

Další datové struktury a operace s nimi — Úvod do jazyka R

Lubomír Štěpánek^{1, 2}



¹Oddělení biomedicínské statistiky & výpočetní techniky
Ústav biofyziky a informatiky
1. lékařská fakulta
Univerzita Karlova v Praze



²Katedra biomedicínské informatiky
Fakulta biomedicínského inženýrství
České vysoké učení technické v Praze

(2017) Lubomír Štěpánek, CC BY-NC-ND 3.0 (CZ)



Dílo lze dále svobodně šířit, ovšem s uvedením původního autora a s uvedením původní licence. Dílo není možné šířit komerčně ani s ním jakkoliv jinak nakládat pro účely komerčního zisku. Dílo nesmí být jakkoliv upravováno. Autor neručí za správnost informací uvedených kdekoli v předložené práci, přesto vynaložil nezanedbatelné úsilí, aby byla uvedená fakta správná a aktuální, a práci sepsal podle svého nejlepšího vědomí a svých „nejlepších“ znalostí problematiky.

Obsah

- 1 Datové struktury
- 2 Matice a práce s nimi
- 3 Pole
- 4 Datové tabulky a práce s nimi
- 5 Seznamy a práce s nimi
- 6 Literatura

Datové struktury

- vektor (vector)
- faktor (factor)
- matice (matrix)
- pole (array)
- tabulka dat (data.frame)
- seznam (list)

Tvorba matic a základní příkazy

- matice (`matrix`) je dvojrozměrným polem, které obsahuje prvky stejného datového typu
- všechny sloupce matice mají shodnou délku, podobně všechny řádky matice mají shodnou délku
- ať je

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

- v R matici A a B získáme příkazy

```
1 | A <- matrix(c(1, 2, 3, 4), nrow = 2,  
2 |                       ncol = 2)  
3 | B <- matrix(c(1, 3, 2, 4), nrow = 2,  
4 |                       ncol = 2)  
5 | B <- matrix(c(1, 2, 3, 4), nrow = 2,  
6 |                       ncol = 2, byrow = TRUE)  
7 | # vždy jeden z argumentů "nrow" či "ncol" je zbytečný
```

Manipulace s maticemi

- at' je

$$C = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix}$$

- v R pak pomocí

```
1 || C <- matrix(letters[1:12], nrow = 3,  
2 || byrow = T)
```

- některé užitečné příkazy

```
1 || is.matrix(C) # TRUE  
2 || class(C) # "matrix"  
3 || mode(C) # "character"; datový typ prvků  
4 || str(C) # chr [1:3, 1:4] "a" "e" "i" ...  
5 || dim(C) # c(3, 4); rozměry matice C
```

Manipulace s maticemi

- at' je

$$C = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix}$$

- další užitečné příkazy

```

1 | colnames(C) <- c("c1", "c2", "c3", "c4")
2 | rownames(C) <- c("r1", "r2", "r3")
3 |           # přidá jmenovky sloupcům i řádkům
4 | C <- unname(C)
5 |           # zbaví sloupce i řádky jmenovek
6 | dimnames(C) <- list(
7 |           c("r1", "r2", "r3"),
8 |           c("c1", "c2", "c3", "c4")
9 |         )
10 |           # opět přidá jmenovky sloupcům i řádkům

```

Manipulace s maticemi

- stále mějme

$$C = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix}$$

- další užitečné příkazy

```
1  rbind(C, c("x", "x", "x", "x"))  
2      # přidání řádku c("x", "x", "x", "x")  
3      # k matici C  
4  cbind(C, c("x", "x", "x"))  
5      # přidání sloupce c("x", "x", "x")  
6      # k matici C  
7  C[-1, ]      # odebrání 1. řádku matici C  
8  C[, -2]      # odebrání 2. sloupce matici C
```


Submatice, indexování, adresace

- ať je

$$C = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix}$$

- v R pomocí

```
1  C <- matrix(letters[1:12], nrow = 3,
2  byrow = T, dimnames = list(
3  c("r1", "r2", "r3"),
4  c("c1", "c2", "c3", "c4")))
```

- adresace

```
1  C[2, 3]           # "g"; prvek 2. řádku, 3. sloupce
2  C["r2", "c3"]    # "g"; prvek 2. řádku, 3. sloupce
3  C[1, ]           # c("a", "b", "c", "d");
4  # tedy vektor 1. řádku matice C
5  # s popisky
```

Submatice, indexování, adresace

- stále mějme

$$C = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix}$$

- adresace

```

1      C[, 3]           # c("c", "g", "k");
2                        # tedy vektor 3. sloupce matice C
3                        # s popisky
4      C[c(1, 3), c(2, 4)]
5                        # matrix(c("b", "j", "d", "l"), 2)
6                        # submatice 1. a 3. řádku,
7                        # 2. a 4. sloupce matice C
8                        # s popisky
9      C["r2", ]        # c("e", "f", "g", "h");
10                        # tedy vektor 2. řádku matice C
11                        # s popisky

```

Submatice, indexování, adresace

- stále mějme

$$C = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix}$$

- adresace

```

1 | C[dim(C)[1], dim(C)[2]]
2 |     # "l"; obecná adresace prvku
3 |     # vpravo dole (např. neznáme-li
4 |     # číselné rozměry matice)
5 | C[5]     # "f"; major-column ordering
6 | C[c(8, 9)] # c("g", "k")
7 | C[13]    # NA
8 | diag(C)   # c("a", "f", "k"); hlavní diagonála
9 | diag(C[, dim(C)[2]:1])
10 |         # c("d", "g", "j"); vedlejší diagonála

```

Intermezzo

- mějme matici

$$\mathbf{X} = \begin{pmatrix} 1 & 3 & 9 & 6 \\ -5 & 0 & 1 & 2 \\ -8 & 4 & 7 & 3 \end{pmatrix}$$

- pomocí R

- založme matici \mathbf{X}
- najdeme prvek třetího řádku druhého sloupce matice \mathbf{X}
- jaký řádkový a sloupcový index mají prvky 0, 7 matice \mathbf{X}
- najdeme minimum matice \mathbf{X} a řádkový a sloupcový index této hodnoty
- najdeme stopu matice \mathbf{X}

Intermezzo

- mějme matici

$$X = \begin{pmatrix} 1 & 3 & 9 & 6 \\ -5 & 0 & 1 & 2 \\ -8 & 4 & 7 & 3 \end{pmatrix}$$

- řešení

```
1      # (i)
2      X <- matrix(c(1, -5, -8, 3, 0, 4, 9, 1,
3                    7, 6, 2, 3), nrow = 3)
4
5      # (ii)
6      X[3, 2]      # 4
7
8      # (iii)
9      c((which(X == 0) - 1) %% dim(X)[1] + 1,
10        ceiling(which(X == 0) / dim(X)[1]))
11      c((which(X == 7) - 1) %% dim(X)[1] + 1,
12        ceiling(which(X == 7) / dim(X)[1]))
```

Intermezzo

- mějme matici

$$X = \begin{pmatrix} 1 & 3 & 9 & 6 \\ -5 & 0 & 1 & 2 \\ -8 & 4 & 7 & 3 \end{pmatrix}$$

- řešení

```

1      # (iii) lépe
2      which(X == 0, ind.arr = TRUE)
3      which(X == 7, ind.arr = TRUE)
4          # which(X == 1, ind.arr = TRUE)
5          # vrátí c(1, 2) i c(2, 3)
6
7      # (iv)
8      min(X); which(X == min(X), arr.ind = T)
9          # -8; c(3, 1)
10
11     # (v)
12     sum(diag(X))
13         # 8 := 1 + 0 + 7

```

Maticová algebra

- bud'te

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \quad b = (1, 0)^T$$

- v R

```
1 || A <- matrix(c(1, 2, 3, 4), nrow = 2)
2 || B <- matrix(c(5, 6, 7, 8), nrow = 2)
3 || b <- c(1, 0)
```

- Hadamardův součin (*element-wise, pairwise*) $A \circ B = \begin{pmatrix} 5 & 21 \\ 12 & 32 \end{pmatrix}$

```
1 || A * B # matrix(c(5, 12, 21, 32), 2)
```

- maticový součin $A \cdot B = \begin{pmatrix} 23 & 31 \\ 34 & 46 \end{pmatrix}$

```
1 || A %*% B # matrix(c(23, 34, 31, 46), 2)
```

Maticová algebra

- bud'te

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \quad \mathbf{b} = (1, 0)^T$$

- transpozice $A^T = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

```
1 ||      t(A)          # matrix(c(1, 3, 2, 4), 2)
```

- jednotková matice $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

```
1 ||      diag(2)       # matrix(c(1, 0, 0, 1), 2)
```

- inverzní matice $A^{-1} = \begin{pmatrix} -2 & 3/2 \\ 1 & -1/2 \end{pmatrix}$

```
1 ||      solve(A)      # matrix(c(-2, 1, 1.5, -0.5), 2)
```


Maticová algebra

- buďte

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \quad b = (1, 0)^T$$

- vlastní čísla matice A jsou $\lambda = (\frac{1}{2}(5 + \sqrt{33}), \frac{1}{2}(5 - \sqrt{33}))^T$

```
1 ||      eigen(A)$values
```

- vlastní vektory matice A jsou například

$$v = [(\frac{1}{4}(-3 + \sqrt{33}), 1)^T, (\frac{1}{4}(-3 - \sqrt{33}), 1)^T]$$

```
1 ||      eigen(A)$vectors      # R vrací ortonormální
2 ||                                # vlastní vektory
```

- řešení x ze soustavy $Ax = b$ je $x = (-2, 1)^T$

```
1 ||      solve(A, b)           # c(-2, 1)
```

Intermezzo

- uvažujme markovský proces s počátečním stavovým vektorem $p^0 = (1, 0, 0)^T$ a přechodovou maticí

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

kde v i -tém řádku jsou vždy po řadě pravděpodobnosti přechodu z i -tého stavu do j -tého stavu pro $\forall i, j \in \{1, 2, 3\}$

- pomocí R
 - ověříme, že matice S je stochastická
 - určíme pravděpodobnosti, s jakými proces dospěl po pěti krocích do prvního, druhého či třetího stavu
 - určíme, zda existuje dynamicky rovnovážný stav procesu

Intermezzo

- uvažujme markovský proces s počátečním stavovým vektorem $p^0 = (1, 0, 0)^T$ a přechodovou maticí

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

kde v i -tém řádku jsou vždy po řadě pravděpodobnosti přechodu z i -tého stavu do j -tého stavu pro $\forall i, j \in \{1, 2, 3\}$

- řešení

```
1 | S <- matrix(c(1/4, 1/2, 1/4,  
2 |                2/3, 0, 1/3,  
3 |                1/2, 1/2, 0), nrow = 3,  
4 |                byrow = TRUE)  
5 | p <- c(1, 0, 0)
```

Intermezzo

- uvažujme markovský proces s počátečním stavovým vektorem $p^0 = (1, 0, 0)^T$ a přechodovou maticí

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

kde v i -tém řádku jsou vždy po řadě pravděpodobnosti přechodu z i -tého stavu do j -tého stavu pro $\forall i, j \in \{1, 2, 3\}$

- řešení

```
1 || # (i)
2 || rowSums(S) # c(1, 1, 1)
```

Intermezzo

- uvažujme markovský proces s počátečním stavovým vektorem $p^0 = (1, 0, 0)^T$ a přechodovou maticí

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

kde v i -tém řádku jsou vždy po řadě pravděpodobnosti přechodu z i -tého stavu do j -tého stavu pro $\forall i, j \in \{1, 2, 3\}$

- řešení

```
1 | # (ii)
2 | install.packages("expm", dependency = T)
3 | library("expm")
4 | p %*% (S %^% 5)      # c(0.437, 0.344, 0.219)
```

Intermezzo

- uvažujme markovský proces s počátečním stavovým vektorem $p^0 = (1, 0, 0)^T$ a přechodovou maticí

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

kde v i -tém řádku jsou vždy po řadě pravděpodobnosti přechodu z i -tého stavu do j -tého stavu pro $\forall i, j \in \{1, 2, 3\}$

- řešení

```

1  # (iii)
2  p %*% (S %^% 10) # c(0.444, 0.333, 0.222)
3  p %*% (S %^% 20) # c(0.444, 0.333, 0.222)
4  p %*% (S %^% 30) # c(0.444, 0.333, 0.222)
5  p %*% (S %^% 40) # c(0.444, 0.333, 0.222)

```

Tvorba polí a práce s nimi

- pole je objekt s vlastnostmi matice, ale libovolným počtem dimenzí
- např.

```
1 | (X <- array(c(1:24), dim = c(4, 3, 2)))
2 | # , , 1
3 | # [,1] [,2] [,3]
4 | # [1,] 1 5 9
5 | # [2,] 2 6 10
6 | # [3,] 3 7 11
7 | # [4,] 4 8 12
8 | # , , 2
9 | # [,1] [,2] [,3]
10 | # [1,] 13 17 21
11 | # [2,] 14 18 22
12 | # [3,] 15 19 23
13 | # [4,] 16 20 24
```

Manipulace s poli

- navigace v poli

```

1 || X[, , 1]      # první vrstva
2 || X[, , 2]      # druhá vrstva
3 || X[3, , 1]     # třetí řádek první vrstvy
4 || X[, 2, ]      # druhé sloupce všech vrstev

```

- pole se příliš nepoužívají, protože vstupem pro analytické funkce jazyka R je obvykle matice či datová tabulka
- pole libovolných rozměrů lze převést na dvourozměrnou matici pomocí dummy proměnné „vrstva“

Tvorba datových tabulek a základní příkazy

- datová tabulka (`data.frame`) je dvojrozměrným polem, které obsahuje v každém sloupci prvky stejného datového typu, ale jednotlivé sloupce se mohou datovým typem lišit
- všechny sloupce datové tabulky mají shodnou délku, podobně všechny řádky datové tabulky mají shodnou délku
- v R je řada vestavěných datových tabulek

```
1 | mtcars
2 | str(mtcars)
3 | class(mtcars)           # "data.frame"
4 | mode(mtcars)            # "list"
5 | is.data.frame(mtcars)   # TRUE
6 | str(iris)
7 |      # 'data.frame': 150 obs. of 5 variables
8 |      # ...
9 | dim(iris)                # c(150, 5)
```

Manipulace s datovými tabulkami

• užitečné příkazy

```
1 data <- mtcars
2 colnames(data)
3 colnames(data) <- paste("c",
4                           1:dim(data)[2],
5                           sep = "_")
6 rownames(data) <- paste("r",
7                           1:dim(data)[1],
8                           sep = "_")
9                               # změní jmenovky sloupcům i řádkům
10 head(data)                 # náhled na prvních 6 řádků
11 head(data, 10)
12                               # náhled na prvních 10 řádků
13 tail(data)                  # náhled na posledních 6 řádků
14 tail(data, 10)
15                               # náhled na posledních 10 řádků
```

Manipulace s datovými tabulkami

- další užitečné příkazy

```
1      rbind(data, rep(0, dim(data)[2]))
2          # přidání řádku c(0, 0, ..., 0)
3          # k data.frameu "data"
4      cbind(data, rep(0, dim(data)[1]))
5          # přidání sloupce c(0, 0, ..., 0)
6          # k data.frameu "data"
7      data.frame(data,
8                  "ahoj" = rep(0, dim(data)[1]))
9          # přidání sloupce c(0, 0, ..., 0)
10         # se jménem "ahoj" k data.frameu
11         # "data"
12      data[-1, ] # odebrání 1. řádku data.frameu
13         # "data"
14      data[, -1] # odebrání 1. sloupce data.frameu
15         # "data"
```

Indexování, adresace

- adresace

```
1      data[2, 3]      # 160; prvek 2. řádku, 3. sloupce
2      data["r_2", "c_3"]
3                                     # 160; prvek pod danými popisky
4      data[1, ]       # c(21, 6, 160, 110, ...);
5                                     # tedy vektor 1. řádku data.frameu
6                                     # "data" s popisky
7      data[, 2]       # c(6, 6, 4, 6, ...);
8                                     # tedy vektor 2. sloupce
9                                     # data.frameu "data" s popisky
10     data$c_5         # c(3.90, 3.90, 3.85, 3.08, ...);
11                                     # tedy vektor 2. sloupce
12                                     # data.frameu "data" s popisky
13     data$c_5[1]      # 3.9;
14                                     # tedy první prvek 2. sloupce
15                                     # data.frameu "data"
```

Indexování, adresace

- adresace

```
1 | data[dim(data)[1], dim(data)[2]]
2 |     # 2; obecná adresace prvku
3 |     # vpravo dole (např. neznáme-li
4 |     # číselné rozměry tabulky)
5 | data[5]     # 5. sloupec, nikoliv major-column
6 |     # ordering
```

Sloupcové přehledy

- může mít smysl znát agregovaný ukazatel nad všemi sloupci

```
1      colSums(data)
2          # součty všech sloupců
3      apply(data, 2, sum)
4          # totéž
5      colMeans(data)
6          # průměry všech sloupců
7      apply(data, 2, mean)
8          # totéž
9      data <- rbind(data, rep(NA, dim(data)[2]))
10         # přidán řádek c(NA, NA, ...)
11      colMeans(data)
12         # c(NA, NA, ...); pro získání průměrů
13         # nutné přidat argument na.rm = TRUE
14      colMeans(data, na.rm = TRUE)
15      apply(data, 2, mean, na.rm = TRUE)
16         # už funguje
```

Tvorba seznamů a základní příkazy

- seznam (`list`) je výčtem prvků, které mohou být různorodého datového typu, včetně listu
- délky jednotlivých prvků seznamu se mohou lišit

```
1 | my_list <- list("a" = c(1:10),
2 |               "b" = mtcars,
3 |               "c" = matrix(1:8, 2),
4 |               "z" = "ahoj")
5 |
6 | str(my_list)
7 | class(my_list)           # "list"
8 | mode(my_list)           # "list"
  | is.list(my_list)        # TRUE
```

Manipulace se seznamy

- užitečné příkazy

```
1 | names(my_list)
2 | names(my_list) <- LETTERS[
3 |     1:length(my_list)
4 | ]           # přejmenovávám prvky listu
5 |
6 | my_list[[length(my_list) + 1]] <- c(T, F)
7 | names(length(my_list)) <- "XY"
8 |           # přidání vektoru c(T, F)
9 |           # k listu "my_list" pod jménem
10 |          # "XY"
```


Indexování, adresace

- adresace

```
1 | my_list[[2]] # 2. prvek listu
2 | my_list[["B"]]
3 | # prvek listu pod jmenovkou "B"
4 | # jde o původní datový typ
5 | # (data.frame)
6 | my_list["B"] # prvek listu pod jmenovkou "B"
7 | # jde (vždy) o list
8 | my_list[c(2, 4)]
9 | # 2. a 4. prvek listu
10 | my_list$C # prvek listu pod jmenovkou "C"
11 | my_list[[1]][2]
12 | # 2; 2. prvek 1. prvku listu
13 | my_list[[2]][3, 5]
14 | # 3.85; z 2. prvku listu vybírám
15 | # prvek o souřadnicích (3, 5)
```

Indexování, adresace pomocí funkce `lapply()`

- adresace

```
1      set.seed(1)
2      my_long_list <- lapply(
3          sample(c(80:120), 100, TRUE),
4          function(x) sample(
5              c(50:150), x, replace = TRUE
6          )
7      ) # list vektorů náhodné délky
8        # generovaných z náhodných čísel
9
10     lapply(my_long_list, "[", 14)
11         # z každého prvku listu (vektoru)
12         # vybírám jen jeho 14. prvek
```

Prvkové přehledy

- může mít smysl znát agregovaný ukazatel nad všemi prvky seznamu

```
1 | lapply(my_long_list, mean)
2 |   # pro každý prvek listu (vektor)
3 |   # vrátím jeho průměr
4 |
5 | lapply(my_long_list, length)
6 |   # pro každý prvek listu (vektor)
7 |   # vrátím jeho délku
```

Literatura



ZVÁRA, Karel. *Základy statistiky v prostředí R*.
Praha, Česká republika: Karolinum, 2013. ISBN 978-80-246-2245-3.



WICKHAM, Hadley. *Advanced R*.
Boca Raton, FL: CRC Press, 2015. ISBN 978-1466586963.

Děkuji za pozornost!

lubomir.stepanek@lf1.cuni.cz

lubomir.stepanek@fbmi.cvut.cz