LEPlmer2018: 1. Uvod u R

Dušica Filipović Đurđević

9 November 2018

> Svaki komentar počinje sa **#**. Sve što sledi nakon # neće biti izvršeno, biće ignorisano. Grupa linija se hešteguje tako što se taj tekst selektuje, pa se onda pritisne **Ctrl+Shft+C**.

> Komande je moguće kucati i direktno u R. Ipak, zbog budućeg prisećanja, bolje je kucati u skriptu, u R studiu, ili u običnom txt editoru (pa sačuvati txt)

> Obratite pažnju na mala i velika slova! Npr. Test nije isto što i test.

R je interpreter

To znači da nije potrebno kompajliranje koda. Ukucana komanda + Enter = izvršena komanda.

Kad se kuca u konzoli: ***Enter***.

Kad se kuca u R studio skriptu: ***Ctrl + Enter*** (da bi bilo izvršeno u konzoli).

R interpretira to što ste ukucali i prikazuje rezultat.

```
"Hello, world!"
```

```
## [1] "Hello, world!"
```

> Jedinica u uglastim zagradama, koja se nalazi pre ispisa pokazuje nam da R naš ispis tretira kao vektor, kao i da ispis počinje prvim elementom tog vektora, koji je, u ovom slučaju i jedini. Uskoro ćete saznati da je ovo vektor koji sadrži podatke tipa character (primetite navodnike).

Šta može R? Kakve su to komande?

R kao digitron

Operatori

Aritmetički operatori

AKA Osnovne računske operacije

```
2+3 # sabiranje
```

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

```
6-4 # oduzimanje
```

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

```
3*8 # množenje
```

```
## [1] 24
```

```
8/2 # deljenje
```

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

```
2^3 # podizanje na neki stepen
```

```
## [1] 8
```

```
sqrt(9) # kvadratni koren
```

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

```
abs(-9) # apsolutna vrednost
```

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

```
log(9) # logaritam
```

```
## [1] 2.197225
```

```
exp(9) # exponent
```

```
## [1] 8103.084
```

Naravno, postoje i mnoge druge…

Logički operatori

```
TRUE # istinito, tačno; PAŽNJA: OBAVEZNO VELIKA SLOVA
```

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

```
FALSE # neistinito, netačno; PAŽNJA: OBAVEZNO VELIKA SLOVA
```

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

```
1<2 # manje od
```

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

```
1<=2 # manje ili jednako
```

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

```
2>1 # veće od
```

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

```
2>=2 # veće ili jednako
```

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

```
2==2 # jedanko sa; PAŽNJA == NIJE ISTO ŠTO I =
```

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

```
2!=3 # nije jednako, različito od
```

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

```
!TRUE # negacija
```

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

```
1|2 # logičko ili
```

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

```
TRUE | FALSE
```

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

```
TRUE | TRUE
```

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

```
FALSE | FALSE
```

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

```
1 & 2 # logičko i
```

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

```
TRUE & FALSE
```

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

```
TRUE & TRUE
```

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

```
FALSE & FALSE
```

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

```
isTRUE(TRUE) # provera istinitosti
```

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

```
isTRUE(FALSE)
```

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

Do sada smo radili sa konkretnim vrednostima

R može da dodeli vrednosti varijablama

Dodeljivanje numeričke vrednosti

```
x <- 1  # x je dodeljena vrednost 1, ali to nije nigde prikazano
x # da bismo prikazali vrednost x
```

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

Ili ovako:

```
y <- 2 + 3
y
```

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

> umesto znaka <- može da se koristi i znak =, ali to nije “posh”

A onda nad x i y možemo da primenimo računske operacije

```
x+y # da dobijemo njihov zbir
```

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

```
z <- x+y # da z dodelimo vrednost zbira x i y
z # da se prikaže vrednost z
```

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

> Vrednosti varijabli koje smo napravili čuvaju se u radnom prostoru (workspace), kako bi moglo da im se kasnije pristupi po potrebi.

Da biste videli koje varijable ste do sada definisali, tj. smestili u radni prostor:

```
ls()
```

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

Da biste proverili kog je tipa neka varijabla:

```
class(x)
```

```
## [1] "numeric"
```

> Da biste znali koju operaciju možete da primenite na nekom podatku, morate znati kog je TIPA taj podatak

Tipovi podataka

- character (“a”, “Hello, world!”, “Jedva čekam da postanem programer!”)
- numeric (2, 2.5)
- integer (2, 3)
- logical (Boolean; TRUE, FALSE, NA)

> Za radoznale: postoje i tipovi podataka kao što su: double, complex…

Da biste proverili tip podatka:

```
class("a")
```

```
## [1] "character"
```

```
class(2.5)
```

```
## [1] "numeric"
```

```
class(2L) # ako želite da budete sigurni da je nešto integer dodate L
```

```
## [1] "integer"
```

```
class(TRUE)
```

```
## [1] "logical"
```

A može i ovako:

```
is.character("a")
```

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

```
is.numeric(2.5)
```

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

```
is.integer(2L)
```

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

```
is.integer(2.5)
```

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

> integer jeste i numeric, ali numeric ne mora da bude integer

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

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

Konverzija

Moguće je promeniti tip podatka

```
as.character(2.5)
```

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

```
as.numeric("2.5")
```

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

```
as.integer(2.5)
```

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

```
as.numeric(TRUE)
```

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

```
as.numeric(FALSE)
```

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

```
as.character(TRUE)
```

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

```
as.numeric("Jedva čekam da postanem programer!")
```

```
## Warning: NAs introduced by coercion
```

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

> Oprez, konverzija nije uvek moguća, postoji hijerarhija!

R operiše sa objektima

Vrste objekata

- vektori (nizovi podataka istog tipa)
- matrice (nizovi vektora; dvodimenzionanlni nizovi podataka smeštenih u redove i kolone)
- liste (nizovi podataka bilo kojih tipova, npr. pomešani brojevi i karakteri, čak i nizovi listi, kombinacija podataka i listi, čak i listi različitih dužina, matrica, data frames, bilo čega)
- data frames (nizovi vektora različitog tipa, ali iste dužine)
- funkcije (ugrađene i one koje sami pišemo)

> Tip podatka i karakteristike koje idu uz klasu objekta određuju vrstu operacije koju možemo nad njim da izvodimo, tj. koju funkciju možemo da primenimo.

Vektori

Ovako napravimo numerički vektor:

```
c(1,2,3,4,5,6,7,8,9,10) # c od combine
```

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

Može i ovako:

```
1:10 # niz od 1 do 10
```

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

ili ovako:

```
seq(1,10,2) # niz od 1 do 10, u koracima od po 2
```

```
## [1] 1 3 5 7 9
```

ili:

```
rep(1, 5) # broj 1 ponovljen pet puta
```

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

slično:

```
rep(1:10,2) # niz od 1 do 10 ponovljen dva puta
```

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

Međutim, može i ovako, vektor karaktera:

```
c("paprika", "cvekla", "luk", "grašak", "kupus")
```

```
## [1] "paprika" "cvekla"  "luk"     "grašak"  "kupus"
```

i vektor brojeva koji označavaju njihove cene:

```
c(50,30,15,45,25)
```

```
## [1] 50 30 15 45 25
```

Da bismo imali bolji uvid u to na šta se odnose koje cene, svakoj ceni dodelimo ime:

```
c("paprika"=50, "cvekla"=30, "luk"=15, "grašak"=45, "kupus"=25)
```

```
## paprika  cvekla     luk  grašak   kupus
##      50      30      15      45      25
```

Može i ovako:

```
cene <- c(50,30,15,45,25)
povrce <- c("paprika", "cvekla", "luk", "grašak", "kupus")
names(cene) <- povrce
cene
```

```
## paprika  cvekla     luk  grašak   kupus
##      50      30      15      45      25
```

Proverimo da li je vektor:

```
is.vector(cene)
```

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

Sveki ojekat ima svoju strukturu (atribute) koju možemo da proverimo:

```
str(cene)
```

```
##  Named num [1:5] 50 30 15 45 25
##  - attr(*, "names")= chr [1:5] "paprika" "cvekla" "luk" "grašak" ...
```

Ono što smo mi uradili koristeći names(cene), jeste dodeljivanje vrednosti vektora povrce odgovarajućim vrednostima atributa names.

Dužina vektora

Dužinu vektora određujemo na sledeći način:

```
length(cene)
```

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

Jedan podatak, tj. jedna vrednost je takođe vektor – vektor čija dužine 1:

```
is.vector(25)
```

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

```
length(25)
```

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

Pristupanje elementima vektora

Uz pomoć indeksa:

```
cene[1] # daje prvi element vektora cene; element sa indeksom 1
```

```
## paprika
##      50
```

> R brojač ne počinje od 0 (što je tipično za programske jezike), već od 1; dakle prvi element vektora ima indeks 1.

Pošto naš vektor cene ima i imena u atributu names, njegovim elementima možemo da pristupimo i preko imena:

```
cene["paprika"]
```

```
## paprika
##      50
```

Da bismo izvukli prva tri elementa:

```
cene[1:3]
```

```
## paprika  cvekla     luk
##      50      30      15
```

Da bismo izvukli drugi, treći i četvrti:

```
cene[2:5]
```

```
## cvekla    luk grašak  kupus
##     30     15     45     25
```

Isto to preko imena:

```
cene[c("paprika", "cvekla", "luk")]
```

```
## paprika  cvekla     luk
##      50      30      15
```

Obratite pažnju na redosled:

```
cene[c("luk", "cvekla")]
```

```
##    luk cvekla
##     15     30
```

```
cene[c(3,4)]
```

```
##    luk grašak
##     15     45
```

```
cene[c(4,3)]
```

```
## grašak    luk
##     45     15
```

Da izostavimo element numeričkog vektora, odnosno izvučemo sve osim nekih:

```
cene[-1] # svi osim prvog
```

```
## cvekla    luk grašak  kupus
##     30     15     45     25
```

```
cene[c(-3,-4)] # svi osim trećeg i četvrtog
```

```
## paprika  cvekla   kupus
##      50      30      25
```

Vrednosti vektora možemo da sortiramo:

```
sort(cene) # u uzlaznom nizu; decreasing = FALSE je default opcija
```

```
##     luk   kupus  cvekla  grašak paprika
##      15      25      30      45      50
```

```
sort(cene, decreasing = TRUE) # u silaznom nizu
```

```
## paprika  grašak  cvekla   kupus     luk
##      50      45      30      25      15
```

Vrednostima vektora možemo da dodelimo rangove:

```
order(cene) # uzlazno
```

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

```
order(cene, decreasing = TRUE) # sizazno
```

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

Matrice

Ovako napravimo numeričku matricu:

```
matrix(1:6, nrow=2) # pravi matricu od šest elemenata (brojevi od 1 do 6) raspoređenih u dva reda i neminovno tri kolone.
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
matrix(1:6, nrow=2, ncol=3) # može i eksplicitno da se navede
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
matrix(1:6, nrow=3) # pravi matricu od šest elemenata (brojevi od 1 do 6) raspoređenih u tri reda i neminovno dve kolone.
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
matrix(1:6, nrow=3, byrow=TRUE) # matrica slična prethodnoj, ali brojevi rastu unutar reda, a ne unutar kolone
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

Može i ovako:

```
cbind(1:3,4:6) # "zalepi" dva vektora kao dve kolone matrice
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

Ili ovako:

```
rbind(1:3,4:6) # "zalepi" dva vektora kao dva reda matrice
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

Na sličan način, postojeća matrica može da se proširi za jedan red ili za jednu kolonu:

```
M <- cbind(1:3,4:6) # sačuvamo matricu u promenljivu M
cbind(M,7:9) # proširimo matricu M ja još jednu kolonu koja sadrži vektor brojeva od 7 do 9
```

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

```
rbind(M,c(3.5,6.5)) # proširimo matricu M za još jedan red koji sadrži vektor brojeva 3.5 i 6.5
```

```
##      [,1] [,2]
## [1,]  1.0  4.0
## [2,]  2.0  5.0
## [3,]  3.0  6.0
## [4,]  3.5  6.5
```

> Primetite da su svi elementi sada u formi decimalnih brojeva (nummeric). To je zbog toga što vektor može da sadrži samo jedan tip podataka, pa R automatski konverziju koju može bezbedno da izvrši, u ovom slučaju integer u numeric.

Na sličan način:

```
rbind(M,c("a","b")) # proširimo matricu M za još jedan red koji sadrži vektor karaktera a i b
```

```
##      [,1] [,2]
## [1,] "1"  "4"
## [2,] "2"  "5"
## [3,] "3"  "6"
## [4,] "a"  "b"
```

> Ovoga puta izvršena je konverzija integer u character

Možemo dati nazive kolonama:

```
colnames(M) <- c("Prva kolona", "Druga kolona")
M
```

```
##      Prva kolona Druga kolona
## [1,]           1            4
## [2,]           2            5
## [3,]           3            6
```

```
rownames(M) <- c("Prvi red", "Drugi red", "Treći red")
M
```

```
##           Prva kolona Druga kolona
## Prvi red            1            4
## Drugi red           2            5
## Treci red           3            6
```

Možemo pristupiti pojedinačnom elementu matrice:

```
M[3,2] # u uglastim zagradama navedemo prvo indeks za red, a potom indeks za kolonu
```

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

Isto možemo postići i koristeći imena (ako smo ih dodelili)

```
M["Treći red", "Druga kolona"]
```

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

Ili kombinaciju:

```
M[3, "Druga kolona"]
```

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

Možemo pristupiti i čitavom redu matrice:

```
M[3,] # navedemo red, izostavimo indeks kolone; PAŽNJA, ZAREZ OBAVEZNO OSTAVITI!
```

```
##  Prva kolona Druga kolona
##            3            6
```

Možemo pristupiti i čitavoj koloni matrice:

```
M[,2] # navedemo red, izostavim indeks kolone; ZAREZ OBAVEZAN!
```

```
##  Prvi red Drugi red Treci red
##         4         5         6
```

Možemo pristupiti i matrici unutar matrice

```
M[2:3, 1] # drugi i treći red za prvu kolonu
```

```
## Drugi red Treci red
##         2         3
```

```
M[2:3, 1:2] # drugi i treći red za prvu i drugu kolonu
```

```
##           Prva kolona Druga kolona
## Drugi red           2            5
## Treci red           3            6
```

Liste

Da bismo formirali listu:

```
list("Petar Petrović", "PS18/0056", 89, 9) # formiranje liste sa četiri elementa: ime studenta (charachter), broj indeksa (character), poeni na ispitu (numeric),  ocena na ispitu (numeric)
```

```
## [[1]]
## [1] "Petar Petrovic"
##
## [[2]]
## [1] "PS18/0056"
##
## [[3]]
## [1] 89
##
## [[4]]
## [1] 9
```

Dobijamo nešto komplikovaniju strukturu (vratićemo se na ovo). Zar nije jednostavnije da napravimo vektor? Da smo pokušali da napravimo vektor sa ovim podacima, desila bi se konverzija u charachter (izgubili bismo mogućnost da primenimo računske operacije nad brojevima, brojevi bi postali običan tekst):

```
c("Petar Petrović", 1, "PS18/0056", 9)
```

```
## [1] "Petar Petrovic" "1"              "PS18/0056"      "9"
```

Dakle, komplikacija je opravdana, da vidimo u čemu se sastoji.

```
list("Petar Petrović", "PS18/0056", 89, 9)
```

```
## [[1]]
## [1] "Petar Petrovic"
##
## [[2]]
## [1] "PS18/0056"
##
## [[3]]
## [1] 89
##
## [[4]]
## [1] 9
```

Ovaj ispis se razlikuje od ispisa koje smo ranije dobijali. Vidimo da ova lista ima četiri elementa, i to četiri vektora sa jednim elementom (čija je dužina 1).

Možemo svakom elementu liste dodeliti ime:

```
list (Ime = "Petar Petrović", Indeks = "PS18/0056", Poeni = 89, Ocena = 9)
```

```
## $Ime
## [1] "Petar Petrovic"
##
## $Indeks
## [1] "PS18/0056"
##
## $Poeni
## [1] 89
##
## $Ocena
## [1] 9
```

A može i ovako:

```
student <- list("Petar Petrović", "PS18/0056", 89, 9) # smestimo listu u promenljivu student1
names (student) <- c("Ime", "Indeks", "Poeni", "Ocena") # u atribut liste student, u atribut names, smestimo vekto
r karaktera
student # prikaz liste
```

```
## $Ime
## [1] "Petar Petrovic"
##
## $Indeks
## [1] "PS18/0056"
##
## $Poeni
## [1] 89
##
## $Ocena
## [1] 9
```

```
str(student) # da vidimo strukturu ovog objekta
```

```
## List of 4
##  $ Ime   : chr "Petar Petrovic"
##  $ Indeks: chr "PS18/0056"
##  $ Poeni : num 89
##  $ Ocena : num 9
```

Element liste može da bude i druga lista. Na primer, podaci o studentu sa kojim naš student radi seminarski rad.

```
student_saradnik <- list(Ime = "Jovan Jovanović", Ocena = 6) # napravimo listu sa podacima o tom, drugom studentu
student <- list (Ime = "Petar Petrović", Indeks = "PS18/0056", Poeni = 89, Ocena = 9, Saradnik = student_saradnik)
# u listu student dodamo još jedan element -- listu student_saradnik
student
```

```
## $Ime
## [1] "Petar Petrovic"
##
## $Indeks
## [1] "PS18/0056"
##
## $Poeni
## [1] 89
##
## $Ocena
## [1] 9
##
## $Saradnik
## $Saradnik$Ime
## [1] "Jovan Jovanovic"
##
## $Saradnik$Ocena
## [1] 6
```

I tako u nedogled…

Međutim, ne treba preterivati, jer je pristupanje elementima liste nešto komplikovanije, a komplikuje se dodatno ako smeštamo listu unutar liste, unutar liste…

Pristupanje podacima liste zahteva dodatnu pažnju. Elementi liste su takođe liste, pa ako tražimo pristup prvom elementu, dobićemo listu:

```
student[1]
```

```
## $Ime
## [1] "Petar Petrovic"
```

Slično:

```
student["Ime"]
```

```
## $Ime
## [1] "Petar Petrovic"
```

Da izdvojimo dva elementa liste:

```
student[c(1,2)]
```

```
## $Ime
## [1] "Petar Petrovic"
##
## $Indeks
## [1] "PS18/0056"
```

Ili:

```
student[c("Ime","Indeks")]
```

```
## $Ime
## [1] "Petar Petrovic"
##
## $Indeks
## [1] "PS18/0056"
```

Ako želimo da ispišemo ime studenta, treba da pristupimo prvom elementu prvog elementa liste:

```
student[[1]]
```

```
## [1] "Petar Petrovic"
```

Ili:

```
student[["Ime"]]
```

```
## [1] "Petar Petrovic"
```

Ako je su elementi liste imenovani, umesto dvostrukih uglastih zagrada, možemo koristiti znak$:

```
student$Ime
```

```
## [1] "Petar Petrovic"
```

Koristeći dvostruke uglaste zagrade ili znak $ možemo i dodavati nove elemente u listu, npr. poene na različitim predispitnim obavezama:

```
pred <- c(Aktivnost = 12, Seminarski = 17, Kolokvijum = 23) # napravimo vektor sa predispitnim poenima
student$Predispitne <- pred # smestimo taj vektor u listu student, kao dodatni element te liste
student
```

```
## $Ime
## [1] "Petar Petrovic"
##
## $Indeks
## [1] "PS18/0056"
##
## $Poeni
## [1] 89
##
## $Ocena
## [1] 9
##
## $Saradnik
## $Saradnik$Ime
## [1] "Jovan Jovanovic"
##
## $Saradnik$Ocena
## [1] 6
##
##
## $Predispitne
##  Aktivnost Seminarski Kolokvijum
##         12         17         23
```

Da pristupimo imenu studenta saradnika, treba da pristupimo prvom elementu petog elementa liste:

```
student[[5]][[1]]
```

```
## [1] "Jovan Jovanovic"
```

Može i ovako:

```
student[[c(5,1)]]
```

```
## [1] "Jovan Jovanovic"
```

Ili ovako:

```
student$Saradnik$Ime
```

```
## [1] "Jovan Jovanovic"
```

A možemo i dodati novi element u podlistu Saradnik:

```
pred_sar <- c(Aktivnost = 12, Seminarski = 11, Kolokvijum = 17)
student$Saradnik$Predispitne <- pred_sar
student
```

```
## $Ime
## [1] "Petar Petrovic"
##
## $Indeks
## [1] "PS18/0056"
##
## $Poeni
## [1] 89
##
## $Ocena
## [1] 9
##
## $Saradnik
## $Saradnik$Ime
## [1] "Jovan Jovanovic"
##
## $Saradnik$Ocena
## [1] 6
##
## $Saradnik$Predispitne
##  Aktivnost Seminarski Kolokvijum
##         12         11         17
##
##
## $Predispitne
##  Aktivnost Seminarski Kolokvijum
##         12         17         23
```

Liste su najneurednija struktura podataka

Data frames

Upravo struktura podataka koju ćemo najčešće koristiti – ono što zovemo žargonski matrica za analizu podataka. Ipak, formalno, u R-u, to nije matrica, jer matrica sadrži podatke istog tipa (npr. samo brojeve, ili samo karaktere). U našoj bazi podataka obično imamo i numeričke podatke (npr. RT, uzrast…) i karaktere (npr. šifra ispitanika, nivoi faktora…). Data frame je sličan matrici, jer sadrži podatke organizovane u redove i kolone (svaki sa jednakim brojem elemenata), a sličan listi, jer sadrži podatke različitih tipova. Ipak, podaci unutar kolone moraju biti istog tipa (što je obično slučaj, jer kolone predstavljaju naše varijable). Dodatno, za razliku od liste, sve kolone u matrici moraju biti iste dužine. Konačno, podacima koji su smešteni u data frame mnogo je lakše pristupati nego podacima u listi.

Da napravimo data frame:

```
Isp <- c("s1", "s2", "s3", "s4") # napravimo vektor sa šiframa ispitanika
Manip <- c("kontrolna grupa", "eksperimentalna grupa", "kontrolna grupa", "eksperimentalna grupa") # napravimo vek
tor sa podatkom o tome kojoj grupi je pripadao ispitanik
Repr <- c(7, 15, 5, 11) # napravimo vektor sa podacima o broju tačno reprodukovanih stimulusa
moji_podaci <- data.frame(Isp, Manip, Repr) # napravimo data frame moji_podaci koji za kolone ima vektore koje smo
napravili
moji_podaci # da prikažemo naš data frame
```

```
##   Isp                Manip Repr
## 1  s1       kontrolna grupa    7
## 2  s2 eksperimentalna grupa   15
## 3  s3       kontrolna grupa    5
## 4  s4 eksperimentalna grupa   11
```

Možemo i da promenimo imena:

```
moji_podaci <- data.frame(Participant = Isp, Manipulation = Manip, Recall = Repr)
moji_podaci
```

```
##   Participant          Manipulation Recall
## 1          s1       kontrolna grupa      7
## 2          s2 eksperimentalna grupa     15
## 3          s3       kontrolna grupa      5
## 4          s4 eksperimentalna grupa     11
```

Ili ovako (opet menjamo imena):

```
names(moji_podaci) <- c("Ispitanik", "Manipulacija", "Reprodukcija")
moji_podaci
```

```
##   Ispitanik          Manipulacija Reprodukcija
## 1        s1       kontrolna grupa            7
## 2        s2 eksperimentalna grupa           15
## 3        s3       kontrolna grupa            5
## 4        s4 eksperimentalna grupa           11
```

Da ispitamo strukturu svog data frame:

```
str(moji_podaci)
```

```
## 'data.frame':    4 obs. of  3 variables:
##  $ Ispitanik   : Factor w/ 4 levels "s1","s2","s3",..: 1 2 3 4
##  $ Manipulacija: Factor w/ 2 levels "eksperimentalna grupa",..: 2 1 2 1
##  $ Reprodukcija: num  7 15 5 11
```

Vidimo da je data frame zapravo lista, ali lista sa vektorima istih dužina!

> Usputna napomena: R vrednosti kolone data frame-a koja sadrži karaktere automatski konvertuje u tzv. faktore (ono što mi zovemo kategoričkim varijablama). Ako iz nekog razloga želite to da izbegnemte, onda kucate: moji_podaci <- data.frame(Ispitanik, Manipulacija, Reprodukcija, stringsAsFactors = FALSE)

Pristupanje elementima data frame-a je jednostavno i slično pristupanju elementima matrice.

Da pristupimo pojedinačnom elementu:

```
moji_podaci[3, 2] # da vidimo kojoj grupi je pripadao ispitanik u trećem redu (s3)
```

```
## [1] kontrolna grupa
## Levels: eksperimentalna grupa kontrolna grupa
```

Isto to na malo drugačiji način:

```
moji_podaci[3, "Manipulacija"]
```

```
## [1] kontrolna grupa
## Levels: eksperimentalna grupa kontrolna grupa
```

Možemo da izvučemo čitav red:

```
moji_podaci[3, ]
```

```
##   Ispitanik    Manipulacija Reprodukcija
## 3        s3 kontrolna grupa            5
```

Možemo da izvučemo čitavu kolonu:

```
moji_podaci[ , 2]
```

```
## [1] kontrolna grupa       eksperimentalna grupa kontrolna grupa
## [4] eksperimentalna grupa
## Levels: eksperimentalna grupa kontrolna grupa
```

ili

```
moji_podaci[ ,"Manipulacija"]
```

```
## [1] kontrolna grupa       eksperimentalna grupa kontrolna grupa
## [4] eksperimentalna grupa
## Levels: eksperimentalna grupa kontrolna grupa
```

Možemo izvući i deo dejta frejma koji sadrži nekoliko redova i nekoliko kolona, a koji i dalje predstavlja data frame.

```
moji_podaci[c(2,4), c("Ispitanik", "Reprodukcija")]
```

```
##   Ispitanik Reprodukcija
## 2        s2           15
## 4        s4           11
```

> Ne zaboravite: data frame je nešto poput liste kolona

Stoga možemo da primenimo strategiju za pristupanje koju smo primenjivali na liste.

Da pristupimo vektoru koji čini treću kolonu:

```
moji_podaci[[3]]
```

```
## [1]  7 15  5 11
```

Ili ovako:

```
moji_podaci[["Reprodukcija"]]
```

```
## [1]  7 15  5 11
```

Ili ovako:

```
moji_podaci$Reprodukcija
```

```
## [1]  7 15  5 11
```

Ako primenimo jednostruke uglaste zagrade, nećemo dobiti vektor, već ponovo data frame, tj. listu koja sadrži samo traženu kolonu:

```
moji_podaci[3]
```

```
##   Reprodukcija
## 1            7
## 2           15
## 3            5
## 4           11
```

```
moji_podaci["Reprodukcija"]
```

```
##   Reprodukcija
## 1            7
## 2           15
## 3            5
## 4           11
```

> Dakle, pažljivo: struktura koju ćemo dobiti zavisi od primenjene strategije izvlačenja.

Podskup data frame-a možemo da izvučemo i tako što ćemo sami postaviti filter.

Filter na osnovu vrednosti neke varijable postavimo ovako:

```
moji_podaci[moji_podaci$Reprodukcija>10,] # tražimo podskup data frame-a koji sadrži samo one redove za koje je vr
ednost u koloni Reprodukcija veća od 10 (prvi element unutar uglaste zagrade koji definiše redove) i želimo sve ko
lone (drugi element unutar uglaste zagrade, onaj iza zareza, onaj koji definiše kolone je izostavljen, što znači d
a zadržavamo sve kolone)
```

```
##   Ispitanik          Manipulacija Reprodukcija
## 2        s2 eksperimentalna grupa           15
## 4        s4 eksperimentalna grupa           11
```

Ili ovako:

```
moji_podaci[1:3,] # tražimo podskup data frame-a koji sadrži samo prva tri reda i sve kolone
```

```
##   Ispitanik          Manipulacija Reprodukcija
## 1        s1       kontrolna grupa            7
## 2        s2 eksperimentalna grupa           15
## 3        s3       kontrolna grupa            5
```

Ili ovako:

```
moji_podaci[c(1,4),] # tražimo podskup data frame-a koji sadrži samo prvi i četvrti red i sve kolone
```

```
##   Ispitanik          Manipulacija Reprodukcija
## 1        s1       kontrolna grupa            7
## 4        s4 eksperimentalna grupa           11
```

Ili ovako:

```
moji_podaci[moji_podaci$Manipulacija=="kontrolna grupa",] # tražimo podskup data frame-a koji sadrži samo ispitani
ke iz kontrolne grupe i sve kolone
```

```
##   Ispitanik    Manipulacija Reprodukcija
## 1        s1 kontrolna grupa            7
## 3        s3 kontrolna grupa            5
```

Ovaj postupak je ekvivalentan sledećem:

```
subset(moji_podaci, Manipulacija=="kontrolna grupa") # tražimo podskup data frame-a koji sadrži samo ispitanike iz
kontrolne grupe i sve kolone
```

```
##   Ispitanik    Manipulacija Reprodukcija
## 1        s1 kontrolna grupa            7
## 3        s3 kontrolna grupa            5
```

Možemo i da ukrstimo kriterijume za filtriranje:

```
moji_podaci[moji_podaci$Manipulacija=="kontrolna grupa" & moji_podaci$Reprodukcija>5,] # tražimo podskup data fram
e-a koji sadrži samo ispitanike iz kontrolne grupe za koje je Reprodukcija veća od 5, zadržavamo sve kolone
```

```
##   Ispitanik    Manipulacija Reprodukcija
## 1        s1 kontrolna grupa            7
```

Možemo da biramo i kolone:

```
moji_podaci[moji_podaci$Manipulacija=="kontrolna grupa", 1:3] # tražimo podskup data frame-a koji sadrži samo ispi
tanike iz kontrolne grupe i prve tri kolone
```

```
##   Ispitanik    Manipulacija Reprodukcija
## 1        s1 kontrolna grupa            7
## 3        s3 kontrolna grupa            5
```

Isto to drugačije:

```
moji_podaci[moji_podaci$Manipulacija=="kontrolna grupa", c(1,2,3)] # tražimo podskup data frame-a koji sadrži samo
ispitanike iz kontrolne grupe i prve tri kolone
```

```
##   Ispitanik    Manipulacija Reprodukcija
## 1        s1 kontrolna grupa            7
## 3        s3 kontrolna grupa            5
```

Isto to drugačije:

```
moji_podaci[moji_podaci$Manipulacija=="kontrolna grupa", c("Ispitanik","Manipulacija","Reprodukcija")] # tražimo p
odskup data frame-a koji sadrži samo ispitanike iz kontrolne grupe i prve tri kolone
```

```
##   Ispitanik    Manipulacija Reprodukcija
## 1        s1 kontrolna grupa            7
## 3        s3 kontrolna grupa            5
```

> Kad savladate ove bazične operacije, možete ih ukrštati međusobno. Nebo je granica!

Možemo dodati kolonu u data frame:

```
iq <- c(101, 124, 110, 98) # napravimo vektor sa vrednostima koje želimo da smestimo u novu kolonu
moji_podaci$IQ <- iq # smestimo vektor iq u kolonu koja će se zvati IQ; isti efekat bismo postigli i sa moji_podac
i[["IQ"]] <- iq
moji_podaci
```

```
##   Ispitanik          Manipulacija Reprodukcija  IQ
## 1        s1       kontrolna grupa            7 101
## 2        s2 eksperimentalna grupa           15 124
## 3        s3       kontrolna grupa            5 110
## 4        s4 eksperimentalna grupa           11  98
```

Kolonu možemo dodati i na ovaj način:

```
Rukost <- c("desnoruk", "levoruk", "levoruk", "desnoruk") # napravimo vektor sa vrednostima koje želimo da stavimo
u novu kolonu
moji_podaci <- cbind(moji_podaci, Rukost)
moji_podaci
```

```
##   Ispitanik          Manipulacija Reprodukcija  IQ   Rukost
## 1        s1       kontrolna grupa            7 101 desnoruk
## 2        s2 eksperimentalna grupa           15 124  levoruk
## 3        s3       kontrolna grupa            5 110  levoruk
## 4        s4 eksperimentalna grupa           11  98 desnoruk
```

Možemo dodati i novi red u data frame, ali tako što ćemo napraviti novi data frame sa jednim elementom (ili više) i moramo imenovati elemente:

```
s5<- data.frame(Ispitanik = "s5", Manipulacija = "kontrolna grupa", Reprodukcija = 10, IQ = 103, Rukost = "desnoru
k") # napravimo novi data frame i imenujemo elemente
moji_podaci <- rbind(moji_podaci, s5) # spojimo stari i novi data frame
moji_podaci
```

```
##   Ispitanik          Manipulacija Reprodukcija  IQ   Rukost
## 1        s1       kontrolna grupa            7 101 desnoruk
## 2        s2 eksperimentalna grupa           15 124  levoruk
## 3        s3       kontrolna grupa            5 110  levoruk
## 4        s4 eksperimentalna grupa           11  98 desnoruk
## 5        s5       kontrolna grupa           10 103 desnoruk
```

Možemo da napravimo novi data frame, pa da ga spojimo sa postojećim:

```
Isp_2 <- c("s6", "s7", "s8", "s9") # napravimo vektor sa šiframa ispitanika
Manip_2 <- c("kontrolna grupa", "eksperimentalna grupa", "kontrolna grupa", "eksperimentalna grupa") # napravimo v
ektor sa podatkom o tome kojoj grupi je pripadao ispitanik
Repr_2 <- c(8, 14, 6, 18) # napravimo vektor sa podacima o broju tačno reprodukovanih stimulusa
IQ_2 <- c(99, 123, 109, 102)
Rukost_2 <- c("desnoruk", "levoruk", "levoruk", "desnoruk")
moji_podaci_2 <- data.frame(Isp_2, Manip_2, Repr_2, IQ_2, Rukost_2) # napravimo novi data frame moji_podaci_2 koji
za kolone ima vektore koje smo napravili, a koji ćemo kasnije dodatni na postojeći
moji_podaci_2 # da prikažemo novi data frame
```

```
##   Isp_2              Manip_2 Repr_2 IQ_2 Rukost_2
## 1    s6       kontrolna grupa      8   99 desnoruk
## 2    s7 eksperimentalna grupa     14  123  levoruk
## 3    s8       kontrolna grupa      6  109  levoruk
## 4    s9 eksperimentalna grupa     18  102 desnoruk
```

```
names(moji_podaci_2) <- c("Ispitanik", "Manipulacija", "Reprodukcija", "IQ", "Rukost") # da bismo spojili dva data
frame-a, moramo u novom dati imena kolonama i to tako da budu ista kao u postojećem
moji_podaci_2 # sad se vide i imena kolona u novomo data frame-u
```

```
##   Ispitanik          Manipulacija Reprodukcija  IQ   Rukost
## 1        s6       kontrolna grupa            8  99 desnoruk
## 2        s7 eksperimentalna grupa           14 123  levoruk
## 3        s8       kontrolna grupa            6 109  levoruk
## 4        s9 eksperimentalna grupa           18 102 desnoruk
```

```
moji_podaci <- rbind(moji_podaci, moji_podaci_2) # sad spojimo stari i novi data frame
moji_podaci # da vidimo kako sad izgleda naš data frame
```

```
##   Ispitanik          Manipulacija Reprodukcija  IQ   Rukost
## 1        s1       kontrolna grupa            7 101 desnoruk
## 2        s2 eksperimentalna grupa           15 124  levoruk
## 3        s3       kontrolna grupa            5 110  levoruk
## 4        s4 eksperimentalna grupa           11  98 desnoruk
## 5        s5       kontrolna grupa           10 103 desnoruk
## 6        s6       kontrolna grupa            8  99 desnoruk
## 7        s7 eksperimentalna grupa           14 123  levoruk
## 8        s8       kontrolna grupa            6 109  levoruk
## 9        s9 eksperimentalna grupa           18 102 desnoruk
```

PRIMETITE: Nakon spajanja dva data frame-a, nismo promenili ime našeg prvobitnog data frame-a. Ako želite da sačuvate originalni, kraći data frame, možete da promenite naziv. Međutim, ne morate, jer nakon ove komande, R tretira data frame moji_podaci kao veliku, spojenu bazu (stara, manja verzija je “prebrisana”). Na sličan način možete da menjate i sadržaj kolona, tj. varijabli, a da ne pravite nove kolone, već zadržite postojeću. Npr. možete da kucate:

```
moji_podaci$Reprodukcija <- log(moji_podaci$Reprodukcija)
```

i primetićete da su vrednosti u koloni Reprodukcija zamenjeni odgovarajućim logaritmovanim vrednostima. Da vratite na staro, kucajte:

```
moji_podaci$Reprodukcija <- exp(moji_podaci$Reprodukcija)
```

i dobićete one prvobitne vrednosti).

Pored ovog uzorkovanja i proširivanja data frame-a, nad podacima unutar njega možemo izvoditi i brojne druge manipulacije

Data frame možemo sortirati po vrednostima neke kolone u uzlaznom ili sizaznom redosledu:

```
moji_podaci[order(moji_podaci$Reprodukcija),] # tražimo data frame moji_podaci, ali tako da budu uključeni oni redovi koji se dobijaju pomoću order vektora kojoi sadrži vrednosti reprodukcije i sadrži sve kolone (jer u polju koje označava kolone nema restrikcije)
```

```
##   Ispitanik          Manipulacija Reprodukcija  IQ   Rukost
## 3        s3       kontrolna grupa            5 110  levoruk
## 8        s8       kontrolna grupa            6 109  levoruk
## 1        s1       kontrolna grupa            7 101 desnoruk
## 6        s6       kontrolna grupa            8  99 desnoruk
## 5        s5       kontrolna grupa           10 103 desnoruk
## 4        s4 eksperimentalna grupa           11  98 desnoruk
## 7        s7 eksperimentalna grupa           14 123  levoruk
## 2        s2 eksperimentalna grupa           15 124  levoruk
## 9        s9 eksperimentalna grupa           18 102 desnoruk
```

Pristupanje nivoima faktora:

```
levels(moji_podaci$Manipulacija)
```

```
## [1] "eksperimentalna grupa" "kontrolna grupa"
```

Krostabulacija (kontingencijske tabele):

```
xtabs(~Manipulacija + Rukost, data= moji_podaci) # znak ~ ima značenje "zavisi od", "u funkciji od"
```

```
##                       Rukost
## Manipulacija           desnoruk levoruk
##   eksperimentalna grupa        2       2
##   kontrolna grupa              3       2
```

Možemo da sumiramo vrednosti:

```
sum(moji_podaci$Reprodukcija)
```

```
## [1] 94
```

Izračunamo prosek:

```
mean(moji_podaci$Reprodukcija)
```

```
## [1] 10.44444
```

Izračunamo Standardnu devijaciju:

```
sd(moji_podaci$Reprodukcija)
```

```
## [1] 4.447221
```

Možemo da izračunamo aritmetičku sredinu odvojeno za nivoe nekog faktora, odnosno različite vrednosti neke kategoričke varijable:

```
mean(moji_podaci[moji_podaci$Manipulacija=="kontrolna grupa",]$Reprodukcija) # računamo prosek za vrednosti varijable Reprodukcija, ali prethodno uzorkujemo data frame tako da izdvojimo samo one redove za koje varijabla Manipulacija ima nivo kontrolna grupa (uz zadržavanje svih kolona, otud prazno mesto iza zareza: NE ZABORVITE ZAREZ)
```

```
## [1] 7.2
```

```
mean(moji_podaci[moji_podaci$Manipulacija=="eksperimentalna grupa",]$Reprodukcija) # isto to za eksperimentalnu grupu
```

```
## [1] 14.5
```

A može i ovako:

```
tapply(moji_podaci$Reprodukcija, moji_podaci$Manipulacija, mean) # prvi argument funkcije tapply() je vektor za ko
ji treba računati nešto, drugi argument je vektor po čijim vrednostima treba izvršiti podelu, a treći argument je
funkcija koju treba primeniti na podatke iz prvog argumenta. U ovom slučaju je mean, ali može da bude i nešto drug
o (sum(), sd(), sqrt(), abs(), itd.)
```

```
## eksperimentalna grupa       kontrolna grupa
##                  14.5                   7.2
```

Postoji i kraća verzija istog postupka:

```
with(moji_podaci, tapply(Reprodukcija, Manipulacija, mean)) # prvi argument funkcije with() definiše data frame na
d kojim treba nešto raditi, drugi definiše ono što radimo nad tim data frame-om
```

```
## eksperimentalna grupa       kontrolna grupa
##                  14.5                   7.2
```

Da razbijemo dosadu možemo da primenimo t test:

```
t.test(Reprodukcija ~ Manipulacija, data=moji_podaci)
```

```
##
##  Welch Two Sample t-test
##
## data:  Reprodukcija by Manipulacija
## t = 4.3445, df = 5.0334, p-value = 0.007285
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   2.989318 11.610682
## sample estimates:
## mean in group eksperimentalna grupa       mean in group kontrolna grupa
##                                14.5                                 7.2
```

Možemo računati proseke i ukrštajući nivoe faktora:

```
with(moji_podaci, tapply(Reprodukcija, list(Manipulacija, Rukost), mean))
```

```
##                       desnoruk levoruk
## eksperimentalna grupa 14.500000    14.5
## kontrolna grupa        8.333333     5.5
```

Rad sa fajlovima

Data frame možemo sačuvati na disku (van R-a):

```
write.table(moji_podaci, file = "Podaci_vezba.txt", append = FALSE, sep = "\t",eol="\n",col.names=TRUE,row.names=F
ALSE)
```

U Windows okruženju, ako se ne specifikuje putanja, R pristupa automatski direktorijumu Documents. U mom slučaju: c: Ako želimo da smestimo fajl na drugu lokaciju, potrebno je da specifikujemo putanju:

```
write.table(moji_podaci, file = "c:/Sam/Dokumenti/Obuke/LEPlmer2018/Podaci_vezba.txt", append = FALSE, sep = "\t",
eol="\n",col.names=TRUE,row.names=FALSE)
```

> Primetite da se u R-u, u Windows okruženju umesto koristi znak /

Kasnije ih možemo otvarati (smeštati u radni prostor R-a). Da otvorimo data frame iz Documents direktorijuma:

```
read.table("Podaci_vezba.txt",sep="\t", header=TRUE)
```

```
##   Ispitanik          Manipulacija Reprodukcija  IQ   Rukost
## 1        s1       kontrolna grupa            7 101 desnoruk
## 2        s2 eksperimentalna grupa           15 124  levoruk
## 3        s3       kontrolna grupa            5 110  levoruk
## 4        s4 eksperimentalna grupa           11  98 desnoruk
## 5        s5       kontrolna grupa           10 103 desnoruk
## 6        s6       kontrolna grupa            8  99 desnoruk
## 7        s7 eksperimentalna grupa           14 123  levoruk
## 8        s8       kontrolna grupa            6 109  levoruk
## 9        s9 eksperimentalna grupa           18 102 desnoruk
```

Da otvorimo data frame iz našeg direktorijuma:

```
read.table("c:/Sam/Dokumenti/Obuke/LEPlmer2018/Podaci_vezba.txt",sep="\t", header=TRUE)
```

```
##   Ispitanik          Manipulacija Reprodukcija  IQ   Rukost
## 1        s1       kontrolna grupa            7 101 desnoruk
## 2        s2 eksperimentalna grupa           15 124  levoruk
## 3        s3       kontrolna grupa            5 110  levoruk
## 4        s4 eksperimentalna grupa           11  98 desnoruk
## 5        s5       kontrolna grupa           10 103 desnoruk
## 6        s6       kontrolna grupa            8  99 desnoruk
## 7        s7 eksperimentalna grupa           14 123  levoruk
## 8        s8       kontrolna grupa            6 109  levoruk
## 9        s9 eksperimentalna grupa           18 102 desnoruk
```

U realnosti, retko ćemo ručno praviti data frame. Umesto toga, učitaćemo ga iz postojećeg fajla sa podacima (.txt, .csv, SQL, Excel, SPSS…).

Pošto ćemo raditi sa velikim data frame-ovima, nećemo ih direktno prikazivati, nego ćemo ih dodeliti nekoj promenljivoj:

```
podaci = read.table("Podaci_vezba.txt",sep="\t", header=TRUE)
```

Naravno, sami smišljate naziv promenljive u koju smeštate data frame, a uobičajena praksa je da se on nazove df

Pisanje skripta štedi vreme u budućnosti

Umesto da svaki put iznova pišete komande, možete da ih pišete u nekom txt editoru i sačuvate kao txt fajl iz kog ćete kasnije kopirati linije koda u R konzolu. Još jednostavnija opcija je da ih pišete u skript prozoru R studio programa i sačuvate skript kao fajl sa ekstenzijom .r

Na ovaj način u budućnosti ne morate sve da pišete ispočetka, dovoljno je da promenite nekoliko kritičnih vrednosti. Ovo je naročito korisno za pravljenje grafikona!