

# Úvod, instalace R a Rstudio, základní datové typy a struktury, operace

17VSADR – Skriptování a analýza dat v jazyce R

Lubomír Štěpánek<sup>1, 2</sup>



<sup>1</sup>Oddělení biomedicínské statistiky  
Ústav biofyziky a informatiky  
1. lékařská fakulta  
Univerzita Karlova v Praze



<sup>2</sup>Katedra biomedicínské informatiky  
Fakulta biomedicínského inženýrství  
České vysoké učení technické v Praze

(2020) 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 Úvod
- 2 Začínáme
- 3 Datové typy
- 4 Datové struktury
- 5 Vektory
- 6 Operace
- 7 Matice
- 8 Datové tabulky
- 9 Seznamy

# Organizace předmětu

- volitelný předmět (2 kreditové body)
- zakončen zápočtem
- rozsah 0 + 2 (vyučovacích hodin týdně)
- neoficiální účast na bázi dobrovolnosti vítána

# Zápočet

- zápočet bude udělen za
  - alespoň rozumnou účast na cvičeních
  - odevzdání alespoň rozumného množství domácích úkolů

# Online supplementum

fakultní stránka předmětu ▶ FBMI

<https://predmety.fbmi.cvut.cz/cs/17VSADR>

githubí stránka předmětu ▶ GitHub

[https://github.com/LStepanek/17VSADR\\_Skriptovani\\_a\\_analyza\\_dat\\_v\\_jazyce\\_R](https://github.com/LStepanek/17VSADR_Skriptovani_a_analyza_dat_v_jazyce_R)

# Literatura



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



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

# Proč bych se právě já měl učit s R

- s vědou (nejen) v biomedicině to myslím vážně
- chtěl bych se věnovat postgraduálnímu doktorskému studiu, kde jistojistě budu potřebovat statistiku
- komerční statistické programy pro mě nejsou dostupné, nebo nejsou dobře použitelné
- sbírám opakovaně data stejného charakteru a rád bych si jejich (před)zpracování automatizoval
- publikuji v odborných časopisech a rád bych do statě článku s Methodology and Statistical Analysis pravdivě psal, že „... all statistical analyses were performed using R language for statistical computing and graphics...“, protože existují důkazy, že citování R či jiných volných statistických nástrojů mnohdy zvyšuje pravděpodobnost citování takového článku



# Proč bych se právě já měl učit s R

- tuším, že věda 2.0 v biomedicině se bude provozovat nejen formou experimentů na živém (in vivo) či v laboratořích (in vitro), ale budou ji tvořit ze značné části počítačové modely a simulace (in silico)
- uvědomuji si, že MS Excel v základním rozhraní neumí doted' vykreslit krabicový diagram
- zpracování dat v tabulkových procesorech a spoléhání se jen na ně je spjato s různými problémy, chybné výstupy z tabulkových procesorů dokonce vyvolaly některé vědecké skandály
- data již nějakou dobu (sám) analyzuji a přemýšlím, který nástroj pro analýzu (s kvalitní dokumentací a živou podporou a komunitou) se začít učit

# Co je R



- R je interpretovaný programovací jazyk
- kombinuje několik paradigmat
  - imperativní
  - funkcionální
  - objektové
- R je *domain specific language* – je určen pro statistickou analýzu dat a jejich grafické zobrazení
- R je open-source, konkrétně *free-as-in-beer* a *free-as-in-speech*

# Stažení a instalace jádra R

- na stránkách R-projectu

<https://www.r-project.org/>

postupně **download R**, vyberme českou doménu a stáhněme desktopově

- poté instalujme dle instrukcí do předvolené složky

# Stažení a instalace RStudio

- RStudio je jedním z grafických IDE (Integrated Development Environment) jazyka R
- na stránkách RStudio

<https://www.rstudio.com/>

postupně **Products > RStudio > Desktop > Open Source Edition > Free > Download**, stáhněme desktopově

- poté instalujme dle instrukcí do předvolené složky

# Ahoj světe!

- do skriptu či konzole napíšme

```
1 || print("hello world")
```

- dostaneme

```
1 || [1] "hello world"
```

# Práce s nápovědou

- nápovědu pro funkci či objekt získáme pomocí příkazu `help()`, kde argumentem je název funkce či objektu

```
1 || help(print)
```

- nebo předsazením symbolu `?` před název funkce či objektu

```
1 || ?print
```

- předsazením symbolů `??` před název funkce či objektu prohledáme veškeré dokumenty nápovědy

```
1 || ??print
```

- vždy je zavolán HTML soubor s volnotextovou nápovědou

# Datové typy

- numerická hodnota (numeric)
- celé číslo (integer)
- komplexní číslo (complex)
- logická hodnota (logical)
- textový řetězec (character)
- NA, NULL, NaN

## Numerická hodnota

- v R jako `numeric`
- libovolné  $x \in \mathbb{R}$  uložené s danou přesností
- odpovídá datovému typu `double` s 64 bitovou přesností, který je běžný v jiných jazycích
- např.

```
1 || 5; -13.8, 4.5578e15
```

- zda je hodnota typu `numeric`, zjistíme pomocí

```
1 is.numeric(-13.8) # TRUE
2 class(-13.8)      # "numeric"
3 class(Inf)         # "numeric"
```

- vhodná pro různorodé operace (viz dále)





# Logická hodnota

- v R jako logical
- libovolné booleovské  $x \in \{\text{TRUE}, \text{FALSE}\}$
- např.

```
1 || TRUE; FALSE; T; F
```

- zda je hodnota typu `logical`, zjistíme pomocí

```
1 is.logical(TRUE)      # TRUE
2 class(FALSE)          # "logical"
3 class("TRUE")         # "character"
4 class(T)              # "logical"
5 class(F)              # "logical"
```

# Textový řetězec

- v R jako character
- libovolná sekvence znaků (extended ASCII) uzavřená mezi jednoduchými či dvojíty uvozovkami
- např.

```
1 || "ahoj"; 'xweiwogw23425ng'; ""
```

- zda je hodnota typu character, zjistíme pomocí

```
1 || is.character("ahoj")      # TRUE
2 || class("bla bla")         # "character"
3 || class("123")             # "character"
4 || class(123)                # "numeric"
5 || is.numeric(Inf)           # TRUE
6 || is.numeric("Inf")         # FALSE
```

- na textový řetězec lze převést libovolnou jinou hodnotu pomocí

```
1 || as.character(123)
```



# Přiřazení hodnoty k proměnné

- přiřadit hodnotu nějaké proměnné lze pomocí jednoduchého rovnítka

```
1 || x = 5
```

- nebo pomocí orientované šipky

```
1 || x <- 5
2 || 5 -> x      # totéž
```

- anebo pomocí funkce `assign()`, kde prvním argumentem je název proměnné (tedy textový řetězec) a druhým hodnota

```
1 || assign("x", 5)      # analogické k x <- 5 či x = 5
```

to se hodí zejména u dynamického iterování (viz později)

# Datové struktury

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

# Tvorba vektorů a základní příkazy

- vektor je jednorozměrný výčet prvků stejného datového typu, nemá orientaci ve smyslu řádek či sloupec
- vektor je objekt typu *tuple*, tedy zachovává pořadí svých prvků (na rozdíl od objektů typu *set*)
- lze vytvořit pomocí generické funkce `c()`, neboli *concatenate*
- např.

```
1 || c() # prázdný vektor
2 || length(c()) # 0
3 || c(3, 1, 2) # vektor o délce 3 a prvcích 3, 1, 2
4 || c("a", "d") # vektor o dél. 2 a prvcích "a", "d"
```

- pomocí funkce `c()` lze vektory i prodlužovat

```
1 || c(c(3, 1, 2), 4) # vektor o prvcích 3, 1, 2, 4
2 || c(3, 1, 2, 4) # zkráceně totéž
```

# Tvorba vektorů a základní příkazy

- vektor tedy lze prodloužit libovolně o jednu či více hodnot

```

1      x <- c(3, 1, 2)
2      length(x)           # 3
3      y <- 1
4      z <- c(2)
5      w <- c(5, 7)
6      x <- c(x, y)         # prodloužení vektoru x
7                           # o hodnotu y
8      w <- c(w, z)         # prodloužení vektoru w
9                           # o vektor z
10                                # jednoprvkový vektor je
11                                # skalárem, jednou hodnotou
12      c <- c(1, 2, 3)
13      c                   # vektor o prvcích 1, 2, 3
14                           # byť je c referovaný termín,
15                           # funkce c je zachována
16                           # a vznikl vektor c

```



# Vektory textových řetězců

- vektory obsahující textové hodnoty, lze je použít např. jako názvy prvků jiného vektoru

```
1 x <- c(3, 1, 2)
2 y <- c("a", "b", "c")
3 names(x) <- y      # pojmenuje prvky
4                    # vektoru x
5
6 x
7 unname(x)          # zbaví prvky vektoru
8                    # x jeho jejich jmen
9
10 setNames(x, y)     # opět pojmenuje
11                   # prvky vektoru x
```

# Subvektory, indexování, adresace

- R indexuje vektory od 1, nikoliv od 0 (první prvek má index 1, druhý index 2, apod.)
- adresujeme pomocí hranatých závorek [ ]

```

1 | x <- c(4, 2, 6, -3)
2 | x[1]                # 4
3 | x[1:2]              # c(4, 2)
4 | x[5]                # NA
5 | x[length(x)]       # -3
6 | x[c(1, 3, 4)]      # c(4, 6, -3)
7 | x[length(x):1]     # c(-3, 6, 2, 4)
8 | rev(x)              # totéž, c(-3, 6, 2, 4)

```

# Logické vektory

- používají se (nejen) k adresování vhodných prvků

```

1 | y <- c(TRUE, TRUE, FALSE, TRUE) # logický
2 |                                     # vektor
3 |
4 | x <- c(3, 1, 2, 5)
5 | x[y]                               # (sub)vektor c(3, 1)
6 | x[c(F, T, F, T)]                 # subvektor c(1, 5)

```

- výhodný je někdy tzv. *recycling*

```

1 | z <- c("R", "G", "E", "F", "I")
2 | z[c(T, F)] # vybere pouze hodnoty
3 |             # na lichých pozicích,
4 |             # tedy "R", "E", "I"
5 |             # neboli vektor
6 |             # c("R", "E", "I")

```

# Faktory

- vektory textových hodnot, kde každá hodnota patří do své kategorie

```
1 x <- factor(  
2   c("muž", "žena", "muž", "muž")  
3 )           # pořadí kategorií je defaultně  
4             # abecední  
5 x <- factor(  
6   c("muž", "žena", "muž", "muž"),  
7   levels = c("žena", "muž")  
8 )           # zde si pořadí kategorií  
9             # určíme sami
```

- nad faktory snadno vytvoříme kontingenční tabulku

```
1 table(x)      # x
2               # žena muž
3               # 1 3
```

# Aritmetické operace

operace	operátor	příklad
sčítání	+	2 + 3
odčítání	-	2 - 3
násobení	*	2 * 3
dělení	/	2 / 3
mocnění	^ či **	2 ^ 3 či 2 ** 3
modulo <sup>1</sup>	%%	7 %% 3
celočíselné dělení	%/%	7 %/% 3

<sup>1</sup>zbytek po celočíselném dělení



# Logické operace

- operace NOT (operátor !)

- vrací výroku opačnou logickou hodnotu, než které výrok nabývá

```
1 || ! TRUE # FALSE
2 || ! 2 > 3 # TRUE
```

- funkce all()

- vrací pro vektor výroků TRUE právě tehdy, jsou-li všechny výroky rovné TRUE

```
1 || all(c(3 > 2, 7 %% 3 == 1, 1 == 0)) # FALSE
2 || all(c(3 > 2, 7 %% 3 == 1, 1 >= 0)) # TRUE
```

- funkce any()

- vrací pro vektor výroků TRUE právě tehdy, je-li alespoň jeden z výroků roven TRUE

```
1 || any(c(3 < 2, 7 %% 3 <= 0, FALSE)) # FALSE
2 || any(c(3 < 2, 7 %% 3 >= 1, FALSE)) # TRUE
```

# Operace porovnávání (komparace)

- pomocí operací porovnávání lze srovnat velikost či pořadí dvou objektů stejného datového typu; datový typ může přitom být v podstatě libovolný
- výsledkem operace porovnání je výrok, tedy hodnota datového typu `logical`
- porovnání typu *je rovno* (`==`, `all.equal()`, `identical()`)

```
1      2 == 3                                # FALSE
2      all.equal(c(1, 2), c(1, 2 + 1e-13),
3                tolerance = 1e-12)
4                                          # TRUE; porovnává vektory
5                                          # volitelnou danou tolerancí
6      identical(c(1, 2), c(1, 2 + 1e-13))
7                                          # FALSE, porovnává objekty
8                                          # a vrací TRUE jen při úplné
9                                          # shodě
```





# Operace porovnávání (komparace)

- porovnání typu *je obsaženo* ve (`%in%`)

```
1 | c(2, 6) %in% c(1:5)           # c(TRUE, FALSE)
2 | "k" %in% LETTERS              # FALSE
3 | "J" %in% letters              # FALSE
4 | "May" %in% month.name         # TRUE
5 | '%in%'("Jan", month.abb)      # TRUE; prefix notace
6 | "a" %in% "abeceda"           # FALSE
```

- ekvivalentem (wrapperem) operace `%in%` je funkce `is.element()`

```
1 | is.element(c(2, 6), c(1:5))
2 |                               # c(TRUE, FALSE)
3 | is.element(c(1:5), c(2, 6))
4 |                               # c(FALSE, TRUE,
5 |                               # FALSE, FALSE, FALSE)
```

- funkce typu *je pravdou* (`isTRUE()`)

```
1 | isTRUE(3^2 > 2^3)           # TRUE
```



# Zaokrouhlování, formátování čísel

- zaokrouhlení čísla  $x$  pomocí `round(x, digits)` na `digits` desetinných míst

```
1 | round(1.4, digits = 0)      # 1
2 | round(-146.655, 2)         # -146.66
```

- zaokrouhlení čísla  $x$  pomocí `signif(x, digits)` na `digits` platných cifer

```
1 | signif(1.458, digits = 1)  # 1
2 | signif(1.458, digits = 2)  # 1.5
3 | signif(1.458, digits = 3)  # 1.46
4 | signif(1.458, digits = 4)  # 1.458
```

- formátování čísla  $x$  pomocí `format(x, nsmall)` na `nsmall` fixních desetinných cifer

```
1 | format(1.45, nsmall = 1)    # "1.45"
2 | format(1.45, nsmall = 2)    # "1.45"
3 | format(1.45, nsmall = 3)    # "1.450"
```

# 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

- atť 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

- 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

```





# 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
4      data[1, ]      # 160; prvek pod danými popisky
5                      # c(21, 6, 160, 110, ...);
6                      # tedy vektor 1. řádku data.frameu
7                      # "data" s popisky
8      data[, 2]      # c(6, 6, 4, 6, ...);
9                      # tedy vektor 2. sloupce
10     data$c_5        # data.frameu "data" s popisky
11                     # c(3.90, 3.90, 3.85, 3.08, ...);
12                     # tedy vektor 2. sloupce
13     data$c_5[1]     # data.frameu "data" s popisky
14                     # 3.9;
15                     # tedy první prvek 2. sloupce
16                     # 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"
9  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
4      # prvek listu pod jmenovkou "B"
5      # jde o původní datový typ
6      # (data.frame)
7      my_list["B"]      # prvek listu pod jmenovkou "B"
8      # jde (vždy) o list
9      my_list[c(2, 4)]
10     # 2. a 4. prvek listu
11     my_list$C          # prvek listu pod jmenovkou "C"
12     my_list[[1]][2]
13     # 2; 2. prvek 1. prvku listu
14     my_list[[2]][3, 5]
15     # 3.85; z 2. prvku listu vybírám
16     # 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 |   # vracím jeho průměr
4 |
5 | lapply(my_long_list, length)
6 |   # pro každý prvek listu (vektor)
7 |   # vracím jeho délku
```

Děkuji za pozornost!

lubomir.stepanek@lf1.cuni.cz

lubomir.stepanek@fbmi.cvut.cz

► GitHub

[https://github.com/LStepanek/17VSADR\\_Skriptovani\\_a\\_analyza\\_dat\\_v\\_jazyce\\_R](https://github.com/LStepanek/17VSADR_Skriptovani_a_analyza_dat_v_jazyce_R)