Predavanje 01 – Uvod, matematične operacije, vektorji

Programski jezik R

R je programski jezik in v širšem pomenu programska oprema, ki je namenjena predvsem delu s podatki, analizi in vizualizaciji.

R je brezplačen, najdemo pa ga lahko na strani https://cran.r-project.org/. Predpostavili bomo, da smo si R že namestili in imamo dostop do najbolj osnovnega komuniciranja v programskem jeziku R – preko t. i. konzole programskega jezika R:

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Znak `>` pomeni, da R čaka na naš ukaz. Programski jeziki so namreč le umetni jeziki, ki smo jih ustvarili za komuniciranje z računalniki, da jim lažje sporočimo, kaj bi radi, da naredijo za nas. Za razliko od naravnih jezikov, so programski jeziki veliko bolj preprosti, obenem pa so bolj zahtevni, saj je računalnik, če povemo po domače, precej neumen. Če mu ne bomo temeljito in brez napake sporočili, kaj želimo, nas ne bo razumel.

Za začetek ga poskusimo pozdraviti s "halo":

```
halo
```

```
## Error in eval(expr, envir, enclos): object 'halo' not found
```

Kot lahko vidimo, našega pozdrava ne razume in sporoči napako. Napak se ne bojimo – računalnik je zelo potrpežljiv in nam ne bo zameril, tudi če se 100x zmotimo.

R kot kalkulator

Sedaj pa je čas, da se naučimo nekaj besed v jeziku R. Začeli bomo s številkami in z osnovnimi računskimi operacijami. Jezik R, kot velika večina programskih jezikov, podpira delo s celimi in decimalnimi števili.

Najprej mu napišimo samo številko 4.5:

```
4.5
```

```
## [1] 4.5
```

R bo za nami ponovil 4.5, kar je naša prva uspešna komunikacija z njim! Obenem smo se naučili pomembne lastnosti jezika R – če mu sporočimo številko ali, kot bomo kasneje videli, izraz ali spremenljivko, R izpiše vrednost.

R podpira osnovne matematične operacije seštevanja, odštevanja, množenja in deljenja, ki so predstavljene s standardnimi simboli `+`, `-`, `*` in `/`.

Poskusimo sešteti 1 + 1:

```
1 + 1
```

```
## [1] 2
```

Kot vidimo, R pravilno izračuna vrednost izraza 1 + 1 in, kot smo že rekli, izpiše vrednost. Na tem mestu bi omenili, da R ni občutljiv na presledke, zato bo enako razumel npr. te ukaze:

```
1 + 1
```

```
## [1] 2
```

```
1 +  1
```

```
## [1] 2
```

```
 1+1
```

```
## [1] 2
```

```
1+  1
```

```
## [1] 2
```

Zaradi bolj pregledne programske kode pa se držimo, da med operacijami vnesemo presledek, pred in za ukazi pa ne vnašamo nepotrebnih presledkov.

Poskusimo še z nekaj izračuni:

```
4.5 - 2.7
```

```
## [1] 1.8
```

```
25 / 5
```

```
## [1] 5
```

```
4 * 4
```

```
## [1] 16
```

```
1643 / 3
```

```
## [1] 547.6667
```

```
5 / 3 + 2
```

```
## [1] 3.666667
```

```
5 / (3 + 2)
```

```
## [1] 1
```

Pri zadnjih dveh izrazih velja omeniti, da R da prednost deljenju pred množenjem pred seštevanjem/odštevanjem. Če želimo najprej sešteti $3 + 2$, potem mu moramo to sporočiti z oklepaji.

Že osnovni R podpira tudi druge uporabne matematične operacije, kot so potenciranje (`^`, ima prednost pred deljenjem), logaritmiranje (`log`, `log2`, `log10`, ...) in trigonometrične funkcije (`sin`, `cos`, ...). Nekaj primerov:

```
3^2
```

```
## [1] 9
```

```
sqrt(4)
```

```
## [1] 2
```

```
log10(100)
```

```
## [1] 2
```

```
sin(0)
```

```
## [1] 0
```

```
cos(0)
```

```
## [1] 1
```

```
log2(2^10)
```

```
## [1] 10
```

```
((sin(3) + cos(4))^2 + 2.5) / 100
```

```
## [1] 0.0276268
```

Spremenljivke

Kalkulatorji nam omogočajo, da si vrednost izračuna tudi shranimo za poznejšo uporabo, kar pogosto poenostavi izračune, če se del izračuna večkrat ponovi. Koncept shranjevanja vrednosti je sestaven del večine programskih jezikov, entitete, ki hranijo vrednosti, pa imenujemo spremenljivke.

Vsaka spremenljivka ima unikatno ime, za katerim se skriva njena vrednost. Kot smo že omenili, če spremenljivko pokličemo po imenu, R samo izpiše njeno vrednost. To lahko poskusimo s `pi`, ki je spremenljivka, ki je že vgrajena v R, in hrani približek te znane matematične konstante. Seveda lahko ta `pi` uporabimo tudi v izrazih:

```
pi
```

```
## [1] 3.141593
```

```
pi^2
```

```
## [1] 9.869604
```

```
sin(pi)
```

```
## [1] 1.224606e-16
```

```
cos(pi)
```

```
## [1] -1
```

Naslednji korak pa je, da se naučimo, kako ustvariti spremenljivko. Najlažji način je, da ji izberemo ime in izbranem imenu priredimo vrednost s pomočjo operatorja za določanje vrednosti `<-`. Pri izbiri imen spremenljivk se zaenkrat omejimo na črke, med katerimi je lahko tudi podčrtaj. Npr. `x`, `y`, `z`, `vmesni_rezultat`, ipd. . .

Poglejmo si, kako spremenljivka x ne obstaja, ko pa ji dodelimo vrednost, jo R izpiše:

```
x
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

```
x <- 5
x
```

```
## [1] 5
```

Vrednost spremenljivke lahko določimo tudi z izrazom, kar nam potem omogoča, da vrednost izraza uporabimo v različnih nadaljnjih izračunih:

```r
x <- 5
y <- 2
z <- (x + y)^2
log(z)
```

```
## [1] 3.89182
```

```r
z^2
```

```
## [1] 2401
```

Imena vseh naših spremenljivk, ki trenutno obstajajo, lahko izpišemo z ukazom `ls()`:

```r
ls()
```

```
## [1] "x" "y" "z"
```

Če želimo spremenljivko odstraniti, kar bomo sicer le redko delali, lahko uporabimo ukaz `rm(<ime spremenljivke>)`:

```r
rm(x)
ls()
```

```
## [1] "y" "z"
```

R ukaze izvaja v vrstnem redu, kot jih dobi. Če se vrednost spremenljivke na nekem mestu spremeni, bodo nadaljnji ukazi to spremembo upoštevali:

```r
x <- 5
x^2
```

```
## [1] 25
```

```r
x <- 4
x^2
```

```
## [1] 16
```

Ko R zapremo, se izgubijo vse informacije o spremenljivkah. Če bi jih želeli uporabiti tudi kasneje, jih lahko shranimo z ukazom `save.image()` in naložimo z ukazom `load()`. Pokažimo to

```r
ls()
```

```
## [1] "x" "y" "z"
```

```r
save.image(file = 'spremenljivke.RData')
rm(list = ls(all = T)) # pobrišemo vse spremenljivke
ls()
```

```
## character(0)
```

```
load('spremenljivke.RData')
ls()
```

```
## [1] "x" "y" "z"
```

Vektorji

V programskih jezikih spremenljivke pogosto organiziramo na način, ki olajša delo z njimi. Takim načinom organizacije pravimo tudi podatkovne strukture. Prva in najpomembnejša struktura, ki jo bomo spoznali, je t.i. vektor – seznam spremenljivk istega tipa.

Vektor lahko naredimo na več načinov. Najprej z združevanjem elementov s funkcijo za združevanje `c()`:

```
c(1, 2, 3)
```

```
## [1] 1 2 3
```

```
c(x, y, z)
```

```
## [1]  4  2 49
```

```
x1 <- c(2, 5, 3)
x1
```

```
## [1] 2 5 3
```

Zadnji primer pokaže, da lahko spremenljivka kaže tudi na podatkovno strukturo in ne samo na eno samo številko.

Če želimo narediti vektor določene dolžine z določenimi elementi, potem lahko uporabimo funkcijo `rep()`, ki ima dva argumenta – vrednost in dolžina:

```
rep(1, 5)
```

```
## [1] 1 1 1 1 1
```

```
rep(x, 10)
```

```
##  [1] 4 4 4 4 4 4 4 4 4 4
```

Do posameznih elementov seznama dostopamo z indeksiranjem z uporabo `[]`. Posamezen element vektorja števil je tudi spremenljivka, zato lahko ji spremenimo vrednost.

```
x1 <- c(25, 3, -1, -20)
x1
```

```
## [1]  25   3  -1 -20
```

```
x1[1]
```

```
## [1] 25
```

```
x1[2]
```

```
## [1] 3
```

```
x1[3]
```

```
## [1] -1
```

```
x1[3] <- 2
x1[3]
```

```
## [1] 2
```

Vektor je najpomembnejša podatkovna struktura v R. Velika prednost Rja je, da lahko operacije izvajamo tudi nad vektorji, ne samo nad posameznimi številkami. Operacija se izvede nad vsakim elementom.

```
x <- c(1, 2, 2, 5)
x^2
```

```
## [1]  1  4  4 25
```

```
x - 5
```

```
## [1] -4 -3 -3  0
```

```
-x
```

```
## [1] -1 -2 -2 -5
```

```
x / 5
```

```
## [1] 0.2 0.4 0.4 1.0
```

Računamo lahko tudi z vektorji. Zaenkrat predpostavimo, da to lahko počnemo samo z vektorjema enake dolžine:

```
x <- c(1, 2, 2, 5)
y <- c(-1, -2, -2, -5)
x + y
```

```
## [1] 0 0 0 0
```

```
x * y
```

```
## [1]  -1  -4  -4 -25
```

Prav tako R vsebuje mnogo uporabnih funkcij nad vektorji, začenši z vsoto, povprečjem, mediano, minimumom, maksimumom in standardnim odklonom:

```
x <- c(1, 2, 2, 5)
sum(x)
```

```
## [1] 10
```

```
mean(x)
```

```
## [1] 2.5
```

```
median(x)
```

```
## [1] 2
```

```
min(x)
```

```
## [1] 1
```

```
max(x)
```

```
## [1] 5
```

```
sd(x)
```

```
## [1] 1.732051
```

V R lahko dostopamo tudi do več elementov hkrati. To lahko storimo na zelo enostaven način – namesto ene same številke indeksiramo z vektorjem številk:

```
x <- c(1, 2, 2, 5)
x[c(1, 3)]
```

```
## [1] 1 2
```

```
x[1:3]
```

```
## [1] 1 2 2
```

V drugem primeru smo uporabili še eno uporabno "besedo" iz programskega jezika R. Z `a:b` dobimo vektor vseh števil med a in b.

```r
1:5
```

```
## [1] 1 2 3 4 5
```

```r
-4:4
```

```
## [1] -4 -3 -2 -1  0  1  2  3  4
```

```r
4:-2
```

```
## [1]  4  3  2  1  0 -1 -2
```

Domača naloga

1. Izrek Pitagore pravi, da v pravokotnem trikotniku velja $a^2 + b^2 = c^2$, kjer sta a in b dolžini katet, c pa dolžina hipotenuze. Napišite kratek program, kjer na začetku dodelimo vrednosti dolžinama a in b, nato pa izračunamo dolžino hipotenuze $c = \sqrt{a^2 + b^2}$ in jo izpišemo. Kaj se izpiše, če a ali b namesto ene številke priredimo vektor številk in izračun ponovimo?

Primer (izračun je izpuščen):

```r
a <- 3
b <- 4
```

```
## [1] 5
```

2. Podan imamo vektor cen izdelkov in vektor enake dolžine, v katerem so zapisane stopnje DDV za izdelke v prvem vektorju. Primer:

```r
cena <- c(153, 4.5, 12, 53.5, 8.7) # cena v €
DDV  <- c(22, 9.5, 22, 22, 9.5) # DDV v %
```

S čim manj operacijami izračunajte skupno ceno brez in skupno ceno z DDV.

Rešitev za primer zgoraj:

```
## [1] 231.7
```

```
## [1] 281.024
```

3. **Težja naloga** V vektorju `temperatura_max` so zapisane napovedi najvišje dnevne temperature za naslednjih 14 dni, podobno v vektorju `temperatura_min` pa najnižje.

```r
temperatura_max <- c(21.1, 19.4, 18.0, 17.8, NA, 16.2, 16.8,
                     18.9, 17.6, 19.2, NA, 21.1, 19.9, 21.7) # najvišja dnevna temperatura
temperatura_min <- c(10.2, 6.0, 10.3, 11.0, 9.1, 7.8, NA,
                     9.0, 9.2, 8.6, 8.1, 9.2, 8.3, 7.6) # najnižja dnevna temperatura
```

Kot opazimo, v prvem vektorju na 5. in 11. mestu in v drugem na 7. mestu se pojavi napis "NA". To je oznaka, ki jo R uporabi, da označi manjkajoče vrednosti. Z vektorji, ki vsebujejo manjkajoče vrednosti, malce težje računamo, saj R ne ve kako naj jih upošteva. Primer:

```r
sum(temperatura_max)
```

```
## [1] NA
```

Kot vidimo, vsota vrednosti vektorja, ki vsebuje NA, vrne prav tako NA. Da bi se izognili tej nevšečnosti, moramo izrecno povedati funkciji, naj ignorira NA-je. Ravno temu služi parameter "na.rm" v funkciji sum.

```r
sum(temperatura_max, na.rm = TRUE)
```

```
## [1] 227.7
```

Odstranjevanje ali polnjenje manjkajočih vrednosti je pogost problem s katerim se srečujemo pri obdelavi podatkov. V tej nalogi imamo en primer takega problema, ki ga lahko rešimo s znanjem, ki smo ga pridobili med tem predavanjem.

a. Zamenjaj manjkajoče vrednosti v vektorjih s povprečno vrednostjo istega vektorja.

Rešitev:

```
##  [1] 21.100 19.400 18.000 17.800 18.975 16.200 16.800 18.900 17.600 19.200
## [11] 18.975 21.100 19.900 21.700
```

```
##  [1] 10.2  6.0 10.3 11.0  9.1  7.8  8.8  9.0  9.2  8.6  8.1  9.2  8.3  7.6
```

b. Izračunaj minimalen in maksimalen dnevni razpon temperature v naslednjih 14 dneh.

```
## [1] "Maksimalen: "
```

```
## [1] 14.1
```

```
## [1] "Minimalen: "
```

```
## [1] 6.8
```