# Generarea unor variabile particulare

## Curs 8

December 4, 2023

# Generarea permutarilor

Fie $X$ o variabilă uniformă discretă, care ia valorile $1, 2, ..., n$:

$$X : \begin{pmatrix} 1 & 2 & ... & n \\ 1/n & 1/n & ... & 1/n \end{pmatrix} \tag{1}$$

Ca să îl generăm pe $X$ vom aplica metoda inversă cazul discret. În acest caz:

$$X = j \text{ dacă } \frac{j-1}{n} \leq U < \frac{j}{n}$$

pentru că $s_i = i/n$ Adică

$$X = j \text{ dacă } j - 1 \leq nU < j$$

$$X = [nU] + 1$$

- Presupunem că dorim să generăm o permutare a numerelor $1, 2, ..., n$ astfel încât toate cele $n!$ ordini posibile sunt egal probabile.
- Alegem mai întâi unul din numerele de la 1 la $n$ și îl punem pe poziția $n$, apoi alegem unul din cele $n-1$ numere rămase și îl punem pe poziția $n-1$, ș.a.m.d.
- Este mai eficient să păstrăm numerele într-o listă și să legem aleator poziția pentru fiecare dintre ele.
- Pas1: Fie $P_1 = 1$, $P_2 = 2$,...,$P_n = n$;
- Pas2: $k = n$;
- Pas3: Generează $U \sim U(0,1)$, $I = [kU] + 1$;
- Pas4: Interschimbă $P_I$ cu $P_k$;
- Pas5: $k = k - 1$. Dacă $k > 1$ mergi la Pas3;
- Ieșire: $(P_1, ..., P_n)$

- Algoritmul de generare a permutărilor se poate folosi și pentru generarea submulțimilor mulțimii $\{1, 2, ..., n\}$. Pentru o submulțime cu $r$ elemente algoritmul se oprește după ce au fost completate pozițiile $n$, $n-1$,..., $n-r+1$. Elementele de pe aceste poziții reprezintă o submulțime aleatoare.
- Generarea de submulțimi aleatoare este foarte importantă în studiile medicale. De exemplu pentru a testa un nou medicament se recutrează 1000 de voluntari. Se ia decizia de a împărți voluntarii în două grupuri de cate 500, un grup de tratament și unul de control. Nici voluntarilor, nici medicilor nu li se spune care dintre voluntari sunt în fiecare grup - test double-blind. Cele două grupuri trebuie să fie cât mai similare posibil.

# Algoritmi care folosesc Metoda inversă, cazul discret

- Variabila $Poisson(\lambda)$:

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

$$p_{i+1} = e^{-\lambda}\frac{\lambda^{i+1}}{(i+1)!} = \frac{\lambda}{i+1}e^{-\lambda}\frac{\lambda^i}{i!} = \frac{\lambda}{i+1}p_i$$

- Algoritm de generare bazat pe ideea de la metoda inversă, cazul discret:
  1. Se generează $U \sim U(0, 1)$;
  2. $i = 0$, $p = e^{-\lambda}$, $s = p$;
  3. Dacă $U < s$, $X = i$, stop;
  4. $p = \lambda p/((i + 1)$, $s = s + p$, $i = i + 1$, mergi la pasul 3.
- Un algoritm mai eficient ar fi să se înceapă căutarea în jurul valorii lui $\lambda$. Dacă $I = [\lambda]$, se calculează $p_I$, apoi $s_I$, se verifică dacă $U \leq s_I$.

- Variabila binomială(n,p)

$$p_i = P\{X = i\} = \frac{n!}{i!(n-i)!} p^i (1-p)^{n-i}$$

$$p_{i+1} = P\{X = i+1\} = \frac{n!}{(i+1)!(n-i-1)!} p^{i+1} (1-p)^{n-i-1} =$$

$$= \frac{n-i}{i+1} \frac{p}{1-p} p_i$$

- Algoritm de generare:
  1. Se generează $U \sim U(0,1)$;
  2. $i = 0$, $c = p/(1-p)$ $pr = (1-p)^n$, $s = pr$;
  3. Dacă $U < s$, $X = i$, stop;
  4. $pr = (c(n-i)/(i+1))pr$, $s = s + pr$, $i = i + 1$, mergi la pasul 3.
- Ca și în cazul variabilei Poisson, algoritmul se poate îmbunătăți dacă se începe căutarea de la $I = [np]$.

# Un algoritm de respingere pentru variabilele discrete

- Presupunem că este cunoscut un algoritm de generare pentru o variabilă discretă cu funcția de probabilitate $\{q_j, j \geq 0\}$ și că dorim să generăm o variabilă discretă cu funcția de probabilitate $\{p_j, j \geq 0\}$.
- Fie $c$ o constantă, astfel încât

$$\frac{p_j}{q_j} \leq c \quad \text{pentru orice } j \text{ cu } p_j > 0$$

.

- Algoritm:
  1. Se generează $Y$ cu funcția de probabilitate $\{q_j, j \geq 0\}$;
  2. Se generează $U \sim U(0,1)$;
  3. Dacă $U < p_Y/cq_y$ atunci $X = Y$, stop. Altfel mergi la pasul 1.
- Teoremă: Algoritmul de mai sus generează o variabilă aleatoare $X$, astfel încât

$$P\{X = j\} = p_j \;\; j = 0, 1, ...$$

În plus, numărul de itera'ii ale algoritmului, necesare pentru a-l obține pe $X$ este o variabilă geometrică de medie $c$.

# Validarea generatorilor

Se verifică faptul că algoritmul de generare produce valori de selecție ale variabilei aleatoare care se dorește a fi generată.

Acest lucru se poate face prin verificarea ipotezei statistice de concordanță:

$$H : X \sim F(x)$$

pe baza unei selecții $X_1, X_2, ..., X_n$ cu $n$ suficient de mare, produse de algoritmul de generare.

Repartiția empirică (sau de selecție) ar trebui să se "asemene" cu repartiția teoretică $F(x) \Rightarrow$ este necesară construcția histogramei rezultate din valorile de selecție $X_1, X_2, ..., X_n$.

**Construcția unei histograme**

Pentru a construi o histogramă atunci când se cunosc valorile de selecție $X_1, X_2, ..., X_n$ se aplică următorul algoritm:

**Algoritm Histograma1**
**Intrare:** $X_1, X_2, ..., X_n$;
P1: Se determină $m = \min\{X_1, X_2, ..., X_n\}$ și
$M = \max\{X_1, X_2, ..., X_n\}$, adică intervalul de variație $[m, M]$;
P2: Se alege $k$ (se recomandă $15 \leq k \leq 40$) reprezentând numărul de intervale ale histogramei;
P3: Se împarte intervalul $[m, M]$ în $k$ intervale $I_i = [a_{i-1}, a_i)$ de lungimi egale, cu $1 \leq i \leq k$, $a_0 = m$, $a_k = M$;
P4: Se determină $f_i =$ nr. valorilor de selecție din intervalul $I_i$, $1 \leq i \leq k$, adică frecvențele absolute.
P5: Pentru $i = 1, 2, ..., k$ se determină frecvențele relative $r_i = \frac{f_i}{n}$
**Ieșire:** Frecvențele relative $r_1, r_2, ..., r_k$.

Prin urmare repartiția empirică obținută este:

$$\begin{pmatrix} I_1, & I_2, & \dots & I_k \\ r_1 & r_2 & \dots & r_k \end{pmatrix} \tag{2}$$

Folosind (10), o histogramă se poate reprezenta grafic astfel: se consideră în plan un sistem de axe $x0y$, pe axa $Ox$ se marchează intervalele $I_i$, $1 \leq i \leq k$ și se construiesc dreptunghiuri care au ca bază intervalele $I_i$ și ca înălțime frecvențele relative $r_i$, $1 \leq i \leq k$.

Implementarea algoritmului Histograma1 creează anumite probleme pentru că determinarea valorilor $m$ și $M$ și apoi construirea intervalelor $I_i$ și calcularea frecvențelor $f_i$ implică memorarea selecției sau repetarea simulării selecției folosind aceeași sămânță a generatorului, acest lucru însemnând consum de memorie sau de timp de calcul.

Idei pentru construcția eficientă a histogramei:

- Generăm mai întâi $n_1 \ll n$ valori de selecție $X_1, X_2, ..., X_{n_1}$ pe care le memorăm;
- Determinăm $a_1 = \min\{X_1, X_2, ..., X_{n_1}\}$, $a_{k-1} = \max\{X_1, X_2, ..., X_{n_1}\}$, $a_0 = a_1$, $a_k = a_{k-1}$;
- Se consideră
$$h = \frac{a_{k-1} - a_1}{k-2}$$
și $a_i = a_{i-1} + h$, $i = 2, 3, ..., k-1$.
- Se iau $f_1 = 0$, $f_k = 0$ și se determină $f_2, f_3, ..., f_{k-1}$ numărul de valori $X_i$ cu $1 \leq i \leq n_1$ din intervalele $I_2, I_3..., I_{k-1}$.
- Se simulează celelalte $n - n_1$ valori de selecție. Fie $X$ o astfel de valoare, atunci:
  - Dacă $X < a_1$ atunci $a_0 = \min\{a_0, X\}$, $f_1 = f_1 + 1$;
  - Dacă $X > a_{k-1}$ atunci $a_k = \max\{a_k, X\}$, $f_k = f_k + 1$;

- Dacă $X \in [a_1, a_{k-1}]$ atunci

$$j := \left[\frac{X - a_1}{h}\right] + 2, \quad f_j := f_j + 1$$

Observăm că această construcție produce $k-2$ intervale egale și două intervale $I_1$ și $I_k$ de lungimi oarecare. Algoritmul rezultat astfel este:

**Algoritm Histograma2**
**Intrare:** $n$, $n_1$, $k$;
P1: Pentru $i = 1, 2, ..., k$ $f[j] := 0$;
P2: Pentru $i = 1, 2, ..., n_1$ se generează $X$, $x[i] := X$;
P3: Se determină $a[1] := \min\{x[1], x[2], ..., x[n_1]\}$,
$a_{k-1} = \max\{x[1], x[2], ..., x[n_1]\}$, $h = (a[k-1] - a[1])/(k-2)$;
P4: Pentru $i = 1, 2, ..., n_1$, $j := trunc((x[i] - a[1])/h) + 2$,
$f[j] := f[j] + 1$;
P5: $a[0] := a[1]$, $a[k] := a[k-1]$;

P6: Pentru $i = 1, 2, ..., n - n_1$ execută

- Se generează $X$;
- Dacă $a[1] \leq X \leq a[k-1]$ atunci $j = trunc\left(\frac{X-a[1]}{h}\right) + 2$, $f[j] := f[j] + 1$;
- Dacă $X < a[1]$ atunci $a[0] := \min\{a[0], X\}$, $f[1] := f[1] + 1$;
- Dacă $X > a[k-1]$ atunci $a[k] := \max\{a[k], X\}$, $f[k] := f[k] + 1$;

**Ieșire:** Frecvențele $f[1], f[2], ..., f[k]$.

## Validarea generatorilor

## 1. Validarea cu histogramă și test de concordanță

Construcția histogramei $\Rightarrow$ obținerea frecvențelor absolute $f_1, f_2, ..., f_k$ pentru fiecare interval de frecvență $[a_0, a_1)$, $[a_1, a_2)$,...,$[a_{k-a}, a_k)$. Pentru a valida generatorul se poate aplica testul de concordanță $\chi^2$ pentru verificarea ipotezei:

$$H : X \sim F(x). \tag{3}$$

- Se calculează probabilitățile:

$$p_1 = F(a_1)$$

$$p_i = F(a_i) - F(a_{i-1}), \quad 2 \leq i \leq k-2$$

$$p_k = 1 - F(a_{k-1})$$

- Se calculează

$$\sum_{j=1}^{k} \frac{(f_i - np_i)^2}{np_i} \tag{4}$$

  unde $n$ reprezintă numărul de valori generate. S-a demonstrat că valoarea obținută în (2) ar trebui să fie valoarea unei variabile aleatoare $\chi^2$ cu $k-1$ grade de libertate.
- Fiind dat riscul de genul I, $\alpha$ (o probabilitate apropiată de 0), se determină din tabele valoarea $\chi^2_{k-1,\alpha}$ numită $\alpha$-cuantilă superioară, astfel încât dacă $X$ este o variabilă aleatoare $\chi^2$ cu $k-1$ grade de libertate, atunci este verificată condiția

$$P(X > \chi^2_{k-1,\alpha}) = \alpha$$

adică probabilitatea de a avea $X > \chi^2_{k-1,\alpha}$ este foarte mică.

- Ipoteza $H$ se acceptă dacă valoarea dată de (2) este mai mică decât $\chi^2_{k-1,\alpha}$ (înseamnă că valoarea calculată poate fi valoarea unei variabile $\chi^2$ cu $k-1$). Altfel, ipoteza $H$ este respinsă.

## 2. Validarea cu medie și dispersie

Presupunem că pentru variabila aleatoare $X$, ale cărei valori au fost generate printr-un algoritm de generare, sunt cunoscute media teoretică și dispersia teoretică:

$$E[X] = m \quad Var[X] = \sigma^2$$

Cu ajutorul valorilor $X_1$, $X_2$,...,$X_n$ obținute din algoritmul de simulare se calculează media empirică de selecție și dispersia empirică de selecție:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}, \quad s^2 = \frac{\sum_{i=1}^{n} X_i^2}{n} - \bar{X}^2.$$

Generatorul este considerat bun dacă pentru $n$ destul de mare ($n > 1000$) valorile mediei $\bar{X}$ și ale dispersiei $s^2$ sunt apropiate de $m$ și $\sigma^2$ ca o consecință a *legii numerelor mari*.