Načítání a ukládání dat do a z R, R jako programovací jazyk

—

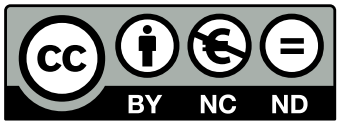
B03128 – Úvod do skriptovacího jazyka R (zkrácená online verze)

Lubomír Štěpánek[1, 2]

[1]Oddělení biomedicínské statistiky
Ústav biofyziky a informatiky
1. lékařská fakulta
Univerzita Karlova v Praze

ČVUT FBMI

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



Obsah

Pracovní složka

- zjištění, která složka je pracovní

```
1 getwd()
```

- výpis obsahu pracovní složky formou vektoru

```
1 dir()
```

- nastavení pracovní složky

```
1 setwd("C:/.../my_working_directory")
```

- nastavení pracovní složky dialogovým oknem

```
1 setwd(choose.dir())
```

Import a export volného textu

- pomocí funkcí `readLines()` a `writeLines()`
- lze tak nahrát libovolná data, která mají textovou reprezentaci

```
my_html <- readLines(
    con = paste(
        "https://ubi.lf1.cuni.cz",
        "uvod-do-skriptovaciho-jazyka-r",
        sep = "/"
    ),
    encoding = "UTF-8"
)
```

Import a export volného textu

- uložení a načtení desktopového dokumentu

```
writeLines(                    # ukládám textový dokument
    text = paste(
        "One R to rule them all",
        "one R to find them",
        "one R to bring them all",
        "and in the darkness bind them",
        sep = "\n"   # separátor typu nový řádek
    ),
    con = "my_text.txt"
)

my_loaded_text <- readLines(
    con = "my_text.txt",
    encoding = "UTF-8"
)                              # načítám textový dokument
```

Import a export dat tabulky

- základní funkcí je `read.table()` a `write.table()`
- obě funkce mají řadu wrapperů (`read.csv()` a `write.csv()`, `read.delim()` a `write.delim()` a další)

```
1   write.table(                    # ukládám data.frame
2       x = mtcars,
3       sep = ";",
4       row.names = FALSE,
5       file = "mtcars.csv"  # anebo "mtcars.txt"
6   )
7
8   my_mtcars <- read.table(
9       file = "mtcars.csv",
10      sep = ";",
11      header = TRUE
12  )                               # načítám data.frame
```

Import a export dat tabulky

```
write.table(x = iris,
            file = "iris.txt")
```

- funkce `read.table()` má spoustu užitečných argumentů

```
my_iris <- read.table(
    file = "iris.txt",
    sep = " ",
    header = TRUE,
    stringsAsFactors = FALSE,
    check.names = FALSE,
                  # vynechá kontrolu korektnosti
                  # popisků sloupců
    colClasses = "character"
                  # přetypuje všechny sloupce
                  # na textové proměnné
)
```

Import a export dat tabulky z MS Excel® (.xlsx)

- vhodný je například balíček openxlsx
  - nutné předem nainstalovat nástroj Rtools z adres (dle platformy)

    https://cran.r-project.org/bin/windows/Rtools/

    https://cran.r-project.org/bin/macosx/tools/
- uložení tabulky do excelového formátu (.xlsx) pomocí funkcí

```
createWorkbook()
addWorksheet(...)
writeData(...)
saveWorkbook(...)
```

- načtení excelové tabulky

```
my_data <- read.xlsx(
  xlsxFile = "moje_tabulka_je_ted_v_excelu.xlsx",
  sheet = 1,    # anebo jméno listu
  colNames = TRUE
)
```

Export konzolového výpisu do souboru

- pomocí `sink()` - `sink()` anebo `capture.output()`

```
1  (muzi <- rnorm(100, mean = 175, sd = 10))
2  (zeny <- rnorm(100, mean = 160, sd = 10))
3
4  t.test(muzi, zeny)
5
6  # výpis z konzole do textového souboru
7  capture.output(
8      t.test(muzi, zeny),
9      file = "t_test.txt"
10 )
11
12 # anebo
13 sink("tohle_je_taky_t_test.txt")
14 t.test(muzi, zeny)
15 sink()
```

Export smysluplného R-kového objektu do TEX-ového kódu

- pomocí balíčku `xtable`

```
1  library("xtable")
2
3  my_linear_model <- lm(mpg ~ hp + cyl,
4                        mtcars)
5
6  xtable(my_linear_model, digits = 4)
7
8  # anebo
9  print(xtable(my_linear_model,
10                digits = 4),
11        floating = FALSE,
12        tabular.environment = "tabular",
13        hline.after = NULL,
14        include.rownames = TRUE,
15        include.colnames = TRUE)
```

Import „exotických“ souborů do R

- naším přítelem je balíček `foreign`
- podporuje načítání dat z formátů
  - Epi Info
  - Minitab
  - S
  - SAS, SPSS, STAT, Systat, Weka

```
1  library("foreign")
2
3  # import dat z SPSS
4  my_data <- read.spss(
5      file = "soubor_z_SPSS.sav",
6      to.data.frame = TRUE
7  )
```

Podmínka *když*

- též zvaná *if-statement*
- rozhodovací ovládací prvek založený na pravdivosti, či nepravdivosti nějakého výroku
- obecná syntaxe

```
1        if(výrok){
2          procedura v případě, že výrok je TRUE
3        }else{
4          procedura v případě, že výrok je FALSE
5        }
```

Podmínka *když*

- například

```
1   x = 1
2
3   if(x == 1){
4     print("x je rovno 1")
5   }
6
7   # anebo
8   x = 2
9
10  if(x == 1){
11    print("x je rovno 1")
12  }else{
13    print("x není rovno 1")
14  }
```

for() cyklus

- ovládací struktura, smyčka pro opakování procedury stejného charakteru
- vhodná tehdy, **víme-li** dopředu počet opakování (iterací) dané procedury
- obecná syntaxe

```
1        for(indexový prostor){
2
3          procedura pro každý prvek indexového
4          prostoru
5
6        }
```

for() cyklus

- například

```
1   for(i in 1:5){
2
3     print(i)
4
5   }
6
7   # anebo
8   for(my_letter in letters){
9
10    print(
11      paste(my_letter, "je fajn", sep = " ")
12    )
13
14  }
```

while() cyklus

- ovládací struktura, smyčka pro opakování procedury stejného charakteru
- vhodná tehdy, **nevíme-li** dopředu počet opakování (iterací) dané procedury
- obecná syntaxe

```
1   index <- 1
2   while(výrok){
3
4     procedura pro každý index a případně
5     měnící výrok;
6     je prováděná, dokud je výrok TRUE
7
8     index <- index + 1
9
10  }
```

while() cyklus

- například

```
1  i <- 1
2  while(i <= 5){
3    print(i)
4    i <- i + 1
5  }
6
7  # anebo
8  my_letters <- letters
9  while(length(my_letters) > 0){
10
11   print(
12     paste(my_letters[1], "je fajn", sep = " ")
13   )
14   my_letters <- my_letters[-1]
15
16 }
```

repeat-until cyklus

- ovládací struktura, smyčka pro opakování procedury stejného charakteru
- vhodná tehdy, **nevíme-li** dopředu počet opakování (iterací) dané procedury
- prakticky ekvivalentní `while()` cyklu
- obecná syntaxe

```
1         index <- 1
2         while(TRUE){
3
4           if(zastavovací podmínka){break}
5
6           procedura pro každý index, která eventuá
                lně
7           založí zastavovaní podmínku
8
9           index <- index + 1
10
```

repeat-until cyklus

- například

```
1  i <- 1
2  while(TRUE){
3    if(i == 6){break}
4    print(i)
5    i <- i + 1
6  }
7
8  # anebo
9  my_letters <- letters
10 while(TRUE){
11   if(length(my_letters) == 0){break}
12   print(
13     paste(my_letters[1], "je fajn", sep = " ")
14   )
15   my_letters <- my_letters[-1]
16 }
```

Varování

- textová hláška vrácená funkcí nebo procedurou
- nejde o maligní chybu

```
log(-5)  # NaN; In log(-5) : NaNs produced
```

- lze zavést i vlastní, například

```
logaritmuj <- function(x){
  # vrací přirozený logaritmus čísla "x"
  if(x <= 0){
    cat(
      "x je nekladné, bude vráceno NaN\n"
    )
  }
  return(suppressWarnings(log(x)))
}

logaritmuj(-5)  # NaN; x je nekladné, bude vráceno NaN
```

Chyby

- ochranný mechanismus funkcí, který zabrání dalšímu provádění kódu

```
"1" + "1"  # Error: non-numeric argument to binary
           # operator
```

- lze zavést i vlastní, například

```
sectiCtverce <- function(a, b){
  # '''
  # vrací součet čtverců čísel "a" a "b"
  # '''
  if(!is.numeric(a)){stop("a musí být číslo!")}
  if(!is.numeric(b)){stop("b musí být číslo!")}
  return(a ^ 2 + b ^ 2)
}

sectiCtverce(1, 2)    # 5
sectiCtverce(1, "2")  # Error: b musí být číslo!
```

Rodina funkcí apply()

- jde o funkce dobře optimalizované tak, že v rámci svého vnitřního kódu „co nejdříve“ volají C++ ekvivalenty R-kové funkce
- díky tomu jsou exekučně rychlé
- nejužitečnější je `apply()` a `lapply()`

```
1  # vrací průměry nad všemi sloupci "mtcars"
2  x <- apply(mtcars, 2, mean)
3
4  # méně šikovně to samé
5  x <- NULL
6  for(i in 1:dim(mtcars)[2]){
7    x <- c(x, mean(mtcars[, i]))
8  }
9  names(x) <- colnames(mtcars)
```

Funkce apply()

- vrací vektor výsledků funkce `FUN` nad maticí či datovou tabulkou `X`, kterou čte po řádcích (`MARGIN = 1`), nebo sloupcích (`MARGIN = 2`)
- syntaxe je `apply(X, MARGIN, FUN, ...)`

```
1  apply(mtcars, 2, mean)
2
3  my_start <- Sys.time()
4  x <- apply(mtcars, 2, mean)
5  my_stop <- Sys.time(); my_stop - my_start # 0.019s
6
7  my_start <- Sys.time()
8  x <- NULL
9  for(i in 1:dim(mtcars)[2]){
10   x <- c(x, mean(mtcars[, i]))
11   names(x)[length(x)] <- colnames(mtcars)[i]
12 }
13 my_stop <- Sys.time(); my_stop - my_start # 0.039s
```

Funkce lapply()

- vrací list výsledků funkce FUN nad vektore či listem X
- syntaxe je lapply(X, FUN, ...)
- **skvěle se hodí pro přepis for() cyklu do vektorizované podoby!**
- vhodná i pro adresaci v listu

```
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
```

Náhrada for cyklu funkcí lapply()

- obě procedury jsou ekvivalentní stran výstupu, `lapply()` je významně rychlejší

```
1  # for cyklus
2  x <- NULL
3  for(i in 1:N){
4    x <- c(x, FUN)
5  }
6
7  # lapply
8  x <- unlist(
9    lapply(
10     1:N,
11     FUN
12   )
13 )
```

Náhrada for cyklu funkcí lapply()

```
1  # for cyklus
2  my_start <- Sys.time()
3
4  for_x <- NULL
5  for(i in 1:100000){for_x <- c(for_x, i ^ 5)}
6
7  my_stop <- Sys.time(); my_stop - my_start # 18.45s
8
9  # lapply
10 my_start <- Sys.time()
11
12 lapply_x <- unlist(lapply(
13   1:100000, function(i) i ^ 5
14 ))
15
16 my_stop <- Sys.time(); my_stop - my_start # 0.10s
```

Literatura

Alain F. Zuur, Elena N. Ieno und Erik Meesters. *A Beginner's Guide to R*. Springer New York, 2009. DOI: 10.1007/978-0-387-93837-0. URL: https://doi.org/10.1007/978-0-387-93837-0.

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

Děkuji za pozornost!

lubomir.stepanek@lf1.cuni.cz
lubomir.stepanek@fbmi.cvut.cz

▸ GitHub

https://github.com/LStepanek/B03128_Uvod_do_skriptovaciho_jazyka_R