1.1 Pravidlo součinu a permutace

Pravidlo součinu: Předpokládejme, že náhodný pokus E_1 může skončit n_1 vzájemně rozlišitelnými výsledky a pro každý z těchto výsledků můžeme provést náhodný pokus E_2, který může skončit n_2 vzájemně rozlišitelnými výsledky. Protom *složený náhodný pokus* E_1E_2, který sestává z provedení E_1 následovaného provedením E_2 může skončit $n_1 \cdot n_2$ rozlišitelnými výsledky.

Poznámka. Výsledky E_1E_2 lze chápat jako uspořádané dvojice $\langle e_1, e_2\rangle$, kde e_1 je výsledek E_1 a e_2 je výsledek E_2.

▷ **Příklad 1.1.** Uvažujme složený náhodný pokus E_1, který spočívá ve výběru studenta z množiny „Alice“, „Bob“, „Cyril“. Zvolenému studentu je jako náhodný pokus E_2 podána jedna z následujících látek: „pivo“, „víno“, „rum“, „destilovaná voda“. Složený náhodný pokus E_1E_2 může skončit $3 \cdot 4$ vzájemně různými výsledky.

Poznámka. Výsedky složených náhodných pokusů lze zaznamenávat diagramy (speciální stromy). Kořen stromu je počátek experimentu, jeho potomci znázorňují výsledky E_1. Každý výsledek z E_1 je zároveň kořenem podstromu, jehož potomci jsou výsledky E_2.

Pravidlo součinu lze chápat obecně pro k náhodných pokusů $E_1, \ldots, E_k$, kde každý náhodný pokus E_i končí n_i rozlišitelnými výsledky. V takovém případě je počet rozlišitelných výsledků složeného náhodného pokusu $E_1 \cdots E_k$ roven součinu $\prod_{i=1}^{k} n_i$.

► **Příklad 1.1.** Restaurace nabízí večerní menu skládající se z několika chodů: polévky, předkrmu, hlavního chodu, moučníku a nápoje. Za předpokladu, že restaurace nabízí 3 různé polévky, 5 různých předkrmů, 20 různých hlavních chodů, 4 různé moučníky a 6 různých nápojů, kolik je potenciálně možné objednat vzájemně odlišných večeří? $[3 \cdot 5 \cdot 20 \cdot 4 \cdot 6]$

▷ **Příklad 1.2.** Uvědomte si, že pomocí pravidla součinu lze jednoduše zdůvodnit některé známé zákony popisující velikosti výsledků operací s konečnými množinami. Mějme například konečné množiny A a B. Pak jejich kartézký součin $A \times B$ má $|A| \cdot |B|$ prvků, což lze pomocí pravidla součinu zdůvodnit tak, že máme-li náhodný pokus E_1 spočívající ve výběru prvku z A, který končí $|A|$ výsledky a následný náhodný pokus E_2 spočívající ve výběru prvků z B, který končí $|B|$ výsledky, pak počet výsledků výběru dvou prvků, prvního z A a druhého z B lze chápat jako počet výsledků složeného náhodného pokusu E_1E_2, tedy $|A| \cdot |B|$.

► **Příklad 1.2.** Mějme množiny A, B a C, kde A a B jsou disjunktní, to jest $A \cap B = \emptyset$. Určete počet prvků množiny $(A \cup B) \times C$. $[(|A| + |B|) \cdot |C|]$

► **Příklad 1.3.** Uvažujte předchozí příklad tak, že A a B nejsou obecně disjunktní a stanovte opět počet prvků $(A \cup B) \times C$. $[(|A| + |B| - |A \cap B|) \cdot |C|]$

► **Příklad 1.4.** Vypočtěte počet všech možných registračních čísel motorových vozidel, která mohou být udělena v rámci ČR. Uvědomte si, že registrační čísla jsou ve tvarech „`NXN NNNN`“, kde `X` je kód kraje a `N` jsou teoreticky libovolná čísla a písmena s výjimkou `O`, `Q`, `W`. $[13 \cdot (26-3+9)^2 \cdot (26-3+10)^4]$

Uvažujme následující problém: máme n prázdných pozic, které máme zaplnit n různými (vzájemně rozlišitelnými) objekty. Otázkou je, kolik je vzájemně různých zaplnění (jinými slovy, kolika způsoby lze n prázdných pozic zaplnit n objekty).

Zřejmě první pozici je možné zaplnit libovolným z n objektů. Jeho výběr můžeme chápat jako výsledek náhodného pokusu výběru jednoho prvku z n možných prvků. Pokud je první pozice zaplněna výsledným prvkem, máme již k dispozici pouze $n-1$ prvků. To jest, druhou pozici (která je nyní první volná) můžeme zaplnit libovolným z $n-1$ prvků, což lze chápat jako výsledek náhodného pokusu výběru jednoho prvku z $n-1$. Na třetí pozici pak máme možnost vybrat jeden z $n-2$ prvků, na čtvrté jeden z $n-3$ prvků a tak dále, až na poslední pozici musíme dát poslední, zbývající objekt. Celkově lze tedy počet způsobů, kterými lze zaplnit n volných pozic pomocí n objektů, chápat jako

počet možných výsledků složeného experimentu $E_1 \cdots E_n$, přitom E_1 končí n výsledky, E_2 končí $n-1$ výsledky, ..., E_n končí jediným výsledkem. Použitím pravidla součinu: $E_1 \cdots E_n$ končí

$$n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$$

vzájemně rozlišitelnými výsledky. Každý z těchto výsledků nazveme *permutace* a jejich celkový počet pro n pozic a n objektů nazveme *počet permutací* n objektů. Počet permutací n objektů značíme $n!$ (čteme n faktoriál). Zřejmě platí, že:

$$n! = \begin{cases} 1 & \text{pokud } n = 0, \\ n \cdot (n-1)! & \text{pokud } n \geq 1. \end{cases}$$

pro každé $n \geq 0$.

Poznámka. Uvědomte si, že mezní případ 0! má rozumný smysl. Na permutace n objektů lze nahlížet jako na uspořádané n-tice hodnot složené z neopakujících se vzájemně rozlišitelných n hodnot. V případě $n = 0$ se tedy jedná o uspořádanou nultici neobsahující žádnou hodnotu (někdy ji značíme $\langle\rangle$), která je právě jedna a z množinově teoretického pohledu je totožná s prázdnou množinou.

1.2 Algoritmus pro výpis permutací

V některých případech je vhodné vypsat všechny permutace n-prvkové množiny A, to jest vypsat všechny uspořádané n-tice hodnot prvků z množiny A. Pro množiny menších rozsahů to lze udělat ručně, pro větší množiny se nabízí využít jednoduchý rekurzivní algoritmus využívající principu *divide et impera*, který redukuje problém nalezení permutací n-prvkové množiny na problém nalezení permutací $n-1$ prvkové množiny:

Můžeme použít následující úvahu: k tomu, abychom vypsali všechny permutace množiny $A = \{a_1, a_2, \ldots, a_n\}$ stačí, abychom:

- vypsali všechny permutace A začínající a_1,
- vypsali všechny permutace A začínající a_2,

  $\vdots$
- vypsali všechny permutace A začínající a_n.

Přitom všechny permutace A začínající a_i lze vypsat tak, že vypíšeme všechny permutace $A - \{a_i\}$, ke kterým na začátek přidáme a_i (předpis rekurze). Mezním případem je případ, kdy $A = \emptyset$, v tom případě je jedinou permutací $\langle\,\rangle = \emptyset$ (limitní podmínka rekurze).

1.3 Vztah permutací a bijektivních zobrazení

Mezi permutacemi množiny A a bijektivními zobrazeními na množině A je úzký vztah – až na malý rozdíl ve formalizaci se jedná o týž pojem.

Připomeňme, že zobrazení $f\colon A \to A$ je bijektivní, pokud platí:

- f je *injektivní*, to jest pokud $f(a) = f(b)$, pak $a = b$; a dále
- f je *surjektivní*, to jest pro každé $b \in A$ existuje $a \in A$ tak, že $f(a) = b$.

▷ **Příklad 1.3.** Uvažujme $A = \{a, b, c\}$. Pak $f : A \to A$, pro které $f(a) = b$, $f(b) = a$, $f(c) = c$ je bijektivní. Naproti tomu $g\colon A \to A$, pro které $f(a) = b$, $f(b) = b$, $f(c) = c$ není bijektivní (toto zobrazení není ani injektivni a ani surjektivní).

Poznámka. Uvědomte si, že pokud je A konečná, pak každá zobrazení $f\colon A \to A$ je buď bijektivní nebo není ani injektivní, ani surjektivní. Pokud by byla A nekonečná, pak bychom mohli najít injektivní $f\colon A \to A$, které není surjektivní i surjektivní $f\colon A \to A$, které není injektivní.

► **Příklad 1.5.** Uveďte příklady injektivních zobrazení $f\colon A \to A$, která nejsou surjektivní a surjektivních zobrazení, která nejsou injektivní. Příklady najděte pro $A = \mathbb{N}$. [injektivní např. $f(n) = n + 1$; surjektivní např. $g(n) = 1$ pro n liché a $g(n) = n/2$ pro n sudé]

Uvažujme množinu $A = \{a_1, \ldots, a_n\}$ a zvolme pevně pořadí, v jakém budeme vypisovat její prvky, například: $a_1, a_2, \ldots, a_n$. Potom každá permutace $b_1 b_2 \cdots b_n$ prvků z množiny A indukuje bijektivní zobrazení $f\colon A \to A$ tak, že $f(a_i) = b_i$ $(i = 1, \ldots, n)$. Obráceně, každé bijektivní zobrazení $f\colon A \to A$ indukuje permutaci $f(a_1)f(a_2)\cdots f(a_n)$.

► **Příklad 1.6.** Mějme množinu $A = \{a, b, c, d\}$ a zvolme a, b, c, d jako pořadí, ve kterém vypisujeme její prvky. Napište jak vypadá bijektivní zobrazení indukované permutací *bcad*. Dále napište, jak vypadá permutace určená bijektivním zobrazením $f(a) = b$, $f(b) = c$, $f(c) = d$ a $f(d) = a$. [$f(a) = b$, $f(b) = c$, $f(c) = a$ a $f(d) = d$; *bcda*]

1.4 Variace

Uvažujeme podobný problém, jako v případě permutací. Máme k dispozici n vzájemně rozlišitelných objektů a máme jimi vyplnit r prázdných pozic. Zřejmě je jasné, že pro $n < r$ je počet řešení 0, protože vždy zůstanou prázdné pozice. Proto tuto patologickou situaci neuvažujeme a vždy předpokládáme, že $n \geq r$ (po zaplnění všech pozic nám některé objektu mohou zbýt).

Počet všech zaplnění lze opět zdůvodnit pravidlem součinu a to podobně jako v předchozím příkladě. První pozice může být zaplněna n objekty, druhá pozice $n-1$ objekty, a tak dále, až r-tá pozice může být zaplněna $n - r + 1$ objekty. Celkový počet možných zaplnění je tedy:

$$n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot (n-r+1)$$

a nazýváme jej *r-prvkové variace z n prvků* a jejich celkový počet pro r pozic a n objektů označujeme $P_{r,n}$.

Například: pro 4 pozice a 9 objektů:

$$P_{4,9} = 9 \cdot 8 \cdot 7 \cdot 6 = 9 \cdot (9-1) \cdot (9-2) \cdot (9-3).$$

Vlastnosti variací: $P_{r,n}$ lze vyjádřit pomocí faktoriálů:

$$P_{r,n} = \frac{n \cdot (n-1) \cdots (n-r+1) \cdot (n-r) \cdots 3 \cdot 2 \cdot 1}{(n-r) \cdot \cdots \cdot 3 \cdot 2 \cdot 1} = \frac{n!}{(n-r)!}$$

Poznámka. Speciální případy: Pokud $r = n$, pak $P_{r,n} = \dfrac{n!}{(n-r)!} = \dfrac{n!}{0!} = \dfrac{n!}{1} = n!$.

To jest, n-prvkové variace z n prvků = permutace z n prvků.

Existuje právě jediná 0-prvková variace z n prvků: $\langle\,\rangle = \emptyset$.

► **Příklad 1.7.** Stanovte celkový počet vzájemně různých čtyřpísmenných kódů, ne kterých se na každé pozici může vyskytovat právě jedno z 26 písmen a to tak, že žádná dvě písmena v jednom kódu nemohou být stejná. [$P_{4,26} = 26 \cdot 25 \cdot 24 \cdot 23 = \frac{26!}{22!} = 358\,800$]

► **Příklad 1.8.** Stanovte počet způsobů, kterými lze zvolit prezidenta, vice prezidenta, sekretáře a pokladníka klubu, který sestává z celkem 10 členů. Předpokládejme, že žádný člen nemůže zestávat víc jak jednu funkci. [$P_{4,10} = 10 \cdot 9 \cdot 8 \cdot 7 = \frac{10!}{6!} = 5\,040$]

1.5 Algoritmus pro výpis variací

Podobně jako v případě výpisu všech permutací dané množiny můžeme navrhnout algoritmus pro výpis všech r-prvkových variací n-prvkové množiny A. V podstatě jde o jednoduchou modifikaci předchozího algoritmu s dodatečnou limitní podmínkou a malou úpravou předpisu rekurze.

K tomu, abychom vypsali všechny r-prvkové variace množiny $A = \{a_1, a_2, \ldots, a_n\}$ stačí, abychom:

- vypsali všechny r-prvkové variace A začínající a_1,
- vypsali všechny r-prvkové variace A začínající a_2,

  $\vdots$
- vypsali všechny r-prvkové variace A začínající a_n.

Nyní zbývá vyřešit problém, jak vypsat všechny r-prvkové variace začínající konkrétním prvkem a_i. Tento problém můžeme ale snadno redukovat na výpočet $(r-1)$-prvkových variací z množiny bez prvku a_i následovně:

- Označme P množinu všech $(r-1)$-prvkových variací množiny $A - \{a_i\}$;
- na začátek každé variace z P přidej a_i.

Opět je vidět, že postup vede na rekurzivní proceduru s limitní podmínkou, kdy $r = 0$ (v tomto případě je výsledkem jediná variace $\langle\,\rangle = \emptyset$). V ostatních případech se výpočet r-prvkových variací z n-prvkové množiny A redukuje na výpočet $(r-1)$-prvkových variací z n množin $A - \{x\}$ o $(n-1)$ prvcích (předpis rekurze).

1.6 Programovací jazyk R

V rámci cvičení se budou některé problémy řešit pomocí programovacího jazyka GNU R. Jedná se o specializovaný jazyk a knihovny funkcí pro podporu prevděpodobnostních simulací a statistické inference. Nainstalujte si balík R (`http://www.r-project.org/`) a vyzkoušejte si následující základní příklady.

Jazyk R používá klasickou (infixovou) notaci pro psaní aritmetických výrazů. Příkaz přírazení se označuje `<-` a lze jej použít i ve tvaru `->`. Vyzkoušejte si následující příklady:

```
x <- 10
x * 20
10 + x * 20
60 -> y
y + 10
```

Kromě skalárních hodnot (čísel) podporuje R práci s vektory. Konstantní vektor lze vytvořit pomocí literálu `c(···)`. Pokud nad vektory provádáme aritmetické operace, (například sčítání, odčítání,...), jsou prováděny po složnách. Operace lze provádět i mezi vektory a skaláry, pak se opět provádí (s konstantní hodnotou) po složkách. Vyzkoušejte si následující příklady:

```
c(10, 20, 30, 40, 50)
v <- c(10, 20, 30, 40, 50)
10 * v
10 * (v + 5)
w <- c(2, 4, 6, 8, 10)
v + w
v * w
v / w
1 / v
```

Stejně tak, jak je tomu například v jazyku PERL, vektory nemohou obsahovat další vektory. Při vytvoření vektoru pomocí `c(···)`, kde jako některou z hodnot uvedeme další vektor, dojde ke zřetězení hodnot a jejich zařazení do výsledného vektoru. Vyzkoušejte si následující příklady:

```
x <- c(10, 20, 30, 40)
y <- c(x, 666, 777, 2 * x)
```

Operace lze provádět i s vektory různých délek. Pokud jde o operace, které se provádějí po složkách, pak se hodnoty kratšího vektoru periodicky opakují, to jest po použití poslední hodnoty vektoru se jako další použije první hodnota. Vyzkoušejte si následující příklady:

```
v <- c(10, 20, 30, 40, 50, 60)
w <- c(2, 4, 6)
v + w
```

Vektory můžeme chápat jako reprezentaci výběrů. Výběrový průměr, rozptyl a směrodatnou odchylku hodnot (ve vektoru) můžeme stanovit pomocí zabudovaných funkcí `mean`, `var` a `sd`. Vyzkoušejte si následující příklady a všimněte si, jak jsme vyjádřili vzorce pro výpočet střední hodnoty, rozptylu a směrodatné odchylky pomocí zabudovaných funkcí `sum` (součet prvků vektory) a `length` (délka vektoru):

```
x <- c(10, 20, 30, 40, 50)
mean (x)
x - mean (x)
(x - mean (x))^2
sum ((x - mean (x))^2)
sum ((x - mean (x))^2) / (length (x) - 1)
var (x)
sqrt (sum ((x - mean (x))^2) / (length (x) - 1))
sqrt (var (x))
sd (x)
```

Při programování v R lze s výhodou použít několik triků pro usnadnění práce. Pokud píšeme zdrojové kódy do samostatného souboru, můžeme jej jednorázově natáhnout a interpretovat pomocí funkce `source` následovně:

```
source ("commands.R")
```

Pokud bychom chtěli vypsat proměnné, na které jsou aktuálně navázány hodnoty v globálním prostředí interpretu, můžeme tak učinit pomocí funkce `objects`:

```
objects ()
```

Vazbu symbolu můžeme zrušit pomocí funkce `rm`:

```
rm (v)
```

Pomocí funkce `assign` lze navázat hodnotu na proměnnou danou svým jménem (řetězec):

```
assign ("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
```

Reference

[1] Baclawski K. P.: *Introduction to Probability with R*
Chapman & Hall/CRC, 2008, ISBN 978–1–42–006521–3.

[2] Devore J. L.: *Probability and Statistics for Engineering and the Sciences*
Duxbury Press, 7. vydání 2008, ISBN 978–0–495–55744–9.

[3] Hendl, J.: *Přehled statistických metod zpracování dat*
Portál, Praha 2006, ISBN 978–80–7367–123–5

[4] Hogg R. V., Tanis E. A.: *Probability and Statistical Inference*
Prentice Hall; 7. vydání 2005, ISBN 978–0–13–146413–1.