Uvod - osnove jezika R

Denis Vlašiček

U ovom dijelu proći ćemo kroz osnove programskog jezika R. Obradit ćemo osnove njegove sintakse te tipove varijabli. Ovdje ćemo proći kroz neke jako bazične stvari. Većina "naprednijih" stvari provlačit će se kroz ostatak radionice.

Osnovne matematičke operacije

Za početak, pogledat ćemo kako možemo izvršavati jednostavne naredbe direktno u R konzoli.

Kad god u R konzolu unesemo naredbu koju R smatra valjanom te pritisnemo `Enter` (također poznat kao `Return`), R će u konzoli izbaciti rezultat izvršavanja te naredbe. Na primjer, ako u konzolu unesemo `2 + 2`, R će izbaciti rezultat te operacije.

```
2 + 2
```

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

Isto možemo napraviti s dijeljenjem (`/`), množenjem (`*`), oduzimanjem (`-`) i potenciranjem (`^`).

```
4 / 2
```

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

```
2 * 3
```

```
## [1] 6
```

```
5 - 1
```

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

```
3^2
```

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

Poseban operator koji nekad zna biti koristan je *modulo* - `%%` - koji vraća cijelobrojni dio dijeljenja dvaju brojeva. Na primjer:

```
# (Linije koje sadrže komentare započinju sa znakom #. R ne interpretira te linije.)
# Obično dijeljenje:
5 / 2
```

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

```
# Modulo
5 %% 2
```

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

Naravno, kao i u pješačkoj matematici, i u R-u je potrebno paziti na grupiranje matematičkih izraza.

`x + y / z` je $x + (y/z)$ je $x + \frac{z}{y}$

`(x + y) / z` je $(x + y)/z$ je $\frac{x+y}{z}$.

Funkcije

Sa samim matematičkim operacijama nećemo daleko doći. R ima i funkcije - operacije koje primaju parametre (eng. *argument*) i vraćaju neke vrijednosti.

Funkcije u R-u imaju opći oblik `funkcija(argument1, argument2, ... , argumentN)`.

Prva funkcija koju ćemo pogledati, a koja nadopunjava matematičke operacije s kojima smo započeli je `sqrt`, kojom možemo dobiti korijen nekog broja.

```
sqrt(4)
```

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

Druga, koja se u R-u javlja **jako** često, je `c` (što je, prema Adleru [2012] skraćeno za *combine*). `c` uzima N argumenata i spaja ih u **vektor**.

```
# navodnici su bitni! ali mogu biti jednostruki ili dvostruki,
# bitno je samo da je riječ omeđena jednakim parom
# npr 'a' je oke, "a' nije oke, ali zato "a" je
print(c('patka', "krava", 'pile', "krumpir"))
```

```
## [1] "patka"   "krava"   "pile"    "krumpir"
```

```
print(c(5, 4, 3, 2, 1))
```

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

Koristeći `c`, stvorili smo dva vektora. Vektori spadaju među osnovne strukture podataka u R-u. Vektori mogu sadržavati proizvoljan broj elemenata **istog tipa**. O tipovima ćemo pričati malo kasnije.

Sada ćemo se pozabaviti varijablama.

Varijable

Kad god smo dosad izvršavali neke funkcije, baratali smo konkretnim vrijednostima (npr. `2 + 2`), a rezultati su ostali lebdjeti u eteru, nedostupni običnim ljudima.

Kako bismo mogli baratati proizovljnim vrijednostima te kako bismo rezultati izvukli iz etera, uvodimo **varijable**.

Varijablu imenujemo (eng. *declare*) tako što neki poluproizvoljan naziv spojimo s nekom vrijednosti, koristeći operator `<-`. Na pimjer:

```
a <- 2
```

Ako sad u konzolu unesemo `a`, konzola će nam vratiti vrijednost te varijable. Isti rezultat dobili bismo ako bismo `a` iskoristili kao argument `print` funkcije (`print(a)`).

```
print(a)
```

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

Vrijednosti varijablama možemo pridavati (eng. *assign*) i koristeći znak `=` (razlikovati od `==`!), no to se **ne preporučuje**. Osim toga, vrijednosti možemo pridavati i s lijeva na desno, koristeći `->`:

```
3 -> b
b
```

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

Imena varijabli mogu sadržavati slova, brojeve, točke (`.`) i underscoreove (čija hrvatska imena ne znam; `_`). Imena varijabli ne mogu započinjati s točkom koju prati broj. Na primjer:

```
.3 <- 5
```

```
## Error in 0.3 <- 5: invalid (do_set) left-hand side to assignment
```

Također, imena varijabli ne mogu biti izrazi koji su rezervirani u samom programskom jeziku, kao što je `for` (koji se koristi za iniciranje petlji).

```
for <- 5
```

```
## Error: <text>:1:5: unexpected assignment
## 1: for <-
##         ^
```

Funkcija koja zna biti zgodna kod imenovanja varijabli je `exists`, koja kao argument prima izraz u navodnicima (što je poznato kao **string**) te vraća `TRUE` ili `FALSE` ovisno o tom je li objekt istog imena pronađen ili ne.

```
# varijabla koju smo ranije stvorili
exists('a')
```

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

```
# ključna riječ, definirana u R-u, ne smijemo koristiti
exists('for')
```

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

```
# ključna riječ, definirana u R-u, ne smijemo koristiti
exists('if')
```

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

```
# ime koje nije iskorišteno
exists('maca')
```

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

Sad kad znamo kako varijablama pripisati vrijednosti, možemo spremiti vektore koje smo ranije napravili koristeći `c`. Neovisno o tome što *možemo* koristiti svašta za imena varijabli, trebali bismo se truditi imena učiniti smislenima. Dugoročno, to će nas poštedjeti puno mentalnog (a nekad i R-ovskog) napora. Također, savjetovao bih da izbjegavate korištenje "hrvatskih" znakova (č, ć, ž, š, đ) u svom kodu; korištenje tih znakova može izazvati snažne glavobolje.

```
domace_zivotinje_i_krumpir <- c('patka', 'krava', 'pile', 'krumpir')

brojevi.5.do.1 <- c(5, 4, 3, 2, 1)
```

Kao i kad smo varijabli `a` pripisali vrijednost `2`, ni sada ne dobivamo nikakav output u konzoli. Ali možemo koristiti `print` ili samo upisati ime varijable u konzolu kako bismo dobili njenu vrijednost.

```
domace_zivotinje_i_krumpir
```

```
## [1] "patka"   "krava"   "pile"    "krumpir"
```

```
brojevi.5.do.1
```

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

Sad kad smo svoje vektore pripisali varijablama, možemo dohvaćati pojedine vrijednosti iz njih. Na primjer, ako želimo dohvatiti prvu vrijednost iz vektora `domace_zivotinje_i_krumpir`, možemo učiniti ovo:

```
domace_zivotinje_i_krumpir[1]
```

```
## [1] "patka"
```

`1` je, u ovom slučaju, **indeks**. U R-u, za razliku od većine drugih programskih jezika, indeksiranje započinje s `1`, a ne s `0`. Za dohvaćanje trećeg elementa iz vektora `brojevi.5.do.1` izvršili bismo:

```
brojevi.5.do.1[3]
```

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

Zadnji element možemo dohvatiti pomoću funkcije `length`, koja vraća duljinu vektora, tj. broj elemenata koji se u njemu nalaze. Na primjer:

```
length(domace_zivotinje_i_krumpir)
```

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

Budući da broj elemenata ujedno označava i posljednji element, možemo učiniti sljedeće:

```
# dohvaćanje pomoću indeksa
domace_zivotinje_i_krumpir[4]
```

```
## [1] "krumpir"
```

```
# dohvacanje pomocu funkcije length()
domace_zivotinje_i_krumpir[length(domace_zivotinje_i_krumpir)]
```

```
## [1] "krumpir"
```

```
# iskoristit ćemo priliku i pokazati kako možemo usporediti dvije vrijednosti
domace_zivotinje_i_krumpir[4] ==
    domace_zivotinje_i_krumpir[length(domace_zivotinje_i_krumpir)]
```

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

Ovo funkcionira jer evaluiranje, odnosno izvršavanje koda `length(domace_zivotinje_i_krumpir)` kao rezultat vraća brojku `4`.

Također, vidjeli smo da možemo koristiti `==` kako bismo provjerili jesu li dva objekta, odnosno dvije varijable jednake. Na primjer

```
2 + 2 == 4
```

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

```
4 == 4
```

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

```
4 == 5
```

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

Treba voditi računa o tome da su `==` i `=` **vrlo različiti**!

Tipovi varijabli

R razlikuje nekoliko osnovnih tipova podataka:

- `character` : "stringovi", tj. tekstualni podaci. Npr. `'patka'`
- `integer` : cijeli brojevi. Npr. `1`
- `numeric` : realni brojevi. Npr. `1.161`
- `logical` : logičke vrijednosti. Postoje ukupno dvije - `TRUE` (može se kratiti u `T`) i `FALSE` (može se kratiti u `F`)

Pogledat ćemo nekoliko primjera ovih tipova, te vidjeti kako možemo provjeriti kojeg je neka varijabla ili vrijednost tipa.

```r
# character
'susjed'
```

```
## [1] "susjed"
```

```r
# da bismo provjerili je li neka vrijednost character, koristimo is.character()
is.character('susjed')
```

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

```r
is.character(domace_zivotinje_i_krumpir[4])
```

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

```r
is.character(1)
```

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

Kod `integer` i `numeric` tipova postoje neke specifičnosti.

```r
# integer
1
```

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

```r
# za provjeravanje koristimo is.integer()
is.integer(1)
```

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

Pozivanje funkcije `is.integer()` s vrijednosti `1` vraća `FALSE`. To je zato jer R brojeve automatski sprema kao `numeric`.

```r
# za provjeravanje je li objekt numeric koristimo is.numeric()
is.numeric(1)
```

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

Kako bismo natjerali R da nam da `integer` vrijednost, možemo staviti `L` na kraj broja:

```r
is.integer(1L)
```

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

Ovo je zgodno znati jer se može dogoditi da funkcija traži `integer`, ali odbija prihvatiti (recimo) `5` kao odgovarajuću vrijednost.

```r
# na kraju, numeric
is.numeric(1.5115)
```

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

Za pisanje decimalnih brojeva **moramo koristiti točku**.

```r
is.numeric(1,4141)
```

```
## Error in is.numeric(1, 4141): 2 arguments passed to 'is.numeric' which requires 1
```

```r
1,5151 + 1
```

```
## Error: <text>:1:2: unexpected ','
## 1: 1,
##      ^
```

Posljednji tip je `logical`:

```
TRUE == T
```

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

```
FALSE == F
```

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

```
is.logical(TRUE)
```

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

```
is.logical(F)
```

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

Za kraj, pogledat ćemo output različitih `is.` funkcija kad im damo različite vrijednosti.

```
is.logical(1)
```

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

```
is.numeric(1L)
```

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

```
is.character(02918)
```

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

```
is.integer(151518)
```

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

Primjećujemo da se `1L` tretira i kao `numeric` tip.

```
is.integer(1L)
```

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

```
is.numeric(1L)
```

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

Isto ne vrijedi za, na primjer, `5.911`:

```
is.integer(5.911)
```

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

```
is.numeric(5.911)
```

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

Nakon upoznavanja s osnovnim tipovima vrijednosti i varijabli, pogledat ćemo osnovne strukture podataka.