Serbia Digital Week

British Embassy Belgrade

Uvod u programski jezik R

Dejan Vujičić

Fakultet tehničkih nauka u Čačku,

5. april 2019. godine

Uvod

- R je programski jezik i softversko okruženje za:
  - Statističku analizu,
  - Grafičko predstavljanje podataka,
  - Pravljenje izveštaja.
- Kreirali su ga **R**oss Ihaka i **R**obert Gentleman sa University of Aukland, New Zealand
- Prva verzija je objavljenja 1993. godine
- Od 1997. godine, *R Core Team* se bavi razvojem ovog programskog jezika

Uvod

- R je interpreterski jezik
- Ključne karakteristike:
  - Jednostavan i efektivan programski jezik koji podržava uslovno izvršavanje, cikluse, korisnički definisane rekurzivne funkcije i olakšan ulaz/izlaz
  - Efikasan rad sa podacima i njihovo smeštanje
  - Podržava operatore za rad sa nizovima, listama, vektorima i matricama
  - Omogućava grafičku predstavu podataka
- R je najpopularniji i najviše korišćen programski jezik za statističke proračune

Instalacija

- Instalaciju preuzeti sa sledeće adrese (poslednja verzija R 3.5.1):

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

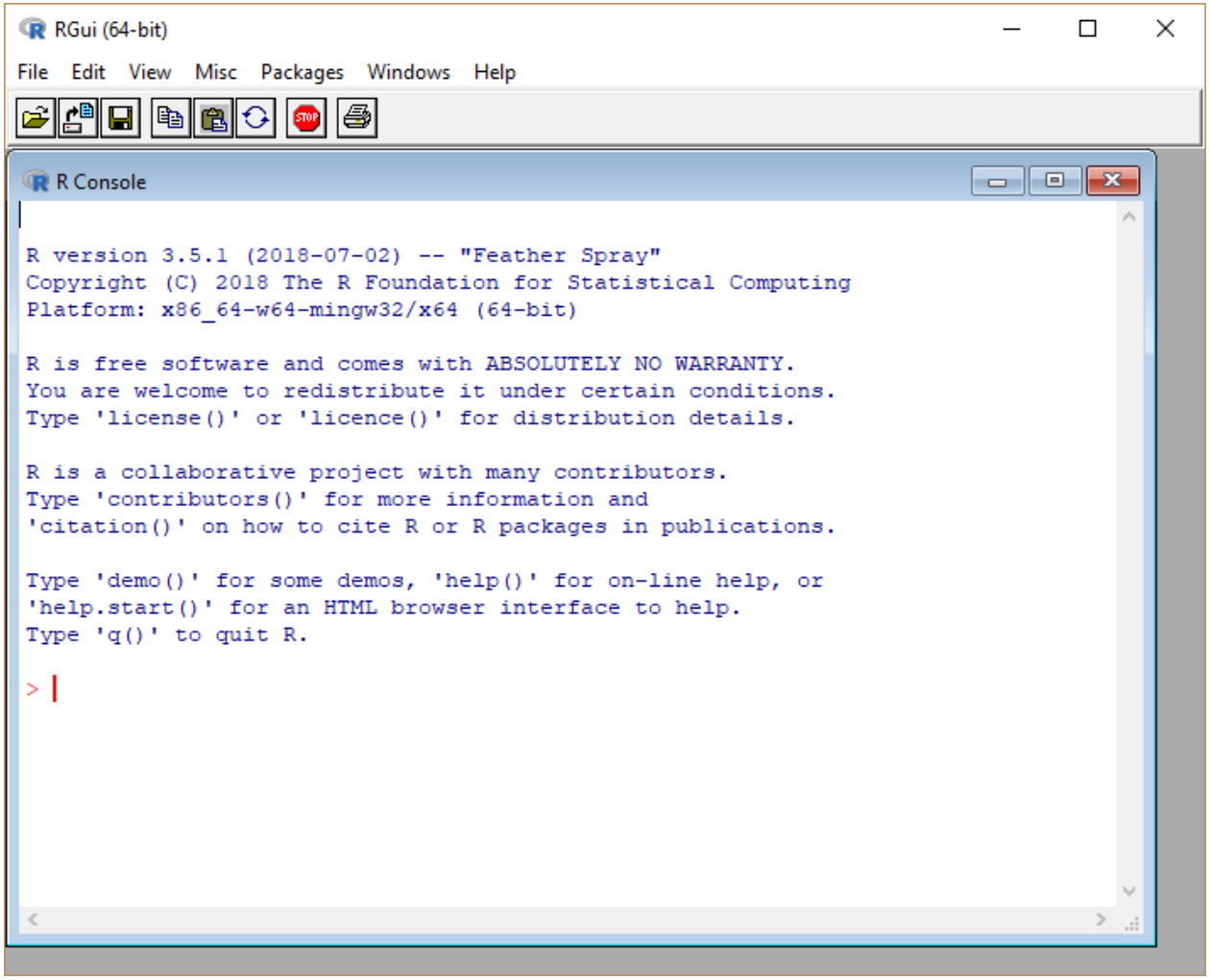
Download R for Windows

base

Download R 3.5.1 for Windows

- U toku instalacije možete izabrati koju verziju želite da instalirate (32-bitnu, 64-bitnu ili obe)

RGui


RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 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.
>

RSTUDIO

- Osnovni R IDE je rudimentaran, pa ćemo koristiti Rstudio
- Preuzeti ga sa sledeće adrese (poslednja verzija 1.1.456):

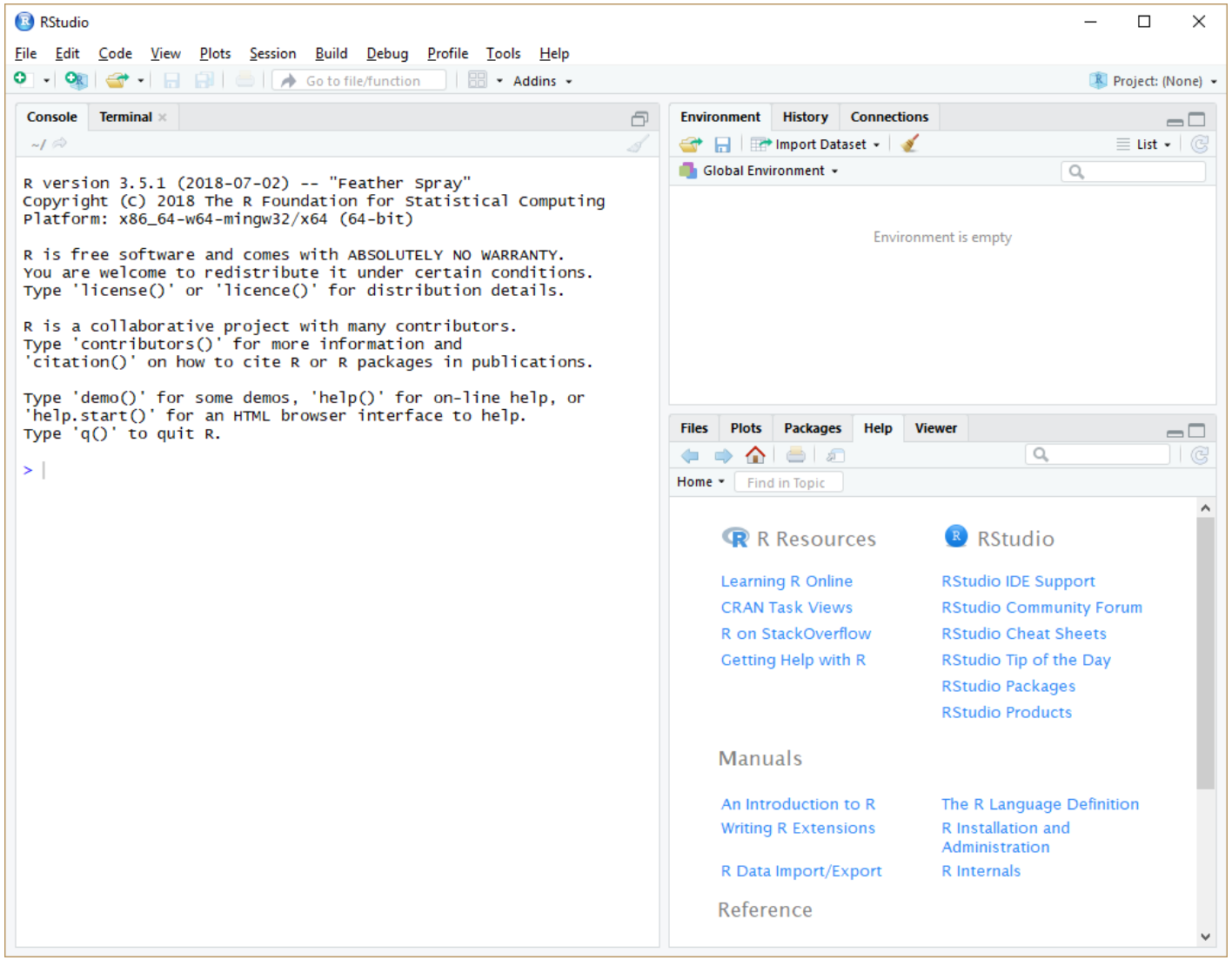
https://www.rstudio.com/products/rstudio/download/

FREE DOWNLOAD

Rstudio 1.1 

/7/8/10

RSTUDIO


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 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.
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Home
Find in Topic
R Resources
Learning R Online
CRAN Task Views
R on StackOverflow
Getting Help with R
RStudio
RStudio IDE Support
RStudio Community Forum
RStudio Cheat Sheets
RStudio Tip of the Day
RStudio Packages
RStudio Products
Manuals
An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals
Reference

Konzola vs. skripta

- Postoje dva osnovna načina izvršavanja instrukcija u R programskom jeziku:
  - *Pisanje u konzoli*: piše se jedna po jedna instrukcija i tako se i izvršava
  - *Pisanje u skripti (fajlu)*: piše se ceo program i kao takav se i izvršava
- Konzola se prepoznaje po znaku prompta >
- Skripta se kreira pomoću *File* → *New File* → **R Script** (ili prečicom Ctrl + Shift + N)

Pisanje u konzoli

- **Primer 1.** Napisati program kojim se ispisuje pozdravna poruka *„Zdravo svete“*.

```
> print("Zdravo svete!")
[1] "Zdravo svete!"
```

- Za ispis korisničkog uputstva o korišćenju ugrađenih funkcija, koristi se funkcija **help()**, npr. **help("print")**

Pisanje skripte

- **Primer 2.** Napisati isti program, samo u skripti.

```
print("Zdravo svete")
```

- Klikom na **Run**, dobija se isti ispis u konzoli kao za prethodni primer
- Skripta ima ekstenziju .R
- **VAŽNO:** Izvršava se deo koda koji je označen. Da bi se izvršio ceo program, potrebno je označiti sve linije koda

Komentari

- Jednolinijski komentari počinju sa znakom #.

```
# Ova linija je komentar
print("Zdravo svete")
```

- Višelinijski komentari zvanično ne postoje, ali se mogu dobiti stavljanjem teksta pod jednostruke ili dvostruke znake navoda:

```
"Ovo je viselinijski komentar
Interperter ce ga izvrsiti,
ali ne dolazi u sukob sa
ostatkom koda"
print("Zdravo svete")
```

Promenljive

- Ime promenljive može da sadrži slova, brojeve i znakove . i _
- Ime promenljive mora da počinje slovom ili tačkom nakon koje ne sledi broj
- Operator dodele vrednosti u programskom jeziku R je <–

```
x <- 4
```

- Alternativno, mogu se koristiti i operatori = i –> za dodelu vrednosti, s tim da je smer dodele u poslednjem slučaju

```
vrednost -> ime_promenljive
```

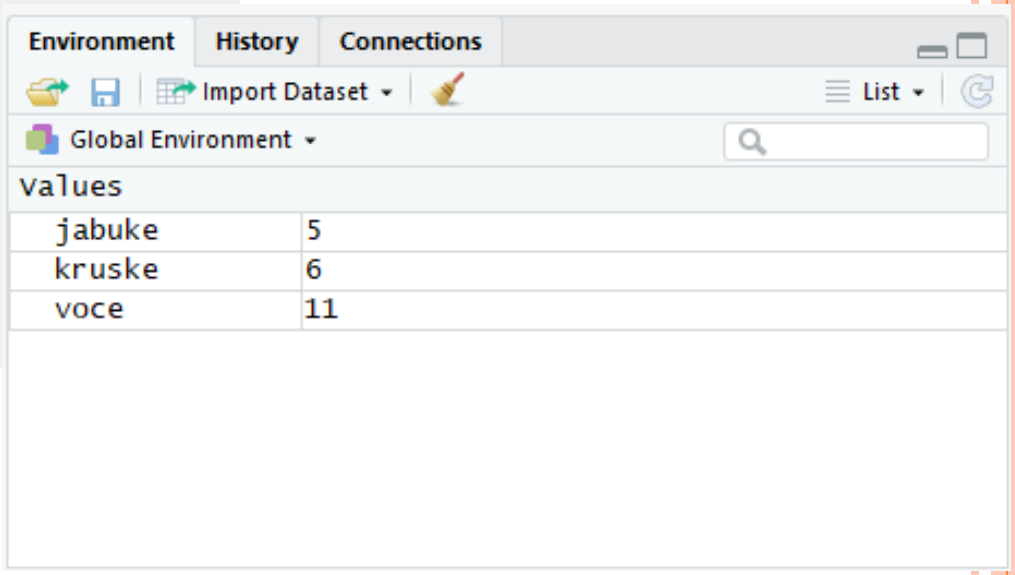
Promenljive

- **Primer 3.** Kreirati promenljivu *jabuke* i dodeliti joj vrednost 5. Potom kreirati promenljivu *kruske* i dodeliti joj vrednost 6. Ispisati sumu obe promeljive i kreirati promenljivu *voce*.

```
# Kreiranje promenljivih jabuke i kruske
jabuke <- 5
kruske <- 6

# Sabiranje ovih promenljivih
jabuke + kruske

# Kreiranje promenljive voce
voce = jabuke + kruske
```



Promenljive

- Ispis svih promenljivih trenutno učitanih se ostvaruje pozivom funkcije **ls():**

```
> ls()
[1] "jabuke" "kruske" "voce"
```

- Pretraga promenljive po pocetku imena vrši se sa parametrom **pattern:**

```
> ls(pattern="ja")
[1] "jabuke"
```

- Brisanje promenljive se vrsi pozivom funkcije **rm():**

```
> rm(voce)
> ls()
[1] "jabuke" "kruske"
```

- Brisanje svih promenljivih i pomoću parametra **list**:

```
> rm(list = ls())
> ls()
character(0)
```

Programski jezik R

Tipovi podataka

Tipovi podataka

- Promenljive se u R-u ne deklarišu po tipu, eksplicitno
- Tipovi podataka:
  - Primitivni tipovi podataka (atomski vektori)
  - Vektori
  - Liste
  - Matrice
  - Nizovi
  - Faktori
  - Tabele

Atomski vektori

- Postoji 6 osnovnih vrsta atomskih vektora
- Zovu se još i primitivnim tipovima podataka
- To su:
  - Logičke vrednosti (TRUE, FALSE)
  - Realni brojevi
  - Celi brojevi
  - Kompleksni brojevi
  - Karakteri
  - „Sirovi“ tipovi podatka (engl. *raw*)

Logički podaci

- Logički tip podataka („logical“):

```
> v <- TRUE
> print(class(v))
[1] "logical"
```

- Metoda **class()** vraća vrstu podatka kojoj promenljiva pripada
- Logičke promenljive mogu imati vrednost TRUE ili FALSE
- Moraju se pisati sa svim velikim slovima
- Logičke operacije:
  - Negacija: !
  - Logičko I: &&
  - Logičko ILI: ||

Realni i celi brojevi

- Realni brojevi potpadaju pod tip podatka „numeric“

```
> v <- 23.5
> print(class(v))
[1] "numeric"
> v <- 4
> print(class(v))
[1] "numeric"
```

- Celi brojevi pripadaju tipu „integer“
- Mora se naglasiti da je podatak ceo broj tako što se doda sufiks L

```
> v <- 2L
> print(class(v))
[1] "integer"
```

Kompleksni brojevi, karakteri

- Kompleksni brojevi se definišu koristeći imaginarnu jedinicu *i*
- Definišu se kao „complex“ tip podataka

```
> v <- 2+4i
> print(class(v))
[1] "complex"
```

- Tipu podatka „character“ pripada i pojedinačni karakteri i stringovi
- Moraju biti ili pod jednostrukim ili dvostrukim navodnicima

```
> a <- 'a'
> b <- "e"
> c <- 'string'
> d <- "23.4"
> print(class(a))
[1] "character"
> print(class(b))
[1] "character"
> print(class(c))
[1] "character"
> print(class(d))
[1] "character"
```

„Sirovi“ tipovi podataka

- Potpadaju pod „raw“ tip podatka
- Karakteri su predstavljeni svojim ASCII vrednostima

```
> v <- charToRaw("Zdravo")
> print(class(v))
[1] "raw"
```

Vektori

- Vektor predstavlja skup podataka istog ili različitog osnovnog tipa
- Ako želite da kreirate vektor sa više od jednim elementom, koristite funkciju **c()** za njihovo kombinovanje
- Indeksi elemenata vektora kreću od 1
- Broj elemenata se dobija pomoću funkcije **length()**
- Pristup pojedinim elementima vektora se obavlja pomoću operatora **[]**
- Vektoru se dodaje nova vrednost tako što se izvrši dodela određenom indeksu, koji ne mora biti naredni indeks
- Mogu se i menjati vrednosti elemenata vektora

Vektori

```
> jabuke <- c('crvene',"zelene",'zute')
> print(jabuke)
[1] "crvene" "zelene" "zute"
> print(class(jabuke))
[1] "character"
> print(jabuke[1])
[1] "crvene"
> print(jabuke[0])
character(0) >
> print(jabuke[2])
[1] "zelene"
> print(jabuke[3])
[1] "zute"
> print(jabuke[4])
[1] NA
> jabuke[4] = 'braon'
> print(jabuke)
[1] "crvene" "zelene" "zute" "braon"
> length(jabuke)
[1] 4
> jabuke[1] = 'plave'
> print(jabuke)
[1] "plave" "zelene" "zute" "braon"
```

Liste

- Lista je R objekat koji može da sadrži mnoštvo različitih elemenata, uključujući vektore, funkcije, pa čak i druge liste
- Lista se kreira pozivom metode **list()**
- Broj elemenata u listi se dobija pomoću funkcije **length()**
- Indeksi kreću od 1, a indeksiranje se vrši sa uglastim zagradama **[]**
- Dodavanje i izmena elemenata su isti kao kod vektora

Liste

```
> lista <- list(c(2,5,3),21.3,sin)
> print(lista)
[[1]]
[1] 2 5 3
[[2]]
[1] 21.3
[[3]]
function (x) .Primitive("sin")
> length(lista)
[1] 3
> lista[4] = "zdravo"
> print(lista)
[[1]]
[1] 2 5 3
[[2]]
[1] 21.3
[[3]]
function (x) .Primitive("sin")
[[4]] [1] "zdravo"
> print(lista[2])
[[1]]
[1] 21.3
```

```
> lista[3] = 5
> print(lista)
[[1]]
[1] 2 5 3
[[2]]
[1] 21.3
[[3]]
[1] 5
[[4]]
[1] "zdravo"
> lista[7] = 6
> print(lista)
[[1]]
[1] 2 5 3
[[2]]
[1] 21.3
[[3]]
function (x) .Primitive("sin")
[[4]]
NULL
[[5]]
NULL
[[6]]
NULL
[[7]]
[1] 6
```

Matrice

- Matrice su dvodimenzionalni skupovi podataka
- Kreiraju se pozivom funkcije **matrix()**
- Mogu se kreirati sa vektorom kao parametrom, gde se specificira broj redova (*nrow*), broj kolona (*ncol*) i da li se podaci smeštaju po redu (*byrow*)
- Metoda **length()** daje ukupan broj elemenata
- Elementima se pristupa pomoću uglastih zagrada **[]**
- Prvi element matrice M ima indeks **M[1,1]**, sledeći **M[1,2]**, itd.
- Štampanje prve vrste matrice M: **M[1,]**
- Štampanje prve kolone matrice M: **M[,1]**

Matrice

```
> M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
> print(M)
     [,1] [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"
> length(M)
[1] 6
> M[1,1]
[1] "a"
> M[1,2]
[1] "a"
> M[1]
[1] "a"
> M[,1]
[1] "a" "c"
> M[1,]
[1] "a" "a" "b"
> M[1,1] = 4
> M
     [,1] [,2] [,3]
[1,] "4"  "a"  "b"
[2,] "c"  "b"  "a"
```

Nizovi

- Nizovi mogu biti proizvoljne dimenzije
- Kreiraju se pozivom funkcije **array()**
- Broj dimenzija se specificira parametrom **dim**
- Ukupan broj elemenata se dobija pomoću funkcije **length()**
- Elementima se pristupa i menjaju im se vrednosti pomoću uglastih zagrada **[]**

Nizovi

- U sledećem primeru se kreiraju dve matrice dimenzije 3 x 3
- Prvoj matrici se pristupa sa **a[,,1]**, a drugoj sa **a[,,2]**

```
> a <- array(c('zelena','crvena'),dim = c(3,3,2))
> print(a)
, , 1
      [,1]     [,2]     [,3]
[1,] "zelena" "crvena" "zelena"
[2,] "crvena" "zelena" "crvena"
[3,] "zelena" "crvena" "zelena"
, , 2
      [,1]    [,2]    [,3]
[1,] "crvena" "zelena" "crvena"
[2,] "zelena" "crvena" "zelena"
[3,] "crvena" "zelena" "crvena"
> length(a)
[1] 18
> a[1,1,1]
[1] "zelena"
> a[1,2,2]
[1] "zelena"
> a[6]
[1] "crvena"
> a[7]
[1] "zelena"
> a[10]
[1] "crvena"
> a[10] = 'zuta'
> a
, , 1
      [,1]     [,2]     [,3]
[1,] "zelena" "crvena" "zelena"
[2,] "crvena" "zelena" "crvena"
[3,] "zelena" "crvena" "zelena"
, , 2
      [,1]     [,2]     [,3]
[1,] "zuta"   "zelena" "crvena"
[2,] "zelena" "crvena" "zelena"
[3,] "crvena" "zelena" "crvena"
```

Faktori

- Faktori su R objekti koji se kreiraju koristeći vektore
- Faktori čuvaju elemente vektora zajedno sa labelama
- Labele predstavljaju različite vrednosti elemenata vektora koji se pojavljuju u faktoru
- Faktor se kreira pomoću funkcije **factor()**
- Broj nivoa u faktoru (jedinstvenih labela) se dobija pomoću funkcije **nlevels()**
- Broj elemenata se dobija pomoću funkcije **length()**
- U faktor se mogu dodavati samo vrednosti koje već postoje u njemu
- Takođe, postojećim elementima se može menjati vrednost samo sa već prisutnim elementima

FAKTORI

```
> boje_jabuka <- c('zelena','zelena','zuta','crvena','crvena','crvena','zelena')
> faktor <- factor(boje_jabuka)
> print(faktor)
[1] zelena zelena zuta crvena crvena crvena zelena
Levels: crvena zelena zuta
> print(nlevels(faktor))
[1] 3
> length(faktor)
[1] 7
> faktor[2] = 'braon'
  Warning message: In `[<-.factor`(`*tmp*`, 2, value = "braon") :
                  invalid factor level, NA generated
> faktor[2] = 'zuta'
> faktor [1]
zelena zuta zuta crvena crvena crvena zelena
Levels: crvena zelena zuta
> faktor[8]='zelena'
> faktor
[1] zelena zuta zuta crvena crvena crvena zelena
[8] zelena Levels: crvena zelena zuta
> faktor[10]='zelena'
> faktor
[1] zelena zuta zuta crvena crvena crvena zelena
[8] zelena <NA> zelena
Levels: crvena zelena zuta
```

Tabele

- Tabele (*data frames*) se kreiraju pomoću funkcije **data.frame()**
- U suštini, tabela je lista vektora jednakih dužina
- Za razliku od matrice, svaka kolona može sadržati različite tipove podataka
- Prva kolona može biti numerička, dok druga znakovna, itd.
- Metoda **length()** vraća broj kolona u tabeli

TAB

```
> BMI <- data.frame( + pol = c("Musko", "Musko","Zensko","Zensko","Musko"),
+ visina = c(182, 171.5, 165, 151, 185),
+ tezina = c(81,93,78,56,92),
+ godine = c(42,38,26,30,32) + )
> BMI
     pol  visina  tezina godine
1   Musko   182.0   81    42
2   Musko   171.5   93    38
3   Zensko  165.0   78    26
4   Zensko  151.0   56    30
5   Musko   185.0   92    32
> BMI[1]
   pol
1 Musko
2 Musko
3 Zensko
4 Zensko
5 Musko
> BMI[2]
  visina
1 182.0
2 171.5
3 165.0
4 151.0
5 185.0
> BMI[,1]
[1] Musko Musko Zensko Zensko Musko
Levels: Musko Zensko
> BMI[,2]
[1] 182.0 171.5 165.0 151.0 185.0
> length(BMI)
[1] 4
```

Tabele (data frames)

- Pravila:
  - Nazivi kolona ne smeju biti prazni
  - Nazivi redova moraju da budu jedinstveni (automatski počinju od 1)
  - Podatak u tabeli može biti broj, faktor ili karakter
  - Svaka kolona bi trebalo da sadrži isti broj podataka

KREIRANJE TABELE

```
# kreiranje tabele
radnici.firme <- data.frame(
  id = c (1:5),
  ime =
c("Marko","Marija","Ana","Luka","Ilija"),
  plata = c(623.3,515.2,611.0,729.0,843.25),
  datum.zaposlenja = as.Date(c("2012-01-01",
"2013-09-23", "2014-11-15", "2014-05-11",
"2015-03-27")),
  stringsAsFactors = FALSE
)
# stampanje tabele
print(radnici.firme)
```

```
> print(radnici.firme)
 id ime    plata  datum.zaposlenja
1 1 Marko  623.30 2012-01-01
2 2 Marija 515.20 2013-09-23
3 3 Ana    611.00 2014-11-15
4 4 Luka   729.00 2014-05-11
5 5 Ilija  843.25 2015-03-27
```

Uvoz tabele

- Ugrađene strukture podataka se mogu dobiti pozivom funkcije **data()**
- Uvoz tabele se prosto vrši navođenjem njenog imena (npr. **mtcars**)

```
Data sets in package 'datasets':

AirPassengers    Monthly Airline Passenger Numbers
                 1949-1960
BJsales          Sales Data with Leading Indicator
BJsales.lead (BJsales)
                 Sales Data with Leading Indicator
BOD              Biochemical Oxygen Demand
CO2              Carbon Dioxide Uptake in Grass
                 Plants
ChickWeight      Weight versus age of chicks on
                 different diets
DNase            Elisa assay of DNase
EuStockMarkets   Daily Closing Prices of Major
                 European Stock Indices, 1991-1998
Formaldehyde     Determination of Formaldehyde
HairEyeColor     Hair and Eye Color of Statistics
                 Students
Harman23.cor     Harman Example 2.3
Harman74.cor     Harman Example 7.4
Indometh         Pharmacokinetics of Indomethacin
```

Rad sa tabelom

- Ispis strukture tabele: **str(radnici.firme)**
- Prikaz statističkih podataka: **summary(radnici.firme)**
- Prikaz kolona: **names(radnici.firme)**
- Broj kolona: **ncol(radnici.firme)** ili **length(radnici.firme)**
- Broj redova: **nrow(radnici.firme)**
- Prikaz kolone: **radnici.firme$ime** ili **radnici.firme[["ime"]]** – vraća vektor sa tipom podataka kolone
- Prikaz kolone: **radnici.firme["ime"]** ili **radnici.firme[2]** – vraća tabelu

Rad sa tabelom

- Prikaz više kolona: **data.frame(radnici.firme$id, radnici.firme$ime)** ili **radnici.firme[,1:3]** ili **radnici.firme[,c(2,4)]**
- Prikaz reda: **radnici.firme[1,]** ili **radnici.firme["1",]** (preko broja ili naziva reda)
- Prikaz više redova: **radnici.firme[1:3,]** ili **radnici.firme[c(2,4),]**
- Prikaz prve i druge kolone drugog i trećeg reda: **radnici.firme[c(2,3),c(1,2)]**
- Prikaz redova čije kolone ispunjavaju neki uslov: **radnici.firme$plata > 600**

Rad sa tabelom

- Dodavanje kolone:
  radnici.firme = cbind(radnici.firme,grad=c("CA","BG","KG","NI","NS")) ili
  radnici.firme$grad = c("CA","BG","KG","NI","NS")
- Dodavanje reda:
  radnici.firme = rbind(radnici.firme,c(6,"Nina",754.6,"2019-03-01","CA"))

Modifikacija podataka u tabeli

- Promena cele kolone: **radnici.firme$grad = "CA"**
- Promena cele kolone: **radnici.firme$grad = c("CA","NI")**
- Promena celog reda: **radnici.firme[1,] = c(7,"Una",700,"2019-03-10","KV")**
- Promena samo jednog podatka: **radnici.firme[1,2] = "Ena"**
- Brisanje kolona: **radnici.firme$grad = NULL** ili **radnici.firme = radnici.firme(-c(1,3,5))**
- Brisanje redova: **radnici.firme = radnici.firme[-2,]** ili **radnici.firme = radnici.firme(-c(1,3,5),)**

Kreiranje podskupa

- Kreiranje tabele koja sadrži samo redove kod kojih je plata veća od 600:
  x = subset(radnici.firme, subset = plata > 600)
- Kreiranje tabele koja sadrži samo redove kod kojih je plata veća od 600 i manja od 700:
  x = subset(radnici.firme, subset = plata > 600 & plata < 700)
- Kreiranje tabele koja sadrži samo redove kod kojih je ime „Marko“ i plata veća od 600:
  x = radnici.firme[which(radnici.firme$ime == "Marko" & radnici.firme$plata > 600),]

Programski jezik R

Ulazna funkcija i operatori

Ulazna funkcija

- Funkcija za unos podataka od korisnika je **readline()**
- Parametar **prompt** ispisuje odgovarajuću poruku, ako je potrebno
- Sve što se unese se karakteriše kao **string**

```
> readline()
Zdravo!

[1] "Zdravo!"
> readline(prompt = "Unesi poruku: ")
Unesi poruku:
Zdravo!

[1] "Zdravo!"
```

Ulazna funkcija

```
> ime = readline("Ime: ")
Ime:
Marko

> god = readline("Godine: ")
Godine:
23

> print(ime)
[1] "Marko"
> print(god)
[1] "23"
> print(god + 5)
Error in god + 5 : non-numeric
argument to binary operator
```

Konverzija tipova podataka

- Za pretvaranje u drugi tip podatka (konverzija, kastovanje), koristi se funkcija **as()**, uz dodatak tipa podatka u koji se vrši konverzija
- Konverzija u ceo broj: **as.integer()**
- Konverzija u realan broj: **as.numeric()**
- Konverzija u string: **as.character()**
- Konverzija u logički podatak: **as.logical()**
- Konverzija u kompleksan broj: **as.complex()**
- Konverzija u vektor: **as.vector()**
- Konverzija u listu: **as.list()**
- Konverzija u niz: **as.array()**
- Konverzija u matricu: **as.matrix()**
- .....

Operatori

- Operatori u programskom jeziku R se mogu klasifikovati u:
  1. Aritmetičke operatore
  2. Relacione operatore
  3. Logičke operatore
  4. Operatore dodele
  5. Ostale operatore

Aritmetički operatori

Operator	Opis
+	Sabiranje
-	Oduzimanje
*	Množenje
/	Deljenje (realno)
^	Stepenovanje
%%	Ostatak pri deljenju
%/%	Celobrojno deljenje

```
> 2 + 3
[1] 5
> 2 - 3
[1] -1
> 2 * 3
[1] 6
> 2 / 3
[1] 0.6666667
> 2 ^ 3
[1] 8
> 2 %% 3
[1] 2
> 2 %/% 3
[1] 0
```

Relacioni operatori

Operator	Opis
==	Jednako
!=	Različito
<	Manje
>	Veće
<=	Manje ili jednako
>=	Veće ili jednako

```
> 2 == 3
[1] FALSE
> 2 != 3
[1] TRUE
> 2 > 3
[1] FALSE
> 2 < 3
[1] TRUE
> 2 >= 3
[1] FALSE
> 2 <= 3
[1] TRUE
```

Logički operatori

Operator	Opis
&	Više-elementno I
|	Više-elementno ILI
!	Negacija
&&	Jedno-elementno I
||	Jedno-elementno ILI

TRUE je bilo koja vrednost različita od nule.
FALSE je vrednost jednaka nuli.

```
> 2 & 3
[1] TRUE
> 2 && 3
[1] TRUE
> 2 || 3
[1] TRUE
> 2 | 3
[1] TRUE
> !2
[1] FALSE
```

Operatori dodele

Operator	Opis
<- <<- =	Leva dodela
-> ->>	Desna dodela

```
> v1 <- c(3, 1, TRUE, 2+3i)
> v2 <<- c(3, 1, TRUE, 2+3i)
> v3 = c(3, 1, TRUE, 2+3i)
> print(v1)
[1] 3+0i 1+0i 1+0i 2+3i
> print(v2)
[1] 3+0i 1+0i 1+0i 2+3i
> print(v3)
[1] 3+0i 1+0i 1+0i 2+3i
> c(3, 1, TRUE, 2+3i) -> v4
> c(3, 1, TRUE, 2+3i) ->> v5
> print(v4)
[1] 3+0i 1+0i 1+0i 2+3i
> print(v5)
[1] 3+0i 1+0i 1+0i 2+3i
```

Ostali operatori

Operator	Opis
:	Operator dodele opsega
%in%	Provera da li element pripada skupu
%*%	Množenje matrice sa transponovanom matricom

Dodela opsega

- Operator dodele opsega **:** kreira elemente sa inkrementom (korakom) 1
- Ako se želi drugačiji inkrement, koristi se funkcija **seq()** sa parametrom **by**

```
> v <- 2:8
> print(v)
[1] 2 3 4 5 6 7 8
> v <- seq(2,8)
> print(v)
[1] 2 3 4 5 6 7 8
> v <- seq(2,8,by=2)
> print(v)
[1] 2 4 6 8
> v <- seq(2,8,by=0.5)
> print(v)
 [1] 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
```

Provera pripadnosti

- Operator **%in%** se koristi za proveru da li element pripada vektoru

```
> v1 <- 8
> v2 <- 12
> t <- 1:10
> print(v1 %in% t)
[1] TRUE
> print(v2 %in% t)
[1] FALSE
```

Programski jezik R

Uslovno izvršavanje

Uslovne strukture

- R programski jezik podržava tri vrste uslovnih struktura:
  1. **if** iskaz
  2. **if...else** iskaz
  3. **switch** iskaz

Provera tipa podataka

- Metodom **is()**, uz dodatak tipa podatka se proverava da li određena promenljiva pripada datom tipu podatka
- Ceo broj: **is.integer()**
- Realan broj: **is.numeric()**
- Karakteri: **is.character()**
- Logička vrednost: **is.logical()**
- Kompleksan broj: **is.complex()**
- Vektor: **is.vector()**
- Matrica: **is.matrix()**
- .....

IF ISKAZ

- Opšti oblik IF iskaza:

```
if (logički_izraz) {
        jedan ili više iskaza
}
```

IF ISKAZ

```
broj <- 5L
if (broj == 5) {
    print("Broj je jednak 5.")
}
if (is.integer(broj)) {
    print("Broj je celobrojnog tipa.")
}
```

```
[1] "Broj je jednak 5."
[1] "Broj je celobrojnog tipa."
```

Složeni logički izrazi

- Kombinovanje izara se postiže operatorima && (I) i || (ILI)
- Može se koristiti i negacija (!)

```
broj <- 5L
if (broj == 5 && is.integer(broj))
{
    print("Broj je jednak 5 i
celobrojnog je tipa.")
}
if (!(broj == 4)) {
    print("Broj je razlicit od
4.")
}
```

```
[1] "Broj je jednak 5 i
celobrojnog je tipa."
[1] "Broj je razlicit od 4."
```

IF...ELSE iskaz

- Opšti oblik:

```
if (logički_uslov) {
        jedan ili više iskaza
} else {
        jedan ili više iskaza
}
```

IF...ELSE ISKAZ

```
x <- c("sta", "je", "istina")

if ("istina" %in% x) {
    print("Istina je nadjena.")
} else {
    print("Istina nije nadjena")
}

x <- 5
if (x < 0) {
    print("Negativan broj.")
} else {
    print("Nenegativan broj.")
}
```

```
[1] "Istina je nadjena."
[1] "Nenegativan broj."
```

Funkcija ifelse()

- Pošto su vektori osnovni gradivni elementi R jezika, funkcijom **ifelse()** se jednostavno mogu evaluirati vektori
- Sintaksa je **ifelse(vektor,x,y)**, gde se:
  - kao prvi parametar prosleđuje vektor,
  - drugi je vrednost koja se menja u vektoru ako je uslov ispunjen,
  - treći je vrednost koja se menja u vektoru ako uslov nije ispunjen
- Rezutat ove funkcije je vektor iste dužine kao vekt ostima

```
> a = c(5, 7, 2, 9)
> ifelse(a %% 2 == 0, "paran", "neparan")
[1] "neparan" "neparan" "paran"   "neparan"
```

SWITCH ISKAZ

- Opšti oblik:

  switch (izraz, slucaj1, slucaj2, slucaj3, ...)

```
> switch(2, "crvena", "zelena", "plava")
[1] "zelena"
> x <- switch(4, "crvena", "zelena", "plava")
> x
NULL
> x <- switch(0, "crvena", "zelena", "plava")
> x
NULL
> switch("boja", "boja" = "crvena", "oblik" = "kvadrat", "duzina" = 5)
[1] "crvena"
> switch("duzina", "boja" = "crvena", "oblik" = "kvadrat", "duzina" = 5)
[1] 5
```

SWITCH ISKAZ

```
y = 3
x = switch(
    y,
    "Dobro jutro",
    "Dobar dan",
    "Dobro vece",
    "Laku noc"
)
print(x)
```

```
[1] "Dobro vece"
[1] "Dobar dan"
```

```
y = "12"
x = switch(
    y,
    "9" = "Dobro jutro",
    "12" = "Dobar dan",
    "18" = "Dobro vece",
    "21" = "Laku noc"
)
print(x)
```

Programski jezik R

Ciklusi

Ciklusi

- Programski jezik R poznaje tri vrste ciklusa:
  1. **repeat**
  2. **while**
  3. **for**
- Upravljačke strukture kojima se može prekinuti izvršenje ciklusa ili nastaviti sa sledećom iteracijom su:
  1. **break**
  2. **next**

Ciklus repeat

- Osnovna sintaksa je:

```
repeat {
        jedan ili više iskaza
        if (logički_uslov) {
                break
        }
}
```

Ciklus repeat

- Sledećim kodom se ispisuje promenljiva *brojac* 5 puta
- Naredbom **break** se izlazi iz ciklusa

```
brojac = 1

repeat {
    print(brojac)
    brojac = brojac + 1
    if (brojac > 5) {
        break
    }
}
```

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

Ciklus while

- Osnovna sintaksa je:

```
while (logički_uslov){
        jedan ili više iskaza
}
```

CIKLUS WHILE

```
brojac = 1

while (brojac < 5) {
    print(brojac)
    brojac = brojac + 1
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
```

Ciklus **for**

- U programskom jeziku R, ciklus **for** se koristi za prolazak (iteriranje) kroz pojedinačne elemente sekvence (stringa, vektora, liste, ...)
- Ključna reč **in** se koristi za uzimanje pojedinačnih vrednosti iz sekvence
- Osnovna sintaksa je:

```
for (vrednost in sekvenca) {
    jedan ili više iskaza
}
```

Ciklus for

```
# funkcija LETTERS daje velika slova
# engleske abecede
v <- LETTERS[1:4]
for (i in v) {
    print(i)
}

# ovaj program prebrojava sve parne
# brojeve u vektoru x
x <- c(2, 5, 3, 9, 8, 11, 6)
brojac <- 0
for (vrednost in x) {
    if (vrednost %% 2 == 0) {
        brojac = brojac + 1
    }
}
print(brojac)
```

```
[1] "A"
[1] "B"
[1] "C"
[1] "D"
[1] 3
```

NAREDBA **BREAK**

- Ova naredba se koristi za izlazak iz ciklusa
- Osim kod ciklusa **repeat**, može se upotrebljavati i kod drugih ciklusa

```
i <- 1
while (i < 10) {
    print(i)
    i = i + 1
    if (i == 5) {
        break
    }
}

x <- 1:10
for (i in x) {
    print(i)
    if (i == 5) {
        break
    }
}
```

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

Naredba next

- Ova naredba se koristi za završetak trenutne iteracije i početak sledeće

```
i <- 0
while (i < 10) {
    i = i + 1
    if (i == 5) {
        next
    }
    print(i)
}

x <- 1:10
for (i in x) {
    if (i == 5) {
        next
    }
    print(i)
}
```

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

Programski jezik R

Funkcije

Funkcije

- Programski jezik R poseduje veliki skup ugrađenih funkcija, a korisnik može definisati i druge
- Funkcija se kreira koristeći ključnu reč **function**
- Osnovna sintaksa funkcije je:

ime_funkcije <- function (argumenati) {

telo funkcije

}

Funkcija bez argumenata

- Sledeća funkcija nema argumente niti povratnu vrednost

```
ispis <- function() {
    print("Pozdrav")
}
```

- Ova funkcija se poziva samo preko imena, sa praznim zagradama za argumente

```
ispis()
```

- U konzoli se ispisuje ono što je definisano u funkciji

```
[1] "Pozdrav"
```

Funkcija sa argumentima

- Sledeća funkcija prima dva argumenta, odvojena zarezima

```
zbir <- function(a, b) {
    print(a + b)
}
```

```
zbir(2, 3)
broj1 <- 5
broj2 <- 6
zbir(broj1, broj2)
zbir(broj1, 10)
```

```
[1] 5
[1] 11
[1] 15
```

Funkcija sa argumentima

- Redosled argumenata se ne mora poštovati, ali se onda moraju znati njihova imena u definiciji funkcije
- Sledeća tri poziva funkcije su ekvivalentna

```
zbir(5, 6)
zbir(a = 5, b = 6)
zbir(b = 6, a = 5)
```

```
[1] 11
[1] 11
[1] 11
```

Podrazumevane vrednosti argumenata

- Funkcija se može definisati sa jednim ili više argumenata sa podrazumevanim vrednostima

```
zbir <- function(a = 5, b = 6) {
    print(a + b)
}
```

- Ova funkcija se može pozvati sa nula ili sa dva argumenta
- Ako se pozove sa nula argumenata, uzimaju se podr

```
zbir() # daje 11
zbir(3, 4) # 7
zbir(a = 7, b = 3) # 10
zbir(b = 4, a = 2) # 6
```

```
[1] 11
[1] 7
[1] 10
[1] 6
```

Povratna vrednost funkcije

- Sintaksa je:

  return (vrednost)

- Može se vratiti samo jedna vrednost (ili više, ali preko složenog tipa podatka, npr. vektora)

Povratna vrednost funkcije

- Funkcija koja vraća zbir argumenata je

```
zbir <- function(a, b) {
    return (a + b)
}

a <- zbir(3, 4)
print(a)
```

```
[1] 7
```

Lokalne i globalne promenljive

- Promenljive koje se koriste unutar funkcije se nazivaju lokalne promenljive za tu funkciju
- Globalne promenljive su promenljive koje su definisane u glavnom programu, ili u konzoli
- Globalne promenljive se mogu čitati, a da bi se menjale, mora se koristiti operator „super“ dodele <<-

Lokalne i globalne promenljive

- Sledeća funkcija neće promeniti vrednost globalne promenljive

```
a <- 5 # globalna promenljiva

funkcija <- function(a) {
    a <- 10 # lokalna promenljiva
}

funkcija(a)
print(a)
```

```
[1] 5
```

Lokalne i globalne promenljive

- Promena globalne promenljive pomoću operatora <<-

```
a <- 5 # globalna promenljiva

funkcija <- function(a) {
    a <<- 10 # promena globalne promenljive
    print(a) # 10
}

funkcija(a)
print(a) # 10
```

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

Vraćanje više vrednosti

- Sledeća funkcija vraća tri vrednosti preko liste

```
multi_return <- function() {
    lista <- list("boja" = "crvena",
                  "velicina" = 20,
                  "oblik" = "okrugao")
    return (lista)
}
```

- Pojedinačnom elementu liste se pristupa preko operatora $

```
lista <- multi_return()

print(lista$boja)
print(lista$velicina)
print(lista$oblik)
```

```
[1] "crvena"
[1] 20
[1] "okrugao"
```

Statističke funkcije

- min(x)
- max(x)
- sum(x)
- range(x)
- cumsum(x)
- cumprod(x)
- diff(x)
- summary(x)
- mean(x)
- median (x)
- sd(x)
- sd(X)
- var(X)
- cor(X)
- quantile(x, 0.75)
- quantile(x)
- rank (x)
- sort(x)
- order(x)

Biblioteke, datoteke i radni direktorijum

- Instaliranje biblioteka: **install.packages()**
- Uvoz biblioteka: **library()**
- Čuvanje datoteke: **save()**
- Učitavanje datoteke: **load()**
- Informacija o radnom direktorijumu: **getwd()**
- Postavljanje radnog direktorijuma: **setwd()**

Hvala na pažnji!