Programovanie v prostredí

I. Úvod do R-ka

Aleš Kozubík

Katedra Matematických Metód a Operačnej Analýzy

18.10.2019

Charakteristika jazyka R

- je jazyk a slobodné softvérové prostredie pre štatistické výpočty a grafiku,
- je dostupné pre viaceré UNIX-ové platformy, Win aj MacOS,
- je alternatívou ku komerčnému nástroju S (jazyk a prostredie), ktorý vyvíja AT&T.

Kde získame R

CRAN - Comprehensive R Archive Network

zoznam zrkadiel na stránke
https://cran.r-project.org/mirrors.html

Pre nás asi najlepšie

Zrkadlá pre download R

Česká republika
https://mirrors.nic.cz/R/
Rakúsko
https://cran.wu.ac.at/

Dokumentácia R

Manuály

https://cran.r-project.org/manuals.html

Ďalšie on-line zdroje

R Cookbook http://www.cookbook-r.com/
R Tutorial http://www.cyclismo.org/tutorial/R/
Quick R https://www.statmethods.net/index.html

R prvé spustenie

Vo Win alebo MacOS dvojitým kliknutím na ikonu v podobe loga jazyka 

My sa budeme zaberať prevádzkovaním pod linuxom, tu spúšťame z príkazového riadku jednoducho príkazom

```
username@host:~$ R
```

Prostredie R sa ozýva symbolom

```
>
```

na začiatku riadku.

Opustenie prostredia R

Činnosť v prostredí R ukončujeme funkciou

```
> q()
```

Pred ukončením sa systém opýta:

```
Save workspace image? [y/n/c]:
```

Treba vložiť správnu odpoveď yes, no alebo cancel.

Uloženie pracovnej plochy

Ak pri opustení prostredia zvolíme možnosť `yes`, celá história našej činnosti sa uloží do súboru `.Rhistory`, ktorý je umiestnený v aktuálnom adresári, z ktorého sme R štartovali.

Pri opätovnom spustení systému R sa môžeme pomocou navigačných kláves vrátiť ku jednotlivým príkazom.

Ak pri opustení prostredia zvolíme možnosť `no`, príkazy stratíme, ale predchádzajúca história ostane zachovaná.

Použitie R ako kalkulátora

Najjednoduchšie je použitie systému ako kalkulátora.

Zadáme numerický výraz a okamžite dostaneme výsledok.

Napríklad

```
> 3*7
[1] 21
>sin(90)
[1] 0.8939967
>mean(c(1,2,3))
[1] 2
>
```

Získanie nápovedy

Na získanie priamej nápovedy slúži funkcia `help()`.

Ako argument sa uvedie názov príkazu resp. funkcie, ku ktorej chceme nápovedu.

Príklad

```
> help(sin)
```

Z režimu nápovedy späť do prostredia R sa dostaneme stlačením q.

Základné dátové typy

Existuje 5 základných dátových typov

- numeric,
- integer,
- complex,
- logical,
- character.

Tie môžeme zlučovať do dátových štruktúr:

- vector,
- matrix,
- list,
- frame.

Dátový typ numeric

Slúži na prácu s desatinnými číslami, je to default dátový typ.
Ak do nejakej premennej priradíme hodnotu desatinného čísla, má automaticky typ `numeric`.
Príklad

```
> x<-12.35
> x
[1] 12.35
>
```

Poznámka

Číslo je reprezentované ako vektor dĺžky 1. Číslo `[1]` znamená, že ide o prvú pozíciu vektora.

Dátový typ numeric

O dátovom type konkrétnej premennej sa presvedčíme pomocou funkcie `class()`

V našom prípade

```
> class(x)
[1] "numeric"
>
```

Ak priradíme do premennej celé číslo (integer), bude typu numeric:

```
> z<-12
> class(z)
[1] "numeric"
> is.integer(z)
[1] FALSE
>
```

Dátový typ integer

Na vytvorenie celočíselnej premennej je v prostredí R potrebné využiť funkciu `as.integer()`

```
> a <- as.integer(12)
> a
[1] 12
> class(a)
[1] "integer"
> is.integer(a)
[1] TRUE
>
```

Dátový typ integer

Pomocou funkcie `as.integer()` získame celočíselnú hodnotu aj z desatinného čísla alebo reťazca obsahujúceho len číslice.
Napríklad

```
> as.integer(2.718)
[1] 2
> as.integer("3.1415926")
[1] 3
>
```

Ale nečíselný reťazec:

```
> as.integer("frcka")
[1] NA
Warning message:
NAs introduced by coercion
>
```

Dátový typ integer

Logické hodnoty je možné funkciou `as.integer()` previesť na hodnoty 0 a 1.

```
> as.integer(TRUE)
[1] 1
> as.integer(FALSE)
[1] 0
>
```

Dátový typ integer

Vložením novej hodnoty je možné zmeniť typ premennej.
K pretypovaniu dochádza bez varovania!

Ukážka:

```
> a<-as.integer(13)
> class(a)
[1] "integer"
> a<-a/3
> class(a)
[1] "numeric"
> a
[1] 4.333333
>
```

Dátový typ complex

Komplexné hodnoty definujeme použitím imaginárnej jednotky `i`.

```
> z<-1+2i
> class(z)
[1] "complex"
>
```

Pozor, hodnota -1 nie je typu complex, preto:

```
> sqrt(-1)
[1] NaN
Warning message:
In sqrt(-1) : NaNs produced
> sqrt(-1+0i)
[1] 0+1i
>
```

Aké by bolo alternatívne riešenie?

Dátový typ logical

Nadobúda 2 hodnoty: `TRUE` alebo `FALSE`.

Často sa definuje napr. ako výsledok porovnania dvoch premenných.

Príklad

```
> x<-10; y<-20
> z<- x>y
> class(z)
[1] "logical"
> z
[1] FALSE
>
```

Dátový typ logical

Štandardné operácie s premennými typu logical sú & (and), | (or) a ! (negácia).

Príklad:

```
> u<-TRUE; v<-FALSE
> u|v
[1] TRUE
> u&!v
[1] TRUE
> u&v
[1] FALSE
>
```

Dátový typ character

Character je objekt používaný na reprezentáciu textových reťazcov.

```
> x<-"facina"
> class(x)
[1] "character"
>
```

Číslo prevedieme na typ character funkciou `as.character()`

```
> as.character(3.1415926)
[1] "3.1415926"
> as.character(u)
[1] "TRUE"
>
```

Dátový typ character

Na spájanie reťazcov používame funkciu `paste()`

Príklad

```
> meno<-"Mike"
> priezvisko<-"Lietavec"
> paste(meno,priezvisko)
[1] "Mike Lietavec"
>
```

Otázka:

Ako zariadime zreťazenie bez medzery?

Dátový typ character

Odpoveď

Pomocou `\help(paste)`si pozrieme nápovedu ku funkcii `paste()`.

Tu sa dozvieme, že jedným z parametrov funkcie `paste` je `sep`, ktorý môžeme nastaviť na požadovanú hodnotu. Implicitná hodnota je `sep=" "`.

Ak chceme zrušiť medzeru, nastavíme `sep=""`.

```
> meno<-"Mike"
> priezvisko<-"Lietavec"
> paste(meno,priezvisko,sep="")
[1] "MikeLietavec"
>
```

Dátový typ character

Podreťazec vyberáme pomocou funkcie `substr()`, ktorej parametrami sú

- pôvodný reťazec,
- hodnota `start` pre určenie začiatku vyberaného podreťazca
- `stop` pre určenie poslednej pozície vyberaného podreťazca.

Príklad

```
> a<-"Dnes␣sme␣si␣zafrcali␣v␣Rku"
> substr(a,start=6,stop=20)
[1] "sme␣si␣zafrcali"
>
```

Dátový typ character

Časť reťazca môžeme nahradiť iným textom pomocou funkcie `sub()` s tromi parametrami:

- `pôvodný text`,
- `nový text`,
- `reťazec`.

Príklad

```
> sub("si","si dobre",a)
[1] "Dnes sme si dobre zafrcali v Rku"
>
```

Pozor na jednoznačnosť, nahrádza sa iba prvý výskyt podreťazca.

Dátová štruktúra vektor

Štruktúra vektor predstavuje postupnosť prvkov rovnakého dátového typu.

Číselný vektor

```
> c(1,2,3)
[1] 1 2 3
>
```

Vektor logických hodnôt

```
> c(TRUE,TRUE,FALSE,TRUE,FALSE)
[1]  TRUE  TRUE FALSE  TRUE FALSE
>
```

Dátová štruktúra vektor

Vektor môže obsahovať aj reťazce

```
> a<-c("aa","bb","cc","dd","ee","ff")
> a
[1] "aa" "bb" "cc" "dd" "ee" "ff"
>
```

Počet prvkov vektora zistíme pomocou funkcie `length()`

```
> length(a)
[1] 6
>
```

Zlučovanie (combining) vektorov

Vektory spájame pomocou funkcie `c()` (c ako combine)

```
> a<-c("aa","bb","cc","dd","ee","ff")
> b<-c(1,2,3)
> c(a,b)
[1] "aa" "bb" "cc" "dd" "ee" "ff" "1"  "2"  "3"
>
```

Pozor na vynútené pretypovanie prvkov vektora! Musí byť zachovaný rovnaký dátový typ prvkov.

Vektorová aritmetika

Pre vektory sú k dispozícii tieto operácie:

- + sčítanie po zložkách,
- − odčítanie po zložkách,
- ∗ pre násobenie, buď vektor číslom, alebo dva vektory po zložkách,
- / delenie po zložkách.

Pozor na recykláciu! Ak operácie vykonávame s vektormi nerovnakej dĺžky, prvky kratšieho vektora sa v rámci operácie recyklujú.
Počet prvkov dlhšieho vektora musí byť celočíselným násobkom počtu prvkov kratšieho vektora.

Vektorová aritmetika - ukážky

```
> a<-c(10,20,30,40)
> b<-c(1,2,3,4)
> a+b
[1] 11 22 33 44
> 5*b
[1]  5 10 15 20
> a-b
[1]  9 18 27 36
> a*b
[1]  10  40  90 160
> a/b
[1] 10 10 10 10
> b<-c(1,2,3,4,5,6,7,8)
> a+b
[1] 11 22 33 44 15 26 37 48
>
```

Vektory - indexácia prvkov

Index prvku sa uvádza v hranatých zátvorkách []

Pozície vo vektore sú číslované od 1.

Ukážka - vybratie štvrtého prvku vektora

```
> a<-c("aa","bb","cc","dd","ee","ff")
> a[4]
[1] "dd"
>
```

Záporná hodnota indexu – vektor, ktorý vznikne odstránením prvku, ktorý leží na pozícii rovnej absolútnej hodnote indexu.

```
> a[-4]
[1] "aa" "bb" "cc" "ee" "ff"
>
```

Vektory - indexácia prvkov

Z daného vektoru je možné vybrať podvektor, obsahujúci prvky na predpísaných pozíciách

Ukážka

```
> a<-c("aa","bb","cc","dd","ee","ff")
> a[c(2,3,5)]
[1] "bb" "cc" "ee"
>
```

Možná je aj takáto konštrukcia

```
> b<-c(1,3,5)
> a[b]
[1] "aa" "cc" "ee"
>
```

Vektory - indexácia prvkov

Indexy sa môžu aj opakovať

Ukážka

```
> a[c(2,2,3,5)]
[1] "bb" "bb" "cc" "ee"
>
```

Možná je aj zmena poradia

```
> a[c(2,1,5,3)]
[1] "bb" "aa" "ee" "cc"
>
```

Vektory - indexácia prvkov

Pomocou symbolu : je možné zadať číselný rozsah indexov od – do.

Ukážka

```
> a[2:4]
[1] "bb" "cc" "dd"
>
```

Záporné znamienko môže byť len pred číslom 0

```
> a[4:-0]
[1] "dd" "cc" "bb" "aa"
> a[-0:3]
[1] "aa" "bb" "cc"
>
```

Vektory - indexovanie typom logical

Príslušné prvky vektora je možné vybrať aj pomocou logických hodnôt `TRUE` a `FALSE`

Ukážka

```
> L<-c(TRUE,FALSE,FALSE,TRUE,FALSE,TRUE)
> a[L]
[1] "aa" "dd" "ff"
>
```

Ak je dĺžka vektora logických hodnôt menšia než dĺžka vektora, z ktorého vyberáme, na zvyšných pozíciách sa implicitne predpokladá hodnota `TRUE`.

Vektory - pomenovanie prvkov vektora

Jednotlivým prvkom vektora je možné priradiť pomenovanie pomocou funkcie `names()`

Ukážka

```
> a<-c("Mike","Lietavec")
> names(a)<-c("Meno","Priezvisko")
> a
      Meno  Priezvisko
    "Mike"  "Lietavec"
> a["Priezvisko"]
Priezvisko
"Lietavec"
>
```

Matica

Ide o dvojrozmernú dátovú štruktúru, usporiadanú do obdĺžnikovej schémy.

Maticu zadávame funkciou `matrix()` s týmito parametrami:

- vektor prvkov matice (v jednom riadku za sebou),
- `nrow` počet riadkov,
- `ncol` počet stĺpcov,
- `byrow` hodnoty `TRUE` a `FALSE` určujú, či má byť matica naplnená po riadkoch, implicitná hodnota je `FALSE`.

Matice – ukážky

```
> A<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=TRUE)
> A
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

```
> A<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=FALSE)
> A
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Matice – extrahovanie riadku, stĺpca

Základom je prístup pomocou dvojice indexov v hranatých zátvorkách.

```
> A[2,3]
[1] 6
> A[3,2] # index mimo rozsah
Error in A[3, 2] : subscript out of bounds
>
```

Ak je zadaný len jeden index, vypíše sa príslušný riadok resp. stĺpec.

```
> A[,2]   # druhy stlpec
[1] 3 4
> A[2,]   # druhy riadok
[1] 2 4 6
>
```

Matice – extrakcia viacerých riadkov, stĺpcov

Pomocou funkcie c()

```
> A[,c(2,1)]
     [,1] [,2]
[1,]    3    1
[2,]    4    2
>
```

Viaceré riadky a stĺpce súčasne

```
> A[c(2,1),c(3,1,2)]
     [,1] [,2] [,3]
[1,]    6    2    4
[2,]    5    1    3
>
```

Matice – prístup k prvkom pomenovaním

Riadky a stĺpce matice pomenujeme pomocou funkcií `dimnames()` a `list()`

```
> dimnames(A)<-list(
+ c("riadok1","riadok2"),                  # mena riadkov
+ c("stlpec1","stlpec2","stlpec3")) # mena stlpcov
> A["riadok2","stlpec3"]
[1] 6
>
```

Matice – násobenie

Rovnako ako pri vektoroch, aj pri maticiach sú všetky operácie definované „po zložkách“.

Preto je treba si dávať pozor, operácia `A*B` zodpovedá vynásobeniu prvkov na rovnakých pozíciách v maticiach `A` a `B`.

„Klasickému“ násobeniu matíc potom zodpovedá operácia `A%*%B`.

Viacrozmerné matice

Viacrozmerné matice zadávame pomocou funkcie `array()`.

Jej argumentmi sú vektor hodnôt a `dim=<vektor>`, čo je vektor jednotlivých rozmerov.

Ako príklad vyskúšajme

```
> x<-1:3
> y<-4:9
> A<-array(c(x,y),dim=c(3,3,2))
> A
```

Zoznam – charakteristika

Zoznam je usporiadaný súbor objektov rôzneho dátového typu.

Umožňuje zlučovať viaceré objekty pod jedným menom.

Zoznam sa vytvára pomocou funkcie `list()`.

Jednotlivým argumentom funkcie `list()` je možné priradiť názvy:

```
list(meno1=objekt1,meno2=objekt2, ...)
```

Zoznam – ukážka

```
> a<-"Nas prvy zoznam"
> b<-c(10,20,30,40)
> c<-c("aa","bb","cc","dd")
> zoznam<-list(title=a,pocet=b,c)
> zoznam
$title
[1] "Nas prvy zoznam"

$pocet
[1] 10 20 30 40

[[3]]
[1] "aa" "bb" "cc" "dd"

>
```

Zoznam – prístup ku položkám zoznamu

Ku jednotlivým položkám zoznamu pristupujeme buď pomocou hranatých zátvoriek alebo menom položky pomocou „dolárovej konštrukcie“ v tvare `<nazov zoznamu>$<meno>`

```
> zoznam[2]
$pocet
[1] 10 20 30 40

> zoznam$pocet
[1] 10 20 30 40
>
```

Zoznam – prístup ku položkám zoznamu

Pri použití vektoru indexov môžeme vybrať viacero položiek

```
> zoznam[c(1,3)]
$title
[1] "Nas prvy zoznam"

[[2]]
[1] "aa" "bb" "cc" "dd"

>
```

Zoznam – prístup ku položkám zoznamu

Ak chceme pristupovať ku jednotlivým prvkom niektorej položky zoznamu, použijeme dvojicu indexov, index položky v zdvojenej hranatej zátvorke, index prvku v jednoduchej.

Môžeme tak aj priamo pozmeniť jednotlivé prvky položiek v zozname.

```
> zoznam[[3]][1]<-c("AA")
> zoznam[3]
[[1]]
[1] "AA" "bb" "cc" "dd"
>
```

Dátová štruktúra frame

Slúži na uchovávanie tabuliek dát.

Svojou konštrukciou predstavuje vektor zoznamov.

Vytvoriť ho môžeme pomocou funkcie `data.frame()`.

```
> a<-c(1,2,3)
> b<-c("aa","bb","cc")
> c<-c(TRUE,FALSE,TRUE)
> d<-data.frame(a,b,c)   # d je frame
> d
  a  b     c
1 1 aa  TRUE
2 2 bb FALSE
3 3 cc  TRUE
>
```

Frame – prístup k položkám

Budeme pracovať so zabudovanou ukážkovou štruktúrou frame `mtcars`

Jej obsah si pozrieme jednoducho:

```
> mtcars
>
```

Prvý riadok obsahuje hlavičku (header).

Ostatné riadky sú údajové (data row), prvý stĺpec obsahuje pomenovania riadkov.

Jednotlivé prvky v riadkoch sa nazývajú bunky (cell).

Frame – prístup k položkám

Obsah jednotlivých buniek môžeme získať buď pomocou indexov v poradí riadok, stĺpec

```
> mtcars[2,3]
[1] 160
>
```

Inou možnosťou je použiť mená riadkov a stĺpcov

```
> mtcars["Mazda␣RX4␣Wag","disp"]
[1] 160
>
```

Frame – rozsah

Počet riadkov štruktúry frame zistíme pomocou funkcie `nrow()`

```
> nrow(mtcars)
[1] 32
>
```

Podobne počet stĺpcov pomocou `ncol()`

```
> ncol(mtcars)
[1] 11
>
```

Namiesto celého obsahu je často užitočné použiť funkciu `head()`, ktorá zobrazí len začiatok tabuľky.

Frame – stĺpce ako vektor

Jednotlivé stĺpce je možné vyberať ako vektory pomocou operátora dvojitých hranatých zátvoriek

S použitím indexu:

```
> mtcars[[1]]
 [1] 21.0 21.0 22.8 ...
>
```

S použitím mena stĺpca

```
> mtcars[["mpg"]]
 [1] 21.0 21.0 22.8 21.4 ...
>
```

```
> mtcars$mpg
 [1] 21.0 21.0 22.8 21.4 ...
>
```

Frame – stĺpce ako nový frame

Ak namiesto dvojitej zátvorky použijeme jednoduchú, výsledkom je stĺpec, ale v štruktúre frame

```
> mtcars[1]
                     mpg
Mazda RX4           21.0
Mazda RX4 Wag       21.0
>

> mtcars["mpg"]
```

Je tu možnosť vybrať viacero stĺpcov

```
> mtcars[c("mpg","cyl")]
                     mpg cyl
Mazda RX4           21.0   6
```

Frame – riadok ako nový frame

Podobne ako stĺpec, čiarkou za indexom musíme určiť, že požadujeme riadok

```
> mtcars[10,]
           mpg cyl  disp  ...
Merc 280  19.2   6 167.6 ...
>
```

S použitím mena riadku

```
> mtcars["Merc␣280",]
           mpg cyl  disp  ...
Merc 280  19.2   6 167.6 ...
>
```

Frame – riadky ako nový frame

Ak chceme vybrať viac riadkov, ich indexy alebo názvy zadáme ako vektor

```
> mtcars[c(3,10),]
             mpg cyl  disp  ...
Datsun 710  22.8   4 108.0  ...
Merc 280    19.2   6 167.6 ...
>

> mtcars[c("Merc 280","Datsun 710"),]
             mpg cyl  disp  ...
Merc 280    19.2   6 167.6 ...
Datsun 710  22.8   4 108.0  ...
>
```

Faktor

Faktor je dátový typ používaný pri práci s kategoriálnymi premennými. (Triedenie do kategórií podľa určitého znaku).

Premenná typu faktor sa vytvára pomocou príkazu `factor()`, jediným argumentom je vektor hodnôt. Vstupom môžu byť numerické aj reťazcové premenné, ale výstupnými úrovňami sú vždy premenné typu `character`.

```
> faktor<-factor(c(1,2,3,4))
> faktor
[1] 1 2 3 4
Levels: 1 2 3 4
>levels(faktor)
[1] "1" "2" "3" "4"
>
```

Faktory a výpočty s nimi

Aj v prípade, že faktory majú numerické hodnoty, ich úrovne sú interpretované ako reťazce.

Na prevod na numerické hodnoty je potrebné použiť funkciu `as.numeric()`.

```
> mean(faktor)
[1] NA
Warning message:
In mean.default(faktor) : ...
> mean(as.numeric(faktor))
[1] 2.5
>
```

Faktory a tabuľky

Faktory sú výhodné pre vytváranie jednocestných tabuliek.

Využívame pri tom funkciu `table()`.

Príklad

```
> a <- factor(c("A","A","B","A","B","B","C","A","C"))
> results <- table(a)
> results
a
A B C
4 3 2
>
```