for(i in 1:n){ D<-ceiling(d*U[i]); k<-I[D]
while(s[k]<=U[i]) k<-k+1; res[i]<-k }
res }
```

Príklad 3.1. *Experimentálne analyzujme časovú náročnosť kombinovanej metódy pre binomické rozdelenie $Bin(n, p)$ s rôznymi hodnotami $d = 1, ..., 10$, $n = 10, 100, 1000$ a $p = 0.1, 0.5, 0.9$. a Poissonovo rozdelenie. Je možné odvodiť približný vzorec pre strednú hodnotu počtu testov podmienky?*

Principálne odlišnou a v niektorých prípadoch veľmi efektívnou metódou na generovanie diskrétnych náhodných premenných nadobúdajúcich konečný počet hodnôt je takzvaná "alias" metóda. Tú však na tejto prednáške nebudeme preberať.

Poznamenajme, že špeciálne rozdelenia ako binomické, geometrické, Poissonovo, hypergeometrické a podobne môžeme generovať aj mnohými inými spôsobmi[13], ktoré sú veľmi efektívne, ale súčasne veľmi komplikované. Na generovanie z týchto rozdelení môžeme v jazyku R použiť funkcie `rbinom`, `rgeo`, `rpois`, `rhypgeo`.

3.3 Metóda diskretizácie spojitých rozdelení

Veľmi jednoducho môžeme generovať realizácie z diskrétnych rozdelení, ktoré dostaneme konštrukciou hornej (prípadne dolnej) celej časti z vhodne zvolených jednoduchých spojitých rozdelení. Najjednoduchšie je takto možné generovať diskrétne náhodné premenné s rovnomerným rozdelením na konečnej množine, t.j. realizácie takej náhodnej premennej X s oborom hodnôt $\{x_1, ..., x_k\}$, pre ktorú $p_i = P[X = x_i] = 1/k$ pre všetky $i = 1, ..., k$. Poznamenajme, že generovanie z takýchto rozdelení je základom štatistických metód označovaných ako "bootstrap".

[13] Napríklad: Zdôvodnite nasledovnú vetu a na základe nej napíšte generátor na Poissonove rozdelenie: Nech $U_1, U_2, ...$ sú nezávislé náhodné premenné s rozdelením $R(0, 1)$. Nech $N = \min\left\{n \in \mathbb{N} : \prod_{i=1}^{n} U_i < e^{-\lambda}\right\}$. Potom $N - 1 \sim Po(\lambda)$

Veta 3.2. *Nech* $\{x_1, ..., x_k\}$ *je množina reálnych čísel a nech* $U \sim R(0,1)$. *Potom* $x_{\lceil kU \rceil}$ *nadobúda hodnoty* $x_1, ..., x_k$, *každú s pravdepodobnosťou* $1/k$.

Realizáciu náhodného výberu veľkosti `n` z rovnomerného rozdelenia na vektore `x` dĺžky `k` získame v jazyku `R` pomocou príkazu `x[ceiling(k*runif(n))]`. Techniku z vety 3.2 je možné zovšeobecniť na prípady, keď sú pravdepodobnosti p_i racionálne čísla pre všetky $i = 1, ..., k$:

Veta 3.3. *Predpokladajme, že* $p_i = l_i/l$ *pre* $i = 1, ..., k$, *kde* $l_i \in \mathbb{N}$ *a* $\sum_{i=1}^{k} l_i = l$. *Nech* v *je vektor dĺžky* l, *pričom pre každé* $m = 1, ..., k$ *platí* $v_{i_m+1} = v_{i_m+2} = ... = v_{i_m+l_m} = x_m$, *kde* $i_1 = 0$ *a* $i_m = \sum_{i=1}^{m-1} l_i$ *pre* $m = 2, ..., k$. *Potom platí: ak* $U \sim R(0,1)$, *tak* $v_{\lceil lU \rceil}$ *nadobúda hodnoty* $x_1, ..., x_k$ *s pravdepodobnosťami* $p_1, ..., p_k$.

Ak `l` je vektor prirodzených čísel $l_1, ..., l_k$, potom generovanie `n` hodnôt vektora `x` s váhami úmernými l_i realizujeme takto: `rep(x,l)[ceiling(sum(l)*runif(n))]`.

Platnosť predchádzajúch dvoch viet je očividná a netreba ich dokazovať. Uvedomíme si, že v niektorých prípadoch môže byť použitie vety 3.3 neefektívne kvôli veľkému rozsahu vektora v.

Geometrické rozdelenie môžeme jednoducho generovať ako diskretizáciu exponenciálneho rozdelenia, ktoré je možné generovať logaritmickou transformáciou rovnomerného rozdelenia:

Veta 3.4. *Nech* $p \in (0,1)$, $U \sim R(0,1)$. *Potom* $\lfloor \ln(U)/\ln(1-p) \rfloor \sim Geo(p)$.

Dôkaz. Jednoducho overíme, že $-\ln(U)$ má hustotou e^{-x} pre $x > 0$, t.j. rozdelenie $Exp(1)$. Nech $X = \lfloor \ln(U)/\ln(1-p) \rfloor = \lfloor -\ln(U)/\ln(1/(1-p)) \rfloor$. Pre akékoľvek $k = 0, 1, ...$ teda máme: $P[X = k] = P[-\ln(U)/\ln(1/(1-p)) \in [k, k+1)] = P[-\ln(U) \in [\ln(1/(1-p))k, \ln(1/(1-p))(k+1)) = \exp(-\ln(1/(1-p))k) - \exp(-\ln(1/(1-p))(k+1)) = (1-p)^k - (1-p)^{k+1} = (1-p)^k p$. □

V jazyku `R` máme na generovanie geometrického rozdelenia k dispozícii funkciu `rgeo`. Len s použitím `runif` vygenerujeme `n` realizácií z geometrického rozdelenia s parametrom `p` príkazom: `floor(log(runif(n))/log(1-p))`.

3.4 Generovanie diskrétnych náhodných vektorov

Ak chceme získať realizácie diskrétneho náhodného vektora $\mathbf{X} = (X_1, ..., X_m)'$ s nezávislými zložkami, stačí postupne vygenerovať realizácie jednotlivých náhodných premenných $X_1, ..., X_m$. Situácia je takto jednoduchá napríklad ak sú $X_1, ..., X_m$ nezávislé náhodné premenné s rozdelením $R(x_1, ..., x_m)$, čím simulujeme náhodný výber z množiny $\{x_1, ..., x_m\}$ s návratom. Komplikovanejšia situácia nastáva, ak sú komponenty nášho vektora závislé, napríklad v prípade simulovania náhodného výberu bez návratu.

V prípade generovania náhodného vektora so závislými komponentami je obvykle dostatočne efektívny nasledovný postup (takzvaná metóda podmienených rozdelení): Najprv vygenerujeme realizáciu x_1 z rozdelenia náhodnej premennej X_1, potom realizáciu x_2 z podmieného rozdelenia $X_2|X_1 = x_1$, potom realizáciu x_3 z podmieného

rozdelenia $X_3|X_1 = x_1, X_2 = x_2$ a tak ďalej.Uvedomme si, že na použitie tejto techniky potrebujme poznať príslušné podmienené rozdelenia, ale aj vedieť z týchto rozdelení generovať realizácie. Uvedieme tvar príslušných podmienených rozdelení potrebných na generovanie multinomického rozdelenia a na generovanie náhodného výberu bez návratu. Generovanie mnohorozmerného hypergeometrického rozdelenia a negatívne multinomického rozdelenia si vyriešite ako úlohu.

Veta 3.5. *Nech* $\mathbf{X} \sim Mult(n, p_1, \ldots, p_m)$, *kde* $p_1, ..., p_m > 0$. *Potom* $X_1 \sim Bin(n, p_1)$. *Nech ďalej* $k_1, ..., k_m \in \{0, 1, ..., n\}$ *sú také, že* $\sum_{j=1}^{m} k_j = n$. *Nech* $i \in \{2, 3, ..., m\}$. *Označme* $n_i = \sum_{j=i}^{m} k_j$ *a* $r_i = p_i / \sum_{j=i}^{m} p_j$. *Potom*

$$P[X_i = k_i | X_1 = k_1, \ldots, X_{i-1} = k_{i-1}] = \binom{n_i}{k_i} r_i^{k_i} (1 - r_i)^{n_i - k_i},$$

T.j. náhodná premenná X_i *za podmienky* $X_1 = k_1, \ldots, X_{i-1} = k_{i-1}$ *má rozdelenie* $Bin(n_i, r_i)$.

Pomocou predchádzajúcej vety môžeme zostaviť nasledovný program na generovanie z multinomického rozdelenia[14]

```
function(n,N,p){
 # Generuje n realizacii z rozdelenia Mult(N,p[1],...,p[m])
 m<-length(p); res<-matrix(ncol=0,nrow=m)
 for(i in 1:n){
   X<-rep(0,m); X[1]<-rbinom(1,size=N,prob=p[1])
   Nrem<-N-X[1]; prem<-1-p[1]
   for(j in 2:m){
     X[j]<-rbinom(1,size=Nrem,prob=p[j]/prem)
     Nrem<-Nrem-X[j]; prem<-prem-p[j] }
   res<-cbind(res,X)}
 res}
```

Všeobecná veta nám tiež umožňuje generovanie náhodných permutácií:

Veta 3.6. *Nech* $\mathbf{X} = (X_1, ..., X_m)^T$ *je náhodný vektor, ktorý má rovnomerné rozdelenie na množine* $\{(k_1, ..., k_m)^T : \{k_1, ..., k_m\} = \{1, ..., m\}\}$. *Potom* X_1 *má rozdelenie* $R(1, ..., m)$ *a pre* $k = 2, ..., m$ *má náhodná premenná* X_k *za podmienky* $X_1 = x_1, ..., X_{k-1} = x_{k-1}$ *rovnomerné rozdelenie na množine* $\{1, ..., m\} \setminus \{x_1, ..., x_{k-1}\}$.

Uvedomíme si, že ak náhodný vektor $\mathbf{X} = (X_1, ..., X_m)^T$ má rovnomerné rozdelenie na množine všetkých permutácií, tak subvektor $(X_1, ..., X_n)^T$, kde $n < m$ má rovnomerné rozdelenie na množine všetkých n-prvkových podmnožín množiny $\{1, ..., m\}$, čiže

[14] Pri praktickom použití programu je vhodné nahradiť `p[j]/prem` výrazom `min(p[j]/prem,1)`, pretože numerické chyby niekedy spôsobujú to, že podiel `p[j]/prem` je máličko väčší ako jedna.

reprezentuje n-prvkový výber bez návratu z množiny $\{1, ..., m\}$. Poznamenajme, že náhodný výber bez návratu je veľmi dôležitý napríklad pre znáhodňovanie pri navrhovaní experimentov.

V jazyku `R` máme procedúru `sample`, ktorá generuje náhodný výber s vracaním aj bez vracania a ako špeciálny prípad aj náhodné permutácie. Ak by sme chceli napísať generátor náhodného výberu bez vracania len s použitím generátora z rovnomerného rozdelenia, môžeme myšlienku vety 3.6 implementovať napríklad nasledovne:

```
function(n,m){
r<-1:m; res<-rep(0,n); U<-runif(n)
for(i in 1:n){
  k<-ceiling((m-i+1)*U[i])
  res[i]<-r[k]
  if(k<(m-i+1)) r[k]<-r[m-i+1] }
res }
```

4 Generovanie realizácií spojitých náhodných premenných a vektorov

4.1 Veta o inverznej transformácii pre spojité rozdelenia

Najjednoduchšou metódou generovania spojitých náhodných premenných so zadaným rozdelením je použiť vetu o inverznej transformácii 2.1; samozrejme, len ak vieme efektívne počítať hodnoty príslušnej kvantilovej funkcie[15].

Príklad 4.1 (Generovanie rozdelenia $Exp(\lambda)$)**.** *Distribučná funkcia rozdelenia* $Exp(\lambda)$ *je* $F(x) = 0$ *pre* $x \leq 0$ *a* $F(x) = 1 - e^{-x/\lambda}$ *pre* $x > 0$*. Kvantilová funkcia je* $G(y) = -\lambda ln(1-y)$ *pre* $y \in (0,1)$*, čiže podľa vety 2.1 platí: Ak* $U \sim R(0,1)$*, tak* $-\lambda ln(1-U) \sim Exp(\lambda)$*. Keďže aj* $1 - U \sim R(0,1)$*, vidíme, že aj* $-\lambda ln(U) \sim Exp(\lambda)$*.*

V jazyku `R` možeme vygenerovať `n` realizácií exponenciálneho rozdelenia s parametrom `lambda` príkazom `rexp(n,rate=1/lambda)`. Ak by sme chceli použiť len generátor z rovnomerného rozdelenia, tak vykonáme príkaz: `-lambda*log(runif(n))`. Kolmogorovov-Smirnovov test, že vektor `y` pochádza z exponenciálneho rozdelenia s parametrom `lambda`, môžeme realizovať takto: `ks.test(y,pexp,rate=1/lambda)`.

Príklad 4.2. *Nech* $m \in \mathbb{N}$*. Uvažujme distribučnú funkciu* $F(x) = 0$ *pre* $x \leq 0$*,* $F(x) = x^m$ *pre* $x \in (0,1)$ *a* $F(x) = 1$ *pre* $x \geq 1$[16]*. Keďže kvantilová funkcia je* $G(y) = y^{1/m}$ *pre*

[15] Poznamenajme, že nie je nutné vedieť vyjadriť kvantilovú funkciu explicitne. Stačí skonštruovať veľmi rýchly iteračný algoritmus, ktorý počíta *funkčné hodnoty* kvantilovej funkcie.

[16] Táto distribučná funkcia zodpovedá rozdeleniu $Be(m+1,1)$ a charakterizuje rozdelenie druhej mocniny normy m-rozmerného náhodného vektora s rovnomerným rozdelením vo vnútri m-rozmernej gule so stredom v počiatku súradnicovej sústavy a polomerom 1 alebo, ekvivalentne, rozdelenie minima m nezávislých $R(0,1)$ náhodných premenných.

$y \in (0,1)$, tak podľa vety 2.1 dostávame: Ak $U \sim R(0,1)$, potom $U^{1/m}$ má distribučnú funkciu F.

Podobne je možné generovať napríklad Cauchyho rozdelenie[17].

4.2 Zamietacia metóda pre spojité rozdelenia

V niektorých prípadoch, keď kvantilovú funkciu nevieme počítať rýchlym algoritmom, môžeme s výhodou použiť takzvanú zamietaciu metódu.

Veta 4.1 (Veta o zamietaní). *Nech $f, h : \mathbb{R} \to [0, \infty)$ sú dve funkcie hustoty, kladné na intervale $I \subseteq \mathbb{R}$ a inde nulové, pričom $c = \sup\{f(x)/h(x) : x \in I\} < \infty$. Nech $Y_1, U_1, Y_2, U_2, \ldots$ sú nezávislé náhodné premenné, pričom $Y_1, Y_2, \ldots$ majú hustotu h a $U_1, U_2, \ldots$ majú rozdelenie $R(0,1)$. Nech*

$$N = \min\{n \in \mathbb{N} : c.h(Y_n)U_n < f(Y_n)\}.$$

Potom náhodná premenná Y_N má hustotu f.

Dôkaz. Pre jednoduchosť urobíme dôkaz len pre tento prípad: $I = (0,1)$ a h je hustota rozdelenia $R(0,1)$, t.j. $h(x) = 1$ pre $x \in (0,1)$, $h(x) = 0$ pre $x \notin (0,1)$.

Zvoľme ľubovoľné $z \in (0,1)$. Zrejme platí

$$P[Y_N < z] = \cup_{k=1}^{\infty} P[Y_N < z, N = k] = \cup_{k=1}^{\infty} P[Y_k < z, N = k].$$

Keďže pre akékoľvek k má náhodný vektor $(Y_k, U_k)^T$ dvojrozmerné rovnomerné rozdelenie na množine $[0,1] \times [0,1]$, je $P[Y_k < z, cU_k < f(Y_k)]$ rovná ploche množiny $B_z = \{(y,u)' \in [0,1] \times [0,1] : y < z \text{ a } c.u < f(y)\}$, teda

$$P[Y_k < z, cU_k < f(Y_k)] = \frac{1}{c}\int_0^z f(y)dy = r_z.$$

Preto $P[Y_1 < z, N = 1] = P[Y_1 < z, c.U_1 < f(Y_1)] = r_z$ a pre $n \geq 2$ je $P[Y_k < z, N = k] = P[Y_k < z, c.U_1 \geq f(Y_1), \ldots, c.U_{k-1} \geq f(Y_{k-1}), c.U_k < f(Y_k)] = P[Y_k < z, c.U_k < f(Y_k)]P[c.U_1 \geq f(Y_1)] \ldots P[c.U_{k-1} \geq f(Y_{k-1})] = r_z(1 - r_1)^{k-1}$. Takže

$$P[Y_N < z] = r_z \sum_{k=1}^{\infty} (1 - 1/c)^{k-1} = r_z c = \int_0^z f(y)dy.$$

Z toho plynie, že f je hustota Y_N. □

Najčastejšie používaný špeciálny prípad predchádzajúcej vety je nasledovný: f je ohraničená hustota kladná len na intervale $I = (a,b)$, kde a, b sú konečné čísla a h je hustota rozdelenia $R(a,b)$. V tomto prípade máme $N = \min\{n \in \mathbb{N} : c.U_n < f(Y_n)\}$, kde $c = \sup\{f(x) : x \in I\}$.

Zdôraznime ešte, že vetu 4.1 je vhodné použiť, len ak vieme rýchlo generovať realizácie náhodných premenných s hustotou h a ak sa hustota h "podobá" na cieľovú hustotu f; len v takom prípade je totiž nízka pravdepodobnosť udalosti $c.h(Y_n)U_n \geq f(Y_n)$. Táto pravdepodobnosť, v angličtine nazývaná *rejection ratio*, určuje asymptotický podiel "zbytočne" vygenerovaných dvojíc (Y_n, U_n).

[17] To je však možné aj bez výpočtu kvantilovej funkcie.

Príklad 4.3. *Uvažujme hustotu* $f(x) = (8/\pi)\sqrt{x}\sqrt{1-x}$ *pre* $x \in (0,1)$ *a* $f(x) = 0$ *pre* $x \notin (0,1)$[18]. *Ak zvolíme* h *hustotu rovnomerného rozdelenia na intervale* $(0,1)$, *tak realizáciu z rozdelenia s hustotou* f *dostaneme nasledovne: Budeme generovať nezávislé realizácie* $Y_1, U_1, Y_2, U_2, \ldots$ *z rozdelenia* $R(0,1)$, *až pokým nenastane prípad* $(4/\pi)U_n < f(Y_n)$. *V tomto okamihu bude* Y_n *reprezentovať realizáciu z rozdelenia s hustotou* f.

4.3 Generovanie realizácií normálneho rozdelenia a príbuzných rozdelení

V jazyku `R` je vygenerujeme `n` realizácií z normálneho rozdelenia so strednou hodnotou `mu` a smerodajnou odchylkou `sigma` pomocou príkazu `rnorm(n,mean=mu,sd=sigma)`. Ako ukážeme, zamietacou metódou je možné generovať realizácie z rozdelenia $N(0,1)$ len pomocou generátora z rovnomerného rozdelenia a následne, vhodnou lineárnou transformáciou, aj z rozdelenia $N(\mu, \sigma^2)$.

Uvažujme najprv hustotu $f(x) = \sqrt{2/\pi} \times e^{-x^2/2}$ pre $x > 0$ a $f(x) = 0$ pre $x \leq 0$, čo zodpovedá hustote náhodnej premennej $|X|$, ak $X \sim N(0,1)$, takzvanému "polnomálnemu" rozdeleniu. Zvolíme za h hustotu rozdelenia $Exp(1)$, t.j. $h(x) = e^{-x}$ pre $x > 0$ a $h(x) = 0$ pre $x \leq 0$. Pomocou štandardných metód matematickej analýzy vypočítame, že

$$c = \sup\{f(x)/h(x) : x > 0\} = \sqrt{2e/\pi}.$$

Podmienka určujúca index n akceptovanej realizácie je

$$\sqrt{2e/\pi} \times e^{-Y_n} U_n < \sqrt{2/\pi} \times e^{-Y_n^2/2},$$

čo je možné ekvivalentne zapísať v tvare

$$U_n < \exp\left(-(Y_n - 1)^2/2\right)$$

Vygenerovať realizáciu premennej z rozdelenia s hustotou f môžeme teda nasledovne: generujeme nezávislé realizácie $Y_1, U_1, Y_2, U_2, \ldots$, pričom $Y_1, Y_2, \ldots$ majú rozdelenie $Exp(1)$ a $U_1, U_2, \ldots$ majú rozdelenie $R(0,1)$. Akonáhle bude splnené $U_n < \exp\left(-(Y_n - 1)^2/2\right)$, bude $Y = Y_n$ konečným výsledkom.

Pomocou realizácie Y s hustotou f môžeme vygenerovať realizáciu z rozdelenia $N(0,1)$ jednoducho tak, že Y vynásobíme s pravdepodobnosťou $1/2$ hodnotou -1 a s pravdepodobnosťou $1/2$ hodnotou 1 (t.j. číslu Y dáme náhodné znamienko). Dostávame nasledovnú vetu:

Veta 4.2. *Nech* $V, Y_1, U_1, Y_2, U_2, \ldots$ *sú nezávislé náhodné premenné, pričom* $P[V = -1] = P[V = 1] = 1/2$, $Y_i \sim Exp(1)$ *a* $U_i \sim R(0,1)$ *pre každé* $i \in \mathbb{N}$. *Nech*

$$N = \min\left\{n \in \mathbb{N} : U_n < \exp\left(-(Y_n - 1)^2/2\right)\right\}.$$

Potom náhodná premenná VY_N *má rozdelenie* $N(0,1)$.

[18] Ide o tzv. beta-rozdelenie s parametrami 3/2 a 3/2.

Zamietací algoritmus generovania `n` realizácií normálneho rozdelenia so strednou hodnotou `mu` a smerodajnou odchýlkou `sigma` môžeme v jazyku `R` zapísať takto:

```
function(n,mu=0,sigma=1){
res<-rep(0,n)
for(i in 1:n){
  ok<-FALSE
  while(!(ok)){
    Y<-rexp(1);U<-runif(1)
    if(U<exp(-(Y-1)^2/2)) ok<-TRUE}
  res[i]<-Y}
rsgn<-2*floor(2*runif(n))-1
sigma*rsgn*res+mu}
```

Iný veľmi rozšírený spôsob generovania realizácií z normálneho rozdelenia využíva to, že dvojrozmerné normalizované normálne rozdelenie $N_2(0, \mathbf{I})$ má polárne súradnice (R, ψ) s ľahko generovateľným rozdelením: $R \sim \sqrt{Exp(1/2)}$ a $\psi \sim R(0, 2\pi)$.

Veta 4.3 (Box-Mullerov generátor)**.** *Nech náhodný vektor $(V_1, V_2)^T$ má rovnomerné rozdelenie na jednotkovej kružnici v rovine a nech $L \sim Exp(1/2)$. Potom náhodné premenné $X_1 = V_1\sqrt{L}$ a $X_2 = V_2\sqrt{L}$ sú nezávislé, obe s rozdelením $N(0, 1)$. Špeciálne, ak U, W sú nezávislé s rozdelením $R(0, 1)$, tak náhodné premenné*

$$X_1 = \cos(2\pi U)\sqrt{-2\ln(W)},\ X_2 = \sin(2\pi U)\sqrt{-2\ln(W)}$$

sú nezávislé, obe s rozdelením $N(0, 1)$.

Implementácia Box-Mullerovho algoritmu je veľmi jednoduchá a nechávame ju ako praktické cvičenie. Uvedomme si ešte, že pomocou normálneho rozdelenia vieme generovať realizácie chíkvadrát rozdelenia (ale to aj pomocou gama rozdelenia; pozri nasledujúcu kapitolu), z ktorých následne môžeme dostať realizácie t-rozdelenia a F-rozdelenia. Existujú aj efektívnejšie (avšak komplikované) metódy na generovanie realizácií z t a F rozdelenia, ale nebudeme ich uvádzať.

4.4 Generovanie Beta a Gama rozdelenia

Definícia 4.1. *Nech $a, b > 0$. Náhodná premenná X má beta rozdelenie s parametrami a, b, ak je X spojitá náhodná premenná s hustotou $f(x) = B^{-1}(a, b)x^{a-1}(1 - x)^{b-1}$ pre $x \in (0, 1)$ a $f(x) = 0$ inak. Symbol B označuje beta funkciu. Značíme $X \sim Be(a, b)$.*

Všimnime si, že beta rozdelenie obsahuje ako špeciálny prípad $R(0, 1)$. Pre špeciálne voľby parametrov a, b je možné generovať realizácie rozdelenia $Be(a, b)$ viacerými spôsobmi (pozri príklad). Univerzálny návod poskytuje nasledovná veta:

Veta 4.4. *Nech $a, b > 0$. Nech X_1, X_2 sú nezávislé náhodné premenné s rozdelením $R(0,1)$. Položme $V = X_1^{1/a}$, $W = X_2^{1/b}$ a $Y = V/(V+W)$. Potom náhodná premenná Y za podmienky $V + W \leq 1$ má rozdelenie $Be(a, b)$.*

V jazyku `R`:

```
function(n,a=1,b=1){
res<-rep(0,n)
for(i in 1:n){
  ok<-FALSE
  while(!(ok)){
    X<-runif(2)
    V<-X[1]^(1/a);W<-X[2]^(1/b)
    if(V+W<1) ok<-TRUE}
  res[i]<-V/(V+W)}
res}
```

Definícia 4.2. *Nech $a, b > 0$. Náhodná premenná X má gama rozdelenie s parametrami a, b, ak X spojitá náhodná premenná s hustotou $f(x) = \Gamma^{-1}(b) a^b x^{b-1} e^{-ax}$ pre $x > 0$ a $f(x) = 0$ inak. Značíme $X \sim G(a, b)$.*

Parameter a nazývame parameter škály a b je parameter tvaru. Trieda gama rozdelení je veľmi široká; ako špeciálny prípad obsahuje rozdelenie $Exp(a) \equiv G(a, 1)$, rozdelenia $\chi^2_n \equiv G(1/2, n/2)$ a mnohé iné. Možno tiež jednoducho ukázať, že ak $X \sim G(1, b)$, tak $a^{-1}X \sim G(a, b)$, čo znamená, že sa stačí zaoberať generovaním rozdelenia $G(1, b)$. Naviac, ak $X_1 \sim G(1, b_1)$ a $X_2 \sim G(1, b_2)$ sú nezávislé náhodné premenné, tak $X_1 + X_2 \sim G(1, b_1 + b_2)$. Ak je teda b reálne číslo, tak $G(1, b)$ je súčtom $\lfloor b \rfloor$ nezávislých náhodných premenných s rozdelením $Exp(1)$ a jednej náhodnej premennej s rozdelením $G(1, r)$, kde $r = b - \lfloor b \rfloor$. To je možné vykonať s pomocou generátora beta rozdelenia.

Veta 4.5. *Nech $a, b > 0$, $n = \lfloor b \rfloor$ a $r = b - n$. Nech $Z_0, Z_1, \ldots, Z_n, Y$ sú nezávislé náhodné premenné, pričom $Z_0, Z_1, \ldots, Z_n \sim Exp(1)$ a $Y \sim Be(r, 1-r)$ ak $r > 0$ alebo $Y = 0$ ak $r = 0$. Ak $n = 0$, položme $Z = 0$ a ak $n \geq 1$ nech $Z = Z_1 + \ldots + Z_n$. Potom platí $a^{-1}(Z + Z_0 Y) \sim G(a, b)$.*

V jazyku `R` zapíšeme výsledný algoritmus nasledovne:

```
function(m,a=1,b=1){
n<-floor(b); r<-b-n; res<-rep(0,m)
for(i in 1:m){
  Y<-0; if(r>0) Y<-rbeta(1,shape1=r,shape2=1-r)
  Z<-0; if(n>0) Z<-sum(rexp(n))
  res[i]<-(Z+rexp(1)*Y)/a}
res}
```

4.5 Generovanie spojitých náhodných vektorov

Veta 4.6. *Nech $X_1, ..., X_m$ sú nezávislé náhodné premenné s rozdelením $N(0,1)$. Nech $\mu \in \mathbb{R}^m$ a nech Σ je pozitívne semidefinitná matica typu $m \times m$. Predpokladajme, že $\Sigma = \mathbf{C}\mathbf{C}^T$, kde $\mathbf{C}$ je matica typu $m \times m$. Potom náhodný vektor $\mathbf{Y} = \mathbf{C}(X_1, ..., X_m)^T + \mu$ má rozdelenie $N_m(\mu, \Sigma)$.*

Maticu $\mathbf{C}$ z predošlej vety je možné určiť viacerými metódami; obvykle sa však používa dolná trojuholníková matica, ktorú je možné z matice Σ vypočítať relatívne rýchlym algoritmom a naviac mierne zjednodušuje aj výpočet súčinu $\mathbf{C}(X_1, ..., X_m)^T$ (Rovnosť $\Sigma = \mathbf{C}\mathbf{C}^T$ v takom prípade nazývame Choleského rozklad.) Takúto dolnú trojuholníkovú maticu vypočítame napríklad pomocou nasledovného algoritmu (prvky matice Σ označíme s_{ij} a prvky matice C označíme c_{ij})

1 Prvkom matice C nad diagonálou priraď hodnotu 0.

2 $c_{11} \leftarrow \sqrt{s_{11}}$ a pre $i = 2$ až m vykonaj: $c_{i1} \leftarrow s_{i1}/c_{11}$

4 $c_{22} \leftarrow \sqrt{s_{22} - c_{21}^2}$

5 Pre $i = 3$ až m vykonaj:

5.1 Pre $j = 2$ až $i - 1$ vykonaj: $c_{ij} \leftarrow (s_{ij} - \sum_{k=1}^{j-1} c_{ik}c_{jk})/c_{jj}$

5.2 $c_{ii} \leftarrow \sqrt{s_{ii} - \sum_{k=1}^{i-1} c_{ik}^2}$

6 Výsledok: C

Často používaný špeciálny prípad dvojrozmerného normálneho náhodného vektora popisuje nasledovná veta:

Veta 4.7. *Nech $X_1, X_2 \sim N(0,1)$ sú nezávislé náhodné premenné, nech $\mu_1, \mu_2, \sigma_1, \sigma_2 \in \mathbb{R}$, $\sigma_1, \sigma_2 > 0$ a $\rho \in [-1, 1]$. Položme $Y_1 = \sigma_1 X_1 + \mu_1$ a $Y_2 = \sigma_2(\rho X_1 + \sqrt{1-\rho^2} X_2) + \mu_2$. Potom náhodný vektor $(Y_1, Y_2)^T$ má združene normálne rozdelenie, pričom $Y_1 \sim N(\mu_1, \sigma_1^2)$, $Y_2 \sim N(\mu_2, \sigma_2^2)$ a korelácia Y_1 a Y_2 je ρ.*

Pomocou normálneho rozdelenia môžeme tiež vygenerovať realizáciu rovnomerného rozdelenia na m-rozmernej guli a na m-rozmernej sfére.

Veta 4.8. *Nech G_m je m-rozmerná jednotková guľa so stredom v bode $0 \in \mathbb{R}^m$. Nech $X_1, ..., X_m, U$ sú nezávislé náhodné premenné, pričom $X_i \sim N(0,1)$ a $U \sim R(0,1)$. Potom $\mathbf{Y} = (\sum_{i=1}^m X_i^2)^{-1/2}(X_1, ..., X_m)^T$ má rovnomerné rozdelenie na hranici gule G_m a $\mathbf{Z} = U^{1/m}\mathbf{Y}$ má rovnomerné rozdelenie na guli G_m.*

Existuje viacero alternatívnych, často efektívnejších spôsobov, ako generovať náhodné vektory rovnomerne na povrchu $G_m(r)$, alebo na $G_m(r)$, najmä pre dimenzie $m = 2, 3$ (ak vynecháme triviálny prípad $m = 1$). Zo širokého spektra rôznych metód spomenieme napríklad nasledovné "zamietacie" metódy pre $m = 2, 3$: generujme $X_1, Y_1, X_2, Y_2, ...$ ako nezávislé realizácie z rozdelenia $R(-1, 1)$, až kým prvýkrát nenastane prípad $X_n^2 + Y_n^2 < 1$. Označme $X = X_n, Y = Y_n$. Potom platí:

- $(X,Y)^T$ má rovnomerné rozdelenie na $G_2(1)$
- $(X^2+Y^2)^{-1/2}(X,Y)^T$ má rovnomerné rozdelenie na hranici $G_2(1)$
- $(X^2+Y^2)^{-1}(X^2-Y^2,2XY)^T$ má rovnomerné rozdelenie na hranici $G_2(1)$
- $(2X\sqrt{1-X^2-Y^2}, 2Y\sqrt{1-X^2-Y^2}, 1-2(X^2+Y^2))^T$ má rovnomerné rozdelenie na hranici $G_3(1)$.

Jednoducho a efektívne je možné generovať aj rovnomerné rozdelenie na "klasickom" simplexe S_m ako aj na "pravdepodobnostnom" simplexe P_m, ako ukazuje nasledovná veta.

Veta 4.9. *Definujme množiny* $S_m = \left\{(x_1,...,x_m)^T \in [0,1]^m : \sum_{i=1}^m x_i \leq 1\right\}$ *a* $P_m = \left\{(x_1,...,x_m)^T \in [0,1]^m : \sum_{i=1}^m x_i = 1\right\}$. *Predpokladajme, že* $U_1,...,U_m$ *sú nezávislé náhodné premenné s rozdelením* $R(0,1)$ *a nech* $Y_1,...,Y_m$ *je usporiadanie týchto náhodných premenných od najmenšej po najväčšiu. Definujme* $Z_1 = Y_1$ *a* $Z_i = Y_i - Y_{i-1}$ *pre* $i = 2,...,m$. *Potom náhodný vektor* $\mathbf{S} = (Z_1,...,Z_m)^T$ *má rovnomerné rozdelenie na* S_m *a náhodný vektor* $\mathbf{P} = (Z_1,...,Z_m, 1-Y_m)^T$ *má rovnomerné rozdelenie na* P_{m+1}.

Z rovnomerného rozdelenia na `m` rozmernom simplexe môžeme teda vygenerovať `n` realizácií nasledovne:

```
function(n,m){
# Funkcia generuje n realizacii na simplexe S_m
res<-matrix(ncol=0,nrow=m)
for(i in 1:n){
  Y<-sort(runif(m)); Z<-c(Y[1],Y[2:m]-Y[1:(m-1)])
  res<-cbind(res,Z)}
res}
```

5 Úvod do metód Monte Carlo

5.1 Základný princíp klasických metód Monte Carlo

Predpokladajme, že nás zaujíma nejaká číselná hodnota, označme si ju z, ktorú nie je možné jednoducho vypočítať analytickými prostriedkami; môže to byť napríklad dôležitá číselná charakteristika komplikovaného systému hromadnej obsluhy, pravdepodobnosť "zruinovania" podmienená zložitým procesom s náhodnými vplyvmi, ale aj hodnota určitého mnohorozmerného integrálu a podobne. Monte Carlo metódy predstavujú zaujímavú možnosť, ako túto číselnú hodnotu odhadnúť pomocou stochastických simulácií.

Základom klasických metód Monte Carlo odhadovania hodnoty z je konštrukcia generátora náhodného výberu $Z_1, Z_2, ..., Z_n$, pre ktorý $E(Z_i) = z$ a $D(Z_i) = \sigma^2 < \infty$.

V takomto prípade totiž pre aritmetický priemer $\bar{Z}_n = \frac{1}{n}\sum_{i=1}^{n} Z_i$ platí $E(\bar{Z}_n) = z$ a $D(\bar{Z}_n) = \sigma^2/n$, čiže $\bar{Z}_n$ je nevychýleným a konzistentným odhadom hľadanej hodnoty z. Navyše, pre z vieme skonštruovať približné intervaly spoľahlivosti - podľa centrálnej limitnej vety má pre veľké n náhodná premenná $\bar{Z}_n$ približne rozdelenie $N(z, \sigma^2/n)$ a na základe zákona veľkých čísel môžeme hodnotu σ^2 odhadnúť pomocou výberového rozptylu $S_n^2 = \frac{1}{n}\sum_{i=1}^{n}(Z_i - \bar{Z}_n)^2 = \frac{1}{n}\sum_{i=1}^{n} Z_i^2 - (\bar{Z}_n)^2$, teda

$$P\left[z \in \left(\bar{Z}_n - u_{1-\alpha/2}\frac{S_n}{\sqrt{n}}, \bar{Z}_n + u_{1-\alpha/2}\frac{S_n}{\sqrt{n}}\right)\right] \approx 1 - \alpha, \tag{2}$$

kde $u_{1-\alpha/2}$ je $100(1-\alpha/2)$-percentný kvantil rozdelenia $N(0,1)$ a $\alpha \in (0,1)$ je zvolená pravdepodobnosť určujúca požadovanú spoľahlivosť[19].

Všimnime si, že dĺžka daného intervalu spoľahlivosti klesá úmerne hodnote $\sqrt{n}$, čiže zvyšovanie počtu realizácií nám prináša relatívne malý nárast presnosti odhadu, preto by sme mali klásť dôraz na to, aby bola čo najmenšia disperzia σ^2 náhodných premenných Z_i, ktorých realizácie generujeme[20].

Poznámka o sekvenčnom výpočte priemeru a výberového rozptylu. Niekedy pri Monte Carlo výpočtoch postupujeme tak, že si nepredpíšeme vopred počet n simulačných behov, no vykonávame simulačné behy, pokým nie je splnené $2u_{1-\alpha/2}\frac{S_n}{\sqrt{n}} < \Delta$, kde Δ je vopred stanovené číslo, čiže pokým dĺžka intervalu spoľahlivosti (2) nie je menšia ako Δ[21]. Výpočet priemeru $\bar{Z}_n$ a výberovej smerodajnej odchýlky S_n po každom novom simulačnom behu na základe vzorcov $\bar{Z}_n = \frac{1}{n}\sum_{i=1}^{n} Z_i$ a $S_n^2 = \frac{1}{n}\sum_{i=1}^{n}(Z_i - \bar{Z}_n)^2$ je ale značne neefektívny. Vhodnejšie je priebežne si udržiavať hodnotu L súčtu všetkých dosiaľ vygenerovaných hodnôt $Z_1, ..., Z_n$ a hodnotu Q súčtu druhých mocnín dosiaľ vygenerovaných hodnôt $Z_1, ..., Z_n$. To znamená že pred začatím simulácií položíme $L = Q = 0$ a po n-tom simulačnom behu vykonáme priradenie $L = L + Z_n$, $Q = Q + Z_n^2$. Zrejme potom $\bar{Z}_n = \frac{1}{n}L$ a $S_n^2 = \frac{1}{n}Q - (\frac{1}{n}L)^2$.

5.2 Simulačný odhad pravdepodobnosti

Dôležitým špeciálnym prípadom aplikácie metód Monte Carlo je odhad pravdepodobnosti $z \in (0,1)$ nejakej udalosti týkajúcej sa analyzovaného systému so stochastickými prvkami. V tomto prípade obvykle vieme priamočiaro skonštruovať simulačný generátor postupnosti nezávislých náhodných premenných $Z_1, ..., Z_n$, pričom Z_i nadobudne 1, ak sledovaná udalosť v i-tom simulačnom behu nastane a 0, ak táto udalosť nenastane, čiže náhodné premenné $Z_1, ..., Z_n$ majú alternatívne rozdelenie $Alt(z)$. Keďže platí $Z_i^2 = Z_i$

[19] Obvykle sa volí $\alpha = 0{,}05$ a v takom prípade $u_{1-\alpha/2} \approx 1{,}96$.

[20] Pochopiteľne, ide aj o to, ako *rýchlo* dokážeme generovať realizácie náhodných premenných Z_i. V zásade platí, že prechod ku generátoru s k-krát nižšou disperziou sa oplatí, pokiaľ čas na vygenerovanie jednej realizácie narastie menej ako k-krát.

[21] Toto vnáša do výsledného intervalu spoľahlivosti ďalšiu nepresnosť, ktorá ale môže byť pri veľkom počte simulačných behov a spojitom, unimodálnom, nie veľmi šikmom rozdelení náhodných premenných Z_i zanedbateľná.

pre všetky, $i = 1, ..., n$, výberová disperzia je rovná $S_n^2 = \bar{Z}_n - (\bar{Z}_n)^2$ a (2) nadobúda tvar

$$P\left[z \in \left(\bar{Z}_n - u_{1-\alpha/2}\sqrt{\frac{\bar{Z}_n - (\bar{Z}_n)^2}{n}}, \bar{Z}_n + u_{1-\alpha/2}\sqrt{\frac{\bar{Z}_n - (\bar{Z}_n)^2}{n}}\right)\right] \approx 1 - \alpha. \qquad (3)$$

Tento interval spoľahlivosti bude veľmi presný pre vysoký počet n simulačných behov a "neextremálnu" hodnotu pravdepodobnosti z. V metódach Monte Carlo obvykle nie je problém s tým, že by sme mali malý rozsah n; bežné sú hodnoty n v miliónoch. Napriek tomu, ak je hodnota z napríklad veľmi blízko nuly alebo jednotky, tak sa môže stať, že uvedený interval spoľahlivosti bude mať ľavý koniec záporný, alebo pravý koniec väčší ako 1[22]. Takýmto patológiám sa môžeme vyhnúť, keď si uvedomíme, že $n\bar{Z}_n$ má rozdelenie $Bin(n, z)$, čiže môžme použiť špeciálne intervaly spoľahlivosti pre parameter "p" (my značíme tento parameter z) binomického rozdelenia. Jednou z možností je takzvaný Clopperov-Pearsonov "exaktný" $100(1-\alpha)$-percentný interval spoľahlivosti

$$P\left[\left(1 + \frac{n+1-X}{XF_1}\right)^{-1} \leq z \leq \left(1 + \frac{n-X}{(1+X)F_2}\right)^{-1}\right] \approx 1 - \alpha, \qquad (4)$$

kde $X = n\bar{Z}_n$ je realizácia príslušného binomického rozdelenia, F_1 je $100(\alpha/2)$-percentný kvantil F-rozdelenia so stupňami voľnosti $2X$ a $2(n - X + 1)$ a F_2 je $100(1 - \alpha/2)$ percentný kvantil F-rozdelenia so stupňami voľnosti $2(X + 1)$ a $2(n - X)$.

[22] Pre odhadovanie pravdepodobností extremálnych udalostí existujú špeciálne metódy, ktoré sú často omnoho efektívnejšie ako priamočiary prístup uvedený v tomto texte. Pre takéto situácie sa niekedy dajú skonštruovať náhodné premenné Z_i, pre ktoré $E(Z_i) = z$, avšak Z_i nie sú nula-jednotkové a ich disperzia je menšia ako $z(1 - z)$. Takýmito metódami sa v tomto texte nebudeme zaoberať.

5.3 Monte Carlo výpočet integrálov

Klasickým využitím metód Monte Carlo je výpočet (presnejšie odhad) hodnoty integrálu

$$z = \int_A h(x)dx.$$

Pre jednoduchosť budeme predpokladať, že $A \subseteq \mathbb{R}^m$ je borelovská množina Lebesguovej miery (dĺžky, plochy, objemu) $\lambda \in (0, \infty)$ a $h : \mathbb{R}^m \to [0, M]$ je spojitá funkcia $M \in (0, \infty)$. Taktiež budeme predpokladať, že $z > 0$.

V súlade s princípmi opísanými v časti 5.1 skonštruujeme generátor realizácie postupnosti nezávislých, rovnako rozdelených náhodných premenných $Z_1, ..., Z_n$, takých, že $E(Z_i) = z$ a $D(Z_i) = \sigma^2 < \infty$, pre $i = 1, ..., n$, na odhad hodnoty z použijeme aritmetický priemer $\bar{Z}_n$ vygenerovaných hodnôt a na interval spoľahlivosti pre z použijeme predpis (2). Pochopiteľne, stačí určiť, ako skonštruujeme realizáciu náhodnej premennej Z s vlastnosťami $E(Z) = z$ a $D(Z) = \sigma^2 < \infty$ v jednom simulačnom behu. Nezávislým opakovaním n takýchto simulačných behov dostaneme realizáciu požadovanej postupnosti $Z_1, ..., Z_n$. Existuje viacero prístupov ku konštruovaniu generátora náhodnej premennej Z s požadovanou strednou hodnotou a konečnou disperziou.

Najjednoduchšou metódou je "zamietacia" metóda (angl. rejection method), v ktorej vygenerujeme náhodnú premennú Z nasledovne: Nech U má rovnomerné rozdelenie na A a nech V má rovnomerné rozdelenie na $[0, M]$. V prípade $V < h(U)$ položíme $Z = \lambda M$ a v opačnom prípade položíme $Z = 0$. Náhodný vektor $(U^T, V)^T$ má očividne rovnomerné rozdelenie na množine $A \times [0, M]$, ktorá má mieru λM. Navyše, udalosť $[Z = \lambda M]$ nastane práve vtedy, keď náhodný vektor $(U^T, V)^T$ padne pod graf funkcie h, takže

$$\begin{aligned} P[Z = \lambda M] &= (\lambda M)^{-1} \int_A h(x)dx = (\lambda M)^{-1} z, \\ E[Z] &= (\lambda M) P[Z = \lambda M] = z, \\ D[Z] &= (\lambda M)^2 P[Z = \lambda M] - z^2 = \lambda M z - z^2 < \infty. \end{aligned} \tag{5}$$

Používanejšou než zamietacia metóda je takzvaná "obyčajná" metóda (angl. crude Monte-Carlo method), ktorá kladie $Z = \lambda h(U)$, kde U je, rovnako ako pri zamietacej metóde, náhodný vektor s rovnomerným rozdelením na množine A. Keďže hustota f_U náhodného vektora U je konštantne $1/\lambda$ na množine A, dostávame

$$\begin{aligned} E(Z) &= E(\lambda h(U)) = \int_A \lambda h(x) f_U(x)dx = z, \\ D(Z) &= \lambda \int_A h^2(x)dx - z^2 \leq \lambda M z - z^2 < \infty. \end{aligned} \tag{6}$$

Z (5) a (6) vidíme, že disperzia pre zamietaciu metódou nie je nikdy lepšia ako pre obyčajnú metódou, hoci v niektorých situáciách môže byť numerický výpočet realizácie náhodnej premennej Z jednoduchší zamietacou metódou. Problémom je, že aj pre obyčajú metódu je $D(Z)$ často vysoké číslo, čo znižuje celkovú presnosť Monte Carlo

odhadu. Ako vieme, zvýšenie počtu n simulačných behov zvyšuje presnosť odhadu len pomaly, preto je často najvýhodnejšie snažiť sa o zmenšenie $D(Z)$.

Zovšeobecnením obyčajnej metódy je takzvaná metóda váhového výberu (angl. importance sampling), ktorá často umožňuje skonštruovať generátor náhodnej premennej s požadovanou strednou hodnotu, ale menšou disperziou ako obyčajná metóda. Predpokladajme, že máme hustotu f na $\mathbb{R}^m$ takú, že $f(x) > 0$ pre $x \in A$ a $f(x) = 0$ ak $x \notin A$, pričom vieme generovať realizácie náhodných vektorov s hustotou f. Predpokladajme tiež, že $h(x)/f(x) < N$ pre všetky $x \in A$ a nejaké $N < \infty$. Metóda váhového výberu predpisuje generovať náhodnú premennú Z takto: $Z = h(X)/f(X)$, kde X je náhodný vektor s hustotou f. Zrejme platí:

$$\begin{aligned} E(Z) &= E(h(X)/f(X)) = \int_A (h(x)/f(x))f(x)dx = z, \\ D(Z) &= \int_A h^2(x)/f(x)dx - z^2 \leq \lambda MN - z^2 < \infty. \end{aligned}$$

Efektívnosť metódy váhového výberu je určená výberom funkcie f[23]. Ukazuje sa, že metóda váhového výberu je tým efektívnejšia, čím "podobnejší" je na množine A tvar hustoty f tvaru funkcie h. Ak by bola "podobnosť tvarov" f a h dokonalá, čiže ak by platilo $f(x) = ch(x)$ pre všetky $x \in A$ a nejakú konštantu c, tak $D(Z)$ je dokonca nulová, aha:

$$\begin{aligned} 1 &= \int_A f(x)dx = \int_A ch(x)dx = cz, \text{ čiže} \\ D(Z) &= \int_A h^2(x)/f(x)dx - z^2 = \int_A h^2(x)/(ch(x))dx - z^2 = z/c - z^2 = 0. \end{aligned}$$

Toto je však len teoretická možnosť, lebo v takejto situácii by sme na vyčíslenie realizácie náhodnej premennej Z potrebovali poznať hodnotu čísla z, ktoré sa snažíme odhadnúť (a vo svetle tohto konštatovania už nie je až také pozoruhodné, že z dokážeme odhadnúť s nulovou disperziou).

[23] Zlý výber funkcie f môže urobiť metódu váhového výberu ešte oveľa menej efektívnou, ako je obyčajná metóda. Premyslite si tiež, pre aké f sa metóda váhového výberu redukuje na obyčajnú metódu.