

Uvod v uporabo programskega okolja R

KAZALO

1	UVOD V R	1
1.1	Delo z R	1
1.1.1	Pomoč	3
1.1.2	Delovno okolje (Workspace).....	3
1.2	Paketi	4
1.3	Osnovne računske operacije v R.....	5
2	PODATKOVNE STRUKTURE	5
2.1	Vektorji	6
2.1.1	Vnos vektorjev v R.....	6
2.1.2	Priklic elementov vektorja	7
2.2	Seznami.....	8
2.2.1	Vnos seznamov v R.....	8
2.2.2	Priklic komponent seznama	9
2.3	Matrike.....	10
2.3.1	Vnos matrik v R.....	10
2.3.2	Priklic elementov matrike	11
2.4	Podatkovni okviri	12
2.4.1	Vnos podatkovnega okvira v R.....	12
2.4.2	Priklic elementov podatkovnega okvira.....	13

KAZALO SLIK

Slika 1: Okno za vnos ukazov (R Console)	1
Slika 2: Okno z naborom ukazov iz izvirne datoteke (Script)	2

KAZALO TABEL

Tabela 1: Funkcije za pomoč	3
Tabela 2: Funkcije za upravljanje delovnega okolja.....	4
Tabela 3: Osnovni operatorji.....	5

1 UVOD V R

R lahko opišemo kot jezik (angl. *language*) in okolje (angl. *environment*) in je namenjen predvsem statističnim izračunom in grafičnim prikazom. Je odprtokodna rešitev za izvedbo različnih analiz podatkov in je podprta s strani velikih in aktivnih raziskovalnih skupnosti širom sveta.

Med drugim je R:

- učinkovito orodje za ravnanje s podatki in shranjevanje,
- nabor operatorjev za izvajanje izračunov na nizih, predvsem matrikah,
- velika, skladna in integrirana zbirka vmesnih orodij za analizo podatkov,
- zmogljivo orodje za grafično predstavitev rezultatov analize podatkov in njihov prikaz ali na računalniku ali na papirju.

R opisujemo tudi kot okolje (angl. *environment*). Pojem “okolje” v bistvu kaže na dejstvo, da je R označen kot popolnoma načrtovan, skladen (koherenten) sistem, namesto značilnega primarnega nalaganja specifičnih in neprilagodljivih orodij, kot to pogosto velja za ostalo programsko opremo za analizo podatkov.

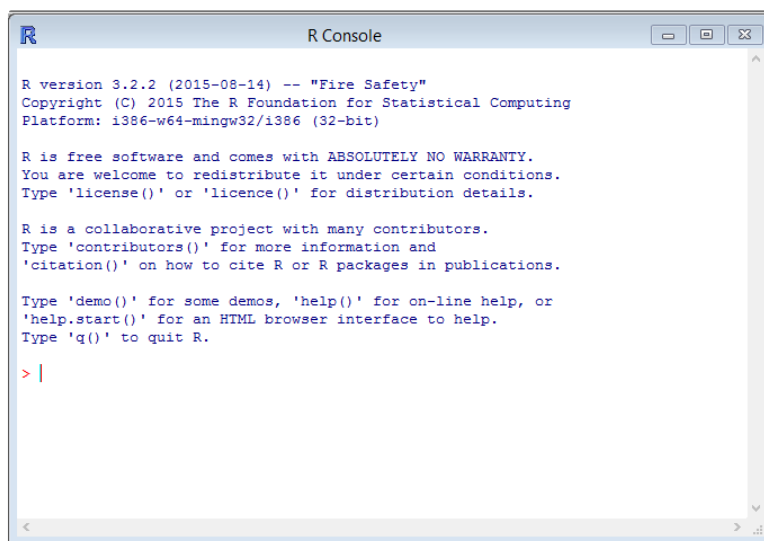
R predstavljajo tudi kot “vodilo” za novo razvijajoče metode za interaktivno analizo podatkov. R okolje se razvija hitro, programerji pa ga stalno razširjajo z velikim naborom paketov. Večina programov, napisanih v R, je v bistvu specifičnih, napisani so namreč za posamezne analize podatkov.

1.1 DELO Z R

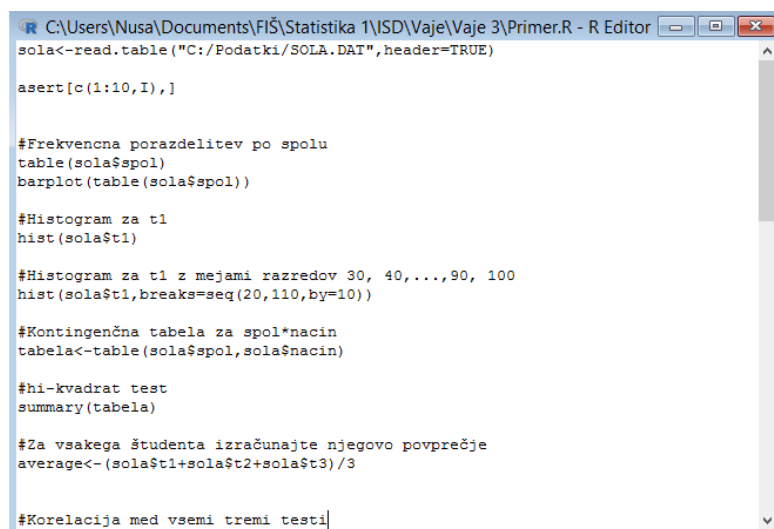
Za R pravimo, da je »case-sensitive« - torej, razlikuje med malimi in veliki črkami. Na primer, črki A in a sta za R različna simbola in bi se torej nanašala na različni spremenljivki.

Ukaze lahko vpisujemo enega za drugim v ukazno vrstico, ki se prične s simbolom `>` (Slika 1), lahko pa zaženemo nabor ukazov iz izvirne datoteke (Slika 2).

Slika 1: Okno za vnos ukazov (R Console)



Slika 2: Okno z naborom ukazov iz izvorne datoteke (Script)



```
C:\Users\Nusa\Documents\FIŠ\Statistika 1\ISD\Vaje\Vaje 3\Primer.R - R Editor
sola<-read.table("C:/Podatki/SOLA.DAT",header=TRUE)

asert[c(1:10,I),]

#Frekvenčna porazdelitev po spolu
table(sola$spol)
barplot(table(sola$spol))

#Histogram za t1
hist(sola$t1)

#Histogram za t1 z mejami razredov 30, 40,...,90, 100
hist(sola$t1,breaks=seq(20,110,by=10))

#Kontingenčna tabela za spol*nacin
tabela<-table(sola$spol,sola$nacin)

#hi-kvadrat test
summary(tabela)

#Za vsakega študenta izračunajte njegovo povprečje
average<-(sola$t1+sola$t2+sola$t3)/3

#Korelacija med vsemi tremi testi
```

Večina funkcionalnosti je na voljo preko vgrajenih funkcij, ustvarjenih s strani uporabnikov. Funkcije so shranjene v t.i. paketih (angl. *packages*).

Vrednosti, ki jih vnašamo v R, je priporočljivo shraniti v določen objekt. To pomeni, da vrednosti priredimo spremenljivki (ali drugemu objektu). Za ta namen R uporablja t.i. prireditvene stavke. Za prirejanje uporabimo znak '<-' ali '=' (uporaba enačaja se sicer odsvetuje). Nato namesto vrednosti uporabljamo ime spremenljivke (objekta), v kateri so vrednosti shranjene.

Na primer, stavek

```
x<-c(1,2,3,4,5)
```

ustvari vektor *x*, ki vključuje števila 1, 2, 3, 4 in 5.

Lahko pa tudi obrnemo smer prireditvenega stavka, pri čemer uporabimo znak '->'. Na primer, stavek

```
c(1,2,3,4,5)->x
```

je ekvivalentnem prejšnjemu.

V R lahko uporabljamo tudi komentarje. Ti so nam v veliko pomoč, ko si želimo poleg ukazov zapisati tudi kakšen komentar za kasneje. Vsak komentar se prične z znakom '#'. Vse, kar sledi temu znaku, R smatra kot komentar in zapis pri izvajanju ukazov ignorira.

Na primer iz zapisa

```
x<-c(1,2,3,4,5) #vrednosti 1, 2, 3, 4 in 5 shranimo v vektor x
```

R prebere ukaz

```
x<-c(1,2,3,4,5)
```

in ustvari vektor *x*, ki vključuje števila 1, 2, 3, 4 in 5. Tekst, ki se nahaja za znakom '#' pa ignorira.

1.1.1 Pomoč

R ponuja širok nabor pomoči, ki nam lahko izjemno olajša delo. Vgrajen sistem pomoči zagotavlja podrobnosti, reference in primere za vse funkcije, ki so zajete v trenutno nameščenih paketih. Tabela 1 prikazuje funkcije, ki jih uporabljamo za klicanje pomoči.

Tabela 1: Funkcije za pomoč

Funkcija	Dejanje
<code>help.start()</code>	splošna pomoč
<code>help('foo')</code> ali <code>?foo</code>	Pomoč za funkcijo <code>foo</code>
<code>help.search('foo')</code> ali <code>??foo</code>	Iskanje primerov niza <code>foo</code> po sistemu pomoči
<code>example('foo')</code>	Primeri funkcije <code>foo</code>
<code>RSiteSearch('foo')</code>	Iskanje niza <code>foo</code> v spletnih priročnikih za pomoč in arhiviranih poštnih seznamih (angl. <i>mailing lists</i>)
<code>apropos('foo', mode='function')</code>	Seznam vseh razpoložljivih funkcij s <code>foo</code> v njihovem imenu
<code>data()</code>	Seznam vseh razpoložljivih primerov podatkov, ki so na voljo v trenutno naloženih paketih
<code>vignette()</code>	Seznam vseh razpoložljivih kratkih opisov trenutno naloženih paketov
<code>vignette('foo')</code>	Prikaz določenih kratkih opisov na temo <code>foo</code> .

1.1.2 Delovno okolje (Workspace)

Delovno okolje vključuje vse objekte (vektorje, matrike, funkcije, podatkovne okvire ali sezname), ki jih je definirala uporabnik.

Pregled funkcij, ki so nam v pomoč pri upravljanju z delovnim okoljem, prikazuje Tabela 2.

.

Tabela 2: Funkcije za upravljanje delovnega okolja

Funkcija	Dejanje
<code>getwd()</code>	prikaže trenutni delovno mapo
<code>setwd('mydirectory')</code>	spremeni trenutno delovno mapo v <code>mydirectory</code>
<code>ls()</code>	prikaže objekte trenutnega delovnega okolja
<code>rm(objectlist)</code>	odstrani (zbriše) enega ali več objektov
<code>help(options)</code>	spoznamo razpoložljive možnosti
<code>options()</code>	pregledamo ali nastavimo trenutne možnosti
<code>savehistory('myfile')</code>	shranimo zgodovino ukazov v <code>myfile</code> (privzeto: <code>.Rhistory</code>)
<code>loadhistory('myfile')</code>	naložimo zgodovino ukazov (privzeto: <code>.Rhistory</code>)
<code>save.image('myfile')</code>	shranimo delovno okolje v <code>myfile</code> (privzeto: <code>.RData</code>)
<code>save(objectlist, file='myfile')</code>	shranimo določene objekte v datoteko
<code>load('myfile')</code>	naložimo delovno okolje v trenutno sejo (privzeto: <code>.RData</code>)
<code>q()</code>	zapustimo (zapremo) R. Pozvani smo k shranjevanju delovnega okolja

1.2 PAKETI

Večina najbolj vznemirljivih značilnosti, ki jih R vključuje, so na razpolago kot opcijski moduli, ki jih lahko prenesemo na računalnik in namestimo. Te module imenujemo paketi.

Paketi so zbirke R-ovih funkcij, podatkov in zbranih kod v natančno opredeljenem formatu. Mapo, v kateri so na našem računalniku shranjeni paketi, imenujemo knjižnica (angl. *library*).

Potrebne pakete lahko poiščemo na CRAN-u (Comprehensive R Archive Network). Pakete najlažje namestimo neposredno iz CRAN-a preko menija Packages ⇒ Install package(s)... ali pa uporabimo funkcijo `install.packages()`.

Paket namestimo le enkrat. Vendar ker so tudi paketi pogosto posodobljeni s strani njihovih avtorjev, je potrebno tudi že nameščene pakete redno posodabljati. Za to lahko uporabimo ukaz `update.packages()`.

Ko imamo pakete nameščene v R, jih lahko uporabljamo tako, da jih naložimo v delovno okolje (angl. *workspace*). Naložimo jih lahko preko menija Packages ⇒ Load package... ali pa uporabimo funkciji `library()` ali `require()`.

1.3 OSNOVNE RAČUNSKE OPERACIJE V R

R lahko uporabljamo za izvajanje osnovnih računskih operacij (npr. namesto kalkulatorja). Osnovne operatorje prikazuje Tabela 3.

Tabela 3: Osnovni operatorji

Operator	Opis
+	seštevanje
-	odštevanje
*	množenje
/	deljenje
^ ali **	potenca
x%%y	ostanek pri deljenju
x%/y	celoštevski rezultat deljenja
sqrt()	kvadratni koren

2 PODATKOVNE STRUKTURE

Ko delamo s podatki, uporabljamo poseben pravokoten niz podatkov, v katerem vrstice predstavljajo opazovanja (angl. *observations*), stolpci pa predstavljajo spremenljivke (angl. *variables*). Takšnemu podatkovnemu nizu rečemo **nabor podatkov**.

Nabor podatkov definiramo s pomočjo objektov, na katerih kasneje izvajamo operacije in/ali funkcije. V programskem okolju R najdemo širok nabor objektov za shranjevanje podatkov.

Med najosnovnejše objekte spadajo:

- **Vektorji** (zaporedja elementov enake oblike),
- **Seznami** (podobni vektorjem, le da lahko vsebujejo elemente različnih oblik),
- **Matrike** (2-dimenzionalni pravokotni objekti z elementi enake oblike),
- **Podatkovni okviri** (podobni matrikam, le da lahko različni stolpci vključujejo različne tipe podatkov)

Objekti se med seboj razlikujejo glede na to, kakšen tip podatkov lahko zajemajo, kako so ustvarjeni, glede na njihovo strukturno kompleksnost ter uporabljeno notacijo, ki se uporablja za identifikacijo in priklic posameznih elementov.

2.1 VEKTORJI

Vektorje definiramo kot zaporedja elementov enake oblike (enakega tipa).

2.1.1 Vnos vektorjev v R

Vektor lahko v R vnesemo na različne načine:

- kot zaporedje s standardnim razkorakom 1:

`a:b`

Pri tem je `a` začetna vrednost zaporedja, `b` pa končna vrednost zaporedja.

Primer: `v1=1:10`

dobimo vektor `v1`, ki vključuje zaporedje števil od 1 do 10:

```
[1] 1 2 3 4 5 6 7 8 9 10
```

- kot zaporedje, v katerem določimo razkorak:

`seq(a,b,by=c)`

Pri tem je `a` začetna vrednost zaporedja, `b` končna vrednost zaporedja, ukaz `by=c` pa pove, da si številke v zaporedju med `a` in `b` sledijo s korakom `c`.

Primer: `v2=seq(1,10,by=0.5)`

dobimo vektor `v2`, ki vključuje zaporedje števil od 1 do 10, s korakom 0.5:

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0  
    6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
```

- kot zaporedje, v katerem določimo razkorak ter dolžino zaporedja:

`seq(a,by=c,length=d)`

Pri tem je `a` začetna vrednost zaporedja, ukaz `by=c` pove, da je korak med številkami zaporedja enak `c`, ukaz `length=d` pa določa, da naj bo zaporedje sestavljeno iz `d` zaporednih števil.

Primer: `v3=seq(1,by=0.5,length=10)`

dobimo vektor `v3`, ki vključuje zaporedje 10-ih števil, ki se začne s številko 1, vsaka zaporedna številka pa je za 0.5 večja od svoje prehodne številke:

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

- s pomočjo funkcije `c()` - concatenate:

Primer: `v4=c(2,4,3,1,5,7)`

dobimo vektor `v4`, ki vključuje navedene številke:

```
[1] 2 4 3 1 5 7
```

Vsi vektorji, prikazani v zgornjih primerih, so primeri številčnih vektorjev. Vektorji pa lahko vključujejo tudi drug tip elementov:

- Vektor znakov: `v5=c('ena','dve','tri')`

```
[1] "ena" "dve" "tri"
```
- Logični vektor: `v6=c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE,FALSE)`

```
[1] TRUE TRUE TRUE FALSE TRUE FALSE FALSE
```

Pomembno!

Čeprav so lahko elementi različnega tipa (številski, znakovni ali logični elementi), pa znotraj posameznega vektorja lahko uporabljamo samo en tip elementov (samo številke, samo znake ali samo logične izraze).

2.1.2 Priklic elementov vektorja

Posamezen element ali podmnožico elementov v vektorju lahko prikličemo s pomočjo numeričnega vektorja položajev, ki ga zapišemo znotraj oglatih oklepajev `[]`.

Primeri: `v4[2]`

nam vrne element, ki se nahaja na 2. mestu v vektorju `v4`:

```
[1] 4
```

```
v4[c(1,3,5)]
```

nam vrne 1., 3. in 5. element vektorja `v4`:

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

```
v4[2:5]
```

nam vrne 2., 3., 4. in 5. element vektorja `v4`:

```
[1] 4 3 1 5
```


2.2 SEZNAMI

Seznami so podobni vektorjem, vendar lahko vsebujejo elemente različnih oblik.

2.2.1 Vnos seznamov v R

Seznane v R vnesemo s pomočjo funkcije `list()`. Splošna oblika funkcije je naslednja:

```
list(object1, object2, ...)
```

pri čemer oznake `object1, object2, ...` predstavljajo objekte kateregakoli tipa, ki jih bomo imenovali tudi komponente seznama.

Primeri:

```
seznam1=list(c(1,3,5),c(TRUE,TRUE,FALSE,TRUE),c('ena','dve','tri'))
```

S tem smo sestavili seznam, poimenovan `seznam1`, ki je sestavljen iz treh objektov (komponent):

```
[[1]]
```

```
[1] 1 3 5
```

```
[[2]]
```

```
[1] TRUE TRUE FALSE TRUE
```

```
[[3]]
```

```
[1] "ena" "dve" "tri"
```

Opcijsko lahko vsak objekt (komponento) seznama tudi poimenujemo:

```
list(name1=object1, name2=object2,...)
```

Primer:

```
seznam2=list('A'=c(1,3,5), 'B'=c(TRUE,TRUE,FALSE,TRUE), 'C'=c('ena','dve','tri'))
```

S tem smo sestavili seznam, poimenovan `seznam2`, ki je sestavljen iz treh komponent, ki smo jih tudi poimenovali.

Imena posamezne komponente seznama lahko izpišemo s pomočjo funkcije `names()`.

Primer: `names(seznam2)`

```
[1] "A" "B" "C"
```

Rezultat je torej seznam, ki smo ga shranili kot `seznam2`:

\$A

```
[1] 1 3 5
```

\$B

```
[1] TRUE TRUE FALSE TRUE
```

\$C

```
[1] "ena" "dve" "tri"
```

Pomembno!

Čprav seznamami lahko vsebujejo elemente različnih oblik, pa mora biti tip elementov v posameznem objektu seznama enak.

2.2.2 Priklic komponent seznama

Posamezne komponente seznama lahko prikličemo na različne načine:

- Uporabimo lahko priklic s pomočjo položaja (zaporedne številke) komponente, ki jih zapišemo znotraj dvojnih oglatih oklepajev `[[]]`:

Primer: `seznam2[[2]]`

Izpiše 2. komponento seznama `seznam2`:

```
[1] TRUE TRUE FALSE TRUE
```

- Določimo lahko imena komponent znotraj dvojnih oglatih oklepajev `[[]]`:

Primer: `seznam2[['B']]`

Izpiše 2. komponento, imenovano 'B', seznama `seznam2`:

```
[1] TRUE TRUE FALSE TRUE
```

2.3 MATRIKE

Matrike so 2-dimenzionalni pravokotni objekti z elementi enake oblike.

2.3.1 Vnos matrik v R

Matrike v splošnem v R vnašamo s pomočjo funkcije `matrix()`. Splošna oblika funkcije je naslednja:

```
matrix(vector, nrow=a, ncol=b, byrow=TRUE, dimnames=list(char_vector_rownames, char_vector_colnames))
```

Pomen: `vector` vključuje elemente matrike,
`nrow` in `ncol` določata dimenzijo matrike (število vrstic in število stolpcev)
`byrow=TRUE` določa, da se elementi, zapisani v vektorju, izpišejo zaporedno po vrsticah; če imamo v vektorju zapisane elemente v vrstnem redu, kot naj se izpišejo po stolpcih vrstice, potem ta ukaz v funkciji uizpustimo.
`dimnames=list(char_vector_rownames, char_vector_colnames)`
pa določa imena dimenzij kot seznam, v katerega vključimo vektor z imeni vrstic ter vektor z imeni stolpcev.

Primeri: `M1=matrix(c(2,4,3,1,5,7), nrow=3, ncol=2)`

Dobimo matriko, dimenzije 3×2 (3 vrstice, 2 stolpca):

```
      [,1] [,2]  
[1,]    2    1  
[2,]    4    5  
[3,]    3    7
```

```
M2=matrix(c(2,4,3,1,5,7), nrow=3, ncol=2, byrow=TRUE)
```

Za razliko od prejšnjega primera, se bodo v tem primeru elementi, podani v vektorju, zapisali v vrstice (v prejšnjem primeru so elementi zapisani po stolpcih):

```
      [,1] [,2]  
[1,]    2    4  
[2,]    3    1  
[3,]    5    7
```

```
M3=matrix(c(2,4,3,1,5,7), nrow=3, ncol=2, byrow=TRUE, dimnames=list(c('V1', 'V2', 'V3'), c('S1', 'S2')))
```

V matriki M3 imamo dodana še imena za vrstice (V1, V2 in V3) ter stolpce (S1 in S2):

	S1	S2
V1	2	4
V2	3	1
V3	5	7

Vse argumente, ki jih uporabljamo v funkciji `matrix()`, lahko definiramo že prej:

```
celice=c(2,4,3,1,5,7)
imev=c('V1','V2','V3')
imes=c('S1','S2')
```

in v funkciji `matrix()` uporabimo kar njihova imena:

```
M4=matrix(celice,nrow=3,ncol=2,byrow=TRUE,dimnames=list(imev,imes))
```

Dobimo enako matriko kot v prejšnjem primeru:

	S1	S2
V1	2	4
V2	3	1
V3	5	7

2.3.2 Priklic elementov matrike

Posamezno vrstico, stolpec ali celico (element) v matriki lahko prikličemo s pomočjo indeksov položaja, ki jih zapišemo znotraj oglatih oklepajev `[]`.

`x[i,]` se nanaša na *i*-to vrstico v matriki *x*

`x[,j]` se nanaša na *j*-ti stolpec v matriki *x*

`x[i,j]` pa se nanaša na element v celici v *i*-ti vrstici in *j*-tem stolpcu matrike *x*.

Primeri: `M2[2,]`

nam vrne elemente 2. vrstice matrike M2:

```
[1] 3 1
```

`M2[,1]`

nam vrne elemente 1. stolpca matrike M2:

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

`M2[3,2]`

nam vrne element, ki se nahaja v 3. vrstici in 2. stolpcu matrike M2:

```
[1] 7
```

`M2[c(2,3),]`

nam vrne elemente, ki se nahajajo v 2. in 3. vrstici matrike M2 (ker smo »obdržali« oba stolpca, je rezultat nova matrika):

```

      [,1] [,2]
[1,]    3    1
[2,]    5    7

```

```
M2[c(2,3),1]
```

nam vrne elemente, ki se nahajajo v 1. stolpcu za 2. in 3. vrstico matrike `M2`:

```
[1] 3 5
```

Pomembno!

Čeprav so lahko elementi matrike različnega tipa (številski, znakovni ali logični elementi), pa znotraj ene same matrike lahko uporabljamo samo en tip elementov (samo številke, samo znake ali samo logične izraze).

2.4 PODATKOVNI OKVIRI

Podatkovni okviri so precej podobni matrikam. Glavna razlika med njima je predvsem ta, da lahko v podatkovnih okvirih različni stolpci vključujejo različne tipe podatkov.

2.4.1 Vnos podatkovnega okvira v R

Podatkovni okvir v R vnesemo s pomočjo funkcije `data.frame()`. Splošna oblika funkcije je naslednja:

```
data.frame(col1,col2,col3,...)
```

pri čemer oznake `col1,col2,col3,...` predstavljajo stolpčne vektorje kateregakoli tipa. Imena posameznega stolpca lahko izpišemo s pomočjo funkcije `names()`.

Primer: Ustvarimo podatkovni okvir, ki bo zajemal naslednje spremenljivke: ID, starost, spol, zaposlitveni status:

Najprej definiramo spremenljivke:

```

ID=1:4
starost=c(25,34,28,52)
spol=c('m','ž','m','ž')
status=c(TRUE,TRUE,FALSE,TRUE)

```

Nato pa s pomočjo definiranih spremenljivk ustvarimo podatkovni okvir:

```
podatki=data.frame(ID,starost,spol,status)
```

Rezultat je naslednji podatkovni okvir:

```

  ID starost spol status
1  1      25   m   TRUE
2  2      34   ž   TRUE
3  3      28   m FALSE
4  4      52   ž   TRUE

```

Pomembno!

Čeprav podatkovni okviri lahko vsebujejo elemente različnih oblik, pa morajo biti elementi v posameznem stolpcu podatkovnega okvira enakega tipa.

2.4.2 Priklic elementov podatkovnega okvira

Posamezne elemente podatkovnega okvira lahko prikličemo na različne načine:

- Uporabimo lahko priklic s pomočjo indeksov položaja, ki jih zapišemo znotraj oglatih oklepajev []:

Primer: `podatki[1:2]`

Ukaz nam izpiše prvi in drugi stolpec (1. in 2. spremenljivko) podatkovnega okvira:

	ID	starost
1	1	25
2	2	34
3	3	28
4	4	52

- Določimo lahko imena stolpcev (spremenljivk) znotraj oglatih oklepajev []:

Primer: `podatki['starost']`

Ukaz nam izpiše podatke za stolpec (spremenljivko) z imenom 'starost':

	starost
1	25
2	34
3	28
4	52

`podatki[c('ID', 'starost')]`

Ukaz nam izpiše podatke za stolpca (spremenljivki) 'ID' in 'starost':

	ID	starost
1	1	25
2	2	34
3	3	28
4	4	52

- Uporabimo znak \$, s katerim določimo spremenljivko v danem podatkovnem okviru:

Primer: `podatki$starost`

Ukaz izpiše podatke za spremenljivko 'starost':

`[1] 25 34 28 52`