Programovanie v prostredí R

VIII. Práca so súbormi a programovanie

Aleš Kozubík

Katedra Matematických Metód a Operačnej Analýzy

20.01.2020

Čo si ukážeme

- Načítanie dát zo súboru
- Ukladanie výsledkov do súboru
- Riadenie behu programu
- Definícia vlastných funkcií

Vstupný formát dát

Najjednoduchší spôsob importovania dát do systému R je v podobe jednoduchého textu (tzv. *plain text*). Takéto súbory je možné editovať napr. v Notepade alebo gedite (Linux).

Existuje viacero textových formátov pre uchovávanie tabuľkových dát:

- `.csv` „comma-separated values“ dáta oddelené čiarkou,
- údaje oddelené tabulátorom, textové súbory `.txt`,
- `.dif` tzv. „data interchange format“.

Sú to najrozšírenejšie formáty pre prenos dát pre štatistické a databázové nástroje.

Importovanie súborov `.csv`

Súbory `.csv` sú najpopulárnejšie pre prenos dát.

Jednotlivé záznamy pozorovaní sú roztriedené do riadkov, pre oddelenie jednotlivých údajov na riadku sú použité čiarky.

Niekedy sa ako oddeľovače používajú bodkočiarky (napr. ak sú čiarky obsahom samotných dát).

Prvý riadok môže, ale nemusí, obsahovať pomenovania jednotlivých stĺpcov (položiek dát)

Pre načítanie súboru máme k dispozícii funkciu `read.csv()`. načítané dáta majú formát `data.frame`.

Funkcia `read.csv()`

Základná syntax pre načítanie súboru v `.csv` formáte je

```
> data<-read.csv("subor")
```

Meno súboru treba zadať s úplnou cestou (pokiaľ nie je v pracovnom adresári).

Funkcia môže mať aj ďalšie argumenty:

- `header=` logická premenná, `F` alebo `T` pre určenie, či súbor obsahuje riadok s menami stĺpcov,
- `sep=""`, do úvodzoviek sa zapíše oddeľovací znak, ak je iný než čiarka,
- `col.names=`, ak neexistuje riadok s pomenovaniami, umožňuje zadať vektor s menami stĺpcov.

Funkcia `read.csv()`

Poznámky:

- `na.string=""` niektoré systémy používajú zástupné symboly pre neexistujúce (nezistené) hodnoty napr. `NULL`, hodnota 9999 alebo bodka `..`Argument umožňuje nastaviť tento údaj.
- Ak sa namiesto desatinnej bodky používa desatinná čiarka, je treba použiť funkciu `read.csv2()`.

Importovanie súborov `.dif`

Pre importovanie súborov vo formáte `.dif` použijeme funkciu

```
> data<-read.DIF(subor)
```

ktorá je podobná funkcii `read.csv()`.

Rozdiel je v tom, že implicitne nepredpokladá existenciu riadku s pomenovaniami stĺpcov. V takom prípade je potrebné deklarovať `header=T`.

Pre exportovaní z Excelu niekedy dochádza ku chybe `More rows than specified in header`, nakoľko Excel používa neštandardný, modifikovaný formát `dif`. vtedy treba použiť argument `transpose=T`.

Importovanie iných textových súborov

Je možné načítať ľubovoľný formát holého textu. K dispozícii je funkcia `read.table()`, ktorá umožňuje načítať akékoľvek dáta, ktoré sú zoradené po jednom pozorovaní na riadku.

Dáta sú načítané do formátu tabuľky `table`.

Na rozdiel od `read.csv()` sa nepredpokladá hlavičkový riadok s menami stĺpcov. Ak existuje, treba to deklarovať argumentom `header=T`.

Import dát z Excelu

Pre importovanie dát z Excelu je navýhodnejšie použiť formát `.csv`.

Pred exportovaním z Excelu je potrebné sa presvedčiť o tom, že:

- nad tabuľkou dát ani naľavo od nej nie sú žiadne prázdne riadky ani stĺpce,
- v tabuľke nie sú žiadne zlúčené bunky,
- hlavička (pomenovania stĺpcov) nemá viac ako jeden riadok,
- v bunkáchj nie je žiadne formátovanie ako bold, italic, orámovanie buniek, farbenie a pod.,

Import dát z Excelu

Pre importovanie dát z Excelu je navýhodnejšie použiť formát `.csv`.

Pred exportovaním z Excelu je potrebné sa presvedčiť o tom, že:

- bunky, kde chýbajú dáta sú prázdne (nesmie tam byť napr. medzera),
- vo veľkých číslach nesmú byť čiarky ako oddeľovače tisícov,
- ak sa používa exponenciálny zápis čísl, tak či je správny formát,
- číselné premenné neobsahujú symboly mien, jednotky alebo percentá,

Import dát z Excelu

Pre importovanie dát z Excelu je navýhodnejšie použiť formát `.csv`.

Pred exportovaním z Excelu je potrebné sa presvedčiť o tom, že:

- záporné hodnoty sú značené znamienkom – (nie napr. červene alebo zátvorkami),
- zošit obsahuje len jeden list resp. exportuje sa vždy len jeden list,
- tisíce nie sú oddelené medzerami (hodnoty by sa načítali ako reťazce).

Import dát z Excelu

Pokiaľ nemáme prístup k excelu alebo inému tabuľkovému procesoru, ktorý podporuje formát `.xls`, je možné použiť doinštalované balíčky.

Takýmito balíčkami sú `xlsx` alebo `xlsReadWrite`, ktoré umožňujú priamo importovať excelovské súbory.

Balíček `xlsx` prináša funkcie `read.xlsx()`,`read.xlsx2()` a `write.xslx()` resp. `write.xlsx2()`.

„Dvojkové“ verzie sú rýchlejšie pri práci s väčšími dátovými súbormi.

Iné zdroje dát

Pre importovanie iných formátov je potrebné nainštalovať balíček `foreign`.

Ten prináša napr. tieto funkcie:

Typ dát	**Prípona**	**Funkcia**
Databázové súbory	`.dbf`	`read.dbf()`
Stata verzie 5 až 12	`.dta`	`read.dta()`
Pracovné listy Minitab	`.mtp`	`read.mtp()`
SAS prenosový formát	`.xport`	`read.xport()`
Octave dátový súbor	`.txt`	`read.octave()`

Exportovanie dát

R umožňuje zápis dát vo formátoch `.csv` hodnôt oddelených čiarkami alebo vo formýte dát oddelených tabulátormi.

Pre ukladanie vo formáte `.csv` slúžia funkcie `write.csv(data,file)` resp. `write.csv2(data,file)` v prípade, že chceme použiť desatinnú čiarku namiesto bodky.

Hodnoty oddelené tabulátormi zapisujeme pomocou funkcie `write.table()`. Tá môže mať viacero argumentov.

Funkcia `write.table()`

Funkcia `write.table()` môže mať tieto argumenty:

- `x` premenná obsahujúca dáta,
- `file` názov výstupného súboru,
- `sep` oddeľovač jednotlivých položiek dát, implicitne tabulátor, t.j. `sep="\t"`,
- `dec` reťazec, ktorý určuje oddelenie desatinnej časti, implicitne `dec="."`,
- `row.names`,`col.names` logické premenné, určujú či dáta obsahujú mená riadkov alebo stĺpcov.

Výstup programu R do súboru

Ak potrebujeme výsledky výpočtov uchovať v súbore, použijeme funkciu `sink()`.

Základná syntax je `sink(file)`.

Všetky výstupy programu R sa potom ukladajú do súboru `file`

Ak súbor s daným názvom existuje, tak je prepísaný. Ak súbor nechceme zmazať, použijeme argument `add=T`, výstupy sú potom pripojené na konie existujúceho súboru.

Spojenie ukončíme funkciou `sink()` bez argumentov.

Aritmetické operácie v R

Pre bežné výpočty máme k dispozícii tieto operátory:

Operátor	**Význam**
+	súčet,
-	rozdiel,
*	súčin,
/	podiel,
^ alebo **	mocnina,
%%	zvyšok po delení (modulo),
%/%	celočíselný podiel.

Matemetické funkcie v R

Pre bežné výpočty máme k dispozícii tieto funkcie:

Funkcia	**Význam**
`abs(x)`	$\lvert x \rvert$,
`sqrt(x)`	$\sqrt{x}$,
`ceiling(x)`	najmenšie celé číslo, väčšie ako x,
`floor(x)`	najväčšie celé číslo, ktoré nie je väčšie ako x,
`trunc(x)`	odstránenie desatinnej časti,
`round(x,digits=n)`	zaokrúhlenie na n desatinných miest,
`signif(x,digits=n)`	úprava na n platných číslic.

Matemetické funkcie v R

Pokračovanie:

Funkcia	**Význam**
`sin(x),cos(x),tan(x)`	goniometrické funkcie,
`log(x)`	prirodzený logaritmus,
`log10(x)`	dekadický logaritmus,
`exp(x)`	e^x,
`gamma(x)`	gama funkcia,
`beta(x,y)`	hodnoty funkcie beta,
`asin(x),acos(x),atan(x)`	cyklometrické funkcie.

Podmienky v R

Podmienka je vlastne výraz, ktorý po vyhodnotení dáva logickú hodnotu `TRUE` aebo `FALSE`.

Pre vytváranie podmienok máme tieto operátory:

Operátor	**Význam**
==	rovná sa? (test na rovnosť),
<	je menšie?
>	je väčšie?
<=	je menšie alebo rovné?
>=	je väčšie alebo rovné?
!=	nerovná sa? (test na nerovnosť),
%in%	testuje, či je hodnota rovná niektorému prvku vektora.

Zložené podmienky

Operátory pre podmienky je možné skladať pomocou logických spojok:

Operátor	**Význam**
&	a,
|	alebo
!x	negácia x

Podmienky – príklady

```
vektor<-c(4,5,2,3)
> 2%in%vektor
[1] TRUE
x <- c(1:10)
> x>8
 [1] F F F F F F F F T T
> x<4|x>7
 [1] T T T F F F F T T T
> x[(x>8)|(x<5)]
[1]  1  2  3  4  9 10
>!(x<5)
 [1] F F F F T T T T T T
```

Podmienený príkaz `if`

Základný tvar príkazu je:

```
if (podmienka) príkaz
```

alebo ak je potrebné vykonať viac príkazov

```
if (podmienka) {
príkazy vykonané ak je podmienka splnená
}
```

Podmienený príkaz `if`

Ukážka

```
> x<-10
> if (x>0) sqrt(x)
[1] 3.162278
> x<--100
> if (x>0) sqrt(x)
>
```

Podmienený príkaz `if else`

Základný tvar príkazu je:

```
if (podmienka) príkaz1 else príkaz 2
```

alebo ak je potrebné vykonať viac príkazov

```
if (podmienka) {
príkazy vykonané ak je podmienka splnená
}
else{
príkazy vykonané ak podmienka nie je splnená
}
```

Podmienený príkaz `if`

Ukážka

```
> x<-10
> if (x>0) sqrt(x)
[1] 3.162278
> x<--100
> if (x>0) sqrt(x) else print("x je zaporne")
[1] "x je zaporne"
>
```

Cyklus `for`

Cyklus `for` je vhodný najmä pre pevný počet opakovaní príkazov.

Základná syntax:

```
for (i in start:koniec){
príkazy, ktoré sa majú vykonať
}
```

Namiesto rozsahu `start:koniec` môže byť zadaný vektor, cez ktorého zložky má riadiaca premenná cyklu prebiehať, napr. `for i in c(1,3,5,8) prikaz`.

Cyklus `for` – ukážka

```
> for(i in 1:10){
+ vys<-i**2
+ text<-paste("i=",i,"druha mocnina je",vys)
+ print(text)
+ }
[1] "i= 1 druha mocnina je 1"
[1] "i= 2 druha mocnina je 4"
.
.
.
[1] "i= 9 druha mocnina je 81"
[1] "i= 10 druha mocnina je 100"
>
```

Cyklus `while`

Vhodný, ak opakované vykonanie príkazov závisí od splnenia nejakej podmienky.

```
while (podmienka) príkaz
```

resp.

```
while (podmienka){
 príkazy na opakovanie
}
```

Cyklus `while` – ukážka

Simulácia hodu kockou, kým nepadne šesťka.

```
> hod<-0
> while(hod!=6){
+ hod<-sample(1:6,1)
+ print(paste("Padlo číslo:",hod))
+ }
[1] "Padlo číslo: 2"
[1] "Padlo číslo: 1"
[1] "Padlo číslo: 1"
[1] "Padlo číslo: 2"
[1] "Padlo číslo: 5"
[1] "Padlo číslo: 4"
[1] "Padlo číslo: 6"
```

Definovanie vlastnej funkcie

Vlastnú funkciu si môže uživateľ definovať takto:

```
meno<-function(argument1,argument2,...,argumentN){
príkazy funkcie
return(výstup)
}
```

Je možné zadať implicitné hodnoty premenných

```
meno<-function(argument1=hodnota1,...,argumentN=hodnotaN){
príkazy funkcie
return(výstup)
}
```

Volanie vlastnej funkcie

Funkcia, ktorú sme definovali je trvale (počas jednej relácie, jedného sedenia) dostupná pre uživateľa.

Voláme ju jej menom so zadanými hodnotami argumentov.

```
meno(hodnota1,hodnota2,...,hodnotaN)
```

Ako výsledok dostávame výstup definovaný v príkaze `return`.

Definícia vlastnej funkcie – ukážka

Definujeme funkciu `cube.root()` pre výpočet tretej odmocniny.

```
> cube.root<-function(x){
+ vys<-x^(1/3)
+ return(vys)
+ }
> cube.root(10)
[1] 2.154435
> cube.root(-10)
[1] NaN
```

Definícia vlastnej funkcie – ukážka

Definujeme funkciu `n.root()` pre výpočet *n*-tej odmocniny.

```
> n.root<-function(x,n){
+ vys<-x^(1/n)
+ return(vys)
+ }
> n.root(10,2)
[1] 3.162278
> n.root(2,10)
[1] 1.071773
> n.root(n=2,x=10)
[1] 3.162278
```

Definícia vlastnej funkcie – ukážka

Definujeme funkciu `n.root()` pre výpočet n-tej odmocniny, ktorá defaultne vypočíta druhú odmocninu.

```
> n.root<-function(x,n=2){
+ vys<-x^(1/n)
+ return(vys)
+ }
> n.root(100)
[1] 10
> n.root(100,3)
[1] 4.641589
>
```

Simulátor hodu kockou kým nepadne *n*

```
> kocka<-function(n){
+ if(n%in%c(1:6)){
+ i<-0
+ hod<-0
+ while(hod!=n){
+ i<-i+1
+ hod<-sample(1:6,1)
+ }
+ text<-paste("Kým padlo číslo",n,"bolo vykonaných celkovo",i,"hodov")
+ }
+ else
+ {text<-paste("Číslo mimo rozsah kocky")}
+ return(text)
+ }
> kocka(3)
[1] "Kým padlo číslo 3 bolo vykonaných celkovo 3 hodov"
> kocka(8)
[1] "Číslo mimo rozsah kocky"
```

Tvorba a zápis skriptu

Skript predstavuje postupnosť príkazov jazyka R, ktoré sa majú postupne vykonať.

Je ich možné editovať v ľubovoľnom textovom editore, ktorý umožňuje ukladať holý text do súboru s príponou R

Skript je potom možné spúšťať viacerými spôsobmi:

- cestou `CMD BATCH` čo je možné v linuxe aj win,
- cestou `rscript.exe` čo je možný postup vo win,
- priamo z GUI, (čo čiastočne popiera účel vytvárania skriptu).

Zápis priamo v GUI

Z ponuky menu zvolíme `File->New Script` a otvoríme editačné okno.

Editujeme skript, napríklad:

```
people<-read.csv("people2.csv")
lm(Height~Hand.Span+Sex,data=people)
```

Skript uložíme s príponou `r`, tj. `meno_suboru.r`.

Spustenie v GUI

V okne, v ktorom editujeme skript vysvietime tú časť skriptu, ktorú chceme spustiť, ak celý tak `Ctrl+A`.

Skript spustíme pomocou kombinácie `Ctrl+R`

Odozva sa zobrazí v interaktívnom okne GUI.

Spustenie z príkazového riadku

Najskôr otvoríme okno s príkazovým riadkom, (v ponuke hľadáme CMD).

Skript spustíme príkazom `rscript meno_suboru.r`. Výstup sa zobrazí na konzole.

Alternatívne môžeme spustiť príkazom
`R CMD BATCH meno_suboru.r`.

POZOR: Cesta ku adresáru, v ktorom je nainštalované R musí byť súčasťou systémovej premennej `path`.

Skript so zápisom do súboru

```
people<-read.csv("people2.csv")
sink("vystup.txt")
lm(Height~Hand.Span+Sex,data=people)
sink()
```

Alebo v pôvodnom tvare (bez `sink`) presmerujeme výstup s konzoly do súboru. Takže skript spustíme takto:
`rscript meno_suboru.r>vystup.txt`