Načítání a ukládání dat

Michal Kvasnička

Práce s daty začíná vždy jejich načtením a často končí jejich uložením. V této lekci se podíváme na to, jak data dostat do R a z R. Představíme také základní funkce pro práci se soubory a adresáři.

Základní práce s adresáři

Funkce `getwd()` vrací cestu k aktuálnímu pracovnímu adresáři, funkce `setwd()` nastaví cestu k pracovnímu adresáři (tj. změní pracovní adresář). V RStudiu stačí napsat část cesty – RStudio nabídne zbytek cesty po stisknutí tabelátoru. Fungují samozřejmě i relativní cesty.

Cesty k souborům je rozumné dávat relativní. Pokud je např. váš pracovní adresář `C:\diplomka\R` a data máte uložená v adresáři `C:\diplomka\data`, je lepší použít k datům relativní cestu `..\data\moje_data.csv`. Pak bude vše fungovat i při přenosu do jiného adresáře, na jiný disk nebo na jiný počítač. Ani to však není ideální, protože různé operační systémy používají různá lomítka a zpětná lomítka používaná ve Windows je při zadávání řetězců v R nutné zdvojit. Abyste se vyhnuli problémům, je rozumné používat funkci `file.path()`, která spojí jednotlivé části cesty k souboru tak, že fungují na každém daném operačním systému:

```
file.path("..", "diplomka", "moje_data.csv")
```

```
## [1] "../diplomka/moje_data.csv"
```

Zde je cesta oddělena normálními lomítky, protože tento text byl zkompilován na Linuxu.

Tabulární textové soubory

Základní způsob výměny dat jsou textové tabulární formáty, kde jednotlivé řádky odpovídají pozorováním a kde jsou sloupce (proměnné) odděleny nějakým oddělovačem, např. mezerou, čárkou, středníkem nebo dvojtečkou. Velká výhoda těchto formátů je, že je možné je načíst téměř do každého software, který pracuje s daty a stejně tak je z něj i vypsat. Zároveň jsou data "čitelná lidmi" a pokud se něco pokazí, je často možné data nějak zachránit. Nevýhodou je, že tyto formáty mohou obsahovat pouze tabulární data (tabulky), nemohou obsahovat metadata (např. atributy), jejich načítání může trvat dlouho (pokud jsou data velká) a že data v těchto formátech nejsou komprimovaná, takže na disku zabírají hodně místa (to je však relativní, viz dále).

Načítání dat

K načítání dat v textových tabulárních formátech slouží v R funkce `read.table()`, která vrací datovou struktur `data.frame`. Tato funkce má velké množství parametrů (přečtěte si dokumentaci!). Mezi základní parametry těchto funkcí patří:

- `file` = jméno souboru nebo spojení (connection), viz dále
- `header` = logická proměnná, zda soubor obsahuje na prvním řádku jména proměnných (sloupců)
- `skip` = počet řádků, které se mají na začátku přeskočit
- `sep` = řetězec obsahující znak, který odděluje jednotlivé sloupce
- `quote` = řetězec obsahující znak, který se používá jako uvozovky k označení textů
- `dec` = řetězec obsahující znak, který odděluje celá čísla a desetiny
- `comment.char` = řetězec obsahující znak, který označuje komentáře
- `na.string`= vektor obsahující řetězce, které se mají nahradit hodnotou `NA`
- `col.names` = vektor názvů sloupců
- `row.names` = vektor názvů řádků

- `nrows` = maximální počet řádků, které má R načíst
- `colClasses` = vektor, který určí, jaký datový typ nebo třídu mají jednotlivé sloupce
- `stringsAsFactors` = logická proměnná, která určí, zda se řetězce mají převést na faktory; implicitně `TRUE`, doporučuji vypnout!

Pro speciální případy (jako je např. formát CSV) nabízí R wrappery nad touto funkcí, zejména funkce `read.csv()` a `read.csv2()`; ty však jen nastavují hodnoty některých parametrů v základní funkci `read.table()`.

Předpokládejme, že chcete načíst data, která byla exportována z LibreOffice Calcu. Soubor se jmenuje "bmi_data.csv" a vypadá takto:

```
id,height,weight,bmi
1,153,55,23.4952368747
2,207,97,22.6376344839
3,173,83,27.7322997761
4,181,92,28.0821708739
5,164,112,41.6418798334
```

Protože se jedná o klasický CSV soubor s hlavičkou v prvním řádku, lze jej načíst takto:

```
bmi_data <- read.csv("data/bmi_data.csv", stringsAsFactors = FALSE)
bmi_data
```

```
##   id height weight      bmi
## 1  1    153     55 23.49524
## 2  2    207     97 22.63763
## 3  3    173     83 27.73230
## 4  4    181     92 28.08217
## 5  5    164    112 41.64188
```

RStudio má pro načítání dat v textových tabulárních formátech šikovné "udělátko": v záložce `Environment` je klikátko `Import Dataset`. Vyberte `From Local File...`. Nástroj odhadne většinu potřebných informací a zbytek je možné konfigurovat.

Načtená data si můžete snadno prohlédnout v záložce `Environment`. Definované proměnné jsou rozdělené do různých kategorií. Kategorie `Data` zahrnuje datasety. Kliknutím na jméno datasetu se otevře prohlížeč dat. V něm si můžete data prohlédnout, třídit a pomocí ikony "nálevky" i filtrovat. Kliknutím na ikonu "šipky" (vlevo vedle jména datasetu) se zobrazí struktura dat (podobně, jako to dělá funkce `str()`). Po otevření datasetu pomocí klikátka `Import Dataset` se prohlížeč data otevře automaticky.

Načítání velkých souborů

Načítání velkých souborů funguje stejně jako načítání malých souborů – se dvěma specifiky: trvá déle a zabírá více operační paměti počítače. R dokáže pracovat pouze s proměnnými, které má v operační paměti. Pokud se pokusíte načíst data, která se do paměti počítače nevejdou, R spadne.

Pomoci mohou tyto triky:

- zjistěte si, kolik řádků má váš data set (v Linuxu vám pomůže prográmek `wc`), kolik má sloupců a jaký mají datový typ (v Linuxu můžete vypsat první řádky datasetu prográmkem `head`), a spočítejte si, kolik paměti budete potřebovat
- pokud v souboru nejsou žádné komentáře, nastavte `comment.char = ""`
- řekněte funkci `read.table()` jakého typu jsou jednotlivé sloupce pomocí parametru `colClasses`, velmi výrazně to zrychlí načítání
- řekněte funkci `read.table()`, kolik řádků má váš datový soubor pomocí parametru `nrows` (můžete mírně nadhodnotit počet řádků, které máte), ušetří to spoustu operační paměti

Roger Peng radí následující trik, jak strojově odhadnout typ jednotlivých sloupců datasetu:

```
sample_dataset <- read.table("dataset.txt", nrows = 100)  # načte 100 řádků
classes <- sapply(sample_dataset, class)  # do vektoru uloží třídy sloupců datasetu
# načte celý dataset s danými datovými typy
full_dataset <- read.table("dataset.txt", colClasses = classes)
```

Další tipy a triky jsou v dokumentaci k funkci `read.table()`.

Naštěstí pro vás, valná většina dat, se kterou budete v blízké budoucnosti pracovat, je tak malá, že nic z těchto triků nebudete muset používat. Například tabulka s milionem řádků a 20 sloupci, které všechny obsahují reálná čísla bude v paměti počítač zabírat $1\,000\;\;000 \times 20 \times 8 = 160\,000\,000$ bytů, tj. asi jen 153 MB.

Ukládání dat

K zapsání dat do tabulárních formátů slouží funkce `write.table()` (a její wrappery `write.csv()` a `write.csv2()`. Detaily jsou v dokumentaci.

Spojení (connections)

Když R čte nebo zapisuje data, nezapisuje přímo do souborů, ale do "spojení" (connections). Spojení zobecňuje myšlenku souboru – spojení může být lokální soubor, soubor komprimovaný algoritmem `gzip` a `bzip` (nikoli obvyklý PK Zip), URL a další. Funkce `read.table()` a její wrappery vytváří spojení pro čtení dat automaticky.

Pokud je datový soubor komprimovaný algoritmem `gzip`, není třeba jej před načtením rozbalovat:

```
bmi_data <- read.csv("data/bmi_data.csv.gz", stringsAsFactors = FALSE)
bmi_data
```

```
##   id height weight      bmi
## 1  1    153     55 23.49524
## 2  2    207     97 22.63763
## 3  3    173     83 27.73230
## 4  4    181     92 28.08217
## 5  5    164    112 41.64188
```

Podobně je možné načíst i data z internetu, stačí nahradit jméno souboru jeho URL:

```
country_codes <- read.csv(
    "http://www.correlatesofwar.org/data-sets/cow-country-codes/cow-country-codes",
    stringsAsFactors = FALSE)
head(country_codes)
```

```
##   StateAbb CCode                StateNme
## 1      USA     2 United States of America
## 2      CAN    20                  Canada
## 3      BHM    31                 Bahamas
## 4      CUB    40                    Cuba
## 5      CUB    40                    Cuba
## 6      HAI    41                   Haiti
```

RStudio umožňuje stahovat textová tabulární data z webu pomocí jednoduchého formuláře. Je dostupný v záložce `Environment` v rámci klikátka `Import Dataset`.

Pokud chcete uložit data pomocí spojení (např. do komprimovaného souboru), je potřeba spojení deklarovat přímo:

```r
gfile <- gzfile("moje_data.csv.gz")
write.csv(bmi_data, file = gfile)
```

Víc detailů ke spojením najdete např. v Spector: *Data Manipulation with R*, s. 23–25 nebo Peng: *R Programming for Data Science*, s. 33–35.

Balík readr

Balík **readr** poskytuje funkce k načítání textových tabulárních datových souborů, které jsou rychlejší než standardní R-kové funkce popsané výše a mají i některé další výhody. Prakticky stačí balík načíst a nahradit standardní funkce funkcemi `read_table()`, `read_csv()`, `read_csv2()` apod. Je však lepší specifikovat typ jednotlivých sloupců, viz vinětа.

Nativní R-kové binární soubory

Textové tabulární datové formáty jsou výborné při předávání dat mezi lidmi. Při vlastním zpracování dat je však výhodnější použít vlastní binární datový formát R. Jeho výhodou je, že data zabírají na disku méně místa, načítají se rychleji, mohou obsahovat metadata včetně atributů a jde v nich ukládat i jiná data, než jsou tabulky (např. seznamy, pole, různé objekty apod.).

K uložení konkrétních proměnných slouží funkce `save()`. Nejdříve se uvedou všechny proměnné, které se mají uložit, a pak cesta k souboru, kam se mají uložit (další parametry jsou popsány v dokumentaci). Data se do těchto souborů obvykle ukládají s koncovkami `.RData` nebo `.rda`.

```r
save(bmi_data, file = "bmi_data.RData")
```

Pokud je proměnných více, oddělí se čárkou:

```r
save(var1, var2, var3, var4, file = "somefile.RData")
```

Někdy chcete uložit všechny proměnné, které máte v paměti R. K tomu slouží funkce `save.image()` (detaily viz dokumentace).

Data uložená pomocí funkcí `save()` a `save.image()` načtete do R pomocí funkce `load()`. Funkce načte všechny proměnné obsažené v daném datovém soubory včetně metadat; proměnným zůstanou jejich původní jména. Jediným důležitým parametrem funkce je cesta k datovému souboru:

```r
load("bmi_data.RData")
```

Načítání dat z balíků

Mnoho R-kových balíků obsahuje data. K načtení těchto dat slouží funkce `data()`. Tato funkce data nevrací, nýbrž je načte na pozadí.

```r
library(ggplot2)
data("diamonds")
head(diamonds)
```

```
##   carat       cut color clarity depth table price    x    y    z
## 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

Funkce data může vypsat i seznam dat obsažených v daném balíku:

```
data(package = "ggplot2")
```

Umožňuje i načíst data bez načtení balíku. K tomu stačí určit jméno balíku:

```
data("economics", package = "ggplot2")
head(economics)
```

```
##         date   pce    pop psavert uempmed unemploy
## 1 1967-07-01 507.4 198712    12.5     4.5     2944
## 2 1967-08-01 510.5 198911    12.5     4.7     2945
## 3 1967-09-01 516.3 199113    11.7     4.6     2958
## 4 1967-10-01 512.9 199311    12.5     4.9     3143
## 5 1967-11-01 518.1 199498    12.5     4.7     3066
## 6 1967-12-01 525.8 199657    12.1     4.8     3018
```

Poznámka: Některé balíky načtou data hned při svém načtení, takže není třeba je zpřístupňovat pomocí funkce `data()`.

Dokumentace k datům funguje stejně jako dokumentace k funkcím. Po načtení dat `economics` můžete dokumentaci zobrazit např. takto: `?economics`.

Data z cizích formátů

Načítání dat z MS Excelu

Nejbezpečnější je načítat data z Excelu tak, že je vyexportujete do CSV souboru a načtete způsobem popsaným výše. Tak máte největší kontrolu nad tím, co se děje.

Data z Excelu je však v R možné načíst i přímo pomocí balíku **readxl** (umí načíst jak soubory `.xls`, tak `.xlsx`). Balík poskytuje dvě funkce: funkce `excel_sheets()` vypíše seznam listů, které Excelový soubor obsahuje; funkce `read_excel()` načte jeden list z Excelového souboru. Víc detailů viz dokumentace.

Data z jichých statistických programů

Někdy jsou data distribuovaná v nativním formátu některého statistického softwaru. K jejich načtení je možné použít zejména balíky **haven** a **foregin**.

Balík **haven** slouží k načítání souborů v nativních formátech programů SPSS, SAS a Stata (v některých případech i k jejich ukládání). K načítání dat poskytuje tyto funkce: `read_dta()` a `read_stata()` čtou soubory Stata DTA, `read_por()`, `read_sav()` a `read_spss()` čtou soubory SPSS POR a SAV a funkce `read_sas()` čte soubory SAS. Detaily viz dokumentace k balíku.

Balík **foreign** umí načítat mnoho dalších datových formátů (a do některých i zapisovat). Detaily viz dokumentace k balíku.

Pokud nějaký formát umí číst oba balíky, doporučuji jako první vyzkoušet balík **haven**.

Připojení databází

Viz Spector: *Data Manipulation with R*, kap. 3.

Data na webu

Ekonomické databáze

R má i balíky, které umí přímo načítat data z některých ekonomických databází. Za zmínku stojí zejména:

- **WDI** umí přímo načíst makroekonomická data z World Development Indicators, které poskytuje Světová banka
- **Quandl** načítá různá ekonomická a finanční data, viz https://www.quandl.com/tools/r
- **quantmod** umí načíst některá finanční data, např. z Fedu, viz http://www.quantmod.com

Balík **Ecdat** poskytuje spoustu datasetů pro ekonometrii.

Google API apod.

Některé balíky v R umožňují číst data z různých služeb Google a podobných internetových firem. Např.

- **ggmap** - využívá mapy z Google k vizualizaci dat, zjištění adres, GPS lokací, dojezdových vzdáleností apod.
- **gtrendsR** – spojení s Google Trends
- **twitteR** – spojení s API Twitteru
- **Rfacebook** – spojení s API Facebooku

Kontrola načtených dat

Když načtete data, je obvykle moudré zkontrolovat, že se načetla všechna data, že se načetla správně a že znamenají to, co si myslíte. Vyplatí se projít minimálně těchto několik kroků:

1. zkontrolujte, že má dataset správný počet řádků (`nrow()`) a sloupců (`ncol()`), tj. že se načetlo vše a nenačetl se nějaký "odpad" uložený na začátku nebo konci datového souboru
2. podívejte se na začátek (`head()`) a konec (`tail()`) datasetu; opět pomůže kontrolovat, zda se nenačetl nějaký zmatek na začátku nebo konci souboru a že vše vypadá tak, jak má
3. zkontrolujte strukturu datasetu (`str()`) – jména sloupců, jejich typy a hodnoty
4. podívejte se na souhrnné statistiky dat (`summary()`): jaké jsou hodnoty proměnných (dávají smysl? jsou správně velké? porovnejte hodnoty s tím, co víte odjinud), kde hodnoty chybí apod.

Další rady se dozvíte v lekci o exploratory data analysis.

Práce se soubory a adresáři v R

Význam a použití níže uvedených funkcí si zjistěte v jejich dokumentaci.

Zjištění a změna pracovního adresáře

```
getwd()
setwd(dir)
```

Seznam existujících souborů apod.

```r
list.files()
dir()
list.dirs()
```

Vytváření, mazání souborů apod.

```r
file.create(..., showWarnings = TRUE)
file.exists(...)
file.remove(...)
file.rename(from, to)
file.append(file1, file2)
file.copy(from, to, overwrite = recursive, recursive = FALSE,
          copy.mode = TRUE, copy.date = FALSE)
file.symlink(from, to)
file.link(from, to)
```

Vytváření, mazání adresářů apod.

```r
dir.exists(paths)
dir.create(path, showWarnings = TRUE, recursive = FALSE, mode = "0777")
Sys.chmod(paths, mode = "0777", use_umask = TRUE)
Sys.umask(mode = NA)
```

Domácí úkol

Upravte soubor `hw03.R`, tak aby:

1. načetl soubor cars.csv z aktuálního adresáře; soubor má v prvním řádku jména sloupců a jednotlivé sloupce jsou odděleny dvojtečkou; nijak dataset neupravujte a výsledek uložte do data.frame `cars`
2. pomocí funkce `cor()` spočítejte Pearsonovu korelaci mezi počtem mil, které auto ujede na 1 galon paliva (`mpg`) a výkonem motoru v koňských silách (`horse`); výsledek uložte do proměnné `mpg_hpw_cor`; hodnot `NA` se zbavte volbou `"complete.obs"`, viz dokumentace funkce
3. načtěte soubor `Cars.sav` ve formátu SPSS, nijak jej neměňte a uložte jej do proměnné `cars2`
4. zkontrolujte, zda jsou všechny sloupce obou datasetů pojmenovány stejně; rozdíly mezi malými a velkými písmeny ignorujte (řetězce na malá písmena převádí funkce `tolower()`); výsledkem bude jediná logická hodnota: `TRUE`, pokud jsou všechny sloupce stejně pojmenované, `FALSE` jinak; výsledek uložte do proměnné `same_header`
5. všechny čtyři vytvořené proměnné uložte do souboru `results.RData`

Poznámky:

- Nezapomeňte ve skriptu načíst všechny potřebné knihovny.
- Pamatujte, že při hodnocení bude váš skript spuštěn s jinými daty – stejný bude jen jejich formát a jména.
- Nezapomeňte výsledky uložit do souboru. Proměnné, které neuložíte, vám nezískají žádné body. Doporučuji vytvořené proměnné ukládat v každém kroku.