Upoznavanje sa sintaksom jezika R i njegova primjena u osnovnoj statističkoj i grafičkoj analizi podataka

S721

srce
Sveučilište u Zagrebu
Sveučilišni računski centar

Izvorni obrazovni materijali „Upoznavanje sa sintaksom jezika R i njegova primjena u osnovnoj statističkoj i grafičkoj analizi podataka (S720)“ dostupni su na poveznici https://www.srce.unizg.hr/stat/s720.

Preradu obrazovnih materijala izvršio je tim Srca u sastavu:

Autorica prerade: Nela Tomić

Recenzentica: mr. sc. Melita Perčec Tadić

Urednica: Sabina Rako

Lektorica: Mia Kožul

Sveučilište u Zagrebu

Sveučilišni računski centar

Josipa Marohnića 5, 10000 Zagreb

edu@srce.hr

ISBN 978-953-8172-43-4 (meki uvez)
ISBN 978-953-8172-44-1 (PDF)

Verzija priručnika: *S721-20201002*

Sadržaj

1. UVOD

U tečaju Upoznavanje sa sintaksom jezika R i njegova primjena u osnovnoj statističkoj i grafičkoj analizi podataka (S720) obrađuju se osnove rada u RStudiju, tipovi i strukture podataka u jeziku R, funkcije i grafička vizualizacija podataka iz osnovnih paketa sustava R, te primjena jezika R u osnovnim statističkim analizama.

Tečaj je namijenjen studentima, djelatnicima visokih učilišta i javnih instituta, zaposlenicima tvrtki i institucija te ostalim zainteresiranima za upoznavanje s jezikom R i njegovoj primjeni u analizi podataka.

Za pohađanje ovoga tečaja potrebno je poznavanje osnova rada s računalom i operacijskim sustavom *MS Windows*, poznavanje osnova rada na Internetu te pasivno služenje engleskim jezikom (razumijevanje jednostavnih tehničkih članaka i uputa). Barem minimalno iskustvo programiranja je prednost.

U ovom priručniku naredbe su pisane proporcionalnim slovima (na primjer, naredba `install.packages()`).

Sintaksa koda (naredbi i komentara) pisana je proporcionalnim slovima u sivoj pozadini:

```
library(help = "base") #pomoć za paket base.
```

Tipke na tipkovnici pisane su proporcionalnim slovima u uglatim zagradama (npr. [Ctrl]). Važni pojmovi su, prilikom prvog spominjanja, pisani podebljano.

Gradivo je uglavnom obrađeno kroz niz primjera i vježbi. Polaznici na raspolaganju imaju isprintani udžbenik, HTML inačicu udžbenika, digitalne radne bilježnice te dodatne materijale u digitalnom obliku. Rješenja zadataka i rezultati izvođenja programskih naredbi dani su u HTML inačici udžbenika.

2. RADNO OKRUŽENJE

2.1. Što je R?

R je besplatno softversko okruženje za statističku analizu i vizualizaciju podataka. Sadrži već ugrađenu široku paletu grafičkih i statističkih tehnika poput onih potrebnih za linearnu i nelinearnu regresiju, klasične statističke testove, analizu vremenskih nizova, klasteriranje podataka, vizualizaciju, strojno učenje, pa sve do statističkih paketa za znanstvene grane kao što su hidrologija, genetika, psihologija i ekonometrija. Kao programski jezik nudi i mogućnost kreiranja vlastitih objekata, funkcija i paketa.

Velika prednost jezika R jest to što ne ovisi o operativnom sustavu, besplatan je, a funkcionalnost razvijenih procedura moguće je kombinirati s mnogim drugim programskim jezicima (*C*/*C++*, *Java*, *Phyton*), formatima zapisa podataka i bazama podataka (*Excel*, *Access*, *SAS*, *Stata*, *SPSS*, *Minitab*…).

Popularnosti R-a pridonijela je i mogućnost izrade širokoga spektra grafičkih prikaza visoke kvalitete u kojem je korisniku omogućeno podešavanje gotovo svih elemenata grafikona.

R se vrlo lako može nadograđivati paketima. Paket je skup funkcija čija je namjena lakše i pojednostavljeno izvršavanje zadatka iz nekog područja. S obzirom na to da je R jezik otvorenoga programskog kôda (engl. *open source*), svakome je dostupna mogućnost izrade i dijeljenja paketa s drugim korisnicima, a to je posljedično omogućilo sve lakšu uporabu jezika R, proširilo njegovu primjenu i pridonijelo njegovoj popularizaciji. Tako se R danas koristi u gotovo svim granama znanosti i industrije, ponajviše u područjima financija, bioznanosti, sporta, trgovine, marketinga, proizvodnje, itd.

Više o sustavu R može se pronaći na poveznici *https://www.r-project.org/*.

2.1.1. Instalacija sustava R

CRAN, *The Comprehensive R Archive Network*, je primarna *web*-stranica za distribuciju i instalaciju sustava R te repozitorij dodatnih R paketa i dokumentacije.

Za instalaciju sustava R, potrebno je na *web*-stranici *http://cran.r-project.org/bin/* odabrati verziju ovisno o operacijskom sustavu na računalu (*Windows*, *Mac*, *Linux*) te krenuti s postupkom instalacije. S obzirom na to da je instalacija na operacijski sustav Windows klasična, ovom prilikom je nećemo detaljnije opisivati.

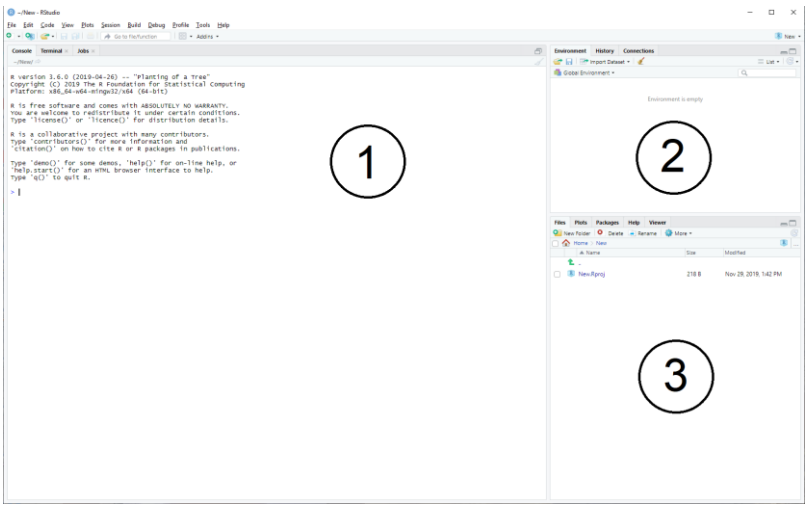
2.2. Upoznavanje s RStudijem

RStudio je korisničko sučelje preko kojeg se najčešće koristi jezik R. Preciznije, RStudio je integrirano razvojno okruženje (engl. *integrated development environment*, IDE) za jezik R. RStudio se može preuzeti na stranici https://www.rstudio.com/. Za rad u RStudiju potrebno je prethodno instalirati i R i RStudio.

Kada se RStudio pokrene klikom na ikonu programa, otvori se prozor s trima oknima:

1. Console
2. Environment
3. Files

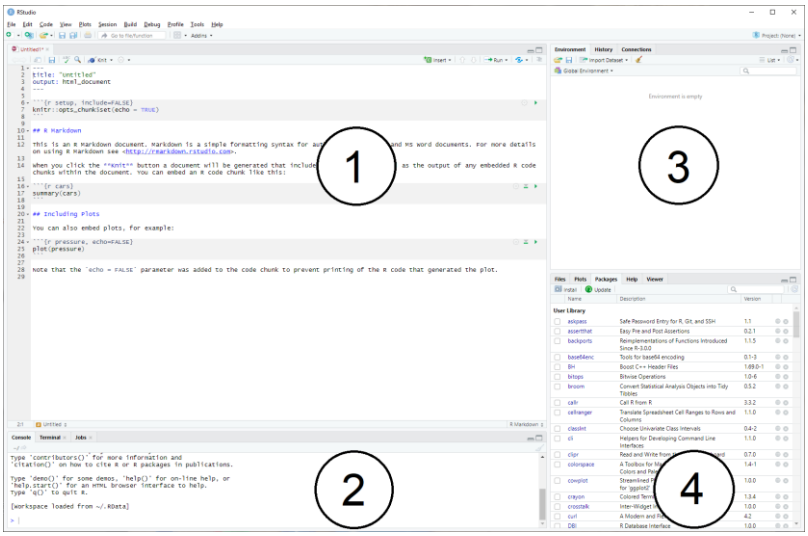
od kojih svaki sadrži nekoliko kartica.



Iako se u konzoli (Console) može pisati programski kôd, RStudio nam nudi bolje mogućnosti kroz Source Panel.

Source Panel ili okno u koje se može pisati i programski kôd i tekst dokumenta nakon prvog pokretanja nije otvoreno. Za otvaranje potrebno je u izborniku odabrati File -> New File i odabrati tip datoteke. Na izboru su R Script, R Notebook, R Markdown, Shiny Web App, itd. Tijekom ovoga tečaja koristit ćemo R Markdown. Nakon odabira tipa datoteke, sučelje programa RStudio sastoji se od četiri okna:

1. Source
2. Console
3. Environment
4. Files



2.2.1. Source

U oknu Source otvaraju se datoteke s kojima radimo.

2.2.2. Console

Konzola (engl. *console*) je okno preko kojeg se izvršavaju sve naredbe sustava R i prikazuju rješenja tih naredbi.

Naredbe u konzolu dospiju ili direktnim upisom u konzolu (nakon znaka >) ili iz drugih okna automatskim prebacivanjem.

Naredbe koje se upisuju direktno u konzolu izvršavaju se nakon pritiska tipke [Enter]. Istovremeno se može napisati i izvršiti samo jedna naredba. Sav kôd iz konzole nestaje nakon prekida sesije.

U slučaju da se u konzoli pojavi simbol +, to je znak da naredba iz konzole ili okna Source nije dovršena i da se čeka nastavak unosa.

2.2.3. Environment

Gornje desno okno **Environment** ili radno okruženje jest preglednik svih objekata (varijable, skupovi podataka, funkcije…) koje korisnik ima na raspolaganju u trenutačnoj sesiji.

U kartici **History** nalazi se popis od maksimalno 512 prethodnih naredbi koje su izvršene u RStudiju. Odabrane naredbe se odavde mogu poslati u Source dokument ili u konzolu na izvršavanje.

2.2.4. Files

Okno **Files** sastoji se od nekoliko korisnih kartica.

Files je mjesto na kojem se može podesiti radni direktorij i pregledati popis svih datoteka. Radni direktorij nekog procesa je hijerarhijska mapa dokumenata vezana za proces. Kad se

unutar procesa stvara ili poziva neka datoteka, njen put (engl. *file path*) kreće od pozicije radnog direktorija (a ne iz korijenskog direktorija (engl. *root directory*)).

Plots služi za prikaz grafikona.

Packages daje pregled dostupnih paketa. Kvačicom su označeni oni paketi koji su i učitani automatski ili pomoću funkcije `library()`. Popis paketa se po potrebi nadopunjava naredbom `install.packages()`.

Kartica **Help** je korisna za pomoć i referenciranje vezano za bilo koji paket, funkciju, ugrađeni podatkovni skup, i bilo što drugo za što postoji službena R dokumentacija. Ako nas na primjer, zanima kako se koristi funkcija `plot()`, upišemo u tražilicu *plot* i dobit ćemo dokumentaciju o toj funkciji. Ovo je vjerojatno jedna od najkorištenijih kartica u RStudiju za početnike.

Kartica **Viewer** služi za pregled lokalnoga *web*-sadržaja.

Primjer: Podesimo radni direktorij na mapu *S721* koja se nalazi na radnoj površini.

Podešavanje radnoga direktorija može se napraviti na nekoliko načina. Jedan način je preko kartice Files. Pronađemo i odaberemo mapu koju želimo postaviti kao radni direktorij, zatim kliknemo na ikonu More koja se nalazi na alatnoj traci unutar karitice i odaberemo “Set As Working Directory”.

Drugi način je pomoću naredbe `setwd()`. Unutar nje u navodnicima upišemo lokaciju (engl. *file path*) željenoga radnog direktorija, s tim da se nazivi mapa i datoteka razdvajaju ili znakom “/” ili “\” jer je standardni "" rezerviran za specijalnu sintaksu markdown datoteke s kojom ćemo se kasnije upoznati.

```
#setwd("Desktop/S721")
```

Nakon podešavanja radnoga direktorija, dobro je napraviti provjeru. To se može napraviti upitom `getwd()`.

2.3. Vrste datoteka u RStudiju

2.3.1. R Script

R Script je jednostavna tekstualna datoteka u koju se upisuju naredbe za R. Ekstenzija datoteka ovoga tipa je “.r”.

Naredbe se izvršavaju istovremenim pritiskom tipki `[Ctrl]+[Enter]`, a rezultat se prikazuje u konzoli. Grafikoni se prikazuju na kartici Plots osim ako se ne specificira drugačije. Komentari se mogu pisati iza znaka #.

Prednost R Scripta nad konzolom je što sadrži pregled svih naredbi koje korisnik lako može ponoviti, ispraviti, doraditi i pohraniti.

Primjer:
Otvorite R Script datoteku, upišite u nju proizvoljan tekst te izračunajte koliko je 123 * 456. Spremite datoteku pod nazivom “proba1.r” u mapu S721.

2.3.2. R Markdown

R Markdown je vrsta R datoteke u kojoj je moguće ispreplitati programski kôd, njegove rezultate i tekst (input ili ulazna datoteka), iz koje se potom može generirati HTML, PDF ili *Word* dokument (output ili izlazna datoteka). Ekstenzija dokumenta je .rmd.

S obzirom na to da je cilj svake analize podataka predstaviti dobivene rezultate u nekoj vrsti dokumenta, R Markdown datoteka prikladano je rješenje jer omogućuje dokumentiranje cijelog postupka koji se provodio nad podacima, kao i njegovo repliciranje i dorađivanje.

Naredbe koje se trebaju izvršiti pišu se unutar blokova. Blok (engl. *chunk*) izrađuje se istovremenim pritiskom tipki `[Ctrl]+[Alt]+[I]`.

Pokretanje jedne naredbe unutar bloka izvršava se istovremenim pritiskom tipki `[Ctrl]+[Enter]`, a pokretanje svih naredbi iz jednog bloka se izvršava istovremenim pritiskom na [Ctrl]+[Shift]+[Enter].

Primjer:
Izračunajte koliko je $25 * 15$. Za izvršavanje operacije u .rmd datoteci potrebno je izraditi blok (engl. *chunk*) pomoću `[Ctrl]+[Alt]+[I]`, upisati naredbu $25 * 15$ te pritisnuti `[Ctrl]+[Enter]`.

```
25*15
```

```
## [1] 375
```

Tekst u .rmd datoteci koji je napisan izvan bloka piše se u običnom formatu, ali postoji sintaksa (jezik Markdown) koja omogućava oblikovanje teksta u izlaznoj datoteci. Na primjer, tekst koji se upiše između dviju zvjezdica bit će napisan *ukošenim slovima*, a tekst koji se upiše između dvostrukih zvjezdica bit će napisan **podebljanim slovima**. Tekst nakon znaka # biti će označen kao naslov poglavlja.

Više o markdown sintaksi može se pronaći na poveznici https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf.

R Markdown dokument konvertira se u pdf, HTML ili *Word* dokument pomoću ikone **Knit** koja se nalazi na alatnoj traci okna Source. Moguće je postaviti dodatne parametre za svaku vrstu izlaznoga dokumenta u zaglavlju .rmd datoteke, no o tome nećemo govoriti u ovom tečaju. Spomenut ćemo samo da se zaglavlje .rmd datoteke zove YAML (/ˈjæməl/, rimuje se s camel) i uvijek počinje i završava trima crticama (`---`).

Komentar u tekstualnom dijelu .rmd datoteke stavlja se unutar znakovnog izraza `<!-- -->`, a komentar unutar bloka piše se nakon znaka #.

U RStudiju mogu se kreirati i mnoge druge vrste datoteka, ali se one neće obrađivati u ovom tečaju.

NAPOMENA: Znak “\” stavlja se ispred znaka za markdown sintaksu ako se želi poništiti njegovo djelovanje i uključiti u tekst doslovce. Tako, na primjer, ako se u tekstu izlaznoga dokumenta treba navesti znak zvjezdice, u R Markdown datoteci umjesto * potrebno je napisati *. Iz ovog razloga se i put do datoteke mora označavati dvostrukim znakom “\\” ili alternativno s “/”. Znakovi koji su u službi sintakse markdown dokumenta zovu se metaznakovi (“\”, “#”, "_“,”^“,”**“,”*"…).

Primjer
1.1 Izradite novu R Markdown datoteku i u nju upišite proizvoljan tekst s naslovom (#).
1.2 Izradite blok i izvršite barem jednu računsku operaciju.
1.2 Odaberite jednu riječ iz teksta koju ćete označiti podebljanim slovima (** **).
1.3 Spremite datoteku u radni direktorij pod nazivom “proba2.rmd”.
1.2 ‘Knitajte’ odnosno generirajte HTML dokument iz datoteke proba2.
1.3 Preko kartice Files u oknu Files provjerite sadržaj radnoga direktorija. Označite datoteku proba2.rmd i izbrišite je klikom na dugme Delete unutar kartice Files.

2.4. Projekti

Na desnom kraju alatne trake nalazi se izbornik vezan za projekte (engl. *Project*). Projekt je imenovano radno okruženje, a praktično je kada korisnik radi na više projekata odjednom. Projekti omogućuju jednostavniji nastavak rada nakon svakog prekida te lakši prelazak iz jednog u drugo radno okruženje bez gubitka podataka i postavki.

Projekti se pohranjuju u datoteke ekstenzije *.Rproj.

2.5. Paketi

Sve funkcionalnosti sustava R grupirane su u paketima.

R paketi se sastoje od skupine funkcija, kompiliranoga kôda i primjera podataka u strogo definiranom formatu. Određeni paketi dolaze sa samom instalacijom sustava R, a ostale nadograđuje korisnik prema svojim potrebama.

Osnovni paketi (engl. *base*) dolaze sa sustavom R i čine njegovu okosnicu sadržavajući osnovne funkcije za aritmetiku, statistiku, grafiku, ulaz i izlaz podataka, osnovne programske potpore, itd. Ovaj skup paketa ne mijenja se između dviju verzija sustava R i ne mogu se pronaći izdvojeni na nekom repozitoriju paketa. Oni su:

- `base`,
- `stats`,
- `datasets`,
- `graphics`,
- `grDevices`,
- `methods`,
- `utils`.

Zatim postoje paketi koji su također dio same instalacije sustava R, ali se njihove verzije mogu nadograđivati između pojedinih verzija i kao takvi mogu se pronaći i pojedinačno na nekom spremištu paketa. Da bi korisnik mogao koristiti funkcije iz takvih paketa, mora ih prvo učitati pomoću funkcije `library()`. Neki od njih su `KernSmooth`, `MASS`, `Matrix`, `boot`, `class`, `cluster`, `codetools`, `foreign`, `lattice`, `mgcv`, `nlme`, `nnet`, `rpart`, `spatial`, `survival`.

Svi ostali paketi moraju se preuzeti s nekog repozitorija (spremišta) paketa pomoću naredbe `install.packages()`, i zatim učitati pomoću naredbe `library()`.

Primjer:

```
#install.packages("wordcloud")
#library(wordcloud)
```

NAPOMENA: Nazivi paketa unutar naredbe `install.packages()` pišu se u navodnicima, dok se unutar funkcije `library()` mogu navesti i bez njih.

Popis svih funkcija iz nekog paketa može se vidjeti u popratnoj dokumentaciji paketa na ovaj način:

```
library(help = "stats")  # popis funkcija iz paketa stats
```

Glavni repozitorij ili spremište paketa nalazi se na *web*-stranici CRAN - The Comprehensive R Archive Network. Trenutačno (jesen 2019.) postoji oko 15,000 paketa na CRAN-u, a cijeli popis se može pronaći na poveznici: https://cran.r-project.org/web/packages/available_packages_by_name.html.

Postoji i pregledniji način pronalaska odgovarajućih paketa, na primjer, na sljedećim poveznicama grupirani su po temi: https://cran.r-project.org/web/views/ https://awesome-r.com/

Već samim uvidom u te popise, može se vidjeti koliko široko se primijenjuje R, od analiza medicinskih slika, kliničkih pokusa i prostornih podataka do statistika za društvene znanosti, obrade prirodnoga jezika i strojnog učenja.

Uz CRAN postoje i drugi repozitoriji, na primjer Bioconductor sadrži spremište paketa specijaliziranih za analizu genomskih podataka.

Paketi se mogu preuzeti i iz drugih izvora, na primjer iz lokalne datoteke ili s GItHub-a od povjerljivog korisnika, no s malo drugačijim naredbama. O tim naredbama nećemo govoriti u početničkom tečaju, no običaj je da se uz informaciju o specifičnom paketu obično navede i točna naredba za njegovo preuzimanje.

U ovom tečaju uglavnom ćemo raditi s funkcijama iz osnovnih paketa.

3. OSNOVNI TIPOVI PODATAKA

Sustav R prepoznaje šest osnovnih ili atomskih tipova podataka:

Tip	Izvorni naziv	Primjeri
realni	double	`5`, `999.111`, `2e5`
cjelobrojni	integer	`1L`, `-100L`, `987654321L`
znakovni	character	`"A"`, `"Učimo R"`, `"99 %"`
logički	logical	`TRUE`, `FALSE`, `T`, `F`
kompleksni	complex	`3+2i`
sirovi	raw	`as.raw(11)`, `charToRaw("Hello")`

Realni tip podatka (engl. *double*) je bilo kakva numerička vrijednost, odnosno vrijednost iz skupa realnih brojeva i svih njegovih podskupova (racionalni, iracionalni i cijeli brojevi). Cjelobrojni tip podatka (engl. *integer*) su cijeli brojevi.

Razlika zapisa cijeloga broja u realnom i cjelobrojnom tipu podatka je u rasponu i zauzimanju memorije. Realni tip podatka definiran je u rasponu +- 1,79e308, dok je raspon integer varijable +- 2.147.483.647, ali zauzima upola manje memorije od realnog tipa.

Cjelobrojne i realne tipove podataka jednim nazivom zovemo numerički tip podatka. U većini slučajeva R će broj pohraniti kao realni tip, a ako baš želimo vrijednost koja je cjelobrojnoga tipa, tada uz broj stavimo oznaku L, na primjer, 42L.

Znakovni tip podatka (engl. *character*) je bilo kakav niz znakova unutar navodnika. Koristi se i izraz string.

Logički tip podatka (engl. *logical*) može poprimiti dvije vrijednosti, `TRUE` ili `FALSE`, a ako nema varijabli koje se zovu `T` ili `F`, vrijednost `TRUE` možemo deklarirati i znakom `T`, a vrijednost `FALSE` znakom `F` (`T` i `F` bez navodnika).

U ovom tečaju nećemo obrađivati kompleksne i sirove tipove podataka jer se oni koriste samo u specifičnim, rijetkim slučajevima.

3.1. Izrada varijabli

Varijable su imenovani spremnici u koje se pohranjuju vrijednosti. Operator kojim se izvršava operacija pohranjivanja, odnosno pridruživanja vrijednosti varijabli je `<-` (skraćeno `[Alt] + [-]`).

Primjer izrade varijabli različitih tipova

```
a <- 1
b <- 1L
c <- "1"
d <- TRUE
```

Tip podatka neke varijable možemo provjeriti funkcijom `typeof()`:

```
typeof(a)
```

```
## [1] "double"
```

```
typeof(b)
```

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

```
typeof(c)
```

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

```
typeof(d)
```

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

Zadatak 1 Izradite po jednu znakovnu, realnu, cjelobrojnu i logičku varijablu i provjerite jesu li ispravnoga tipa podatka.

3.2. Funkcije `is.` i `as.`

Jezik R ima funkcije `is.` i `as.` koje u kombinaciji s tipom podatka (`is.double()`, `is.integer()`, `is.numeric()`, `is.logical()`, `is.vector()`…) mogu provjeriti je li neki objekt određenog tipa, odnosno pretvoriti (ukoliko je to moguće) objekt iz jednog tipa u drugi (`as.double()`, `as.integer()`, `as.numeric()`, `as.logical()`, `as.vector()`…).

Primjer

```
is.integer(b)  # funkcijom is. testiramo je li objekt nekog tipa
```

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

```
b <- as.character(b)  # funkcijom as. konvertiramo objekt u drugi tip
is.character(b)
```

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

Funkcije `is.` i `as.` mogu se kombinirati s gotovo bilo kojim R objektom (matrix, list, data.frame…).

Zadatak 2
Pretvorite logičku varijablu d u cjelobrojni tip podatka. Može li se napraviti pretvorba u obrnutom smjeru, tj. iz cjelobrojnog (i generalno, numeričkog) u logički tip podatka?

3.3. Specijalne vrijednosti

Sustav R ima definiranu vrijednost za beskonačnost i označava se s `Inf` (engl. *infinity*). Može se dobiti kao $n/0$ pri čemu je n bilo koji broj veći od 0.

```
1/0
```

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

```
1/Inf
```

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

```
Inf + Inf
```

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

```
Inf - Inf
```

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

```
is.numeric(Inf)
```

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

```
# ne postoji funkcija is.Inf, nego obrnuto:
is.finite(Inf)
```

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

`Inf` je numerički, realni tip podatka.

Nedostajuće vrijednost u sustavu R označene su oznakom `NA` (not available). Nedefinirane vrijednosti označene su s `NaN` (not a number), na primjer, $0/0$ je nedefinirana vrijednost.

Funkcije kojima ispitujemo je li neka vrijednost nedostajuća ili nedefinirana su redom `is.na()` odnosno `is.nan()`.

```
x <- NA
typeof(x)
```

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

```
is.na(0/0)
```

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

```
is.nan(0/0)
```

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

NAPOMENA: Iako postoje funkcije za testiranje `is.na()` i `is.nan()`, ne postoje funkcije za pretvorbu u `NA` ili `NaN`, tj. ne postoje `as.na()` ili `as.nan()`.

Usporedimo rezultate idućih dviju naredbi:

```
is.nan(5)
```

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

```
is.nan("Ana")
```

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

Funkcija `is.nan()` testira je li numerička vrijednost nedefinirana, ali nije alat kojim se testira je li neka varijabla numerička. Za to nam služi funkcija `is.numeric()`.

```
is.numeric("Ana")
## [1] FALSE
```

```
is.numeric(5)
## [1] TRUE
```

4. OSNOVNE STRUKTURE PODATAKA

U analizi podataka praktično je podatke organizirati i zapisati u definirane strukture koje proizlaze iz matematičkih struktura kakve su primjerice vektori i matrice. Različite strukture podatka zahtijevaju korištenje različitih operacija i funkcija, stoga je bitno upoznati se s osnovnim strukturama podataka u sustavu R.

Strukture podataka mogu se podijeliti prema dimenzionalnosti i homogenosti. Homogene strukture podataka su one u kojima su svi podaci istog tipa, dok su heterogene one koje mogu sadržavati različite tipove podataka.

Dimenzija	Homogene strukture	Heterogene strukture
1	Vektori (*Vector*)	Liste (*List*)
2	Matrice (*Matrix*)	Podatkovni okviri (*Data Frames*)
N	Polja/Nizovi (*Arrays*)	

Funkcijom `class()` može se ispitati kakve je vrste neki objekt (numeric, character, logical, list, matrix, array, data.frame), a detaljnija struktura tog objekta dobije se funkcijom `str()`. Primjeri slijede nakon upoznavanja s pojedinom strukturom podataka.

4.1. Vektori

4.1.1. Izrada vektora

Vektor je uređeni niz elemenata istoga tipa. Izrađuje se pomoću funkcije `c()` (combine).

Primjer izrade vektora različitih tipova:

```
a <- c(TRUE, FALSE, T, F)
b <- c("Krk", "Pag", "Cres")
c <- c(7, 4, 8, 8, 5, 10)
```

Zadatak 3
Izradite po jedan znakovni, realni, cjelobrojni i logički vektor koji sadrže po tri elementa i provjerite kojega su tipa pomoću funkcije `typeof()`.

4.1.2. Pomoćne funkcije za izradu vektora

U slučaju potrebe za izradom duljih vektora, postoje pomoćne funkcije koje to olakšavaju:

Operator dvotočka : generira niz ‘od:do’:

```
a <- 11:15
a

## [1] 11 12 13 14 15
```

Funkcija **`seq()`** je slična operatoru "`:`", ali nudi dodatne varijacije:

```
seq(1, 10)
```

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

```
seq(1, 10, by = 2)
```

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

```
seq(1, 10, length.out = 4)
```

```
## [1]  1  4  7 10
```

Funkcija **`rep()`** replicira zadane vrijednosti na zadani način:

```
rep(c(1, 2, 3), times=2)
```

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

```
rep(c(1, 2, 3), each=2)
```

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

```
rep(c(1, 2, 3), length.out=5)
```

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

Za popis argumenata neke funkcije i pomoć pri korištenju, može se upisati naziv funkcije u karticu Help unutar okna File ili izvršiti naredbu `?+[naziv funkcije]`, na primjer, `?rep`.

NAPOMENA: Kada se unutar funkcije zadaje vrijednost nekom atributu, koristi se operator pridruživanja `=`, a ne `<-`.

Vektor se može izraditi i funkcijom `vector(type, n)`. Njom se dobije trivijalni vektor navedenoga tipa i duljine `n`:

```
vector("character", 5)
```

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

Spajanje vektora može se napraviti pomoću funkcije `c()`:

```
a <- c(1, 2, 3)
b <- c(4, 5, 6)
c <- c(a, b)
c
```

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

Zadatak 4

4.1 Izradite vektor `a` koji sadrži sve negativne cijele brojeve između -10 i -1.
4.2 Izradite vektor `b` koji će sadržavati sve parne brojeve od 1 do 100.
4.3 Realne brojeve u rasponu od -5 do 5 želimo razvrstati u pet razreda. Odredite granice tih razreda.
4.4 Izradite vektor `c` koji sadrži niz brojeva od 0 do 30, bez brojeva od 16 do 20.
4.5 Izradite vektor `d` koji će sadržavati 15 vrijednosti `TRUE` i 15 vrijednosti `FALSE`.

Što se dogodi kada se u funkciji za izradu vektora `c()` navedu vrijednosti različitoga tipa?

Primjer: Odredite tip vektora `l`, `m` i `n`:

```
l <- c(TRUE, 2)
l

## [1] 1 2

m <- c(2, "A")
m

## [1] "2" "A"

n <- c(2L, "A")
n

## [1] "2" "A"
```

Tip vektora možemo ispitati pomoću funkcije `typeof()`:

```
typeof(l)

## [1] "double"

typeof(m)

## [1] "character"

typeof(n)

## [1] "character"
```

Ako se u funkciji za izradu vektora `c()` navedu vrijednosti različitoga tipa, doći će do automatske pretvorbe slabijih tipova podataka u jači tip podatka, pri čemu je jači tip podatka onaj koji zadržava više informacija o svakom elementu.

Redoslijed pretvorbe ide u smjeru:

logical -> integer -> double -> character

Primjer:

```
TRUE   ->   1L   ->   1   ->   "1"
```

4.1.3. Atributi

Strukture podataka mogu imati atribute poput naziva elemenata ili drugih oznaka koje korisnik sam definira.

Nazivi elemenata vektora mogu se zadati na više načina, na primjer funkcijom `names()`:

```
v1 <- c("Sergej", 9, "sladoled", T)
names(v1) <- c("ime", "starost", "najdraža_hrana", "jedinac")
```

```
#vektor v1 sada izgleda ovako:
v1
```

```
##            ime        starost najdraža_hrana        jedinac
##       "Sergej"            "9"     "sladoled"         "TRUE"
```

Nazivi elemenata vektora mogu se odrediti i prilikom izrade vektora:

```
v2 <- c(ime="Alex", starost="10", najdraža_hrana="hamburger", jedinac="FALS
E")
```

Atributi se generalno mogu dodati uz pomoć funkcije `attr()`.

```
attr(v1,"jezik") <-"hrv"
attr(v2, "jezik") <- "eng"
```

Pri ispisu vektora ispisat će se i svi njegovi atributi:

```
v1
```

```
##            ime        starost najdraža_hrana        jedinac
##       "Sergej"            "9"     "sladoled"         "TRUE"
## attr(,"jezik")
## [1] "hrv"
```

Atributi se mogu na uredniji način izlistati i uz pomoć funkcije `attributes()`:

```
attributes(v1)
```

```
## $names
## [1] "ime"           "starost"       "najdraža_hrana" "jedinac"
##
## $jezik
## [1] "hrv"
```

NAPOMENA: Ako vektor ima ikakve dodatne atribute osim `names`, funkcija `is.vector()` ga neće prepoznati kao vektor:

```
is.vector(v1)
```

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

4.1.4. Duljina vektora

Duljina vektora (broj elemenata u vektoru) dobiva se funkcijom `length()`:

```
length(v1)
```

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

Duljina znakova pojedinih elemenata unutar vektora dobiva se funkcijom `nchar()`

```
nchar(v1)
```

```
##            ime        starost najdraža_hrana        jedinac
##              6              1              8              4
```

```
nchar(c(TRUE, FALSE, T, F))
```

```
## [1] 4 5 4 5
```

```
nchar(c(9999L, 66L, -5L))
```

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

Zadatak 5
5.1 Izradite proizvoljni logički vektor naziva `joga` čiji elementi redom imaju nazive: `pon`, `uto`, `sri`, `čet` i `pet`, i ispišite ga.
5.2 Ispitajte tip, klasu i strukturu vektora `joga` pomoću funkcija `typeof()`, `class()` i `str()`.

4.1.5. Selektiranje elemenata vektora

Elementi vektora selektiraju se operatorom `[`.

Primjer:

```
a <- LETTERS
a
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q"
## [18] "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
```

```
a[2]
```

```
## [1] "B"
```

S obzirom na to da operator `[` prima samo jedan argument, u slučaju kada je potrebno izdvojiti više elemenata iz vektora, njihovi indeksi (redni brojevi pozicija u vektoru) sažmu se u novi vektor.

```
#izbor 5., 10. i 15. elementa
a[c(5, 10, 15)]
```

```
## [1] "E" "J" "O"
```

```
#izbor svih elemenata između 11. i 15. člana, uključujući i njih
a[11:15]
```

```
## [1] "K" "L" "M" "N" "O"
```

```
#svaki drugi element, počevši od prvog:
a[seq(1, 26, by = 2)]
```

```
##  [1] "A" "C" "E" "G" "I" "K" "M" "O" "Q" "S" "U" "W" "Y"
```

Na ovaj način mogu se mijenjati vrijednosti pojedinih elemenata vektora:

```
a[1] <- "?"
a
```

```
##  [1] "?" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q"
## [18] "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
```

Ako su elementi vektora imenovani, pojedini elementi mogu se dohvatiti i uz pomoć naziva.

Zadatak 6
Iz zadanog vektora `r` selektirajte:

- deseti element,
- prvih pet elemenata,
- promijenite vrijednost prvoga, drugog i trećeg elementa na "-" i ispišite cijeli vektor,
- zadnji element.

```
r <- letters
```

4.1.6. Brisanje elemenata iz vektora

Element vektora briše se tako da mu se indeks postavi na negativnu vrijednost:

Primjer: Ispišite sve elemente vektora `x` osim drugog elementa.

```
x <- c("A", "B", "C", "D")
x[-2]

## [1] "A" "C" "D"
```

Primjer: Iz vektora `y` izbacite prva dva elementa.

```
y <- 10:20
y <- y[-c(1,2)]
y

## [1] 12 13 14 15 16 17 18 19 20
```

Zadatak 7
7.1 Iz vektora `ljubimci` izbacite uljeza.

```
ljubimci <- c("pas", "mačak", "banana", "papiga", "ribica")
```

7.2 Ispišite vektor `v` bez prvog i zadnjeg elementa.

```
v <- LETTERS
```

4.1.7. Operacije nad vektorima

Vektorizirane funkcije su one koje nad vektorom izvršavaju operaciju tako da je automatski izvrše nad svakim njegovim elementom. S obzirom na to da se tako ponašaju gotovo svi operatori i velik broj funkcija, kaže se da je sustav R vektoriziran. Takav sustav računanja u R-u uvelike olakšava i skraćuje broj koraka potrebnih za neku analizu.

Primjer: Udvostručite vrijednost svakog elementa vektora `t`.

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

## [1]  2  4  6  8 10
```

Primjer: Svaki element vektora `t` uvećajte za 10.

```
t+10
```

```
## [1] 11 12 13 14 15
```

Primjer: Koji su rezultati sljedećih operacija?

```
t^2
```

```
## [1]  1  4  9 16 25
```

```
t*3+5
```

```
## [1]  8 11 14 17 20
```

Primjer: Izračunajte zbroj svih elemenata (`sum()`), minimalnu (`min()`), maksimalnu (`max()`) i srednju vrijednost (`mean()`) vektora `t`:

```
sum(t)
```

```
## [1] 15
```

```
min(t)
```

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

```
max(t)
```

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

```
mean(t)
```

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

Zadatak 8
8.1 Izradite cjelobrojni vektor `r` na način da sadrži sve cjelobrojne vrijednosti od 10 do 50, osim sekvence od 26 do 34. Koliko elemenata sadrži taj vektor?
8.2. Zbrojite sve elemente vektora `r` iz prethodnog zadatka.
8.3. Svaki element vektora `r` podijelite s 2 pa uvećajte za 5.

Primjer: Operacije nad dvama vektorima.

```
t <- c(1, 2, 3, 4, 5)
s <- c(0, 2, 4, 6, 8)

t+s
```

```
## [1]  1  4  7 10 13
```

```
t/s
```

```
## [1]       Inf 1.0000000 0.7500000 0.6666667 0.6250000
```

```
t*s
```

```
## [1]  0  4 12 24 40
```

```
sum(t*s)

## [1] 80

t>s

## [1]  TRUE FALSE FALSE FALSE FALSE

t!=s

## [1]  TRUE FALSE  TRUE  TRUE  TRUE

t==s

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

4.1.8. Princip recikliranja

Što ako vektori nad kojima vršimo operaciju nisu jednakih duljina?

```
t <- c(1, 2, 3, 4, 5)
s <- c(3, 6, 9)

t+s

## [1]  4  8 12  7 11

t/s

## [1] 0.3333333 0.3333333 0.3333333 1.3333333 0.8333333

t*s

## [1]  3 12 27 12 30

sum(t*s)

## [1] 84

t>s

## [1] FALSE FALSE FALSE  TRUE FALSE

t!=s

## [1] TRUE TRUE TRUE TRUE TRUE

t==s

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

Ako dva vektora nad kojima se izvršava vektorizirana operacija nisu jednakih duljina, sustav R će ih napraviti jednako dugima tako da reciklira odnosno replicira kraći vektor dok ne postane jednake duljine kao dulji vektor. U slučaju kada manjeg vektora na kraju treba odrezati, R će dati upozorenje, ali i rezultat.

Zadatak 9
Pokušajte predvidjeti rezultate sljedećih operacija te usporedite sa stvarnim rezultatima:

```
t <- c(1, 2, 3, 4)

t + 1
t + c(1, 2)
t + c(1, 2, 3)

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

4.1.9. Logičko selektiranje elemenata vektora

Nakon upoznavanja s principima vektoriziranja i recikliranja, može se lakše shvatiti logičko selektiranje, koje će svakome tko radi u sustavu R biti od velike koristi.

Elementi vektora mogu se birati i logičkim vektorima te logičkim upitima unutar operatora selektiranja `[]`, pri čemu vrijedi da je `TRUE` vrijednost koja će biti selektirana, a `FALSE` vrijednost koja neće biti selektirana.

Primjer:

```
t <- c(2, 4, 6, 8, 10)
s <- c(TRUE, TRUE, FALSE, TRUE, FALSE)

t[s]
```

```
## [1] 2 4 8
```

S obzirom na to da operacije `<`, `>`, `==`, `!=` rezultiraju logičkim vektorom…

```
t>2
```

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

…sljedeća naredba daje odgovor na upit o tome da se izdvoje elementi vektora `t` čija je vrijednost veća od 2:

```
t[t>2]
```

```
## [1]  4  6  8 10
```

Ako se logičko selektiranje udruži s principom recikliranja, može se postaviti sljedeći upit:

```
t <- c(1, 2, 3, 4, 5, 6,  7, 8, 9, 10)
t[c(T, F)]
```

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

Rezultat ovog upita je vektor čiji su elementi jednaki svakom drugom elementu vektora `t`.

Zadatak 10

10.1 Izradite vektor `m` kao niz cijelih brojeva od 1 do 30. Zatim selektirajte svaki treći element počevši od prvog (1., 4., 7.,…).
10.2 Izradite vektor `k` kao niz brojeva od 1 do 100, pomnožite ga s 3, pa zatim izdvojite sve elemente veće od 280.
10.3 Iz vektora `f` izdvojite sva imena koja se sastoje od tri slova (nchar()).

```
f <- c("Ivan", "Iva", "Branko", "Mia", "Ana")
```

4.1.10. Zadaci za samostalni rad

Zadatak 1: Izradite logički vektor `L`, duljine 5 s trima vrijednostima `TRUE` i dvijema `FALSE`.

Zadatak 2: Koliki je zbroj elemenata vektora `L` iz prethodnog zadatka?

Zadatak 3: Izradite cjelobrojni vektor `r` na način da nosi sve cjelobrojne vrijednosti od 1 do 100, osim sekvence od 5 do 10. Koliko elemenata sadrži taj vektor?

Zadatak 4: Selektirajte prvih pet elemenata vektora iz prethodnog zadatka.

Zadatak 5: Selektirajte sve elemente veće od 20.

Zadatak 6: Svaki treći element vektora `r` iz trećeg zadatka izmijenite u 0.

4.2. Matrice i polja

Matrice su dvodimenzionalni vektori, dok su polja višedimenzionalni vektori. S obzirom na to da ljudi općenito nisu naviknuti na rad s višedimenzionalnim skupovima podataka, polja su rjeđe korištene strukture, pa ćemo se u ovom poglavlju više fokusirati na svojstva matrica.

Matrica se izrađuje funkcijom `matrix()`:

```
matrix(data = 1:12, nrow = 3, ncol = 4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

U prozor Help u oknu Files upišite matrix da dobijete opis te funkcije. Vidimo da matrica posjeduje atribute dimenzije `nrow` i `ncol`, odnosno broj redaka i stupaca, te ima opciju kojom se određuje smjer popunjavanja. Zadano je da se matrica popunjava po stupcima, ali korisnik to može promijeniti po potrebi ako postavi `byrow = TRUE`.

```
matrix(data = 1:12, nrow = 3, ncol = 4, byrow=TRUE)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

Prethodna naredba može se napisati u skraćenoj formi (izostavljanjem naziva parametara) ako se poštuje redoslijed parametara kakav je naveden u Help dokumentaciji:

```
matrix(1:12, 3, 4, T)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

Nazivi redaka i stupaca mogu se zadati pomoću atributa `dimnames`. To je lista koja se sastoji od dvaju vektora od kojih prvi sadrži nazive redaka, a drugi nazive stupaca. U nastavku tečaja detaljnije će se obraditi lista kao struktura podataka.

Primjer:

```
matrix(1:4, 2, 2, dimnames=list(c("Gore", "Dolje"), c("Lijevo", "Desno")))
```

```
##       Lijevo Desno
## Gore       1     3
## Dolje      2     4
```

Drugi način pozivanja i izvedbi svih gornjih funkcija bio bi ovakav:

```
#definiramo vektor
m <- 1:12

#zadamo mu dvije dimenzije 3 x 4
dim(m) <- c(3,4)

#pridružimo matrici m nazive redaka i stupaca
rownames(m) <- c("A", "B", "C")
colnames(m) <- c("1. stupac", "2. stupac", "3. stupac", "4. stupac")

#ispišemo matricu
m
```

```
##   1. stupac 2. stupac 3. stupac 4. stupac
## A         1         4         7        10
## B         2         5         8        11
## C         3         6         9        12
```

Primjer: Provjerimo rezultate sljedećih funkcija:

```
class(m)
```

```
## [1] "matrix"
```

```
typeof(m)
```

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

```
is.matrix(m)
```

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

Zadatak 11
Izradite matricu `M1` dimenzija 3 × 3 koja se sastoji od svih pozitivnih parnih brojeva manjih od 20. Imenujte retke i stupce te matrice po vlastitom izboru.

4.2.1. Selektiranje elemenata matrice

S obzirom na to da je matrica dvodimenzionalna struktura podataka, njene elemente dohvaćamo pomoću operatora `[,]` koji prihvaća dva argumenta. Prvi se odnosi na redni broj retka, a drugi na redni broj stupca.

Primjer: Prisjetimo se matrice `m` iz prethodnog primjera, te izdvojimo element koji se nalazi u prvom retku i trećem stupcu:

```
m[1,3]
```

```
## [1] 7
```

Ako se u operatoru pridruživanja izostavi prvi argument, kao na primjer `m[, 3]` dobit će se treći stupac. Ako se u operatoru pridruživanja izostavi drugi argument, na primjer `m[3,]`, dobit će se treći redak.

Primjer: Ispišite prvi redak matrice `m`.

```
m[1,]
```

```
## 1. stupac 2. stupac 3. stupac 4. stupac
##         1         4         7        10
```

Zadatak 12
12.1 U matrici `m` zamijenite element iz 3. retka i 1. stupca brojem 100 te ispišite rezultat.
12.2 Drugi redak matrice `m` pomnožite s 10 te ispišite cijelu matricu.

Elemente matrice možemo dohvaćati i pomoću naziva redaka i stupaca, ako su zadani.

```
m["A", "2. stupac"]
```

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

Ispis retka `B`:

```
m["B",]
```

```
## 1. stupac 2. stupac 3. stupac 4. stupac
##         2         5         8        11
```

Ispisivanje nekoliko stupaca, na primjer prvog i drugog iz matrice `m` može se dobiti na sljedeći način:

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

```
##   1. stupac 2. stupac
## A         1         4
## B         2         5
## C         3         6
```

Treći način dohvaćanja elemenata matrice je pomoću logičkih upita pri čemu vrijedi da `TRUE` odgovara vrijednosti koja će biti selektirana, a `FALSE` vrijednost koja neće biti selektirana.

Primjer: Ispišite sve elemente matrice `m` veće od 50.

```
m[m>50]
```

```
## integer(0)
```

Zadatak 13
Iz vektora `m2` izradite matricu `M2` dimenzija 5 × 3, a potom iz te matrice izdvojite sve elemente manje od 30.

```
m2 <- seq(1, 90, 6)
```

4.2.2. Spajanje matrica

Matrici se može dodati novi redak, stupac ili druga matrica odgovarajućih dimenzija pomoću funkcije `cbind()` za spajanje preko stupaca te `rbind()` za spajanje preko redaka. Pritom se mora paziti na to da se odgovarajuće dimenzije objekata podudaraju.

Primjer spajanja matrica preko redaka:

```
m <- matrix(1:6, nrow = 2, ncol = 3, byrow=TRUE)
n <- matrix(letters[1:12], nrow = 4, ncol = 3, byrow=TRUE)

rbind(m, n)
```

```
##      [,1] [,2] [,3]
## [1,] "1"  "2"  "3"
## [2,] "4"  "5"  "6"
## [3,] "a"  "b"  "c"
## [4,] "d"  "e"  "f"
## [5,] "g"  "h"  "i"
## [6,] "j"  "k"  "l"
```

Primjer spajanja matrica preko stupaca:

```
M <- matrix(1:12, nrow = 4, ncol = 3)
N <- matrix(LETTERS[1:16], nrow = 4 , ncol = 4)

MN <- cbind(M, N)
MN
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,] "1"  "5"  "9"  "A"  "E"  "I"  "M"
## [2,] "2"  "6"  "10" "B"  "F"  "J"  "N"
## [3,] "3"  "7"  "11" "C"  "G"  "K"  "O"
## [4,] "4"  "8"  "12" "D"  "H"  "L"  "P"
```

Zadatak 14
Provjerite tip i klasu matrice `M`, `N` i `MN`.

Ako se na matricu primijeni funkcija `str()`, dobiju se sljedeće informacije o njenoj strukturi: tip podataka unutar matrice, dimenzije i ispis prvih nekoliko elemenata.

```
str(N)
```

```
##  chr [1:4, 1:4] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N"
"O" ...
```

4.2.3. Transponiranje matrice

Zamjena redaka i stupaca matrice, tzv. transponiranje matrice provodi se pomoću funcije `t()`.

```
n
```

```
##      [,1] [,2] [,3]
## [1,] "a"  "b"  "c"
## [2,] "d"  "e"  "f"
## [3,] "g"  "h"  "i"
## [4,] "j"  "k"  "l"
```

```
t(n)
```

```
##      [,1] [,2] [,3] [,4]
## [1,] "a"  "d"  "g"  "j"
## [2,] "b"  "e"  "h"  "k"
## [3,] "c"  "f"  "i"  "l"
```

4.2.4. Polja

Radnje analogne onima koje se provode nad matricama mogu se provoditi i nad poljima. Za izradu polja koristi se funkcija `array()`, dimenzije se definiraju funkcijom `dim()`, a imenuju funkcijom `dimnames()`. Polje se može transponirati funkcijom `abind()` iz paketa `abind`.

S obzirom na to da se polja rjeđe koriste u praksi, vježbanje operacija nad poljima ostavljamo za samostalni rad.

4.3. Liste

4.3.1. Izrada, selektiranje i imenovanje elemenata liste

Liste su objekti koji mogu sadržavati elemente različitih tipova, uključujući i druge objekte poput liste, matrice, funkcije, itd. Liste se stvaraju pomoću funkcije `list()`. Liste su jednodimenzionalne strukture podataka, iako elementi unutar liste mogu biti višedimenzionalni.

```
lista <- list(10, "Ana", FALSE, matrix(1:20, nrow=5, ncol=4))
lista
```

```
## [[1]]
## [1] 10
##
## [[2]]
## [1] "Ana"
##
## [[3]]
## [1] FALSE
##
```

```
## [[4]]
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

Svaki element liste označen je dvostrukom uglatom zagradom `[[ ]]` i tako se može i selektirati.

```
lista[[2]]
```

```
## [1] "Ana"
```

Imenovanje elemenata liste može se izvršiti, kao i kod vektora, funkcijom `names()` ili izravno tijekom izrade liste.

Prvi način imenovanja elemenata liste:

```
names(lista) <- c("broj", "ime", "status", "mjerenja")
lista
```

```
## $broj
## [1] 10
##
## $ime
## [1] "Ana"
##
## $status
## [1] FALSE
##
## $mjerenja
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

Drugi način imenovanja elemenata liste:

```
lista2 <- list(broj = 5, ime = "Mateo", status = TRUE, mjerenja = matrix(ro
und(runif(20, 0, 20)), 4, 5))
lista2
```

```
## $broj
## [1] 5
##
## $ime
## [1] "Mateo"
##
```

```
## $status
## [1] TRUE
##
## $mjerenja
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   18    7    8   17    1
## [2,]   14   12    5    9   10
## [3,]   10    4    2   19   14
## [4,]   18   16    6   16   19
```

NAPOMENA: U slučaju imenovanja elemenata pri samoj izrade liste, nazivi elemenata nisu navedeni unutar navodnika, dok se kod korištenja funkcije imenovanja elemenata liste `names()` nazivi elemenata moraju navesti u navodnicima.

Ako su elementi liste imenovani, mogu se selektirati uz pomoć operatora `$`:

```
lista$mjerenja
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

Zadatak 15
15.1 Dohvatite element iz prvog stupca i trećeg retka matrice `mjerenja` u listi `lista`.
15.2 Izradite listu koja sadrži pet elemenata po vlastitom izboru i ispišite je.
15.3 Promijenite drugi element te liste.

4.3.2. Dodavanje i brisanje elemenata liste

Novi element liste može se dodati direktno pridruživanjem `<-`:

```
lista[[5]] <- "Peti element"
```

Ako se novi element želi i imenovati, dodavanje i imenovanje novog elementa može se napraviti u jednom koraku.

Primjer: Listi `lista` dodajmo novi element naziva `dodatak` i pridružimo mu vrijednost “20 %”:

```
lista$dodatak <- "20 %"
```

Element liste i drugih objekata brišemo tako da mu pridružimo vrijednost `NULL`. Obrišite prvi element liste `lista` i ispišite je.

```
lista[[1]] <- NULL
```

4.3.3. Izrada liste koja u sebi nosi dvije liste

Najjednostavniji način spajanja dviju lista je pomoću funkcije `c()`. S obzirom na to da je lista jednodimenzionalni objekt, primjena funkcije `c()` ima smisla.

```
listaA <- list(ime="Ana", prezime="Anic", godina=14)
listaB <- list(razred=8, ocjene=c(5, 4, 3), muzicka="TRUE", grad="Zagreb")

listaC <- c(listaA, listaB)
```

Dvije liste mogu se spojiti i pomoću funkcije `list()`, no rezultat je onda lista s dva elementa, od kojih je svaki opet lista. Tu treba biti oprezan s indeksiranjem.

```
listaD <- list(listaA, listaB)
```

Lista će uvijek biti preglednija ako su svi elementi imenovani:

```
listaE <- list(A = listaA, B = listaB)
listaE

## $A
## $A$ime
## [1] "Ana"
##
## $A$prezime
## [1] "Anic"
##
## $A$godina
## [1] 14
##
##
## $B
## $B$razred
## [1] 8
##
## $B$ocjene
## [1] 5 4 3
##
## $B$muzicka
## [1] "TRUE"
##
## $B$grad
## [1] "Zagreb"
```

Vraćanje liste natrag u atomski vektor provodi se funkcijom `unlist()`. Usporedite donja dva rezultata.

```
unlist(listaC)

##      ime  prezime   godina   razred  ocjene1  ocjene2  ocjene3  muzicka
##    "Ana"   "Anic"     "14"      "8"      "5"      "4"      "3"   "TRUE"
##     grad
## "Zagreb"

unlist(listaD)

##      ime  prezime   godina   razred  ocjene1  ocjene2  ocjene3  muzicka
##    "Ana"   "Anic"     "14"      "8"      "5"      "4"      "3"   "TRUE"
```

```
##     grad
## "Zagreb"
```

S obzirom na to da funkcija `unlist()` pretvara listu u vektor, rezultati obaju gornjih slučajeva bit će jednaki.

4.3.4. Atributi liste

Atributi liste i ostalih objekata u R-u ispituju se pomoću funkcija `attributes()` i `attr()`.

```
attributes(listaA)
```

```
## $names
## [1] "ime"     "prezime" "godina"
```

Traženje vrijednosti pojedinog atributa (primjer names):

```
attr(listaA, "names")
```

```
## [1] "ime"     "prezime" "godina"
```

Zadatak 16

16.1 Podsjetite se kako izgledaju `listaD` i `listaE`. Pokušajte procijeniti rezultat pozivanja sljedećih naredbi. Izvršite naredbe i provjerite preklapa li se Vaša procjena s dobivenih rezultatima.

```
listaD[[2]]
listaD[[2]][2]
listaD[[2]][[2]]
listaD[[2]]$ocjene
listaD[[2]][[2]][1]
listaD[[2]]$ocjene[1]

listaE$A
listaE$A$ime
listaE$B$ocjene[2]
```

16.2 Napišite dohvaćanje trećeg elementa liste `A` iz `listaE` na što je moguće više načina.
16.3 Ispitajte tipove podataka, klasu i strukturu `listaE`.
16.4 Dodajte listi `listaD` treći element koji će biti matrica 2 × 2 s vrijednostima 1:4.
16.5 Dodajte listi `B` unutar liste `listaE` element `škola` i pridružite mu vrijednost “Gimnazija”. Ispišite listu `listaE`.
16.6 Ispitajte atribute listi `listaD` i `listaE`.

4.4. Podatkovni okviri

Podatkovni okviri su dvodimenzionalni skupovi podataka koji sadrže elemente različitih tipova, pri čemu svaki njegov stupac unutar sebe sadrži isti tip podatka i svi stupci su istih dimenzija. Ovo je najčešći oblik čuvanja podataka i gotovo svaka analiza u R-u će podrazumijevati rad s podatkovnim okvirom.

Primjer:

Film	Godina	Zarada	Distributor
Black Panther	2018	700.059.566	Buena Vista
Star Wars: The Last Jedi	2017	620.181.382	Buena Vista
Rogue One: A Star Wars Story	2016	532.177.324	Buena Vista
Star Wars: The Force Awakens	2015	936.662.225	Buena Vista
American Sniper	2014	350.126.372	Warner Bros

Stupci podatkovnog okvira predstavljaju varijable opažanja, a svaki redak predstavlja po jednu opservaciju.

Svaki stupac možemo shvatiti kao vektor, a svaki redak kao listu. Također, cijeli podatkovni okvir možemo shvatiti kao listu vektora istih duljina, stoga podatkovni okviri posjeduju svojstva i matrice i liste. Iz tog razloga nad njima možemo primijenjivati funkcije poput `names()`, `colnames()`, `rownames()`, `dim()`, `length()`, `ncol()`, `nrow()`, `rbind()`, `cbind()`, itd.

4.4.1. Izrada podatkovnog okvira

Uobičajen način izrade podatkovnog okvira je ili ‘ručno’ pomoću funkcije `data.frame()`, ili učitavanjem iz vanjske datoteke.

Za izradu podatkovnog okvira pomoću funkcije `data.frame()` potrebno je imati vektore koji će poslužiti kao varijable. U sljedećem primjeru koriste se vektori `Ime`, `Dob`, `VozackaDozvola` i `Grad` kako bi se izradio podatkovni okvir `grupa`:

Prvi način izrade podatkovnog okvira:

```
Ime <- c("Ana", "Iva", "Leo", "Teo", "Mateo")
Dob <- c(25, 31, 28, 33, 30)
VozackaDozvola <- c(FALSE, TRUE, FALSE, TRUE, TRUE)
Grad <- c("Zagreb", "Split", "Zagreb", "Rijeka", "Rijeka")

grupa <- data.frame(Ime, Dob, VozackaDozvola, Grad)
grupa
```

```
##     Ime Dob VozackaDozvola   Grad
## 1   Ana  25          FALSE Zagreb
## 2   Iva  31           TRUE  Split
## 3   Leo  28          FALSE Zagreb
## 4   Teo  33           TRUE Rijeka
## 5 Mateo  30           TRUE Rijeka
```

Drugi način izrade podatkovnog okvira:

```
grupa2 <- data.frame(Ime = c("Ema", "Branko", "David"), Dob = c(26, 27, 22)
, VozackaDozvola = c(TRUE, FALSE, TRUE), Grad = c("Bjelovar", "Zadar", "Kut
ina"))
grupa2
```

```
##      Ime Dob VozackaDozvola     Grad
## 1    Ema  26           TRUE Bjelovar
## 2 Branko  27          FALSE    Zadar
## 3  David  22           TRUE   Kutina
```

Ispitajmo svojstva ovoga podatkovnog okvira sljedećim funkcijama:

```
dim(grupa)
```

```
## [1] 5 4
```

```
length(grupa)
```

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

```
nrow(grupa)
```

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

```
ncol(grupa)
```

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

Uočimo da se `lenght()` sada odnosi na broj stupaca, odnosno `length()` = `ncol()` kada se radi o podatkovnim okvirima.

Zadatak 17
Ispitajte tip i klasu podatkovnog okvira `grupa`.

Nazivi stupaca odnosno varijabli mogu se zadati ili promijeniti funkcijom `names()`. Isto se može napraviti i funkcijom `colnames()`.

```
names(grupa2) <- c("Name", "Age", "DrivingLicence", "City")
grupa2
```

```
##     Name Age DrivingLicence     City
## 1    Ema  26           TRUE Bjelovar
## 2 Branko  27          FALSE    Zadar
## 3  David  22           TRUE   Kutina
```

Nazivi redaka zadaju se funkcijom `rownames()`.

```
rownames(grupa2)<- c("1.", "2.", "3.")
grupa2
```

```
##      Name Age DrivingLicence     City
## 1.    Ema  26           TRUE Bjelovar
## 2. Branko  27          FALSE    Zadar
## 3.  David  22           TRUE   Kutina
```

Napomena: Broj stupaca, odnosno `length()` i `ncol()` ostaju nepromijenjeni bez obzira na to imaju li redci naziv ili ne.

4.4.2. Selektiranje elemenata podatkovnog okvira

S obzirom na to da podatkovni okviri posjeduju svojstva i matrica i listi, pojedine elemente možemo dohvatiti operatorima `[,]` i `$`. Dvostruke uglate zagrade `[[` tehnički se mogu koristiti, ali se u praksi rjeđe primijenjuju na podatkovnim okvirima.

Primjeri dohvaćanja 2. stupca podatkovnog okvira `grupa`:

```
grupa["Dob"]
grupa[2]
grupa[[2]]
grupa$Dob
grupa[,2]
```

Jednodimenzionalni operator `[` kao rezultat vraća podatkovni okvir, dok ostali operatori vraćaju vektor, stoga izbor selektiranja može ovisiti o tome kakav se objekt želi dobiti kao rezultat.

Dohvaćanje pojedinih elementa unutar stupaca može se izvršiti na bilo koji od sljedećih načina:

```
grupa$Dob[1]
```

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

```
grupa[1,2]
```

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

```
grupa[[2]][1]
```

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

U većini slučajeva, od većeg interesa u analizi podataka jest rad nad cijelim stupcem ili dijelom stupca koja zadovoljava neki kriterij, a ne pojedini element, stoga su principi vektorizacije i recikliranja ovdje od velike pomoći. Sve što smo naučili o tome u poglavlju o vektorima, može se primijeniti i na svaki stupac podatkovnog okvira.

Primjeri: 1. Izračunajte prosjek godina osoba iz skupine `grupa` (funkcijom `mean()`).

```
mean(grupa$Dob)
```

```
## [1] 29.4
```

2. Koliko osoba iz skupine `grupa` ima vozačku dozvolu (koristite funkciju `sum()`)?

```
sum(grupa$VozackaDozvola)
```

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

3. Izračunajte prosjek godina osoba koje imaju vozačku dozvolu.

```
mean(grupa$Dob[grupa$VozackaDozvola==TRUE])
```

```
## [1] 31.33333
```

4.4.3. Podskup podatkovnog okvira

Često će od interesa biti podskup podatkovnog okvira koji zadovoljava neki kriterij. Podskup skupa se dobiva naredbom `subset()` koja se također može primijeniti i na vektorima i matricama.

Primjer: Izaberite sve osobe iz podatkovnog okvira grupa koje nisu iz Zagreba:

```
subset(grupa, Grad != "Zagreb")
```

```
##     Ime Dob VozackaDozvola   Grad
## 2   Iva  31           TRUE  Split
## 4   Teo  33           TRUE Rijeka
## 5 Mateo  30           TRUE Rijeka
```

Zadatak 18
Izračunajte prosjek godina svih osoba iz podatkovnog okvira `grupa` koje su iz Zagreba.

4.4.4. Dodavanje novih elemenata

Ako je potrebno nadopuniti podatkovni okvir dodatnim retkom ili redcima, to se može napraviti funkcijom `rbind()`, pri čemu je najsigurnije da se nove retke pretvori u novi podatkovni okvir (funkcijama `as.data.frame()` ili `data.frame()`). Nazivi stupaca obaju podatkovnih okvira moraju biti definirani i ujednačeni.

```
novi <- data.frame(Ime = "Aron", Dob = 34, VozackaDozvola = TRUE, Grad = "Z
agreb")
grupa <- rbind(grupa, novi )
grupa
```

```
##     Ime Dob VozackaDozvola   Grad
## 1   Ana  25          FALSE Zagreb
## 2   Iva  31           TRUE  Split
## 3   Leo  28          FALSE Zagreb
## 4   Teo  33           TRUE Rijeka
## 5 Mateo  30           TRUE Rijeka
## 6  Aron  34           TRUE Zagreb
```

4.4.5. Sortiranje podatkovnog okvira

Za početak upoznajmo se s funkcijom `order()`.

Funkcija `order(x)` vraća indekse u takvom redoslijedu u kojem bi vektor `x` bio sortiran.

Primjer:

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

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

Primjer:

```
order(grupa$Dob)

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

Za sortiranje cijeloga podatkovnog okvira prema nekoj varijabli, koristimo kombinaciju funkcije `order()` i operatora `[`. Pri tome treba voditi računa o tome da je podatkovni okvir dvodimenzionalan, te da operator `[` zahtjeva dva argumenta razdvojena zarezom (`[redak, stupac]`).

Primjer: Sortirajmo podatkovni okvir `grupa` po godinama, odnosno po varijabli `Dob`.

```
grupa[order(grupa$Dob),]

##     Ime Dob VozackaDozvola   Grad
## 1   Ana  25          FALSE Zagreb
## 3   Leo  28          FALSE Zagreb
## 5 Mateo  30           TRUE Rijeka
## 2   Iva  31           TRUE  Split
## 4   Teo  33           TRUE Rijeka
## 6  Aron  34           TRUE Zagreb
```

Logika iza ove naredbe jest da se u operatoru `[,]` prvom argumentu, koji se odnosi na retke, proslijede indeksi u redoslijedu u kojem bi dali sortiranu varijablu `Dob`. Drugi argument koji se odnosi na stupce ostaje prazan što znači da se ispisuju svi stupci.

Primjer: Sortirajte silazno podatkovni okvir `grupa` po varijabli `Dob`.

```
grupa[order(grupa$Dob, decreasing = T),] # argumentom decreasing funkcije o
rder() reguliramo hoće li niz biti uzlazan ili silazan

##     Ime Dob VozackaDozvola   Grad
## 6  Aron  34           TRUE Zagreb
## 4   Teo  33           TRUE Rijeka
## 2   Iva  31           TRUE  Split
## 5 Mateo  30           TRUE Rijeka
## 3   Leo  28          FALSE Zagreb
## 1   Ana  25          FALSE Zagreb
```

4.4.6. Brisanje elemenata iz podatkovnog okvira

Za brisanje elemenata iz podatkovnog okvira koriste se različite metode ovisno o tome što se briše.

- Brisanje stupaca provodi se pomoću `NULL` objekta ili operatora minus (`-`):

```
grupa$Grad <- NULL
grupa
```

```
##     Ime Dob VozackaDozvola
## 1   Ana  25          FALSE
## 2   Iva  31           TRUE
## 3   Leo  28          FALSE
## 4   Teo  33           TRUE
## 5 Mateo  30           TRUE
## 6  Aron  34           TRUE
```

```
grupa[-2]
```

```
##     Ime VozackaDozvola
## 1   Ana          FALSE
## 2   Iva           TRUE
## 3   Leo          FALSE
## 4   Teo           TRUE
## 5 Mateo           TRUE
## 6  Aron           TRUE
```

```
grupa
```

```
##     Ime Dob VozackaDozvola
## 1   Ana  25          FALSE
## 2   Iva  31           TRUE
## 3   Leo  28          FALSE
## 4   Teo  33           TRUE
## 5 Mateo  30           TRUE
## 6  Aron  34           TRUE
```

Za trajno brisanje, potrebna je operacija pridruživanja. Operator minus "-" samo daje prikaz s izostavljenim elementom, stoga je za trajno brisanje potrebno obaviti i pridruživanje.

```
grupa <- grupa[-2]
grupa
```

```
##     Ime VozackaDozvola
## 1   Ana          FALSE
## 2   Iva           TRUE
## 3   Leo          FALSE
## 4   Teo           TRUE
## 5 Mateo           TRUE
## 6  Aron           TRUE
```

- Brisanje redaka može se odraditi pomoću operatora minus “-”:

```
grupa <- grupa[-4,]
grupa
```

```
##     Ime VozackaDozvola
## 1   Ana          FALSE
## 2   Iva           TRUE
## 3   Leo          FALSE
## 5 Mateo           TRUE
## 6  Aron           TRUE
```

Primjer izostavljanja svih neparnih redaka:

```
grupa[-c(1,3,5),]
```

```
##     Ime VozackaDozvola
## 2   Iva           TRUE
## 5 Mateo           TRUE
```

- Pojedinačne vrijednosti unutar podatkovnog skupa brišu se pomoću logičke konstante za nedostajuću vrijednost NA.

```
grupa$VozackaDozvola[grupa$Ime=="Iva"] <- NA
grupa
```

```
##     Ime VozackaDozvola
## 1   Ana          FALSE
## 2   Iva             NA
## 3   Leo          FALSE
## 5 Mateo           TRUE
## 6  Aron           TRUE
```

```
grupa[5,2] <- NA
grupa
```

```
##     Ime VozackaDozvola
## 1   Ana          FALSE
## 2   Iva             NA
## 3   Leo          FALSE
## 5 Mateo           TRUE
## 6  Aron             NA
```

NAPOMENA: Ako se uklanjanje vrijednosti nekog elementa obavlja tako da mu se pridruži prazno polje "“, cijeli stupac će se pretvoriti u tip character, što nije uvijek poželjno, stoga je preporuka staviti NA umjesto”" za prazno polje.

Zadatak 19

19.1 Izradite podatkovni okvir `topLista` pomoću zadanih vektora `Film`, `Godina`, `Zarada` i `Distributer`.

```
Film <- c("Black Panther", "Star Wars: The Last Jedi", "Rogue One: A Star W
ars Story", "Star Wars: The Force Awakens", "American Sniper")
Godina <- c(2018, 2017, 2016, 2015, 2014)
Zarada <- c(700059566, 620181382, 532177324, 936662225, 350126372)
Distributer <- c("Buena Vista", "Buena Vista", "Buena Vista", "Buena Vista"
, "Warner Bros")
```

19.2 Ispišite prva tri retka i prva dva stupca podatkovnog okvira `topLista`.
19.3 Sortirajte podatkovni okvir `topLista` silazno po zaradi.
19.4 Izbrišite stupac `Distributer`.
19.5 Iz podatkovnog skupa `topLista` uklonite prvi zapis (redak).
19.6 Ispišite podatke filmova koji su zaradili više od pola milijarde dolara.

4.4.7. Učitavanje podatkovnih okvira

Drugi način izrade podatkovnog okvira je učitavanjem vanjske datoteke (.txt, .csv…) uz pomoć funkcija `read.csv()`, `read.csv2()`, `read.table()` ili drugih.

Izbor funkcije može ovisiti o separatoru koji je korišten za razdvajanje polja u datoteci.

"csv" (*comma separated values*) je datotečni nastavak za datoteke u kojoj su zapisi najčešće međusobno razdvojeni zarezom ili točkom sa zarezom ";". U slučaju kada su polja razdvojena zarezom, datoteka se najjednostavnije učita naredbom `read.csv()`, a u slučaju kada su polja odvojena točka-zarezom, datoteka se najjednostavnije učita naredbom `read.csv2()`. Funkcija `read.table()` je pogodan izbor kada su podaci razdvojeni bilo kakvom prazninom.

Za sve navedene funkcije, R nudi niz argumenata kojima se mogu podešavati separator, decimalna točka/zarez, navodnici, kodiranje, nedostajuće vrijednosti, a mogu se pronaći u dokumentaciji u oknu Help.

U sljedećem primjeru učitat ćemo datoteku `deniro.csv` koja se nalazi u radnom direktoriju.

```
deniro <- read.csv("Podaci/deniro.csv")
```

NAPOMENA: Za datoteke koje se ne nalaze u radnom direktoriju, treba navesti putanju do datoteke.

Nakon što se učita datoteka, uvijek je dobro provjeriti osnovnu strukturu i njen sadržaj. To se može napraviti pomoću naredbi `head()`, `tail()`, `summary()`, `str()`, itd.

```
head(deniro)
tail(deniro)
summary(deniro)
str(deniro)
```

Ovdje se radi o podatkovnom okviru (engl. *data.frame*) koji prikazuje filmove Robert De Nira s godinom izlaska i ocjenom kritike. Zapis ima 87 opservacija odnosno redaka, te tri varijable odnosno tri stupca koji redom predstavljaju godinu, broj bodova i naziv filma.

Funkcija `str()` navedi i kojega je tipa svaki stupac. Možemo uočiti da je varijabla `Title` tipa Factor, a ne character. No, što je faktor?

4.4.8. Faktori

Faktori su varijable koje mogu poprimiti konačan broj vrijednosti (engl. *levels*), i svaka takva vrijednost predstavlja po jednu kategoriju odnosno razinu. Faktori se zato još zovu i kategorijske varijable. Neke funkcije prihvaćaju samo takve varijable jer zahtijevaju konačan broj unaprijed definiranih vrijednosti, stoga je bitno razlikovati varijable koje jesu faktori i one koje to nisu.

Faktori se mogu izraditi pomoću funkcije `factor()`:

```
x <- factor(c("lijevo", "desno", "desno", "lijevo"))
```

Razine faktora su izvedene automatski ako ih sami ne specificiramo, a možemo ih provjeriti funkcijom `levels()`. To je vektor jedinstvenih vrijednosti koje neki faktor može poprimiti. Ako sami ne specificiramo redoslijed, bit će poredani uzlazno.

```
levels(x)
```

```
## [1] "desno"  "lijevo"
```

Razina može biti i više nego što ih je prikazano u faktoru, na primjer:

```
x <- factor(c("lijevo", "desno", "desno", "lijevo"), levels = c("lijevo", "
sredina", "desno"))
x
```

```
## [1] lijevo desno  desno  lijevo
## Levels: lijevo sredina desno
```

Uočimo da kad sami specificiramo razine, uzlazni poredak se gubi.

Ako je potrebno znati broj razina nekoga faktora, koristimo funkciju `length()`:

```
#ukupni broj razina faktora x:
length(levels(x))
```

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

```
#korišteni broj razina faktora x
length(unique(x))
```

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

Funkcija `unique()` vraća jedinstvene vrijednosti objekta.

Što se dešava kada pokušamo izmijeniti faktor?

```
x[1] <- "sredina"
x
```

```
## [1] sredina desno   desno   lijevo
## Levels: lijevo sredina desno
```

```
x[1] <- "gore"
x
```

```
## [1] <NA>   desno  desno  lijevo
## Levels: lijevo sredina desno
```

Faktor možemo mijenjati, ali samo vrijednostima koje su predefinirane kao razine (`levels`).

Ako želimo dodati novu razinu već postojećem faktoru, to možemo napraviti funkcijom `levels()`:

```
levels(x) <- c(levels(x), "dolje", "gore")
x
```

```
## [1] <NA>   desno  desno  lijevo
## Levels: lijevo sredina desno dolje gore
```

Faktori mogu biti i u obliku brojeva, međutim sustav ih doživljava samo kao nazive kategorija te se s takvim brojevima ne mogu obavljati matematičke operacije.

```
y <- factor(c(5,3,4,3,3))
y
```

```
## [1] 5 3 4 3 3
## Levels: 3 4 5
```

```
y[1]+y[2]
```

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

Često će biti korisno takve faktore prebaciti natrag u brojeve, a to možemo napraviti pomoću funkcija `as.numeric()` i `as.character()`.

```
as.numeric(y)
```

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

U ovom slučaju, `as.numeric()` će vratiti samo redne brojeve razina u redoslijedu u kojem se nalaze u faktoru. O čemu se tu radi?

Da bi se dobio odgovor na to pitanje, treba pogledati strukturu faktora:

```
str(y)
```

```
##  Factor w/ 3 levels "3","4","5": 3 1 2 1 1
```

Još jedan primjer:

```
boje <- factor(c("Plava","Crvena", "Crvena", "Plava", "Crvena"))
str(boje)
```

```
##  Factor w/ 2 levels "Crvena","Plava": 2 1 1 2 1
```

Naime, svaki faktor je pohranjen kao cjelobrojni vektor koji predstavlja redne brojeve svojih razina.

S obzirom na to, da bi se faktor prebacio u numeričku vrijednost, moramo prvo prebaciti faktor u znakovni tip pomoću `as.character()`, te onda prebaciti taj rezultat u numerički pomoću `as.numeric()`:

```
as.numeric(as.character(y))
```

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

Ukoliko želimo stupac podatkovnog okvira prebaciti u faktore, to možemo napraviti funkcijom `as.factor()`.

```
deniro$Title <- as.factor(deniro$Title)
str(deniro)
```

```
## 'data.frame':    87 obs. of  3 variables:
##  $ Year : int  1968 1970 1970 1971 1973 1973 1974 1976 1976 1977 ...
##  $ Score: int  86 17 73 40 98 88 97 41 99 47 ...
##  $ Title: Factor w/ 87 levels "15 Minutes","1900",..: 28 12 33 13 47 10
74 78 67 2 ...
```

Funkcije `read.csv()` i `read.csv2()` imaju zadanu postavku prema kojoj se tijekom učitavanja svaki znakovni stupac prebacuje u faktore. Bolja praksa je sam odlučiti koji će stupci biti faktorski, a koji znakovni, pa se obično te funkcije pozivaju na sljedeći način:

```
deniro <- read.csv("Podaci/deniro.csv", stringsAsFactors = FALSE)
str(deniro)
```

```
## 'data.frame':    87 obs. of  3 variables:
##  $ Year : int  1968 1970 1970 1971 1973 1973 1974 1976 1976 1977 ...
##  $ Score: int  86 17 73 40 98 88 97 41 99 47 ...
##  $ Title: chr  "Greetings" "Bloody Mama" "Hi Mom!" "Born to Win" ...
```

4.4.9. Funkcija `table()`

Funkcija `table()` proizvodi kontingencijsku tablicu, tj. tablicu koja nudi uvid u brojčano stanje elemenata. Drugim riječima, ona popisuje čega sve ima i koliko te je kao takva korisna za analize podatkovnih okvira.

`table()` prima jedan ili više argumenata, koji su idealno faktorske varijable ili barem varijable koje bi imalo smisla pretvoriti u faktore (na primjer decimalne brojeve ne bi stavili kao argument funkciji `table()`).

Za potrebe demonstracije kreirajmo podatkovni okvir `uzorak` koji sadrži podatke o spolu, boji kose i boje očiju nekih osoba.

```
oci <- rep(c("Plave", "Smeđe", "Zelene"), 5)
kosa <- rep(c("Smeđa", "Crna", "Plava", "Crvena", "Sijeda"), 3)
spol <-  c(rep("Muški", 7), rep("Ženski", 8))

uzorak <- as.data.frame(cbind(spol, kosa, oci))
uzorak
```

```
##      spol   kosa    oci
## 1   Muški  Smeđa  Plave
## 2   Muški   Crna  Smeđe
## 3   Muški  Plava Zelene
## 4   Muški Crvena  Plave
## 5   Muški Sijeda  Smeđe
## 6   Muški  Smeđa Zelene
## 7   Muški   Crna  Plave
## 8  Ženski  Plava  Smeđe
## 9  Ženski Crvena Zelene
## 10 Ženski Sijeda  Plave
## 11 Ženski  Smeđa  Smeđe
## 12 Ženski   Crna Zelene
## 13 Ženski  Plava  Plave
## 14 Ženski Crvena  Smeđe
## 15 Ženski Sijeda Zelene
```

Primjer funkcije `table()` kada primi jedan argument:

```
table(uzorak$spol)
```

```
##
##  Muški Ženski
##      7      8
```

```
table(uzorak$kosa)
```

```
##
##   Crna Crvena  Plava Sijeda  Smeđa
##      3      3      3      3      3
```

```
table(uzorak$oci)
```

```
##
##  Plave  Smeđe Zelene
##      5      5      5
```

Primjer funkcije `table()` kada primi dva argumenta:

```
table(uzorak$spol, uzorak$kosa)
```

```
##
##          Crna Crvena Plava Sijeda Smeđa
##   Muški     2      1     1      1     2
##   Ženski    1      2     2      2     1
```

Primjer funkcije `table()` kada primi tri argumenta:

```
table(uzorak$spol, uzorak$kosa, uzorak$oci)
```

```
## , ,  = Plave
##
##
##          Crna Crvena Plava Sijeda Smeđa
##   Muški     1      1     0      0     1
##   Ženski    0      0     1      1     0
##
## , ,  = Smeđe
##
##
##          Crna Crvena Plava Sijeda Smeđa
##   Muški     1      0     0      1     0
##   Ženski    0      1     1      0     1
##
## , ,  = Zelene
##
##
##          Crna Crvena Plava Sijeda Smeđa
##   Muški     0      0     1      0     1
##   Ženski    1      1     0      1     0
```

Primjer: U kojoj godini/godinama je Robert De Niro snimio najviše filmova?

```
sort(table(deniro$Year), decreasing = TRUE)
```

```
##
## 1991 2013 1987 1990 1993 1996 1997 2000 2005 2010 2011 2012 2015 1970 1973 1976
##    4    4    3    3    3    3    3    3    3    3    3    3    3    2    2    2
## 1977 1984 1989 1992 1995 1998 1999 2001 2002 2004 2007 2008 1968 1971 1974 1978
##    2    2    2    2    2    2    2    2    2    2    2    2    1    1    1    1
## 1980 1981 1983 1985 1986 1988 1994 2003 2006 2009 2014 2016
##    1    1    1    1    1    1    1    1    1    1    1    1
```

Zadatak 20

20.1 Koje bodove su najčešće dobivali filmovi Roberta De Nira od kritike?

20.2 Funkcijom `order()` sortirajte cijeli podatkovni okvir deniro silazno prema osvojenim bodovima (`Score`).

20.3 Koji je film Roberta De Nira osvojio najvišu ocjenu kritike i kada je snimljen? Za rješavanje ovoga zadatka preporučuje se uporaba funkcije `which.max()`. Proučite funkciju u dokumentaciji u oknu Help.

4.4.10. Podatkovni okviri ugrađeni u R

Treći način dolaska do podatkovnih okvira su skupovi podataka koji su već ugrađeni u sustav R ili u pakete.

Primjer: Podatkovni okvir `iris` jedan je od korištenijih i dio je ugrađene zbirke podataka "datasets" sustava R. Poziva se naredbom `data(iris)`.

```
data(iris)
data(mtcars)
```

Popis svih dostupnih podatkovnih skupova može se dobiti naredbom `data()`:

```
data()
```

Za svaki takav podatkovni skup postoji dokumentacija u kojoj su dane informacije o skupu i varijablama. Možemo ih dobiti standardnim putem pomoću naredbe "`?`" ili direktnim upisom u karticu Help.

```
?iris
```

5. FUNKCIJE SUSTAVA R

Funkcija je skupina izraza koja izvršava neki zadatak. Osnovna sintaksa svake funkcije je

```
ime_funkcije <- function(argument1, argument2, ...){

izraz1
izraz2
...

}
```

Svaka funkcija ima svoj naziv po kojem se poziva.

Argumenti su ulazni podaci koji trebaju funkciji da izvrši neki zadatak. Funkcija može, a i ne mora imati argumente, i argumenti mogu imati unaprijed zadane vrijednosti.

`izraz1`, `izraz2`… se zajedno zovu tijelo funkcije i organizirani su tako da izvršavaju zadatak za koji je funkcija namijenjena.

Prednost sustava R je to što ima velik broj već unaprijed definiranih funkcija. Svaka takva funkcija pripada nekom paketu. Neke funkcije su ugrađene u sustav R putem osnovnih paketa, kao na primjer `mean()` za izračun srednje vrijednosti i `c()` za izradu vektora koji pripadaju paketu `{base}`, druge se nalaze u vanjskim paketima koji su specijalizirani za lakše obavljanje određenih zadataka. Po potrebi i sam korisnik može definirati svoje funkcije prema gornjoj sintaksi.

Unutar sustava R s funkcijama se može činiti sve što i s vektorima - mogu se dodijeliti nekoj varijabli, spremiti u listu, biti argumenti nekih drugih funkcija. Čak je moguće izraditi bezimenu funkciju te funkciju unutar funkcije. Funkcije čine najmoćniji dio sustava R.

Unutar istog paketa ne mogu postojati dvije funkcije istoga naziva, no mogu postojati funkcije istoga naziva iz različitih paketa. U tom slučaju, funkcija iz paketa koji je zadnji učitan će se automatski izvršavati, a ako se želi osigurati na koji paket se odnosi, funkcija se poziva pomoću sintakse `ime_paketa::ime_funkcije()`.

Primjer.
`stats::filter()`
`dplyr::filter()`

5.1. Dokumentacija funkcija

S obzirom na velik broj već postojećih funkcija i na činjenicu da se često jedna funkcija može koristiti na više načina, gotovo svaki korisnik R-a se često nađe u situaciji gdje mora pretraživati i istraživati funkcije.

Početno mjesto za upoznavanje s nekom funkcijom jest njena dokumentacija. U njoj se može naći sintaksa funkcije, opis argumenata, primjeri, itd.

Primjer: Pogledajmo dokumentaciju za funkciju `matrix()` direktnim upisom riječi *matrix* u karticu Help ili naredbom `?matrix`:

```
?matrix
```

Iz cijele dokumentacije posebnu pozornost trebamo obratiti na dijelove `Description`, `Usage`, `Arguments` i `Examples`.

```
matrix {base}

Description
matrix creates a matrix from the given set of values.

Usage
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

Na vrhu dokumentacije u vitičastim zagradama naveden je naziv paketa kojem funkcija pripada. U poglavlju `Description` dan je opis zadatka koji funkcija obavlja, a u poglavlju `Usage` navedeno je kako se konkretno funkcija poziva. Tu se vidi koje argumente funkcija može primiti te koji od njih imaju unaprijed zadane vrijednosti. Dalje u poglavlju `Arguments` detaljnije su pojašnjeni svi argumenti funkcije, navedene su i sve opcionalne napomene, a na samom kraju u poglavlju `Examples` dani su i primjeri. Takva dokumentacija postoji za svaku funkciju iz sustava R i iz paketa koji se mogu preuzeti s CRAN stranice.

Funkcije posjeduju imenovane argumente. Neki od njih imaju zadane predodređene vrijednosti (engl. *default values*), a svim drugima korisnik treba postaviti neku vrijednost da bi se funkcija mogla pozvati. Argumenti koji imaju predodređene vrijednosti ne trebaju se navoditi u pozivu funkcije, osim ako im se ne želi promijeniti zadana vrijednost. Preporuka je da se korisnik upozna sa svakom funkcijom prije korištenja.

Iz primjera funkcije `matrix()` vidimo da su svi argumenti unaprijed zadani, tako da je moguće pozvati funkciju `matrix()` bez argumenata, no time se dobije trivijalna 1 × 1 matrica s `NA` elementom.

Argumenti se s funkcijom spajaju (engl. *match*) na tri načina: prema nazivu, prema djelomičnom nazivu ili prema poziciji.

Primjer: poziv funkcije matrix prema načinu spajanja argumenata.

- spajanje prema nazivu

```
matrix(data = 1:12, nrow = 3, ncol = 4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
# kod spajanja prema imenu, redoslijed argumenata je nebitan
matrix(ncol = 4, data = 1:12, nrow = 3)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

- spajanje prema djelomičnom nazivu

```
matrix(da = 1:12, nr = 3, nc = 4) # navedemo najkraću verziju naziva argumenta tako da ono bude jedinstveno za tu funkciju (npr. 'd = ' se može odnositi i na data i na dimnames tako da se za argument data treba koristiti 'da', 'dat' ili 'data')
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

- spajanja prema poziciji

```
matrix(1:12, 3, 4) # kod spajanja prema poziciji, pozicija argumenata, odnosno redoslijed je presudan
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

Zadatak 21
21.1 Otvorite dokumentaciju funkcije `seq()`.
21.2 Iskopirajte sve primjere iz dokumentacije funkcije `seq()` (pod `Examples`) i izvršite ih.
21.3 Isprobajte sva tri načina kojima, pomoću funkcije `seq()`, možete ispisati niz od svakog četvrtog broja između 0 i 100 (0, 4, 8, 12, …, 96, 100). Ova funkcija prihvaća samo jedan dodatni argument istovremeno i pomoću svakog od njih (`by`, `length.out`, `along.with`) probajte dobiti traženi niz. Za pomoćni vektor, gdje je potrebno, koristite `letters`.

5.2. Funkcije iz grupe apply

Funkcije iz obitelji apply alternativna su rješenja za petlje, odnosno mehanizam koji koristimo umjesto višestrukog izvršavanja istoga programskog kôda.

Na primjer, u slučaju potrebe za ponavljanjem iste funkcije po redcima ili stupcima nekog objekta, umjesto petlje kojom bi se prolazilo po svakom retku/stupcu, koristi se neka iz obitelji ovih funkcija. Izbor funkcije ovisi o tome nad kojim se objektom vrši radnja i kakav objekt želimo za rezultat.

Od funkcija iz obitelji apply na ovom tečaju obradit će se `apply()`, `lapply()`, `sapply()` i `tapply()`, te dodatna funkcija `aggregate()` povezana s ovom skupinom.

5.2.1. Funkcija `apply()`

Funkcija `apply()` koristi se nad poljima, a radi jednostavnosti ovdje ćemo se ograničiti na dvodimenzionalna polja, odnosno matrice. Funkcija se koristi na sljedeći način:

`apply(X, MARGIN, FUN)` gdje je

- `X` matrica,
- `MARGIN` je indikator retka ili stupca,
- `FUN` je funkcija koja će se primijenjivati na margine (`MARGIN`) matrice `X`.

Primjer: Zbrojimo elemente svakoga stupca matrice `M`:

```
M <- matrix(1:12, 3, 4)
M
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
apply(M,  1, sum) #zbroj po redcima
```

```
## [1] 22 26 30
```

```
apply(M,  2, sum) #zbroj po stupcima
```

```
## [1]  6 15 24 33
```

Tako izračunati redak ili stupac može se dodati matrici `M`.

```
total <- apply(M, 2, sum) #zbrajamo po stupcima
M <- rbind(M, total)
M
```

```
##       [,1] [,2] [,3] [,4]
##          1    4    7   10
##          2    5    8   11
##          3    6    9   12
## total    6   15   24   33
```

Zadatak 22
Iz zadanog vektora `sample` izradite matricu `A` dimenzija 5 × 4 te joj dodajte stupac `Srednja_Vrijednost` koji sadrži srednju vrijednost svakoga retka (funkcija `mean()`).

```
sample <- sample(100, 20)
```

5.2.2. Funkcija `lapply()`

`lapply()` je proširenje funkcije `apply()` koje se može primijeniti i na podatkovne okvire i liste, a rezultat se vraća isključivo u obliku liste (odakle dolazi i "l" u nazivu funkcije).

Poziva se na sljedeći način:

`lapply(X, FUN, ...)` gdje je

- `X` vektor ili lista, a sve ostalo će biti automatski pretvoreno u listu,
- `FUN` funkcija koja će se primijenjivati nad X,
- `...` dodatne opcije funkcije FUN

NAPOMENA: U pozivu funkcije `lapply()` argument `FUN` nalazi se na drugom mjestu, dok je u pozivu funkcije `apply()` na trećem mjestu.

Primjer: Izdvajanje redaka ili stupaca iz liste matrica

Izradimo listu matrica `ABC`:

```
A <- matrix(1:12, 4, 3)
B <- matrix(11:30, 5, 4)
C <- matrix(LETTERS[1:24], 6, 4)
ABC <- list(A, B, C)
ABC
```

```
## [[1]]
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
##
## [[2]]
##      [,1] [,2] [,3] [,4]
## [1,]   11   16   21   26
## [2,]   12   17   22   27
## [3,]   13   18   23   28
## [4,]   14   19   24   29
## [5,]   15   20   25   30
##
## [[3]]
##      [,1] [,2] [,3] [,4]
## [1,] "A"  "G"  "M"  "S"
## [2,] "B"  "H"  "N"  "T"
## [3,] "C"  "I"  "O"  "U"
## [4,] "D"  "J"  "P"  "V"
## [5,] "E"  "K"  "Q"  "W"
## [6,] "F"  "L"  "R"  "X"
```

Iz liste matrica `ABC` izdvojimo drugi redak iz svake matrice:

```
lapply(ABC, "[", 2,)
```

```
## [[1]]
## [1]  2  6 10
##
## [[2]]
## [1] 12 17 22 27
##
## [[3]]
## [1] "B" "H" "N" "T"
```

Iz liste matrica `ABC` izdvojimo prvi stupac iz svake matrice:

```
lapply(ABC, "[", ,1)
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 11 12 13 14 15
##
## [[3]]
## [1] "A" "B" "C" "D" "E" "F"
```

Zadatak 23
Pomoću funkcije `lapply()` pretvorite matrice iz liste `ABC` u tekstualni tip podatka (funkcijom `as.character()`).

Ako se nakon konverzije podaci žele zadržati u obliku matrica, onda je bolje konverziju napraviti na sljedeći način:

```
lapply(ABC, apply, 2, as.character )
```

```
## [[1]]
##      [,1] [,2] [,3]
## [1,] "1"  "5"  "9"
## [2,] "2"  "6"  "10"
## [3,] "3"  "7"  "11"
## [4,] "4"  "8"  "12"
##
## [[2]]
##      [,1] [,2] [,3] [,4]
## [1,] "11" "16" "21" "26"
## [2,] "12" "17" "22" "27"
## [3,] "13" "18" "23" "28"
## [4,] "14" "19" "24" "29"
## [5,] "15" "20" "25" "30"
##
```

```
## [[3]]
##      [,1] [,2] [,3] [,4]
## [1,] "A"  "G"  "M"  "S"
## [2,] "B"  "H"  "N"  "T"
## [3,] "C"  "I"  "O"  "U"
## [4,] "D"  "J"  "P"  "V"
## [5,] "E"  "K"  "Q"  "W"
## [6,] "F"  "L"  "R"  "X"
```

Ovim postupkom je svaki stupac u matrici konvertiran u tip character funkcijom `apply()`, a onda smo to ponovili za svaku matricu u listi pomoću funkcije `lapply()`.

5.2.3. Funkcija `sapply()`

Funkcija `sapply()` radi isto što i `lapply()`, osim što za rezultat vraća najjednostavniju moguću strukturu podataka. Na primjer, ako je moguće umjesto liste vratiti odgovor u obliku vektora, `sapply()` će to i napraviti. “s” u nazivu dolazi od simplify.

Primjer: Razlika rezultata funkcija `sapply()` i `lapply()`

```
sapply(ABC, min)
```

```
## [1] "1"  "11" "A"
```

```
lapply(ABC, min)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 11
##
## [[3]]
## [1] "A"
```

5.2.4. Funkcije `tapply()` i `aggregate()`

Analiza podataka često zahtjeva računanje ili uspoređivanje svojstava prema podgrupama nekoga skupa podataka, na primjer, da se podaci podijele po spolu, državi, gradu, razredu, itd., i to često dovodi do zanimljivih otkrića. Tu je od velike pomoći `tapply()`. Funkcija `tapply()` podijeli podatkovni skup u kategorije, te za svaku kategoriju izračuna traženu funkciju.

```
tapply(X, INDEX, FUN = NULL)
```

- `X` - podatkovni skup, po mogućnosti vektor, za koji je moguće napraviti podjelu po kategorijama,
- `INDEX` - lista elemenata po kojoj se određuju kategorije,
- `FUN` - funkcija koja se primijenjuje na svaku podgrupu.

Prisjetimo se skupa podataka `iris`. To je podatkovni skup o duljinama i širinama čašičnih listića (engl. *sepal*) i latica (engl. *petal*) danih za tri vrste iris cvjetova. Vrste cvjetova (engl. *species*) su faktori.

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1
1 1 1 1 1 ...
```

Primjer: Usporedite srednje vrijednosti duljina latica (engl. *petal length*) za sve tri vrste cvijeta iris.

```
tapply(iris$Petal.Length, iris$Species, mean)
```

```
##     setosa versicolor  virginica
##      1.462      4.260      5.552
```

Ako se ovaj postupak želi ponoviti za više varijabli (stupaca) koristi se funkcija

```
aggregate(X, by, FUN)
```

gdje je

- `X` - podatkovni okvir nad kojim se želi izvršiti podjela u podgrupe i izračunati neka funkcija,
- `by` - lista faktora prema kojoj se rade podgrupe (često se mora pretvoriti u listu (list(X$by)),
- `FUN` - funkcija.

Primjer: Izračun srednjih vrijednosti za sve četiri varijable (duljine i širine latica i čašičnih listića) ovisno o vrsti cvijeta iris.

```
aggregate(iris[,1:4], list(Vrste = iris$Species), mean)
```

```
##        Vrste Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     setosa        5.006       3.428        1.462       0.246
## 2 versicolor        5.936       2.770        4.260       1.326
## 3  virginica        6.588       2.974        5.552       2.026
```

Zadatak 24
Prisjetite se kako izgleda podatkovni skup `deniro`. Pomoću funkcije `tapply()` nađite u kojoj godini je De Niro snimio najviše filmova. Za prebrojavanje koristite funkciju `length()`.

5.3. Popis operatora u sustavu R

U nastavku slijedi popis operatora, aritmetičkih i logičkih, za izvršavanje operacija nad varijablama koje bi mogle biti korisne u daljnjem radu.

Aritmetički operatori:

- + zbrajanje
- - oduzimanje
- * množenje
- / dijeljenje
- ^ ili ** potenciranje
- %% modulo (ostatak pri cjelobrojnom dijeljenju).

Logički operatori:

- <, <=, >, >=, == manje, manje ili jednako, veće, veće ili jednako, jednako
- != različito
- ! negacija (!x - nije x)
- x | y - x ili y
- x & y - x i y.
- || - skalarni i (nije vektoriziran)
- && skalarni ili (nije vektoriziran).

Primjer: Ispišite sve parne brojeve iz vektora a.

```
# Za testiranje parnih brojeva koristimo operator modulo %%:
a <- seq(1, 100, 3)
a[a %% 2 == 0]
```

```
##  [1]   4  10  16  22  28  34  40  46  52  58  64  70  76  82  88  94 100
```

Zadatak 25
Svaki element vektora b kvadrirajte, a potom ispišite samo neparne elemente.

```
b <- 1:10
```

Zadatak 26
Pokušajte predvidjeti rezultate sljedećih logičkih operacija, a potom ih izvršite i usporedite sa svojim odgovorom.

```
x <- c(1, 0, 1, 1, 0)
y <- c(0, 1, 0, 1, 1)

x==y
x>y
x!=y
!x
x&y;      x&&y
x|y;      x||y
```

5.4. Ostale korisne funkcije

Svaka funkcija pripada nekom paketu, a paketi se obično pronalaze i učitavaju ovisno o potrebama zadataka koji se obavljaju. Međutim, u gotovo svakom radu, dobro je znati i neke pomoćne, sporedne funkcije koje olakšavaju rad u sustavu R.

5.4.1. Oblikovanje brojeva

- `options(digits=n)` - globalna postavka za ispis brojeva s minimalno n značajnih mjesta. Značajna mjesta su decimalna mjesta za brojeve između $< -1{,}1 >$, te zbroj dekadskih i decimalnih mjesta za ostale brojeve, pri čemu veći prioritet imaju dekadska mjesta.

Primjer:

```
options(digits = 7)
1/3
```

```
## [1] 0.3333333
```

```
options(digits = 2)
1/3
```

```
## [1] 0.33
```

```
options(digits = 2)
100/3
```

```
## [1] 33
```

```
options(digits = 4 )
100/3
```

```
## [1] 33.33
```

```
options(digits = 7) #vraćanje na standardnu postavku
```

- Popis ostalih globalnih postavki može se naći u Help kartici upitom “options”.
- `format()` - služi za oblikovanje objekta za prikladniji ispis u kojem se mogu definirati postavke poput decimalne točke odnosno decimalnog zareza, broja decimala, ispisa broja u eksponencijalnom obliku (argument `scientific`), itd.

```
x=1000000/3
format(x, big.mark = ".", decimal.mark = ",", nsmall = 4)
```

```
## [1] "333.333,3333"
```

```
format(5.4e16)
```

```
## [1] "5.4e+16"
```

```
format(5.4e16, scientific = FALSE)
```

```
## [1] "54000000000000000"
```

- `round()` - funkcija kojom definiramo precizan broj decimalnih mjesta u rezultatu (ne samo ispisu) nekoga broja

```
round(10/3, 1)*2
```

```
## [1] 6.6
```

```
round(10/3*2, 1)
```

```
## [1] 6.7
```

5.4.2. Generiranje slučajnih brojeva

- `sample()` - generator slučajnih brojeva (s ili bez ponavljanja: `replace = TRUE|FALSE`)

Primjer: Ponovite naredbu `sample(100,4)` nekoliko puta. Što uočavate?

```
sample(100, 4)
```

```
## [1]  5 44 19 83
```

- `set.seed()` - služi za definiranje početne vrijednosti generatora slučajnih brojeva, odnosno, ovime se osigurava da se uvijek ponovi isti slučajan uzorak.

Primjer: Izvršite cijeli donji blok nekoliko puta. Što uočavate ovdje?

```
set.seed(1)
sample(100,4)
```

```
## [1] 68 39  1 34
```

5.4.3. Help funkcije

- `apropos(" ")` - daje popis svih funkcija koje u sebi imaju navedeni pojam.

```
apropos("read")
```

```
##  [1] ".readRDS"         "read.csv"         "read.csv2"        "read.dcf"
##  [5] "read.delim"       "read.delim2"      "read.DIF"         "read.fort
ran"
##  [9] "read.ftable"      "read.fwf"         "read.socket"      "read.tabl
e"
## [13] "read_chunk"       "read_demo"        "read_rforge"      "readBin"
## [17] "readChar"         "readCitationFile" "readClipboard"    "readline"
## [21] "readLines"        "readRDS"          "readRegistry"     "readRenvi
ron"
## [25] "Sys.readlink"
```

- `library(help="ime_paketa")` - popis svih funkcija iz navedenog paketa.

```
library(help="base")
```

5.4.4. Funkcije za ispis na ekran

- `print()` - ispis na ekran.

```
vektor <- c(1, 2, 3)
print(vektor)

## [1] 1 2 3
```

- `cat()` - najjednostavniji mogući ispis na ekran, uz mogućnost spajanja objekata za ispis (concatenate and print).

```
cat(vektor, " -> Ovo je naš vektor.")

## 1 2 3  -> Ovo je naš vektor.
```

5.4.5. Funkcije za brisanje i pohranjivanje objekata

- `rm()` ili `remove()` - brisanje datoteka, varijabli, funkcija iz radne okoline.
- `rm(list = ls())` - brisanje svih objekata iz radnog okruženja.

```
rm(vektor)
```

- `save()` - pohranjivanje objekata iz radnog okruženja u vanjsku datoteku (obično joj se dodijeli ekstenzija .Rdata). Takvi objekti mogu se učitavati nazad u radno okruženje funkcijama `load()`, `attach()` ili u nekim slučajevima `data()`.

Primjer: Kreirajmo proizvoljne vektore `a` i `b`, i pohranimo ih u datoteku `vektori`.

```
a <- c("A", "B", "C")
b <- seq(1, 100, 7)
save(a, b, file = "vektori.RData")  # pohranjivanje objekata a i b
rm(a, b)                            # brisanje objekata a i b
load("vektori.Rdata")               # ponovno učitavanje objekata a i b
```

- `save.image()` - pohranjivanje cijeloga radnog okruženja koje se kasnije može ponovno učitati. Obično se pohranjuje u datoteku ekstenzije .RData.

```
save.image("Funkcije.RData")  # spremanje radnog okruženja
rm(list = ls())               # brisanje radnog okruženja
load("Funkcije.RData")        # ponovno učitavanje radnog okruženja
```

6. VIZUALIZACIJA PODATAKA

Kod gotovo svake analize podataka redoslijed radnji je takav da se podaci prvo učitaju i pripreme za obradu, potom se ispitaju neka osnovna svojstva i struktura podataka s nekom od funkcija `summary()`, `head()`, `tail()`, `str()`, i/ili drugih, a gotovo uvijek nakon toga preporučuje se podatke grafički prikazati jer na taj način analitičar/ka može dobiti jasniju sliku o podacima i vezama između varijabli te ideju o nastavku analize.

Postoji nekoliko paketa za grafički prikaz podataka. Na ovom tečaju obrađuje se grafika iz osnovnih paketa sustava R (`base` i `lattice`) koji su njegov sastavni dio. S vremenom su se razvili i puno napredniji paketi, kao na primjer `ggplot2` (vjerojatno najpopularniji), `ggvis`, `rgl` (za interaktivne 3D vizualizacije), `googleVis` (nastao iz Gapmindera), itd., no njihove mogućnosti ostavljamo polazniku da ih sam istraži.

6.1. Grafički sustav osnovnih paketa

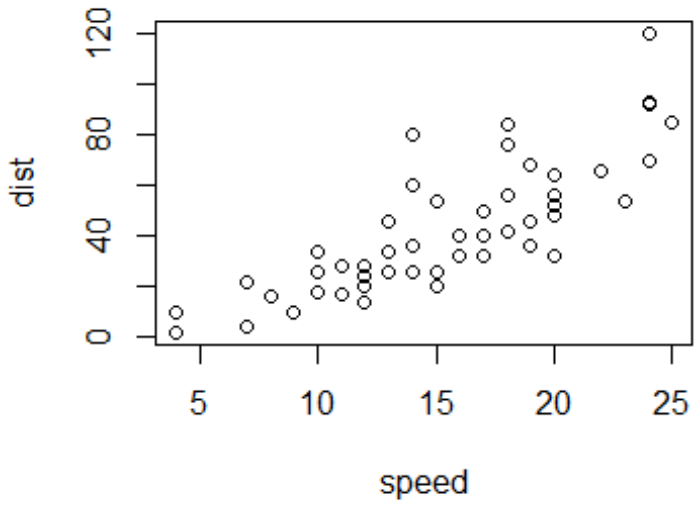
Za crtanje grafikona koristit ćemo podatkovni okvir `cars` koji se sastoji od dvaju numeričkih stupaca, prvi naziva `speed` koji predstavlja brzinu automobila, i drugi naziva `dist` koji predstavlja duljinu zaustavnoga puta.

6.1.1. Plot

Osnovna naredba za crtanje grafa je `plot()`.

Primjer: Primjena naredbe `plot()` nad podatkovnim okvirom `cars`.

```
plot(cars)
```

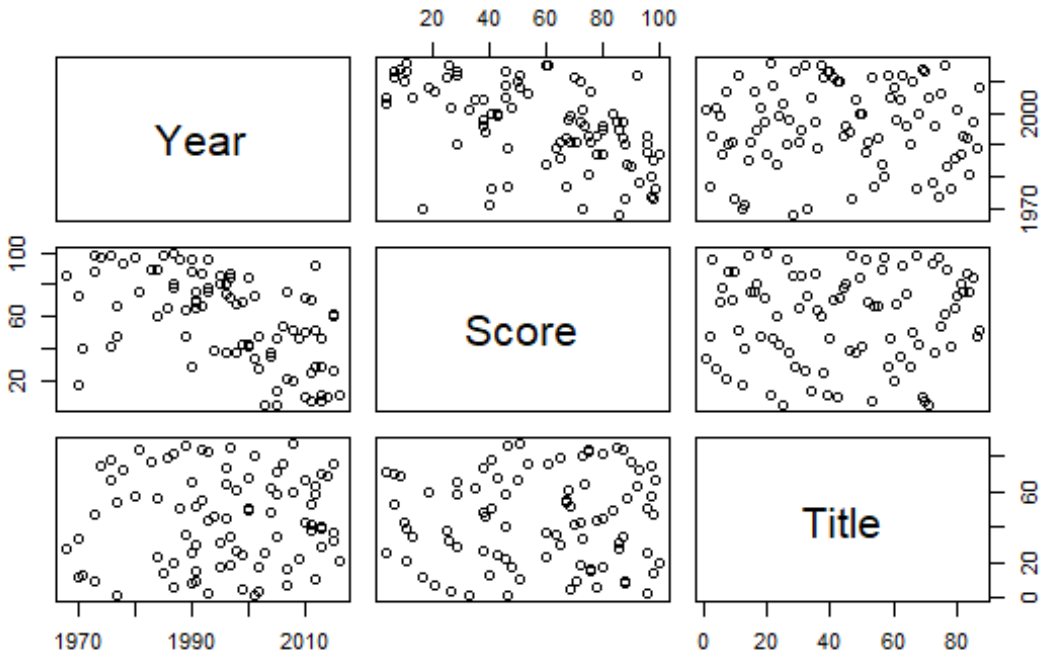


Čak i kada se funkciji `plot()` zada samo podatkovni okvir, bez specificiranja osi x i y, R će prikazati smisleni odgovor. Tako je ovdje prepoznato da je `cars` podatkovni okvir od dvaju stupaca, te je prvi preslikan na os x, a drugi na os y.

S obzirom na to da nismo zadali nazive osima x i y, R je iskoristio nazive stupaca za to, sam je postavio raspon osi x i y te postavio prikladne oznake na osima po svom izboru.

U slučaju da podatkovni okvir ima više od dvaju stupaca, `plot()` će nacrtati po jedan grafikon za svaki par varijabli.

```
deniro <- read.csv("Podaci/deniro.csv")
plot(deniro)
```



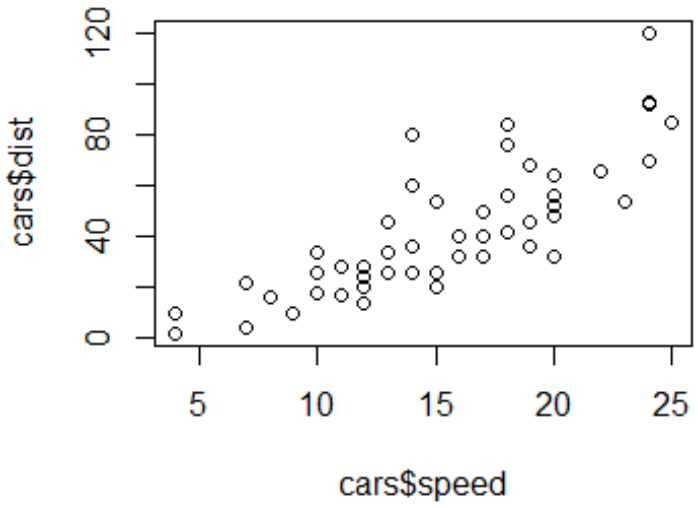
Saznajmo više o funkciji `plot()` uvidom u dokumentaciju. Upišite plot u karticu Help ili izvršite naredbu `?plot`.

```
?plot
```

Funkcija `plot()` ima dva glavna argumenta, x i y koordinate, te uz njih može primiti dodatne grafičke argumente.

x i y argumente možemo zadati na sljedeći način:

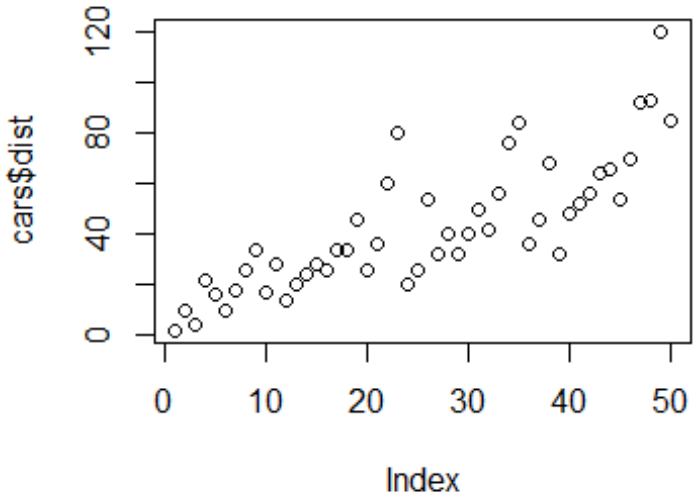
```
plot(x = cars$speed, y = cars$dist)
```



U ovom slučaju R nije siguran što su nazivi osi x i y, pa koristi nazive argumenata za to, uključujući i znak $.

Ako se funkciji plot proslijedi samo jedan vektor, njegove vrijednosti projiciraju se na os y, dok su na osi x indeksi (redni brojevi) podataka.

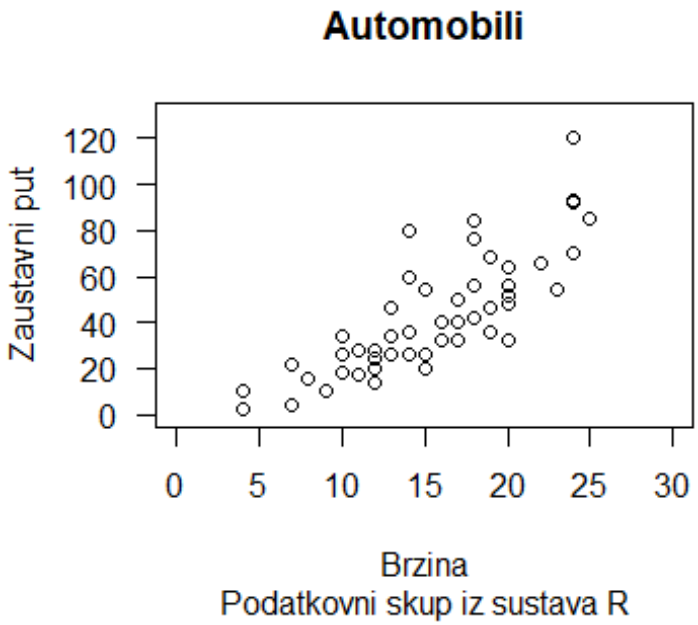
```
plot(cars$dist)
```



6.1.2. Argumenti funkcije plot

Osim preslikavanja podataka na grafikon, poželjno ga je i dodatno urediti. U tu svrhu dodaju se razni argumenti kako bi se podesili dodatni elementi grafikona.

```
plot(cars,
     xlab="Brzina", ylab="Zaustavni put",     # nazivi osi x i y
     main="Automobili",                       # naslov grafikona
     sub="Podatkovni skup iz sustava R",      # podnaslov grafikona
     xlim = c(0,30), ylim = c(0, 130),        # podešavanje raspona osi
     las = 1)                                 # orijentacija oznaka na osima
```



Boja točaka može se podesiti argumentom `col` (*color*). R ima osam standardnih boja koje se mogu pozvati rednim brojevima od 1 do 8 (npr. `col = 2`).

Osim tih osam, R ima dodatnih 657 imenovanih boja koje se mogu pozvati eksplicitno nazivom, na primjer `col="white"`. Popis svih naziva boja može se naći u datoteci rcolorsheet.pdf u radnom direktoriju.

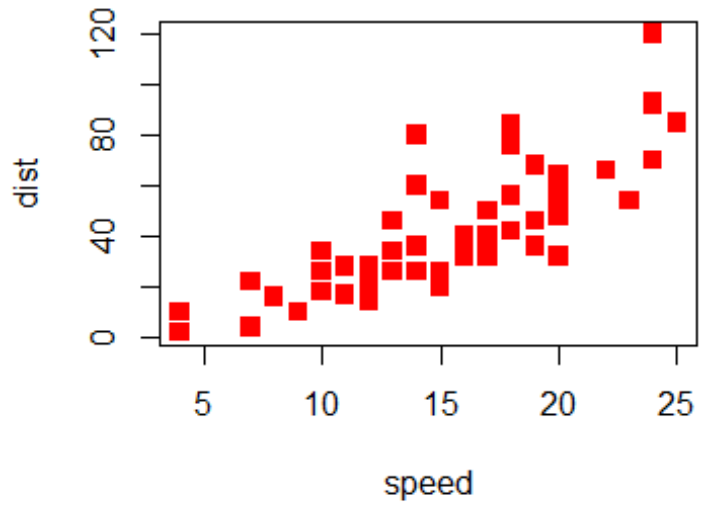
Znak točke grafa određuje se parametrom `pch` (*plotting character*). Znak može biti bilo koji tekstualni znak, na primjer, “A”, “5”, “?” ili numerički kod od 0 do 25, koji predstavljaju sljedeće simbole:

Više detalja o znakovima se može pronaći u R dokumentaciji pozivom naredbe `?points`.

Veličina točaka podešava se argumentom `cex`. `cex` je broj koji predstavlja povećanje veličine (magnifikator) u odnosu na standardnu vrijednost kada je `cex = 1`. Na primjer, `cex = 1.5` povećava točku za 50 % u odnosu na standardnu veličinu, dok `cex = 0.7` smanjuje točku na 70 % standardne veličine.

Na grafičkom prikazu podatkovnoga skupa `cars`, promijenimo znak točke u ispunjeni kvadrat crvene boje uvećan za 30 %.

```
plot(cars,
     pch=15,     # znak točke
     col="red",  # boja točke
     cex=1.3)    # veličina točke
```



Argument `cex` može se proširiti i na ostale elemente grafikona, na primjer argumentima `cex.axis`, `cex.lab`, `cex.main`, `cex.sub` podešavaju se redom veličine naziva na osima, naziva osi, naslova i podnaslova grafikona.

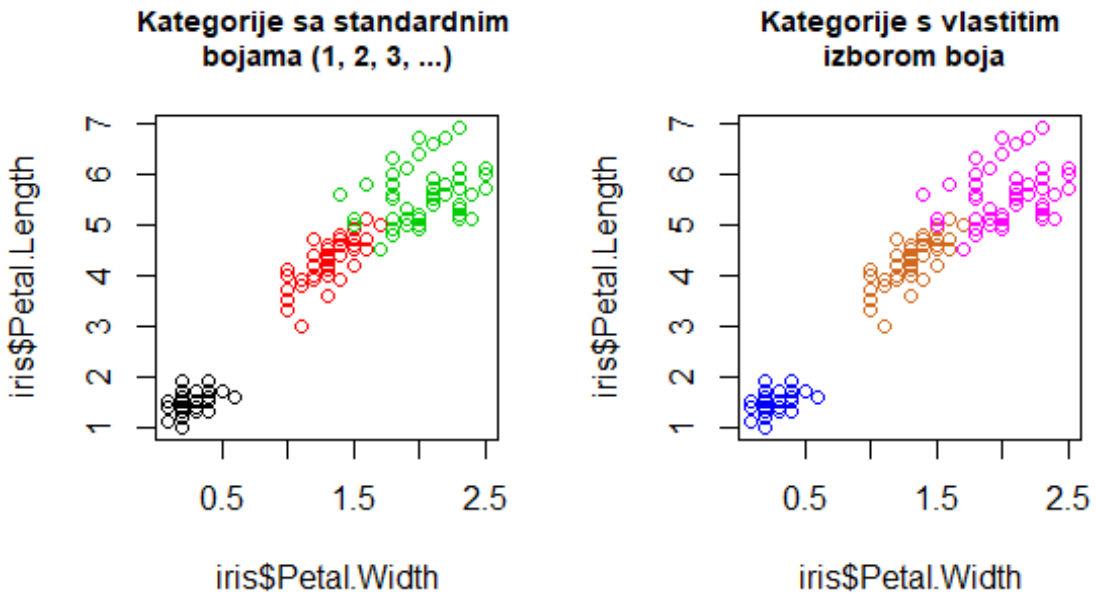
Zadatak 27
Nacrtajte graf podatkovnoga skupa `cars` i dodajte mu barem tri argumenta po vlastitom izboru vezana za točke.

Zadatak 28 Učitajte podatkovni okvir `iris` s kojim smo se već ranije upoznali.

a) Nacrtajte graf kojemu su vrijednosti osi x jednake `Petal.Width`, a vrijednosti osi y `Petal.Length` iz podatkovnog okvira `iris`,
b) naslov grafikona neka bude “Latice irisa”,
c) postavite nazive osima x i y na “Širina latica”, odnosno “Duljina latica”,
d) vrsta točke neka bude ispunjeni kružić plave boje,
e) postavite orijentaciju svih oznaka na osima uspravno,
f) naslov grafikona uvećajte za 50 %.

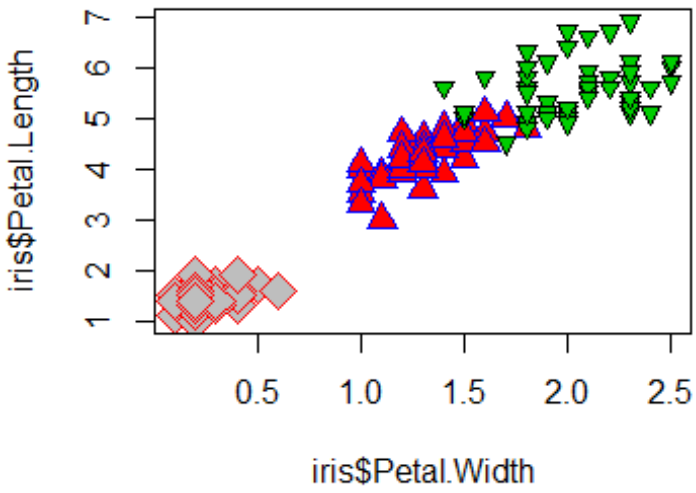
Ako se točke grafa trebaju označiti ovisno o razinama neke kategorijske varijable, to se može napraviti na više načina, na primjer ovisno o izboru boja:

```
par(mfrow=c(1, 2))

plot(iris$Petal.Width, iris$Petal.Length, col=iris$Species, main = "Kategor
ije sa standardnim \nbojama (1, 2, 3, ...)", cex.main=0.9)

plot(iris$Petal.Width, iris$Petal.Length, col=c(4, "chocolate", 6)[iris$Spe
cies], main =  "Kategorije s vlastitim\n izborom boja", cex.main=0.9)
```



Ta tehnika može se primijeniti i na ostalim argumentima, kao na primjer `pch`, `bg`, `cex`, itd.

```
plot(iris$Petal.Width, iris$Petal.Length,
     col=c(2, 4, 1)[iris$Species],
     pch=c(23, 24, 25)[iris$Species],
     bg=c(8, 2, 3)[iris$Species],
     cex=c(2, 1.5, 1)[iris$Species])
```

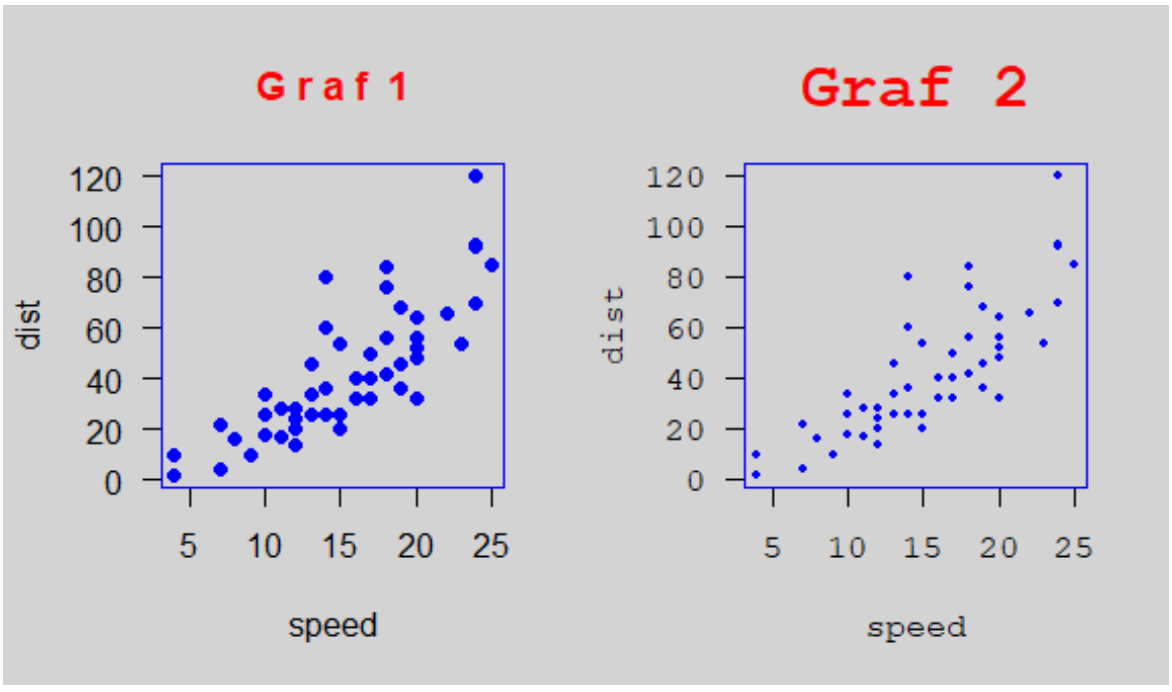


Postoji još niz drugih argumenata kojima se mogu podesiti postavke grafikona, a one se mogu naći pozivom naredbe `?par` (*graphical parameters*).

Većina argumenata opisana u `par` mogu se staviti kao argumenti funkcije `plot()`, kao što smo to dosad i radili, no `par` se može pozivati i odvojeno, specijalno ako se želi specificirati parametre koje može poprimiti samo funkcija `par()` te ako se žele postaviti globalne postavke za više grafikona. U tom slučaju `par()` se mora navesti prije grafičkih funkcija na koje se odnosi, a cijeli komplet funkcija (par + plots) mora se izvršiti istovremeno unutar istog bloka. Izvršavanje svih naredbi iz bloka ('chunka') provodi se istovremenim pritiskom na `[Ctrl]+[Shift]+[Enter]` dok je kursor unutar bloka.

Primjer:

```
par(mfrow = c(1,2), pch = 16, bg = "lightgray", col=4, col.main=2, las=1)
plot(cars, main="G r a f  1")
plot(cars, main = "Graf 2", family="mono", cex=0.5, cex.main = 2)
```



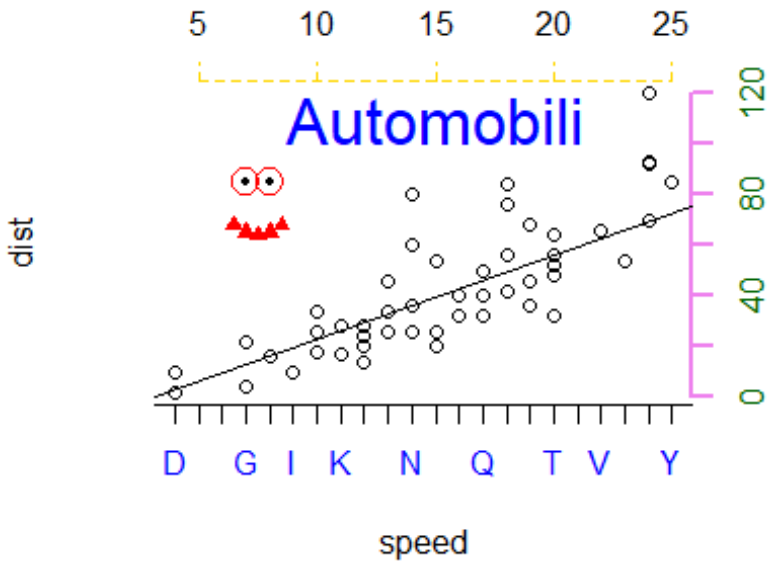
6.1.3. Grafičke funkcije nižega reda

Funkcija koja samostalno može proizvesti grafikon (kao na primjer `plot()`) zove se grafička funkcija višega reda, dok se funkcija koja dodaje sloj na grafikon, ali ne može proizvesti neovisan grafikon, zove grafička funkcija nižega reda.

Grafičke funkcije nižega reda moraju se navesti u istom bloku kao i grafička funkcija višeg reda na koju se odnose i istovremeno se izvršiti.

U nastavku su primjeri nekih grafičkih funkcija nižega reda koje dodaju određeni sloj grafikonu `plot(cars)`.

```
plot(cars, axes=F)

#osi (Prije ručnog dodavanja osi preporučuje se postaviti argument axes = F
ALSE u funkciju plot da se izbrišu standardno zadane osi)
axis(1, 1:26, LETTERS, col.axis = "blue")
axis(3, col = "gold", lty = 2, lwd = 1.5)
axis(4, col = "violet", col.axis="dark green", lwd = 2)


# pravac y = a + bx
abline(a= -10, b=3.3)

#točke
points(x=c(7, 8),y=c(85, 85), col=2, pch=1, cex=2)
points(x=c(7, 8),y=c(85, 85), col="black", pch=16, cex=0.5)
points(x=c(6.5, 7, 7.5, 8, 8.5),y=c(68, 65, 64, 65, 68), col=2, pch=17)

# tekst
text(15,110, "Automobili", cex=2, col=4)
```



Da sažmemo, navest ćemo neke od češće korištenih grafičkih argumenata:

Argument	Pojašnjenje
`pch`	vrsta simbola
`lty`	vrsta linije
`lwd`	debljina crte
`col`	boje korištene u crtanju (mogu se definirati nazivom, brojem ili heksadecimalnim brojem)
`cex` (`cex`, `cex.axis`, `cex.lab`, `cex.main`, `cex.sub`)	multiplikator veličine točaka, odnosno navedenih naslova
`xlim`, `ylim`	podešavanje raspona osi x i y, zadaje se u obliku vektora `c(od, do)`
`las`	orijentacija teksta na osima grafikona
`frame`, `bty`	podešavanje okvira grafikona (`bty` dolazi od engl. box type)
`xlab`, `ylab`	nazivi osi
`main`	naziv grafikona
`bg`	boja pozadine grafikona unutar `par()` funkcije, a boja ispune točke za `pch` između 21:25 unutar `plot()` funkcije
`mar`	određuje širinu margina brojem linija teksta
`mfrow = c(m, n)`	za crtanje više grafikona na istoj slici, koji se postavljaju u m × n matricu i popunjavaju po redcima (poziv moguć samo unutar funkcije `par()`)
`mfcol = c(m, n)`	kao `mfrow`, ali se matrica grafikona popunjava po stupcima (poziv moguć samo unutar funkcije `par()`).

Grafikonu možemo dodavati elemente pomoću sljedećih funkcija:

Funkcija	Pojašnjenje
`points()`	ucrtava točke na grafikon kojima možemo zadati koordinate, boju, oblik, veličinu, itd.
`lines()`	crta liniju koja prolazi zadanim točkama
`abline(a, b)`	dodaje pravac jednadžbe $y = a + bx$
`segments()`	crta segment između zadanih točaka
`arrows()`	isto kao segments(), ali sa strelicom na kraju
`axis()`	funkcija za podešavanje osi i oznaka na osima (prije toga staviti `axes = FALSE` u `plot()`)
`title()`	dodaje naslov grafikona
`text()`	dodaje tekst na proizvoljnom mjestu grafikona
`mtext()`	dodaje tekst na margine grafikona
`legend()`	dodaje i podešava legendu

Za pravilnu uporabu svakog argumenta i funkcije preporučuje se pročitati uputstva iz dokumentacije.

Zadatak 29

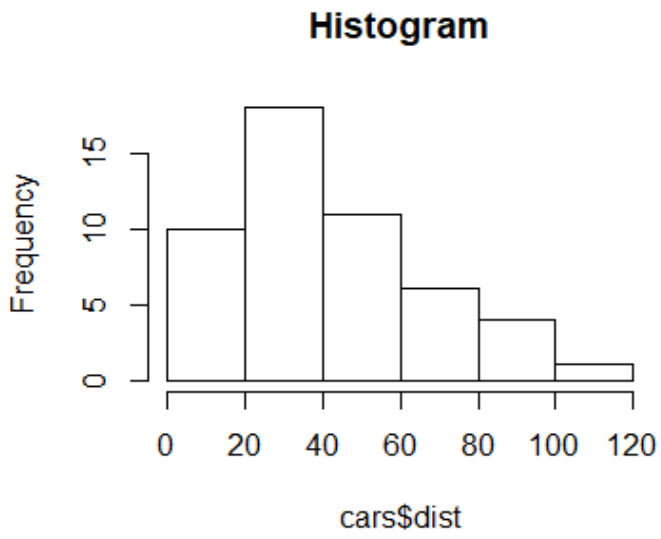
1. Nacrtajte dva grafikona koja prikazuju vezu između:
   a) duljine i širine cvjetnih čašićnih listića (engl. sepal) - 1. grafikon
   b) duljine i širine latica (engl. petal) - 2. grafikon
   cvjetova irisa iz podatkovnog okvira iris.
2. Postavite globalne postavke grafikona pomoću funkcije `par()` takve da
   c) oba grafikona budu u jednom retku
   d) pozadina grafikona bude siva
   e) veličina naslova grafikona veća za 50 % od standardne.
3. Unutar svakog grafikona
   a) neka se vrste cvjetova razlikuju po bojama
   b) osi x i y nazovite “Širina”, odnosno "Duljina
   c) naslove grafikona postavite na “Sepal”, odnosno “Petal”.
4. Unutar grafikona Sepal dodajte dva pravca čije su jednadžbe $y = 1.5x$, donosno $y = 0.5 + 2x$.
5. Na grafikonu Petal
   a) dodajte isprekidani pravac jednadžbe $y = 0.75 + 2.5x$
   b) dodajte jednu točku čije su koordinate srednje vrijednosti svih izmjerenih širina, odnosno duljina latica
   c) neka ta točka srednje vrijednosti bude crvena, ispunjena i za pola veća od ostalih točaka.

6.1.4. Histogram

Histogram je vrsta grafa kojim se prikazuje razdioba (kontinuirane) numeričke varijable. Na osi x navedeni su intervali, a na osi y preslikane su frekvencije, odnosno broj jedinki u svakom intervalu.

U sustavu R histogram se može dobiti naredbom `hist()`.

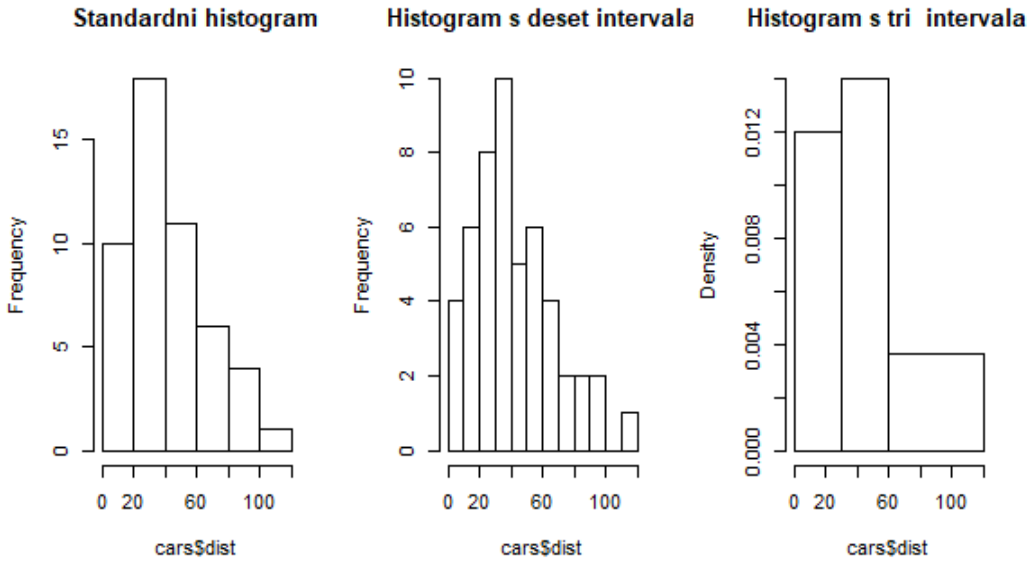
```
hist(cars$dist, main="Histogram")
```



S obzirom na to da izgled histograma varira ovisno o širini intervala, korisnik sam može podešavati broj intervala.

Primjer:

```
par(mfrow = c(1,3))
hist(cars$dist, main="Standardni histogram")
hist(cars$dist, breaks = 10, main="Histogram s deset intervala")
hist(cars$dist, breaks = c(0, 30, 60, 120), main="Histogram s tri  interval
a")
```

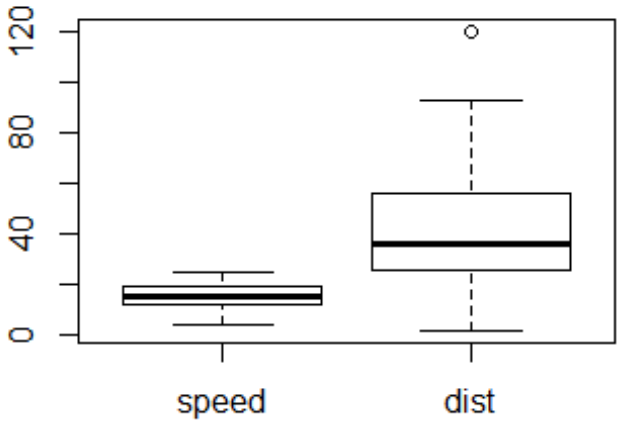


Zadatak 30
Nacrtajte histograme za sve četiri numeričke varijable podatkovnog okvira `iris`. Pomoću funkcije `par()` postavite sva četiri grafikona u istu sliku s rasporedom 2 × 2, i dajte prikladne nazive osima x (na primjer Sepal Length umjesto iris$Sepal.Length). Svaki histogram obojite bojama po vlastitom izboru.

6.1.5. Dijagram pravokutnika (*boxplot*)

Dijagram pravokutnika ili brkata kutija (engl. *box and whisker plot*) je vrsta grafa koja prikazuje nekoliko mjera deskriptivne statistike na jednom grafu - u središnjem se pravokutniku nalazi 50 % podataka, i to onih između prvog i trećeg kvartila ($Q1$ i $Q3$), brkovi prikazuju raspon ostatka skupa, ali tako da ne prelazi granice od $Q1 - 1.5\ IQR$ odozdo i $Q3 + 1.5\ IQR$ odozgo (gdje je $IQR = Q3 - Q1$, interkvartilni raspon). Sve točke koje prelaze te granice označene su kružićem i zovu se ekstremne vrijednosti (engl. *outliers*). Linija koja presijeca središnji pravokutnik predstavlja medijan.

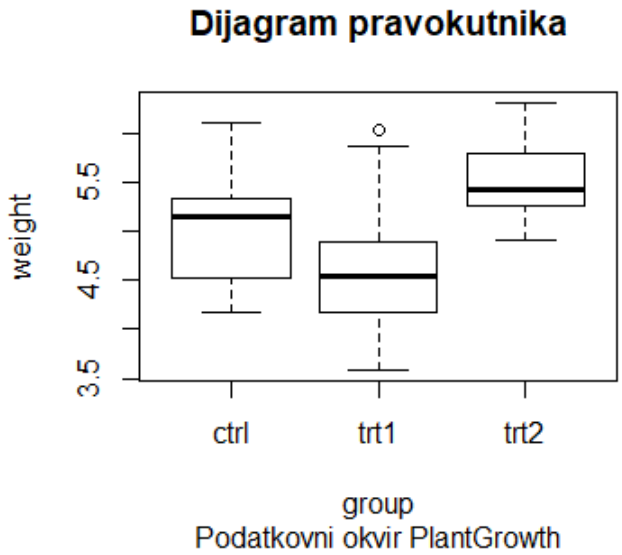
```
boxplot(cars)
```



`boxplot()`, kao i dosta drugih funkcija u R-u (na primjer `plot()`, `aggregate()`…), može primiti argumente u obliku formule, odnosno izraze sa znakom ~ koji ukazuje na odnos između varijabli nekoga podatkovnog skupa. Tako možemo pozvati funkciju `boxplot()` s argumentima `Y ~ X` gdje se kategorijska varijabla `X` preslikava na os x, a raspon vrijednosti `Y` po svakoj kategoriji na os y.

U sljedećem primjeru koristit ćemo podatkovni okvir `PlantGrowth` koji sadrži podatke o težini uroda biljaka koje su podvrgnute jednoj od triju vrsta tretmana. Podatkovni okvir sastoji se od dviju varijabli, `weight` i `group`.

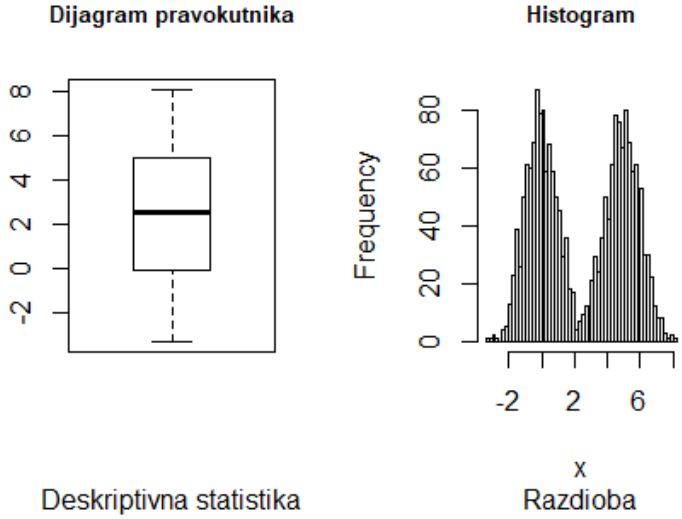
```
boxplot(weight ~ group, data = PlantGrowth,
        main = "Dijagram pravokutnika", sub="Podatkovni okvir PlantGrowth")
```



NAPOMENA: Dijagram pravokutnika daje grafički prikaz nekih mjera deskriptivne statistike, no ne prikazuje razdiobu podataka.

Primjer: Dijagram pravokutnika ne pokazuje razdiobu varijable.

```
n <- seq(-5,5,l=1000)
x1 <- rnorm(n, 0, 1)
x2 <- rnorm(n, 5, 1)
x <- c(x1, x2)

par(mfrow=c(1, 2))
boxplot(x, main ="Dijagram pravokutnika", sub = "Deskriptivna statistika")
hist(x, 50, main ="Histogram", sub = "Razdioba")
```



Zadatak 31
U jednom pozivu funkcije `boxplot()` prikažite dijagram pravokutnika za sve četiri numeričke varijable iz podatkovnog okvira `iris`.

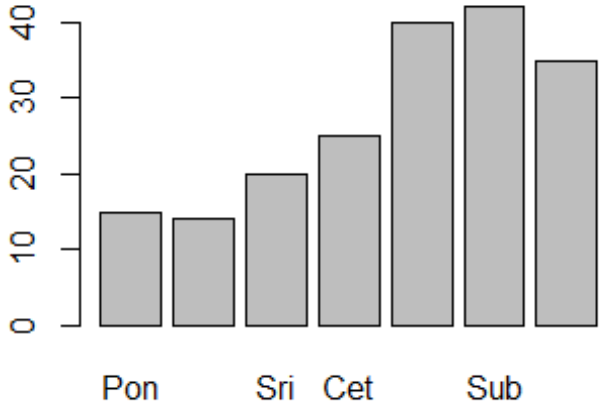
6.1.6. Stupčasti grafikon (*barplot*)

Stupčasti grafikon (*barplot*) je vrsta grafikona koja prikazuje vezu između kategorijske i numeričke varijable, pri čemu se kategorijska varijabla prikazuje na osi x.

Funkciji `barplot()` potrebno je proslijediti vektor ili matricu s frekvencijama koje će se preslikati na os y, no postoje funkcije za izradu stupčastih grafikona iz nekih drugih paketa (na primjer ggplot2) koje same izračunaju potrebne frekvencije iz zadanog podatkovnog skupa.

Primjer stupčastoga grafikona za vektor:

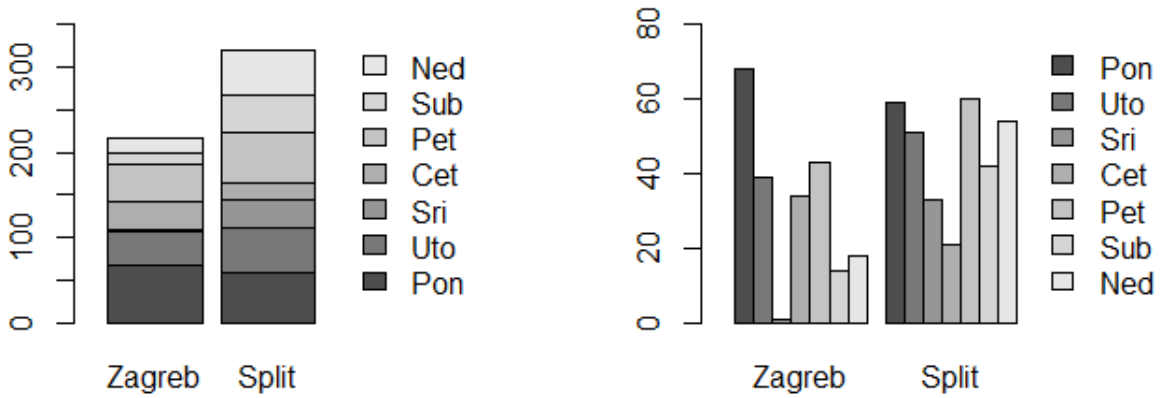
```
Broj_putnika <- c(Pon = 15, Uto = 14, Sri = 20, Cet = 25, Pet = 40, Sub = 4
2, Ned = 35)

barplot(Broj_putnika)
```



Ako se funkciji `barplot()` proslijedi matrica, dobit će se stupčasti dijagram sa stupcima naslaganima jedni na drugima. Ako se žele stupci koji su poredani jedan do drugog, argument `beside` postavi se na `TRUE`.

```
#izrada matrice:
set.seed(1)
Broj_putnika2 <- matrix(sample(70, 14), 7, 2)
rownames(Broj_putnika2) <- c("Pon", "Uto", "Sri", "Cet", "Pet", "Sub", "Ned
")
colnames(Broj_putnika2) <- c("Zagreb", "Split")

par(mfrow=c(1,2))

# Primjer stupčastog grafikona za matricu gdje su stupci poredani jedan na
drugom:
# beside = FALSE
```

```
barplot(Broj_putnika2,
        beside = FALSE,
        width = 0.5,
        xlim = c(0,2),
        ylim = c(0,350),
        legend.text = TRUE,
        args.legend = list(x="right", bty="n"))

#Primjer stupčastog grafikona za matricu gdje su stupci poredani jedan do d
rugog:
# beside = TRUE
barplot(Broj_putnika2,
        beside = TRUE,
        width = 0.1,
        xlim = c(0,2),
        ylim=c(0, 80),
        legend.text = TRUE,
        args.legend = list(x="right", bty="n", inset=-0.15))
```



Zadatak 32

Na temelju podatkovnog okvira `iris` stvorena je matrica srednjih vrijednosti dimenzija 3 × 4 za sve četiri varijable i sve tri vrste cvijeta iris. Iz te matrice (`srednje_vrijednosti_iris`) napravite stupčasti dijagram u kojem su stupci poredani jedan pored drugog. Boje dijagrama odaberite po vlastitom izboru.

```
srednje_vrijednosti_iris <- aggregate(iris[1:4], list(iris$Species), mean)
srednje_vrijednosti_iris <- as.matrix(srednje_vrijednosti_iris[,2:5])
```

6.2. Paket `lattice`

Paket `lattice` je napredniji dodatak osnovnoj grafici sustava R s boljim standardnim postavkama i boljim mogućnostima prikaza veza između više varijabli.

Sintaksa funkcija iz paketa `lattice` je

```
vrsta_grafa(formula|uvjet, podatkovni_skup)
```

gdje je `vrsta_grafa` jedna od mogućnosti iz donje tablice, `formula` određuje koje varijable se prikazuju na osima (npr. `~x` ili `y~x`), a varijable iza znaka "|" navode prema kojim uvjetima (faktorima) se prikazuju varijable iz formule, s tim da je prikaz podijeljen na zasebnim grafikonima za svaku razinu faktora iz varijable `uvjet`. Na primjer, `~x|A` traži prikaz varijable `x` ovisno o razinama faktora `A` i prikazuje rezultat na onoliko grafikona koliko ima razina u faktoru `A`, dok `y~x|A*B` traži prikaz veze između `y` i `x` ovisno o faktorima `A` i `B`, a konačan broj grafikona je umnožak razina faktora `A` i `B`.

Vrste grafikona

Vrsta grafikona	Opis	Formula
`barchart`	stupčasti dijagram	$x \sim A$ ili $A \sim x$
`bwplot`	dijagram pravokutnika	$x \sim A$ ili $A \sim x$
`cloud`	3D točkasti dijagram	$z \sim x * y \mid A$
`contourplot`	3D kontura	$z \sim x * y$
`densityplot`	graf funkcije gustoće vjerojatnosti	$\sim x \mid A * B$
`dotplot`	točkasti graf	$\sim x \mid A$
`histogram`	histogram	$\sim x$
`levelplot`	3D graf razina	$z \sim y * x$
`parallel`	graf paralelnih koordinata	podatkovni okvir
`splom`	točkasti graf matrice	podatkovni okvir
`stripplot`	linijski (1D) graf	$A \sim x$ ili $x \sim A$
`xyplot`	graf dviju varijabli	$y \sim x \mid A$
`wireframe`	3D uokvireni graf	$z \sim y * x$

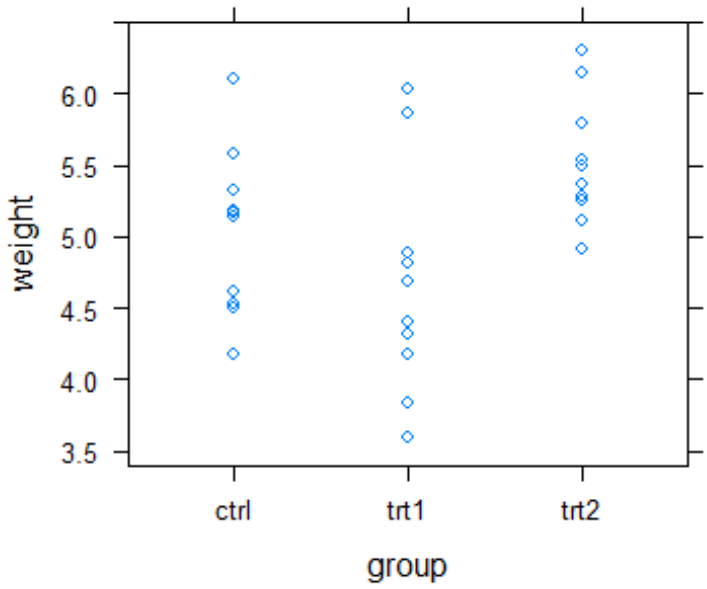
Postavke koje su vrijedile za `plot()` i `par()` generalno ne vrijede na `lattice` funkcijama, no neki argumenti su im zajednički, na primjer `col`, `main`, `xlab`, `ylab`, `pch`, `lty`, `cex`…

S obzirom na to da je paket `lattice` dio sustava R, ali nije jedan od osnovnih paketa, potrebno ga je prvo učitati funkcijom `library()`.

```
library(lattice)
```

Primjer: Na već poznatom skupu `PlantGrowth` upoznat ćemo se s funkcijom `xyplot(y ~ x, data, ...)` koja prikazuje vezu dviju varijabli.

```
data("PlantGrowth")
xyplot(weight~group, PlantGrowth)
```

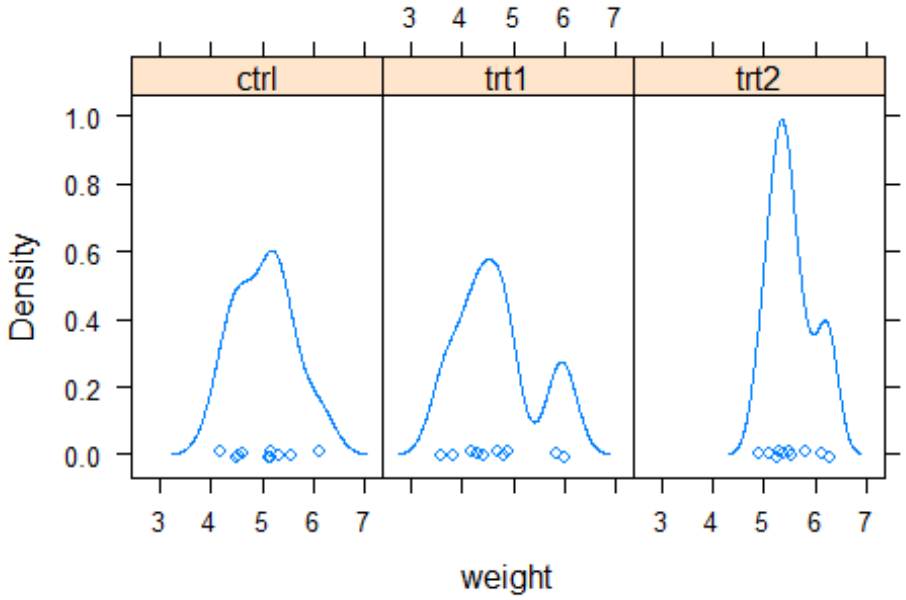


Zadatak 33
Funkcijom `xyplot()` nacrtajte grafikon koji prikazuje vezu između varijabli `Petal.Length` i `Petal.Width` iz podatkovnog okvira `iris`. Grafikonu dodajte naslov "Irisove latice".

6.2.1. densityplot()

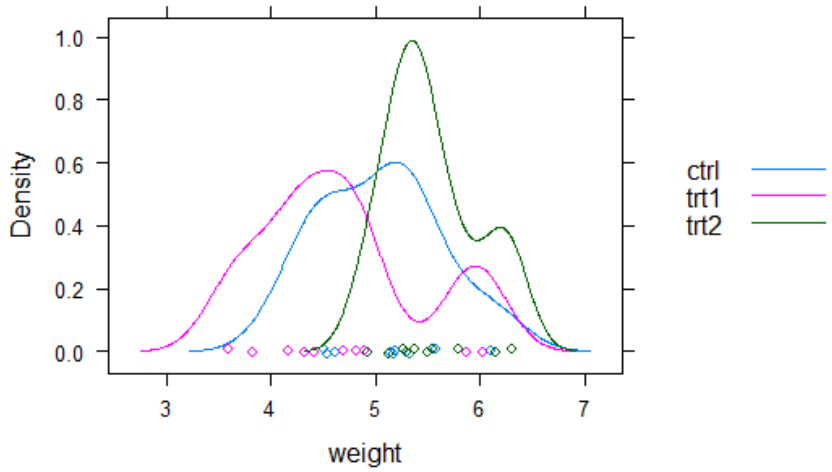
Za prikaz razdiobe podataka prikladna je funkcija `densityplot(~x, data, ...)`. Formula prima samo jednu varijablu, no omogućuje raščlanjivanje po faktorima.

```
densityplot(~weight|group, PlantGrowth)
```



Ako se radi lakše usporedbe sve kategorije trebaju prikazati na istom grafikonu, onda se naredba preformulira na sljedeći način:

```
densityplot(~weight, PlantGrowth,
            groups = group,                    # razdvaja graf prema faktori
ma
            auto.key = list(space="right"))    # pozicionira legendu
```



Zadatak 34

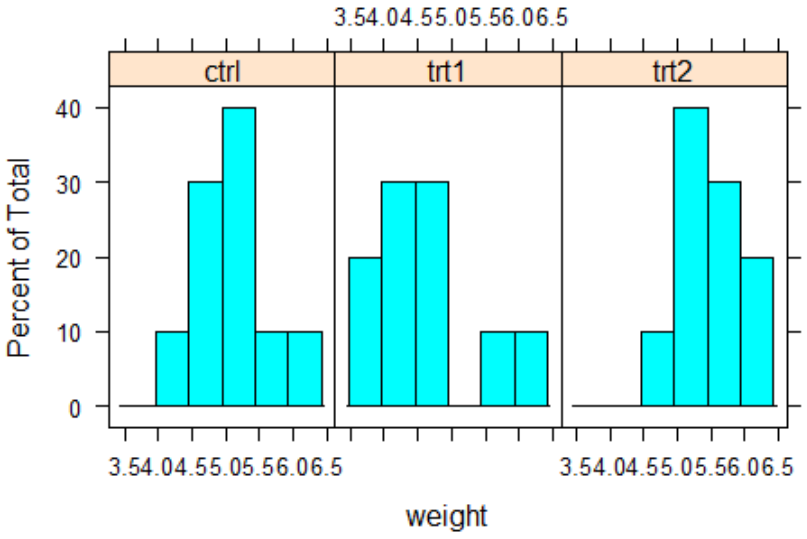
34.1 Funkcijom `densityplot()` nacrtajte funkciju gustoće vjerojatnosti varijable `Petal.Width` iz podatkovnog okvira `iris`. Graf nacrtajte isprekidanom crvenom linijom.

34.2 Funkciju gustoće vjerojatnosti iz prethodnog zadatka prikažite u ovisnosti o vrsti cvijeta.

6.2.2. Histogram

U paketu `lattice` histogram se može dobiti istoimenom funkcijom `histogram(~x, data, ...)`. Formula prima samo jednu varijablu, a podaci se mogu dodatno razvrstati po faktorima.

```
histogram(~weight|group, PlantGrowth)
```



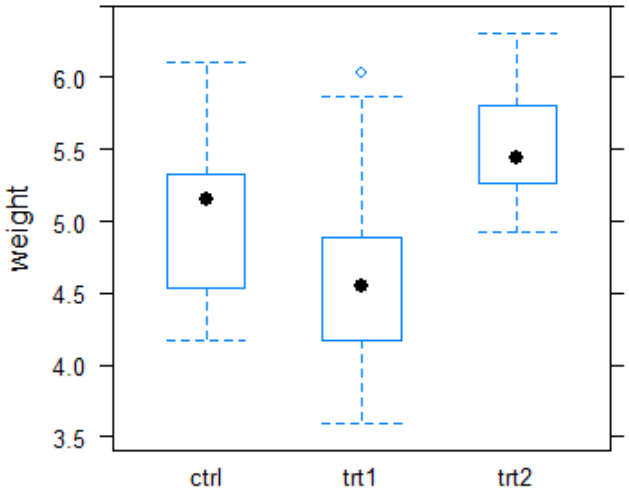
Broj razreda u histogramu može se podesiti argumentom `nint`.

Zadatak 35
Nacrtajte histogram za varijablu `Petal.Width` podatkovnog okvira `iris` posebno za svaku vrstu cvijeta.

6.2.3. Dijagram pravokutnika

U paketu `lattice` dijagram pravokutnika dobiva se naredbom `bwplot(y ~ x, data)`, gdje je poželjno da varijabla x bude faktorska.

```
bwplot(weight ~ group , PlantGrowth)
```



Zadatak 36
Nacrtajte dijagram pravokutnika za varijablu `Petal.Length` podatkovnog okvira `iris` koji prikazuje po jedan brkati pravokutnik za svaku vrstu cvijeta.

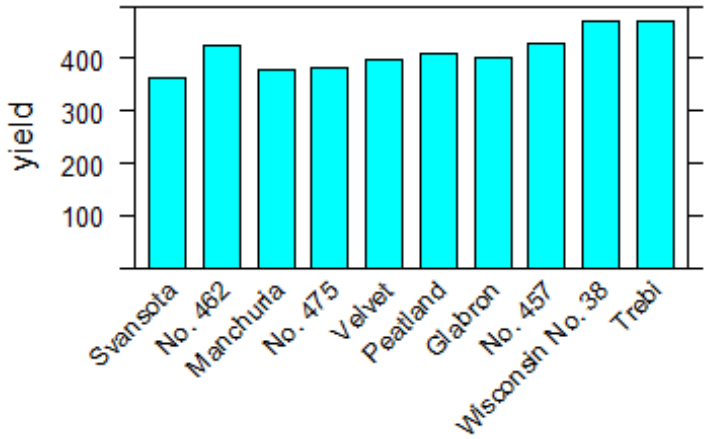
6.2.4. Stupčasti grafikon

Za stupčasti grafikon (`barchart()`), koristit ćemo podatkovni okvir `barley` iz sustava R koji sadrži podatke o berbama različitih vrsta ječma izmjerenih na različitim lokacija u različitim godinama. Ovaj podatkovni skup je prikladan za istraživanje mogućnosti prikaza stupčastih grafikona jer ima tri kategorijske varijable: `variety`, `year` i `site`.

```
data(barley)
```

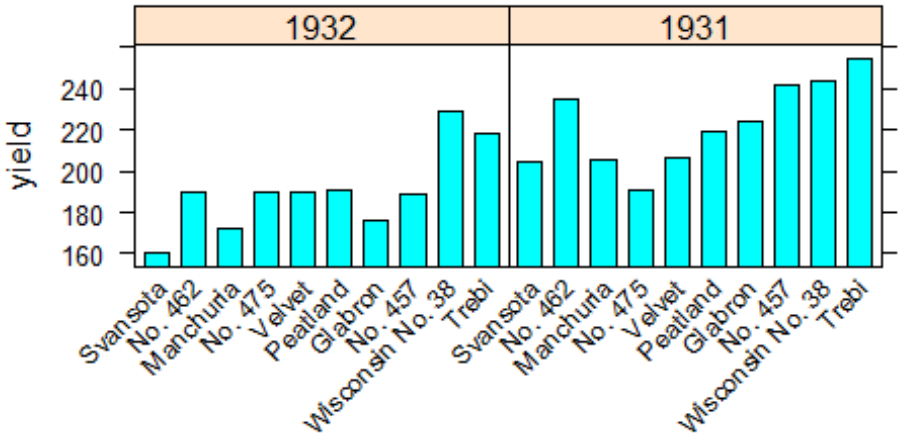
Funkciji `barchart()` potrebno je proslijediti agregirani, odnosno zbrojeni podatkovni okvir.

```
barley_sum <- aggregate(yield~variety, barley, sum)

barchart(yield ~ variety,  data = barley_sum,
         ylim=c(0, 500),                        #podešavanje raspona osi y
        scales = list(x = list(rot = 45)))   #rotacija naziva kategorija na
osi x
```



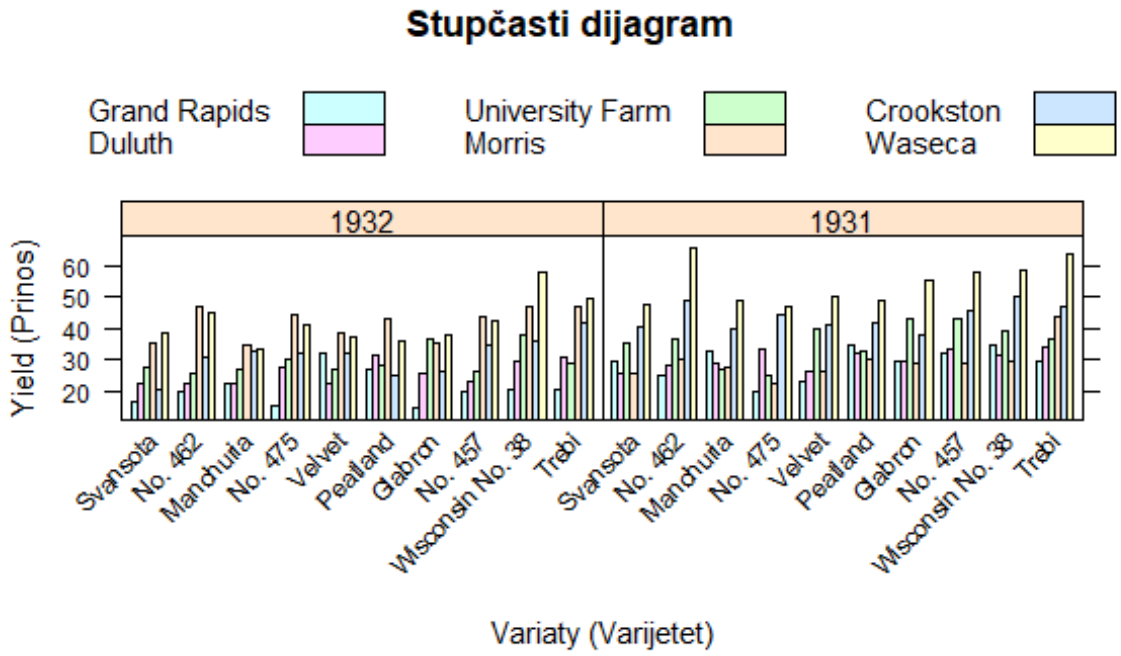
Podatke možemo dalje raščlanjivati po kategorijama. Prvo ih se agregira, a onda crta.

```
barley_sum2 <- aggregate(yield~variety*year, barley, sum)
barchart(yield ~ variety|year,  data = barley_sum2,
         scales = list(x = list(rot = 45)))
```



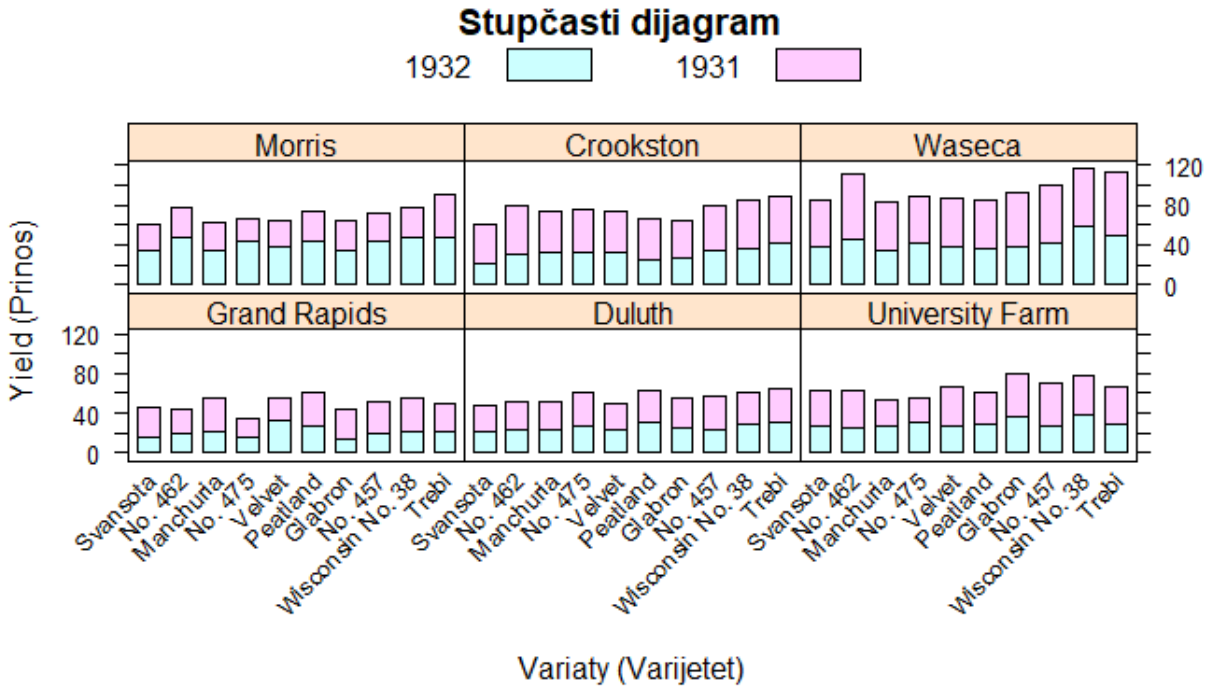
Kada se podaci raščlanjuju do zadnjeg detalja, nema potrebe za agregacijom:

```
barchart(yield ~ variety|year,  data = barley,
         groups = site,
         main = "Stupčasti dijagram \n",
         xlab = "Variaty (Varijetet)",
         ylab = "Yield (Prinos)",
         scales = list(x = list(rot = 45)),
         auto.key = list(rectangles = TRUE, space = 'top', columns = 3),
                                    #postavke legende
         box.width = 0.8,           #širina stupaca
         par.settings = list(fontsize=list(text=11)))
                                    #veličina naziva kategorija na osi x
```



Ili:

```
barchart(yield ~ variety|site,  data = barley,
         groups = year,
         stack = TRUE,                  #stupci su naslagani jedan na drugog
         main = "Stupčasti dijagram",
         xlab = "Variaty (Varijetet)",
         ylab = "Yield (Prinos)",
         scales = list(x = list(rot = 45)),
         auto.key = list(rectangles = TRUE, space = 'top', columns = 2))
```



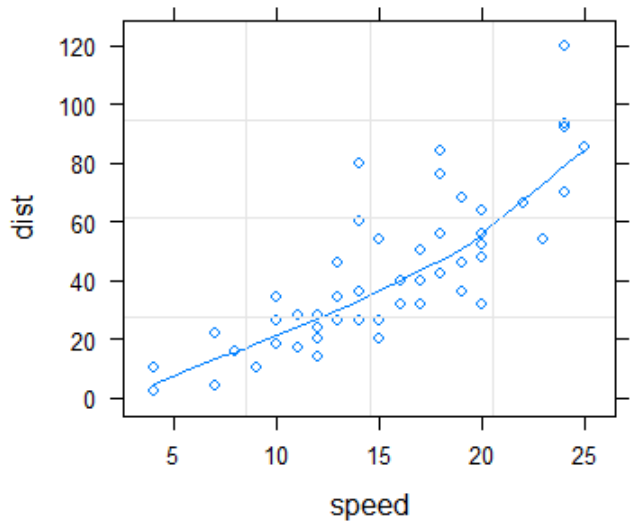
Zadatak 37
Prikažite podatke iz podatkovnog okvira `barley` u stupčastom dijagramu tako da na os x preslikate varijablu `site`, a na os y varijablu `yield`. Neka podaci budu podijeljeni po sorti, te neka stupci budu razdijeljeni po godinama i poredani jedan pored drugog. Zarotirajte nazive na osi x i uključite legendu u grafikon.

6.2.5. Panel funkcije

Grafičke funckije iz paketa `lattice` mogu se nadograđivati raznim opcijama pomoću **panel funkcija**. Korisnik može sam definirati panel funkciju te ih kombinirati s predefiniranim ugrađenim panel funkcijama poput `panel.grid` za ucrtavanje mreže, `panel.abline` za dodavanje linije, itd.

Primjer: Na sljedećem primjeru prikazana je sintaksa i pozivanje panel funkcije za `xyplot()`

```
panel.smoother <- function(x, y) {
  panel.xyplot(x, y) # crta sve točke xyplot grafikona
  panel.loess(x, y)  # ucrtava glatku krivulju
  panel.grid()       # dodaje mrežu
}

xyplot(dist~speed, cars, panel=panel.smoother)
```



Više o mogućnostima paketa `lattice` može se pronaći u materijalima koje je napisao autor paketa Deepayan Sarkar: https://www.isid.ac.in/~deepayan/R-tutorials/labs/04_lattice_lab.pdf.

6.3. Boje u sustavu R

Boje u sustavu R označavaju se na jedan od tri načina:

- cijeli broj koji označava redni broj boje u paleti `palette()`
- naziva boje iz popisa od 657 imenovanih boja sustava R
- heksadecimalni zapis oblika `#rrggbb` ili `#rrggbbaa` pri čemu znamenke `rr`, `gg`, `bb` predstavljaju heksadecimalni zapis udjela crvene, zelene i plave boje koji može ići od 0 do 255, a `aa` je stupanj prozirnosti pri čemu je 00 potpuno prozirno, a 255 potpuno neprozirno.

Kod crtanja grafikona, pravilnom uporabom boja moguće je u znatnoj mjeri predočiti vezu unutar podataka koje crtamo. Međutim, biranje i kombiniranje boja može uzeti puno vremena, pogotovo ako je potreban velik broj boja. Zbog toga je posebna pažnja dana razvoju, upravljanju i specifikaciji boja na grafikonima. Uz nekoliko opcija koje nam nude osnovni paketi `graphics` i `grDevices`, postoji i niz korisničkih paketa koji rad s bojama čine vrlo jednostavnim i praktičnim, a korisniku ostavljaju prostor za kreativnost.

6.3.1. Palete boja

Standardne boje u R-u koje možemo pozivati rednim brojevima od 1 do 8 (kao na primjer u argumentu `col = 2`) dio su standardne palete boja, no to se može mijenjati funkcijom `palette()`.

Možemo definirati vlastiti niz boja proizvoljne duljine koju kasnije koristimo u grafikonima.

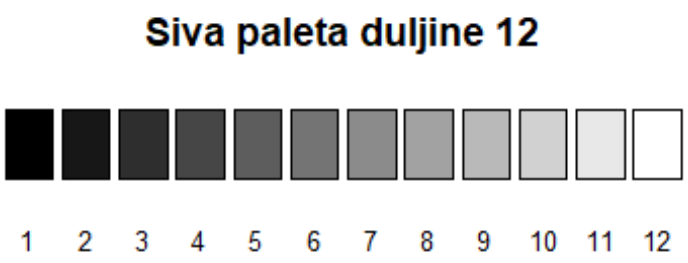
```
palette(c("palegreen", "seagreen1","turquoise", "lightseagreen","plum", "orchid"))

barplot(rep(1,12), col = 1:12, xlab="", ylab="",
        names.arg = 1:12, cex.names = 0.8, axes = FALSE,
        main="Ručno izrađena paleta boja po vlastitom izboru" )
```

Specijalno za sive nijanse moguće je paletu definirati na sljedeći način:

```
palette((gray(seq(0,1,len = 12))))
palette()

##  [1] "black"   "gray9"   "gray18"  "#464646" "#5D5D5D" "#747474" "#8B8B8B"
##  [8] "#A2A2A2" "#B9B9B9" "gray82"  "gray91"  "white"
```

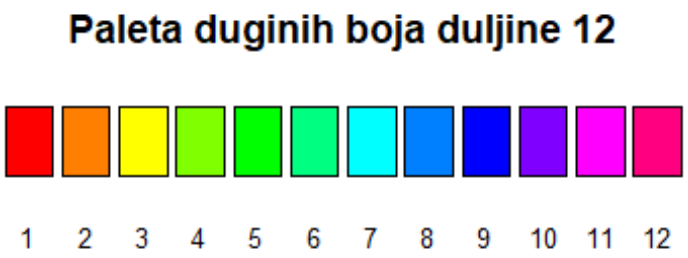
Boje koje nemaju standardni naziv dane su u heksadecimalnom zapisu.

```
barplot(rep(1,12), col = 1:12, xlab="", ylab="", axes = FALSE,
        names.arg = 1:12, cex.names = 0.8,
         main="Siva paleta duljine 12" )
```



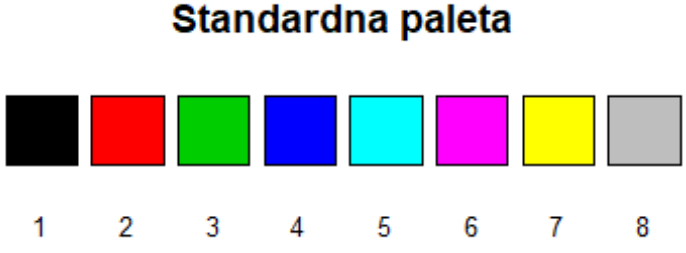
Postoji i paleta duginih boja, odnosno `palette(rainbow(n))` gdje je `n` broj boja, i može poprimiti do 1024 nijanse boja.

```
palette(rainbow(12))

barplot(rep(1,12), col = 1:12, xlab="", ylab="", axes = FALSE,
        names.arg = 1:12, cex.names = 0.8,
         main="Paleta duginih boja duljine 12" )
```



Na popisu paleta mogu se naći i `heat.colors()`, `terrain.colors()`, `topo.colors()` i `cm.colors()`. Povratak na standardnu paletu poziva se naredbom:

```
palette("default")

barplot(rep(1,8), col = 1:8, xlab="", ylab="", axes = FALSE,
        names.arg = 1:8, cex.names = 0.8,
         main="Standardna paleta" )
```

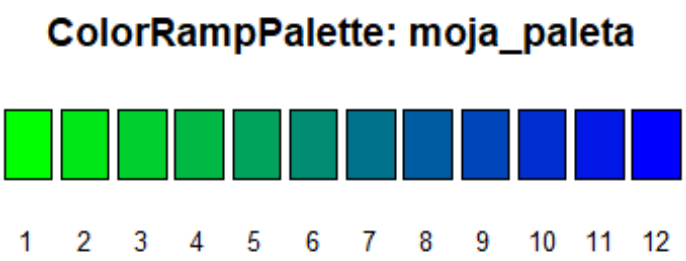


6.3.2. ColoRampPalette

Još jedan način na koji se preko funkcija osnovnih paketa može doći do nove palete boja jest funkcijom `colorRampPalette()` iz paketa `grDevices` koja za argument prima dvije ili više boja kao granice između kojih se interpoliraju ostale nijanse boja. `colorRampPalette()` vraća funkciju kojoj se za argument proslijedi cijeli broj kojim određujemo ukupan broj nijansi u paleti.

Primjer:

```
moja_paleta <- colorRampPalette(c("green", "blue"))    # moja_paleta je fun
kcija

palette(moja_paleta(12))  # funkciji moja_paleta prosljeđujemo broj 12 kao
ukupan broj boja u paleti

barplot(rep(1,12), col = 1:12, xlab="", ylab="", axes = FALSE,
        names.arg = 1:12, cex.names = 0.8,
         main="ColorRampPalette: moja_paleta" )
```



Zadatak 38
Na jedan od gore opisanih načina izradite vlastitu paletu boja i nacrtajte stupčasti dijagram y~x s već postojećim vektorima x i y koji ćete obojati vlastitom paletom.

6.3.3. RColorBrewer

Jedan od popularnijih paketa za boje je `RColorBrewer` koji sadrži nekoliko unaprijed kreiranih paleta, a koje se dijele na tri kategorije:

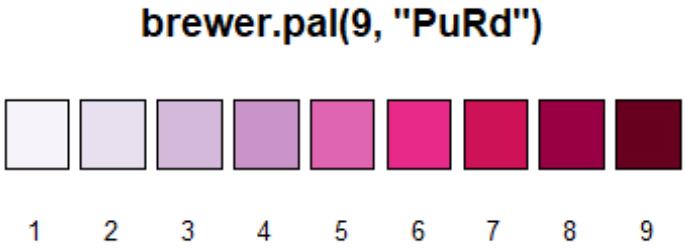
1. sekvencijalne – za kontinuirane varijable, svijetle boje za niske, a tamne boje za visoke vrijednosti
2. kvalitativne – za kategorijske varijable, ciljaju maksimalnoj vizualnoj razlici između kategorija
3. divergentne – za varijable koje divergiraju od neke centralne vrijednosti ili imaju raspon od negativnih prema pozitivnim vrijednostima. Svijetle boje su namijenjene vrijednostima u sredini raspona, dok su niske i visoke vrijednosti prikazane u kontrastnim tamnim tonovima.

Pregled paleta nalazi se u datoteci Slika/RColorBrewer.png, a može ih se dohvatiti naredbom `display.brewer.all()` nakon instalacije i učitavanja paketa `RColorBrewer`.

Paleta se dohvaća naredbom `brewer.pal(n, "ime_palete")` gdje je n broj željenih boja iz navedene palete.

Primjer:

```
barplot(rep(1,9), col = brewer.pal(9, "PuRd"),
        xlab="", ylab="", axes = FALSE,
        names.arg = 1:9, cex.names = 0.8,
         main='brewer.pal(9, "PuRd")' )
```



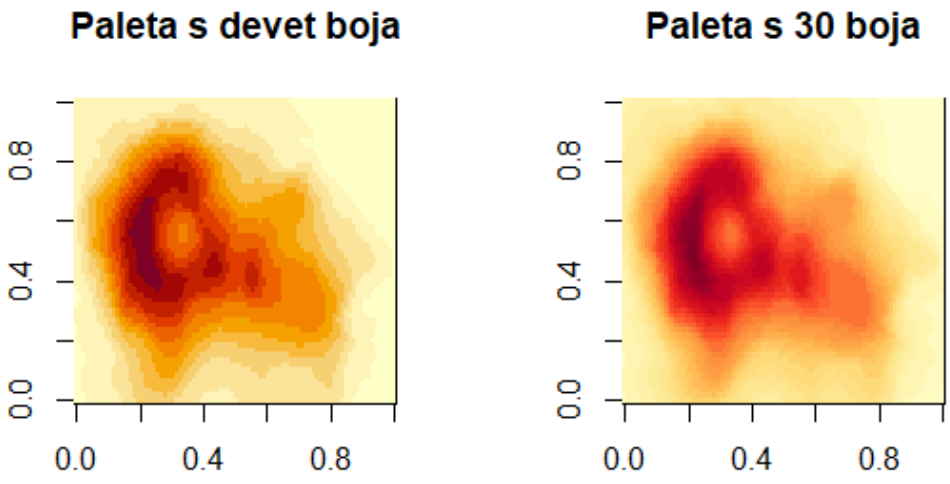
6.3.4. Funkcija `image()`

Funkcija `image()` služi za trodimenzionalni prikaz ili prikaz prostornih podataka oblika `(x, y, z)` u kojem je z reprezentiran bojom. Za potrebe prikaza, iz ugrađene baze podataka sustava R preuzet ćemo matricu `volcano` koja sadrži topografske podatke o vulkanu Maungawhau, upoznati se s njenom strukturom i nacrtati grafikon funkcijom `image()`.

```
data(volcano)
str(volcano)
```

```
##  num [1:87, 1:61] 100 101 102 103 104 105 105 106 107 108 ...
```

```
par(mfrow=c(1,2))

image(volcano, main="Paleta s devet boja")

paleta30 <- colorRampPalette(brewer.pal(9, "YlOrRd"))(30)
image(volcano, col=paleta30, main="Paleta s 30 boja")
```



Kvaliteta vizualizacije je bolja na desnoj slici u kojoj se koristilo više nijansi boja. Broj boja unutar neke palete regulira se funkcijom `colorRampPalette()` koja za rezultat vraća funkciju. Ako uz tu funkciju navedemo cijeli broj u zagradi, kao što smo gore naveli (30), onda to postaje vektor s fiksnim brojem nijansi boja iz navedene palete.

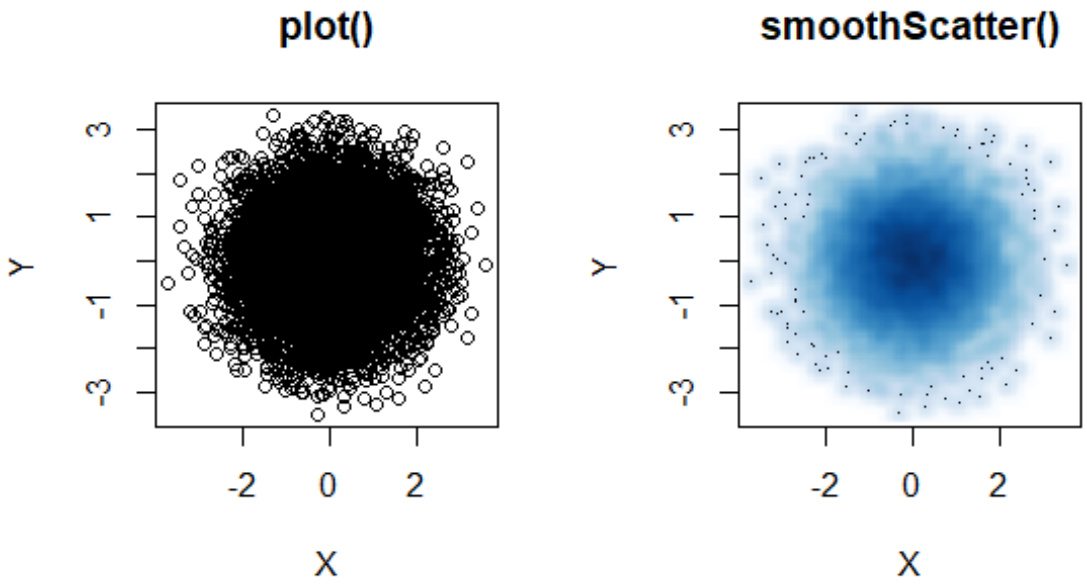
Zadatak 39
Ponovite gornji zadatak odabirom nove palete i broja boja. Isprobajte nekoliko varijanti.

6.3.5. Funkcija `smoothScatter()`

U slučajevima kada je gustoću točaka potrebno reprezentirati nijansama boja, dobra opcija grafičkoga prikaza je funkcija `smoothScatter()`.

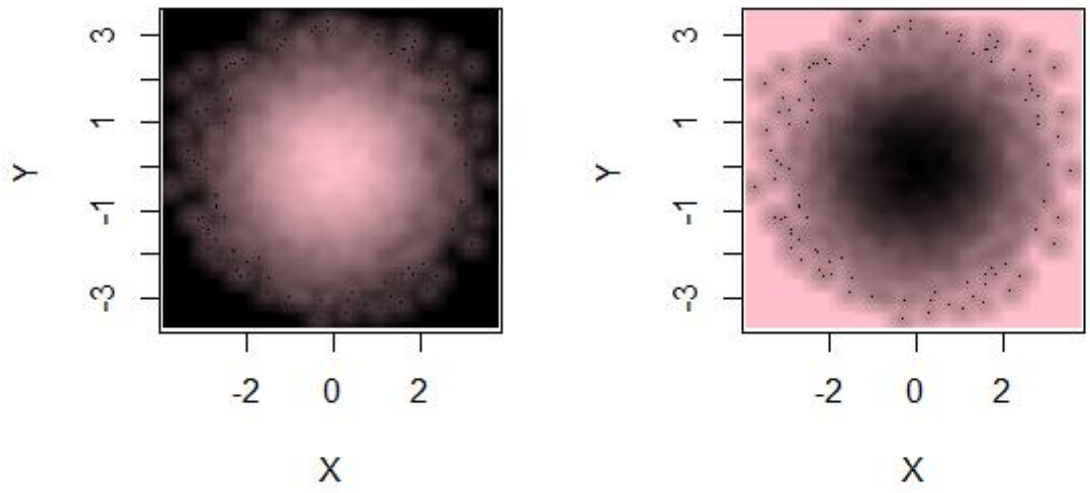
Za potrebe prikaza, izradit ćemo slučajan niz od 10,000 točaka iz standardne normalne razdiobe te usporediti rezultate funkcija `plot()` i `smoothScatter()`.

```
X <- rnorm (10000)
Y <- rnorm (10000)
Z <- cbind(X,Y)

par(mfrow=c(1,2))
plot(X,Y, main="plot()")
smoothScatter(X,Y, main="smoothScatter()")
```



Boje u smoothScater tipu grafikona mijenjamo argumentom `colramp` koji prihvaća samo funkcije nastale iz `colorRampPalette()`. Mijenjanjem nijansi boja može se postići efekt dubine ili ispupčenosti.

```
par(mfrow=c(1,2))
rozocrna_paleta <- colorRampPalette(c("pink", "black"))
crnoroza_paleta <- colorRampPalette(c("black", "pink"))

smoothScatter(X, Y, colramp = crnoroza_paleta)
smoothScatter(X, Y, colramp = rozocrna_paleta)
```



Zadatak 40
Napravite smoothScater tip grafikona iz već zadanih vektora X i Y s paletom boja po vlastitom izboru.

6.4. Pohranjivanje grafikona

Grafikoni koji se proizvedu u RStudiju mogu se pohraniti u datoteku da bi se kasnije mogli koristiti u drugim dokumentima ili podijeliti s drugim korisnicima. Najčešći oblik pohranjivanja je u oblicima .jpeg, .png te .pdf. Za pohranu takve datoteke potrebno je otvoriti novu datoteku određenoga tipa naredbama `jpeg()`, `png()` ili `pdf()` te u nju upisati naziv nove datoteke. U tu datoteku se zatim nacrta grafikon, jedan ili više njih, te se završi naredbom `dev.off()`.

```
jpeg("histogrami.jpeg")
hist(iris$Petal.Length, xlab = "Petal Length", col="plum")
dev.off()


png("histogrami.png")
hist(iris$Petal.Length, xlab = "Petal Length", col="plum")
dev.off()


pdf("histogrami.pdf")
hist(iris$Petal.Length, xlab = "Petal Length", col="plum")
dev.off()
```

Nove datoteke pohranjene su u radnom direktoriju.

7. MANIPULACIJA TEKSTOM U R-u

U cijelom procesu analize podataka, najviše vremena se obično provede u ‘uređivanju’ podataka. Podaci se smatraju urednima kada se dovedu u oblik gdje svaki redak predstavlja jednu opservaciju, a svaki stupac jednu varijablu mjerenja. Često je potrebno sintetizirati više skupova podataka iz različitih izvora u isti format, a često je slučaj da se podaci razbijaju na dijelove te onda sastavljaju u skupove drugačijih oblika kako bi bili uredni.

Obrada znakovnih podataka često podrazumijeva homogeniziranje formata, ispravljanje pogrešaka te razne manipulacije i transformacije podataka. R ima bogat skup funkcija dizajniran za manipulaciju znakovnih podataka čime se olakšava čišćenje, formatiranje i obrada informacija. Upoznat ćemo se s osnovama manipulacije znakovnih podataka u R-u.

7.1. Regularni izrazi

Regularni izrazi mogu se definirati kao uzorak znakova koji opisuje skup nizova ili jednostavnije rečeno, regularni izrazi daju upute funkciji na koji način da pariraju i zamijene određene znakove.

Ako je, na primjer, potrebno izvršiti neku radnju vezanu za numeričke znakove unutar neke znakovne varijable, umjesto da se poziva funkcija za svaku znamenku posebno, 1, 2, 3… dovoljno je staviti regularni izraz `\d`, `[[:digit]]` ili `[0-9]` koji će obuhvatiti sve znamenke. Ovo je primjer kako regularni izrazi skraćuju posao i poopćavaju slučajeve.

U sustavu R regularni izrazi postoje u dvama oblicima: produljeni (engl. *extended*) i oni nalik jeziku Perl (engl. *Perl-like*). Kao što ćemo vidjeti, u mnogim funkcijama za manipulaciju tekstom postoji argument s predefiniranom vrijednosti `perl=TRUE/FALSE`. U ovom tečaju obradit ćemo samo produljeni oblik.

Regularni izrazi konstruiraju se analogno aritmetičkim izrazima, korištenjem raznolikih operatora.

Tablica odabranih regularnih izraza za korištenje u R naredbama:

Element	Značenje
\\d ili [[:digit:]] ili [0-9]	bilo koja znamenka
\\D ili [[:alpha:]] ili [A-Za-z]	bilo koje slovo
[[:alnum:]]	bilo koji alfanumerički znak (slovo ili znamenka)
[[:punct:]]	bilo koji znak interpunkcije
\\s ili [[:space:]]	bilo kakva praznina
\\.	točka
[abc]	samo navedeni znakovi
[^abc]	svi znakovi osim navedenih
(ab|cd)	niz ‘ab’ ili niz ‘cd’
.	bilo koji znak osim novog reda
?	prethodni znak je opcionalan
+	jedno ili više ponavljanja prethodnoga znaka
*	nijedno ili više ponavljanja prethodnoga znaka
{n}	prethodni znak se ponavlja točno n puta
{n,}	prethodni znak se ponavlja n ili više puta
{n,m}	prethodni znak se ponavlja barem n puta, ali ne više od m puta
^	oznaka za početak stringa
$	oznaka za kraj stringa
|	ili
( )	zagrade za grupiranje
[ ]	zagrade za klasu znakova
\	znak za izlaz iz načina rada regularnog izraza.

Primjeri s regularnim izrazima bit će obrađeni u nastavku u sklopu funkcija namijenjenih radu sa znakovnim tipovima podataka.

7.2. Osnovne funkcije za rad sa znakovima

Niz znakova se često naziva i string. Znakovni vektor možemo izraditi, uz već spomenutu funkciju `c()`, i pomoću funkcije `character()`.

`is.character()` je funkcija kojom se provjerava je li neki vektor znakovnoga tipa, a funkcijom `as.character()` može se bilo koji vektor pretvoriti u znakovni.

`touper()` mijenja mala slova stringa u velika tiskana, dok njena inverzna funkcija `tolower()` radi obrnuto.

```
tekst <- "Šifra za LOCK je QWERT123qwert."
toupper(tekst)

## [1] "ŠIFRA ZA LOCK JE QWERT123QWERT."

tolower(tekst)

## [1] "šifra za lock je qwert123qwert."
```

Osnovni paketi sustava R posjeduju niz funkcija za prepoznavanje i selektiranje dijelova znakovnoga vektora kao što su primjerice: `substr()`, `strsplit()`, `grep()`, `grepl()`, `sub()`, `gsub()`, `strsplit()` te mnoge druge.

- **`substr()`** je funkcija za izdvajanje znakova iz stringa na točno zadanim lokacijama.

```
substr(tekst,start=3,stop=5)

## [1] "fra"

substr(tekst,start=7,stop=8)

## [1] "za"
```

- **`strsplit(x, split)`** je funkcija koja razdvaja znakovni vektor ili string `x` na manje podstringove, a granice razdvajanja su određene argumentom `split` koji je također znakovnoga tipa. Sadržaj iz stringa `split` nije uključen u nove podstringove.

```
rijeci <- strsplit(tekst, split="je",fixed=TRUE)
rijeci

## [[1]]
## [1] "Šifra za LOCK "  " QWERT123qwert."

str(rijeci)

## List of 1
##  $ : chr [1:2] "Šifra za LOCK " " QWERT123qwert."
```

Rezultat funkcije `strsplit()` je lista koja u sebi sadrži znakovne vektore.

```
a <- c("San Diego", "New York", "Sao Paolo", "Cape Town")

aSplit <-strsplit(a,split=" ",fixed=TRUE)
aSplit

## [[1]]
## [1] "San"   "Diego"
##
## [[2]]
## [1] "New"  "York"
##
## [[3]]
## [1] "Sao"   "Paolo"
##
## [[4]]
## [1] "Cape" "Town"
```

- **`paste()`** i **`paste0()`** – kombinira nekoliko znakovnih podataka u jedan string pri čemu je standardni znak razdvajanja bjelina " “, ali može se definirati i bilo koji drugi string za razdvajanje argumentom `sep`. Koristi se i funkcija `paste0()` kod koje je standardni znak razdvajanja”" (odnosno, bez razdvajanja).

Primjer:

```
paste(aSplit[[1]][1], aSplit[[1]][2])

## [1] "San Diego"

paste0(aSplit[[1]][1], aSplit[[1]][2])

## [1] "SanDiego"
```

Primjer:

```
paste(1:5, "dio", sep = ". ")

## [1] "1. dio" "2. dio" "3. dio" "4. dio" "5. dio"
```

Funkcije za identificiranje uzorka u tekstu

- **grep(uzorak, x, ...)** – vraća indekse elemenata ili elemente vektora x koji sadrže podstring uzorak.

```
voce <- c('banana', 'jabuka', '5m0kva', 'mandarina', 'kru?ka')

grep('r',voce)  # vraća vektor s indeksima elemenata koji sadrže traženi uzorak

## [1] 4 5

grep('r', voce, value=TRUE)  # vraća vektor s elementima koji sadrže traženi uzorak

## [1] "mandarina" "kru?ka"
```

Primjer s regularnim izrazima:

```
grep('[[:digit:]]',voce,value=TRUE)

## [1] "5m0kva"

grep('[[:punct:]]',voce,value=TRUE)

## [1] "kru?ka"

grep('?',voce)

## [1] 1 2 3 4 5

grep("\\?",voce)

## [1] 5

#drugi način: argument fixed kaže da se navedeni string uzme doslovno, a ne kao regularni izraz
grep('?', voce, fixed=TRUE, value=TRUE)

## [1] "kru?ka"

grep('an|ab',voce, value =TRUE)

## [1] "banana"    "jabuka"    "mandarina"
```

- **grepl(uzorak, x, ...)** – vraća logički vektor koji s TRUE i FALSE vrijednostima indicira koji elementi vektora x imaju traženi uzorak

```
grepl('na',voce)
```

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

Primjer: iz podatkovnog okvira deniro izdvojite sve filmove koji u sebi sadrže riječ "city".

```
grep("city", deniro$Title, value = TRUE, ignore.case = TRUE) #vraća samo naslove filmova
```

```
## [1] "Night and the City" "City by the Sea"
```

```
deniro[grep("City", deniro$Title),] #vraća cijeli redak podatkovnog okvira
```

```
##    Year Score              Title
## 34 1992    67 Night and the City
## 56 2002    48    City by the Sea
```

Zadatak 41

41.1 Funkcijom library() učitajte paket ggplot2, a potom funkcijom data() učitajte skup msleep iz paketa ggplot2 koji sadrži podatke o karakteristikama spavanja sisavaca i upoznajte se s njegovom strukturom i sadržajem.
41.2 Funkcijom grep() izdvojite sve zapise primata (gdje order poprima vrijednost "Primates") i pohranite ih u novi podatkovni okvir naziva primati.
41.3 Iz podatkovnog okvira primati, izdvojite sve zapise primata čiji nazivi (varijabla name) završavaju samoglasnikom.

Funkcije za supstituiranje uzorka u tekstu

- **sub(uzorak, zamjena, x)** – supstituira prvi podstring uzorak s novim nizom znakova zamjena u svakom elementu znakovnoga vektora x.

```
sub('a','$', voce)
```

```
## [1] "b$nana"    "j$buka"    "5m0kv$"    "m$ndarina" "kru?k$"
```

- **gsub(uzorak, zamjena, x)** – supstituira sve podstringove uzorak s novim stringom zamjena u svakom elementu znakovnoga vektora x.

```
gsub('a','$',voce)
```

```
## [1] "b$n$n$"    "j$buk$"    "5m0kv$"    "m$nd$rin$" "kru?k$"
```

Primjer funkcija sub() i gsub() s regularnim izrazima:

```
sub('[[:alpha:]]','$',voce)
```

```
## [1] "$anana"    "$abuka"    "5$0kva"    "$andarina" "$ru?ka"
```

```
gsub('[[:alpha:]]','$',voce)
```

```
## [1] "$$$$$$"    "$$$$$$"    "5$0$$$"    "$$$$$$$$$" "$$$?$$"
```

Zadatak 42

U podatkovnom okviru `primati`, u varijabli `vore` zamjenite "omni" sa "svejedi", "carni" s "mesojedi" i "herbi" s "biljojedi". Ispišite podatkovni okvir `primati` i provjerite je li cijeli stupac pravilno izmijenjen.

Funkcije za lociranje uzorka u tekstu

- **`regexpr(uzorak, x)`** – vraća cjelobrojni vektor iste duljine kao `x` koji sadrži početne pozicije prvih podudaranja uzorka `uzorak` unutar svakog elementa vektora `x`. Ako nema podudaranja, vraća `-1`.

```
voce
```

```
## [1] "banana"    "jabuka"    "5m0kva"    "mandarina" "kru?ka"
```

```
regexpr('na',voce)
```

```
## [1]  3 -1 -1  8 -1
## attr(,"match.length")
## [1]  2 -1 -1  2 -1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

- **`gregexpr()`** – vraća listu iste duljine kao vektor `x` čiji je svaki element istog oblika kao rezultat funkcije `regexpr()`, ali daje startne pozicije svih podudaranja unutar svakog elementa vektora `x`.

```
voce
```

```
## [1] "banana"    "jabuka"    "5m0kva"    "mandarina" "kru?ka"
```

```
gregexpr('na',voce)
```

```
## [[1]]
## [1] 3 5
## attr(,"match.length")
## [1] 2 2
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
##
## [[2]]
## [1] -1
## attr(,"match.length")
## [1] -1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
##
## [[3]]
## [1] -1
## attr(,"match.length")
```

```
## [1] -1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
##
## [[4]]
## [1] 8
## attr(,"match.length")
## [1] 2
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
##
## [[5]]
## [1] -1
## attr(,"match.length")
## [1] -1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

Funkcije za izvlačenje uzorka iz teksta

- **`regmatches(x, m)`** – ili izdvaja ili supstituira one podstringove iz `x` o kojima `m` sadrži informacije gdje je `m` rezultat funkcije `(g)regexpr()`.

Izdvajanje podstringova:

```
voce <- c('banana', 'jabuka', '5m0kva', 'mandarina', 'kru?ka')
m <- regexpr("na", voce)   # vraća samo prve pronalaske uzorka "na" u svako
m elementu vektora "voce"
m
```

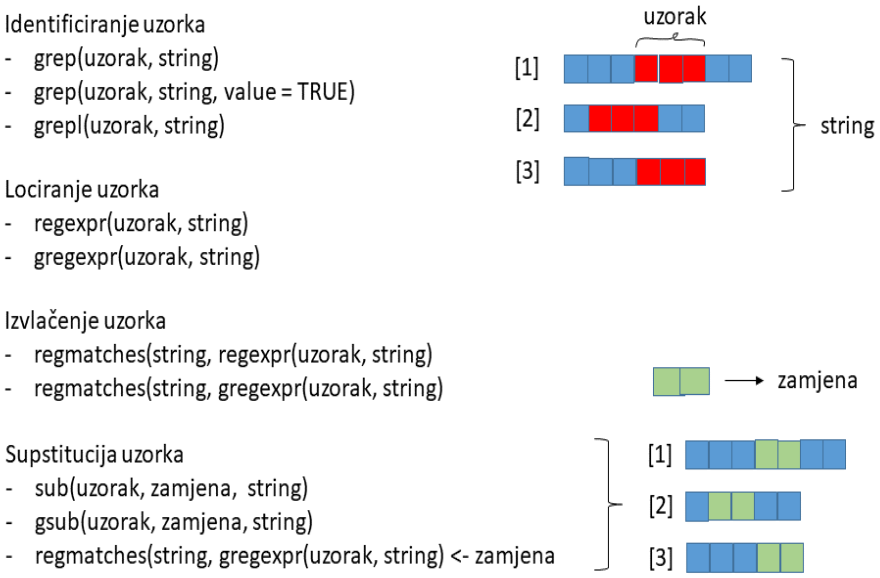
```
## [1]  3 -1 -1  8 -1
## attr(,"match.length")
## [1]  2 -1 -1  2 -1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

```
regmatches(voce,m)
```

```
## [1] "na" "na"
```

Za supstituciju, funkciji `regmatches()` potrebno je pridružiti supstituiranu vrijednost:

```
m <- gregexpr("na", voce)  # vraća sve pronalaske na kojima se pojavljuje u
zorak "na"
regmatches(voce,m) <- "._."
voce
## [1] "ba._.._."   "jabuka"     "5m0kva"     "mandari._." "kru?ka"
```



7.3. Učitavanje sadržaja s *weba*:

Nekada je skup podataka koji se analizira dostupan putem weba, te je korisno znati da R pruža mogućnost direktnog učitavanja datoteka s *web*-izvora. Prednost takvog učitavanja se očituje kod podataka koji se redovito ažuriraju, pa se prije svake analize preuzimaju aktualni podaci, a ne oni koji su se nekad davno pohranili lokalno.

Prije ovakvog načina preuzimanja podataka, potrebno je prethodno provjeriti uvjete korištenja.

Primjer učitavanja skupa podataka koji je javno dostupan (licenca *Creative Commons*) na *web*-stranici https://data.gov.au/data/dataset/abc-local-stations:

```
abc <- "http://www.abc.net.au/local/data/public/stations/abc-local-radio.csv"

radio <- read.table(abc,
                    header = TRUE,
                    sep ="," ,
                    stringsAsFactors = FALSE)



radio_2 <- read.table(abc,
                    header = TRUE,
                    sep ="," ,
                    stringsAsFactors = TRUE)
```

7.4. Primjer analize naziva časopisa

U varijablu `biomed` učitat ćemo tablicu iz radnog direktorija koja sadrži podatke o velikom broju znanstvenih časopisa, a zatim ćemo analizirati nazive časopisa iz te tablice na način da ćemo istražiti koje su najčešće korištene riječi u njihovim nazivima.

```
biomed <-read.table("Podaci/biomedcentraljournallist.txt", header=TRUE, sep
=",", na.strings="NA", dec=".", strip.white=TRUE, stringsAsFactors = FALSE)

str(biomed, vec.len = 1) # vec.len = 1 argument prikazuje manje podatka za
svaku varijablu, za ljepši ispis
```

```
## 'data.frame':    855 obs. of  7 variables:
##  $ Publisher     : chr  "Springer" ...
##  $ Journal.name  : chr  "3 Biotech" ...
##  $ Abbreviation  : chr  "3 Biotech" ...
##  $ ISSN          : chr  "2190-5738" ...
##  $ URL           : chr  "http://www.springer.com/13205" ...
##  $ Start.Date    : int  2011 2015 ...
##  $ Citation.Style: chr  "n/a" ...
```

S obzirom na to da nas zanimaju samo nazivi časopisa, izdvojit ćemo ih u poseban vektor.

```
nazivi <- biomed$Journal.name

#Radi lakše usporedbe, pretvorit ćemo sva slova u mala
nazivi <- tolower(nazivi)

#uklonit ćemo sve interpunkcijske znakove iz naziva
nazivi <- gsub('[[:punct:]]', "", nazivi)

#radi uklanjanja interpunkcije između riječi stvorio se višak bjelina, pa ć
emo to popraviti
nazivi <- gsub("\\s+", " ", nazivi)
```

Sada ćemo razdvojiti nazive na pojedinačne riječi.

```
rijeci <- strsplit(nazivi, " ")  #rezultat je lista, pa ćemo je pretvoriti
u vektor
rijeci <- unlist(rijeci)
```

Funkcija `table()` dat će kontingencijsku tablicu, odnosno tablicu koja sadrži prebrojavanja svih kombinacija elemenata iz arugmenta. Ako je odmah pretvorimo u podatkovni okvir pomoću `as.data.frame()`, rezultat će biti kontingencijska tablica koja za prvi stupac ima nazive svih elemenata koje je izbrojala.

```
tablica <- as.data.frame(table(rijeci))
```

Preostalo je još sortirati tablicu silazno, čime ćemo dobiti uvid u najčešće korištene riječi u nazivima znanstvenih časopisa u danom uzorku.

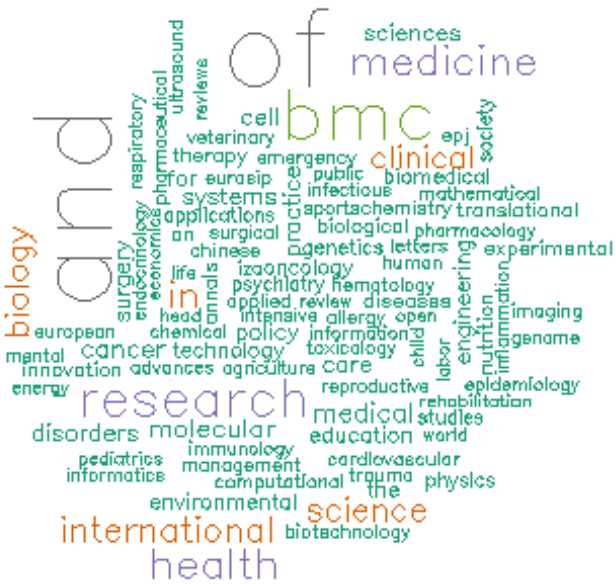
```
tablica <- tablica[order(tablica$Freq, decreasing=T),]

head(tablica)
```

```
##       rijeci Freq
## 371  journal  234
## 474       of  221
## 30       and  215
## 98       bmc  126
## 578 research   85
## 310   health   61
```

Prikažimo grafički zastupljenost riječi u nazivima časopisa pomoću paketa i funkcije `wordcloud`.

Paket `wordcloud` je prvo potrebno instalirati i učitati.

```
#install.packages("wordcloud")
library(wordcloud)

wordcloud(tablica$rijeci, tablica$Freq,
          #dodatni argumenti:
          min.freq=2, max.words=100, random.order=T,
          rot.per=.15, colors=brewer.pal(8, "Dark2"),
          vfont=c("sans serif","plain"))
```



8. RAD S DATUMIMA

Radi važnosti pravilnoga definiranja datuma prilikom vizualizacije i analize prostorno vremenskih podataka proći ćemo kroz osnove za rad s datumima koje su unutar mogućnosti osnovnih paketa sustava R.

U sustavu R postoje tri klase kojima se može zabilježiti vrijeme:

- klasa `Date` – pohranjuje datum kao broj dana od 1.1.1970 UTC.
- klase `POSIXct` – pohranjuje datum i vrijeme kao broj sekundi od 1.1.1970 UTC.
- klasa `POSIXlt` – pohranjuje datum i vrijeme u obliku liste (hour, min, sec, mon,…) čime je lakše dohvatiti pojedini element datuma.

Unatoč 'numeričkom' načinu pohranjivanja datuma u memoriju, svaka od ovih klasa ima svoj standardni način na koji ispisuje datum, a pohrana datuma u memoriji kao broj dana ili sekundi omogućava lakše i točnije obavljanje matematičkih operacija nad datumima, kao na primjer računanja razlike između dviju vremenskih točaka.

Funkcija `as.Date()` prebacuje datum zapisan kao tekst (engl. *character*) u datum klase `Date`, a funkcije `as.POSIXct()`/`as.POSIXlt()` prebacuju datum i vrijeme zapisane kao tekst u vremenski zapis tipa `POSIXct`/`POSIXlt`. Sve tri funkcije nude mogućnost odabira raznih formata datuma i vremena, dok funkcije `as.POSIXct()` i `as.POSIXlt()` omogućuju i kontrolu vremenske zone.

8.1. Klasa `Date`

Standardan način formata kojim se zapisuju i ispisuju datumi u sustavu R je oblika `YYYY-mm-dd` (godina-mjesec-dan).

```
datum <- "2019-10-23"
class(datum)

## [1] "character"

datum <- as.Date(datum)   #znakovni tip podatka se mora pretvoriti u klasu
Date
class(datum)

## [1] "Date"
```

Ako se neki zapis želi pohraniti kao datum, ali je zapisan u nestandardnom obliku, onda se `as.Date()` poziva s dodatnim argumentom `format` u kojem se format datuma opiše pomoću sljedeće sintakse:

Šifra	Značenje	Šifra	Značenje
%d	dan u mjesecu (broj)	%B	mjesec (puni naziv)
%m	mjesec (broj)	%y	godina (2 znamenke)
%b	mjesec (skraćeni naziv)	%Y	godina (4 znamenke)

Primjer:

```
datum <- "23.10.2019"  #znakovni tip (character)
datum <- as.Date(datum, format = "%d.%m.%Y")  #datum je sada postao klase D
ate
datum  #pri ispisu će imati standardni format datuma
```

```
## [1] "2019-10-23"
```

Funkcijom `weekdays()` može se saznati koji je dan u tjednu određeni datum, funkcija `months()` može ispisati naziv mjeseca iz datuma, a funkcijom `quarters()` može se saznati kojem od četiriju kvartala u godini pripada zadani datum.

```
weekdays(datum)
```

```
## [1] "srijeda"
```

```
months(datum)
```

```
## [1] "listopad"
```

```
quarters(datum)
```

```
## [1] "Q4"
```

Ako iz datuma želimo izdvojiti samo godinu, to možemo napraviti pomoću funkcije `format()`:

```
format(datum, "%Y")
```

```
## [1] "2019"
```

Zadatak 43
Saznajte koji dan u tjednu su sljedeći datumi.
NAPOMENA: Datumi se prvo moraju pretvoriti u objekt klase `Date`, pa se tek onda može ispitati koji su dan u tjednu.

```
datum1 <- "16/10/00"
datum2 <- "27. travanj, 2001"
datum3 <- "02/14/2016"
```

Ako na zapis klase `Date` primijenimo funkciju `unclass()`, datum će se pretvoriti u broj dana od 1. 1. 1970.

```
unclass(datum1)
```

```
## [1] "16/10/00"
```

Funkcija `Sys.Date()` vraća trenutačni datum u obliku klase `Date`, a funkcija `date()` trenutačni datum i vrijeme u znakovnom tipu podatka.

8.2. Klase `POSIXlt` i `POSIXct`

Kada želimo pohraniti i datum i vrijeme, koriste se objekti klase `POSIXct` ili `POSIXlt`.

`POSIXct` pohranjuje datum kao broj sekundi od ponoći 01.01.1970. (ct = calendar time).

`POSIXlt` pohranjuje datum kao listu koja sadrži dan, mjesec, godinu, sat, minutu, sekundu, itd… (lt = local time), ali mjeri vrijeme drugačije, na primjer godine broji od 1900.-te, mjesece označava brojevima 0-11, itd. Otvorite karticu Help i upišite POSIXlt za više detalja.

Standardni format ispisa je isti u obama slučajevima: `%Y-%m-%d %H:%M:%S tz` gdje je tz vremenska zona (engl. *time zone*).

Vektor se pretvara u objekt tipa `POSIXct`/`POSIClt` funkcijama `as.POSIXct()`/`as.POSIXlt()`.

```
datum <- "1990-01-21 18:47:22"

datum_LT <- as.POSIXlt(datum)
datum_LT

## [1] "1990-01-21 18:47:22 CET"

datum_CT <- as.POSIXct(datum)
datum_CT

## [1] "1990-01-21 18:47:22 CET"
```

Iako se iz njihove strukture i ispisa rezultata ne vidi gotovo nikakva razlika, `datum_LT` i `datum_CT` nisu isti objekti i u kartici Environment stavljeni su pod različite kategorije.

S obzirom na to da je `datum_LT` lista, pomoću operatora `$` moguće je dohvatiti pojedine elemente tog datuma, kao na primjer godina, mjesec, dan u mjesecu, sat, itd.

```
datum_LT$year #broji godine od 1900.-te

## [1] 90

datum_LT$mon  #mjesece označava brojevima 0-12

## [1] 0

datum_LT$hour #sate označava brojevima 0-23

## [1] 18

datum_LT$mday

## [1] 21
```

8.3. `strptime()` i `strftime()`

Funkcija `strptime()` djeluje jednako kao `as.POSIXlt()`, ali se primijenjuje samo na znakovnim tipovima podataka. (Prijašnje dvije fukcije `as.POSIXlt()` i `as.POSIXct()` mogu se primijeniti nad raznim tipovima podataka).

```
datum <- "18/01/2019"

datum_S <- strptime(datum, "%d/%m/%Y")
```

Funkcija `strftime()` vraća objekte tipa `POSIXlt`, `POSIXct` i `Date` u znakovni tip podatka. Isto se može napraviti i funkcijom `as.character()`, ali funkcije `strptime()`/`strftime()` imaju bogatiji izbor formata koje je moguće pronaći u dokumentaciji.

```
strftime(datum_S)

## [1] "2019-01-18"

strftime(datum_S, format = "%j")  # Redni broj dana u godini (rezultat je z
nakovni tip, ne numerički)

## [1] "018"

strftime(datum_S, format = "%A")  # Puni naziv dana u tjednu

## [1] "petak"

strftime(datum_S, format = "%c")  # Vrsta formata ovisno o regionalnim post
avkama na računalu

## [1] "pet sij 18 00:00:00 2019"
```

8.4. Vremenske zone

Vremenske zone kontroliraju se argumentom `tz`. Popis mogućih vremenskih zona (Olsonove vremenske zone) dobije se naredbom `OlsonNames()`, a vremensku zonu iz regionalnih postavki računala dobije se naredbom `Sys.timezone()`.

```
Sys.timezone()

## [1] "Europe/Warsaw"
```

Zadatak 44
Dan je podatkovni okvir `djeca` koji se sastoji od imena i datuma rođenja petero djece.

```
Ime <- c("Leonid", "Ania", "Marta", "Mateo", "Jakov")
Datum <- c("04/12/2015", "31/10/2017", "03/04/2016", "18/09/2018", "14/08/2
016")


djeca <- as.data.frame(list(Ime = Ime, DatumRođenja =Datum), stringsAsFacto
rs = FALSE)
```

```
djeca
```

```
##      Ime DatumRođenja
## 1 Leonid   04/12/2015
## 2   Ania   31/10/2017
## 3  Marta   03/04/2016
## 4  Mateo   18/09/2018
## 5  Jakov   14/08/2016
```

44.1 Pretvorite stupac `DatumRođenja` u objekt klase `Date`.
44.2 Dodajte novi stupac naziva `Dan` koji će navesti na koji dan u tjednu je svako dijete rođeno (koristite funkciju `strftime()`).
44.3 Dodajte još jedan stupac naziva `BrojDana` koji će prikazati starost svakoga djeteta u danima.

9. UVOD U STATISTIKU UZ SUSTAV R

Statistika se kao znanstvena disciplina bavi razvojem metoda prikupljanja, opisivanja i analiziranja podataka te primjenom tih metoda u procesu donošenja zaključaka.

Primijenjena statistika je znanost učenja iz podataka na način da kontrolira i komunicira nesigurnost u zaključcima te na taj način pruža bitne smjernice za upravljanje znanstvenim i društvenim napretkom. U svakodnevnom životu riječ statistika koristimo kada brojčanim vrijednostima pokušavamo opisati bitne karakteristike nekoga skupa podataka (populacije/uzorka).

Statističari primijenjuju statističke metode i način razmišljanja u raznim znanstvenim, društvenim i poslovnim područjima kao što su astronomija, biologija, obrazovanje, ekonomija, inženjerstvo, genetika, marketing, medicina, psihologija, javno zdravstvo, sport i drugo.

"Najbolja stvar u bavljenju statistikom je ta da statističar ima priliku igrati se u svačijem dvorištu." (John Tukey, Bell Labs, Sveučilište Princeton).

9.1. Osnovni pojmovi u statistici

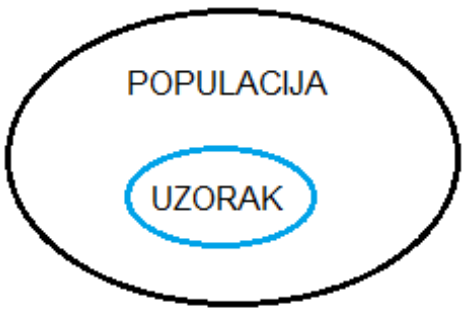
9.1.1. Statistička populacija

U statistici, populacija je ukupan skup jedinki koje su predmet nekog interesa i o kojima se želi donijeti zaključak. Populacija može biti konačna ili beskonačna.

Primjeri statističkih populacija:

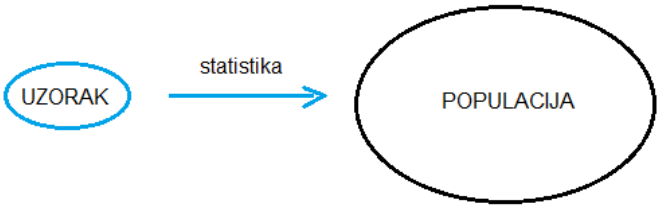
Ako želimo znati udio sredovječnih muškaraca koji nemaju srčani udar nakon što su konzumirali određeni lijek, onda populaciju čine svi sredovječni muškarci koji su primali ovu terapiju. Ako želimo znati postotak loše istokarenih dijelova nekog aparata u određenoj seriji proizvoda, tada populaciju čine svi istokareni dijelovi iz određene serije.

S obzirom na to da je mjerenje svih jedinki u populaciji najčešće nemoguće, populacija se uzorkuje procesom selekcije podskupa. Uzorak se odabire na način da bude reprezentativan za populaciju koja se istražuje.



9.1.2. Statističko zaključivanje

Statističko zaključivanje je zaključivanje o populaciji od interesa, a zasniva se na reprezentativnom uzorku iz populacije.



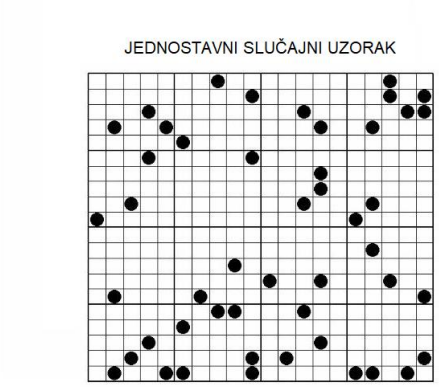
9.1.3. Uzorak

Uzorci se dijele na vjerojatnosne i nevjerojatnosne.

Nevjerojatnosni uzorci se ne smiju koristiti za statističko zaključivanje jer izbor jedinki u uzorku ne predstavlja promatranu populaciju. Jedinke u takvim uzorcima su odabrane jer su bile pogodne, dostupne, odabrane preko prethodnih jedinki, ili su ciljano odabrane kao neka podskupina. Primjer takvih uzoraka su ankete.

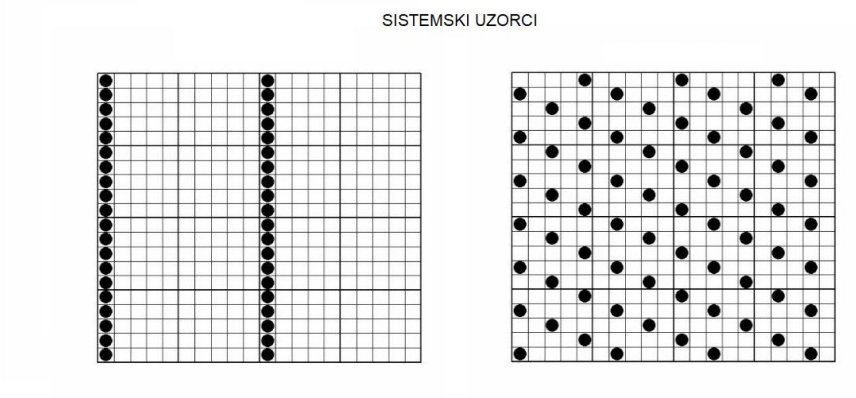
Vjerojatnosni uzorci su odabrani tako da se mogu koristiti za statističko zaključivanje. Postoji pet vrsta takvih uzoraka:

Jednostavni slučajni uzorak - dobije se tako da se slučajno odabere n jedinki iz populacije veličine N pri čemu svaka jedinka ima jednaku vjerojatnost da bude izabrana u uzorak.



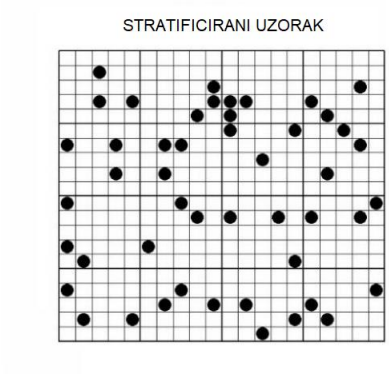
Prednosti: jednostavnost.
Nedostatci: postoji šansa da uzorak nije dovoljno precizan; uzorak može biti skup za realizaciju.

Sistematski uzorak - dobije se tako da se prva jedinka koja ulazi u uzorak odabere slučajno; nakon toga odabire se svaka n-ta jedinka iz populacije veličine N (npr. svaka deseta, svaka šesta).



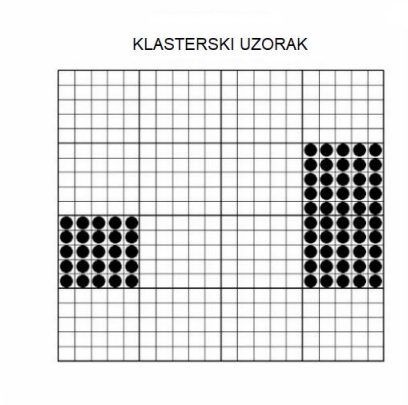
Prednosti: jednostavno za objašnjenje i sprovođenje.
Nedostatci: netočan ako postoji obrazac u podacima koji se ponavlja ciklički.

Stratificirani uzorak - populacija se prvo podijeli u nepreklapajuće podskupove (stratume). U ovisnosti o veličini podskupova, iz svakog se podskupa odabere slučajni uzorak odgovarajuće veličine.



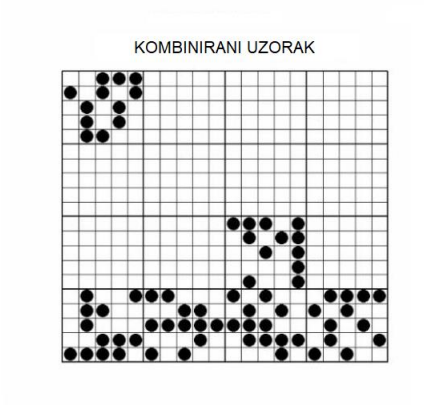
Prednosti: točnost, osigurava reprezentativnost s obzirom na relevantne varijable.
Nedostatci: zahtijeva određeno predznanje o populaciji koja se uzorkuje.

Klasterski uzorak - nakon podjele populacije na nepreklapajuće klastere, slučajno se odabere određeni broj klastera, nakon čega se uzorkuju sve jedinice unutar odabranih klastera.

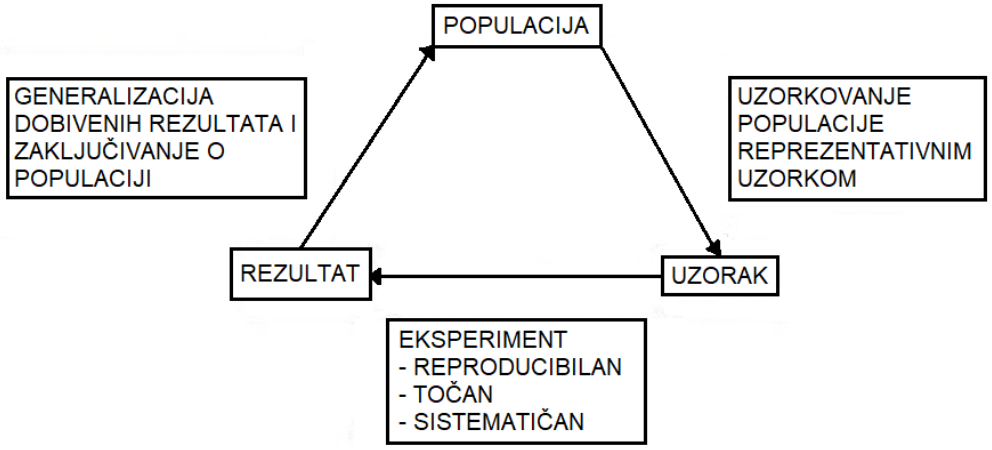


Prednosti: manji troškovi.
Nedostatci: neprecizan ako postoji velika varijabilnost među klasterima.

Kombinirani uzorak - dobije se slično kao klaster uzorka, no iz svakog slučajno odabranoga klastera odabere se slučajni ili sistematski uzorak.



Ako imamo neki od vjerojatnosnih uzoraka tada proces statističkoga zaključivanja možemo detaljnije opisati dijagramom koji slijedi:



9.1.4. Zavisnost dvaju uzoraka

Uzorci mogu biti zavisni i nezavisni.

- **Nezavisni uzorci** su oni kod kojih rezultati mjerenja i opažanja na jednom uzorku ne utječu na rezultate mjerenja i opažanja istih osobina na drugom uzorku.

Primjer:

1. Slučajnim izborom, na početku semestra, biramo 50 studenata druge godine, izabranoga smjera i fakulteta Sveučilišta u Zagrebu i testiramo njihovo znanje iz matematike.

2. Slučajnim izborom, na kraju semestra, biramo 50 studenata druge godine izabranoga smjera i fakulteta Sveučilišta u Zagrebu i testiramo njihovo znanje iz matematike.

- **Zavisni uzorci** su takvi da su mjerenja u jednom uzorku povezana s mjerenjima u drugom uzorku. Dobiju se ako se, na primjer, mjere ili opažaju osobine od interesa na istom skupu jedinki u različito vrijeme.

Primjer:

1. Slučajnim izborom, na početku semestra, biramo 50 studenata druge godine izabranoga smjera i fakulteta Sveučilišta u Zagrebu i testiramo njihovo znanje iz matematike.
2. Na kraju semestra, na istih 50 studenata druge godine izabranoga smjera i fakulteta Sveučilišta u Zagrebu testiramo znanje iz matematike.

9.1.5. Varijable

Varijabla je svojstvo ili stanje čije vrijednosti variraju. Vrijednosti varijable mijenjaju se u skladu s različitim čimbenicima. Vrijednosti nekih varijabli mijenjaju se brzo, poput vrijednosti dionica na burzi, dok se vrijednosti drugih varijabli mijenjaju rijetko, primjerice promjena imena iste osobe tijekom života.

Konceptualno, varijable dijelimo na nezavisne i zavisne (ponekad zvane varijablama ishoda). Nezavisne varijable zovu se eksperimentalne kada se njima manipulira u istraživanju, ili prediktorske varijable kojima se ne manipulira u istraživanju. Cilj je ustanoviti utjecaj nezavisnih na zavisnu varijablu.

PRIMJER: Učitelja matematike zanimaju faktori (varijable) koji utječu na rezultate testova matematike njegovih učenika. Iz iskustva pretpostavlja da bi na rezultate mogla utjecati količina vremena uložena u rješavanje zadataka te opća inteligencija učenika (IQ).

Varijable mogu biti:

- **Nezavisne** (eksperimentalne/prediktorske): vrijeme učenja, IQ
- **Zavisna** (varijabla ishoda): ishod testa iz matematike.

Varijable se dijele na:

- **Diskretne**: one koje mogu poprimiti konačan broj vrijednosti (npr. ocjena, razred, boja…)
- **Neprekidne** ili **kontinuirane**: koje mogu poprimiti neprebrojivo mnogo vrijednosti (npr. duljina, širina, vrijeme…)

Mjerenje varijabli

Za uspostavu odnosa između varijabli, varijable se moraju mjeriti. Mjerenje je konceptualizacija - pridruživanje brojeva vrijednostima varijable. Povezivanje koncepta (ideje) s mjerenjem (tehnikom) čini vrijednosti varijable empirijski vidljivima. Proces mjerenja varijabli zahtijeva skale mjerenja koje možemo shvatiti kao shemu za numeričku reprezentaciju vrijednosti varijable.

Mjerenje može biti direktno za varijable kao što su visina, temperatura, težina ili pak indirektno kao što su primjerice motivacija, inteligencija, pogodnost staništa za vrstu, znanje, itd. Ovisno o tome koji tip varijable mjerimo, moramo izabrati adekvatno oruđe, mjerni instrument, kao što su primjerice vaga, test, termometar, upitnik osobnosti i drugo.

Tipovi mjernih skala

Mjerne skale se razlikuju po tipu varijabli koje predstavljaju te svojstvima i interpretaciji brojeva sadržanih u skali. Tipovi mjernih skala su:

1. Nominalna
2. Ordinalna
3. Intervalna
4. Omjerna.

Vrsta skale kojom se mjeri varijabla određuje koje statističke postupke možemo koristiti prilikom njihove analize.

- **Nominalna skala** – vrijednosti dodijeljene varijabli predstavljaju opisne kategorije, ali nemaju uređene numeričke vrijednosti u odnosu na veličinu. Služe za kategorizaciju jedinki u konačan broj grupa koje se međusobno ne preklapaju (disjunktne grupe).

PRIMJERI: tip proizvoda, spol, brand, država, dan u tjednu…
RAČUNSKE OPERACIJE: prebrojavanje jedinki unutar grupa, proporcije.

- **Ordinalna skala** – podatke je moguće poredati po veličini ili prema intenzitetu. Pokazuju relativan položaj elemenata u grupi, opservacije nisu ekvidistantne.

PRIMJERI: ocjena, stupanj obrazovanja, kvaliteta proizvoda.
RAČUNSKE OPERACIJE: >,<, medijan

- **Intervalna skala** – vrijednosti varijable su numeričke, na kontinuiranoj skali (realni brojevi). Jednake razlike u izmjerenim vrijednostima varijable predstavljaju jednake razlike mjerenoga svojstva, tj. opservacije su ekvidistantne; ne postoji ishodište.

PRIMJER: temperatura na Fehrenhajtovoj ili Celzijusovoj skali, nadmorska visina.
RAČUNSKE OPERACIJE: oduzimanje, zbrajanje i ostalo, ali ne i omjeri vrijednosti.

- **Omjerna skala** – vrijednosti varijable su numeričke. Skala posjeduje ishodište (apsolutnu nulu) koje predstavlja odsutnost mjernoga svojstva.

PRIMJER: masa, visina, temperatura na Kelvinovoj skali.
RAČUNSKE OPERACIJE: oduzimanje, zbrajanje, množenje i dijeljenje.

Nominalna i ordinalna skala još se nazivaju i *kategorijskim skalama* te služe za mjerenje diskretnih tipova varijabli.

Intervalna i omjerna skala se jednim nazivom nazivaju *metričkim skalama* te služe za mjerenje kontinuiranih tipova varijabli.

9.1.6. Deskriptivna vs. inferencijalna statistika

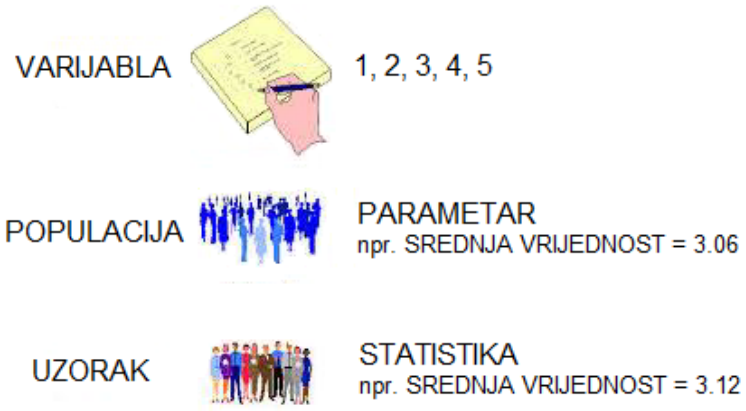
Deskriptivna statistika - obuhvaća postupke uređivanja, tabličnog i grafičkog prikazivanja podataka te izračunavanje opisnih statističkih pokazatelja.

Inferencijalna statistika - temelji se na teoriji vjerojatnosti i proučava metode kojima se pomoću dijela informacija (reprezentativnog uzorka) donose zaključci o cjelini (populaciji).

9.2. Deskriptivna statistika

Metodama deskriptivne statistike opisujemo odabrani uzorak ili populaciju što uključuje računanje statističkih pokazatelja poput aritmetičke sredine, maksimuma, minimuma, itd. Generalizacija zaključaka nije moguća.

Sumarni pokazatelj za cijelu populaciju naziva se **parametar**, dok se sumarni pokazatelj uzorka naziva **statistika**.



Analiza podataka veoma često završava samo deskriptivnom statistikom, bilo zbog nereprezentativnog uzorka, nedovoljnog broja varijabli ili nedovoljnoga znanja praktičara. Istraživači koriste deskriptivnu statistiku kako bi izvještavali o svom uzorku ili populaciji. Sumirajući informacije, deskriptivna statistika ubrzava i pojednostavljuje proces razumijevanja karakteristika bilo populacije bilo dijela populacije.

Sumiranje podataka o nekom skupu, populaciji ili uzorku, možemo podijeliti na mjere:

- centralne tendencije podataka
  - srednja vrijednost
  - medijan
  - mod
- varijabilnosti podataka
  - apsolutna varijabilnost
    - standardna devijacija
    - varijanca
    - raspon
    - interkvartilni raspon
  - relativna varijabilnost
    - koeficijent korelacije.

Unutar sustava R postoji mnoštvo funkcija korisnih za opis podataka. Primjere u R-u izvodit ćemo na prvih deset stupaca podatkovnog okvira `starwars` koji sadrži niz podatka o likovima iz filmova Star Wars, a koji se nalazi u paketu `dplyr`.

Za početak potrebno je učitati paket `dplyr`, podatkovni okvir `starwars`, te se upoznati sa skupom `starwars`.

```
library(dplyr)
data(starwars)
#zadnja tri stupca podatkovnog okvira su liste, zato ćemo radi jednostavnij
eg ispisa na ekran skup skratiti na prvih deset stupaca
starwars <- starwars[,1:10]
```

9.2.1. Mjere centralne tendencije

Srednja vrijednost

Srednja vrijednost, prosjek ili aritmetička sredina varijable $X = (x_1, x_2, \ldots, x_n)$ računa se formulom

$$\overline{X} = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

Funkcija kojom je računamo u R-u je `mean()`.

```
X <- 1:10
mean(X)

## [1] 5.5
```

Ukoliko skup podataka ima nedostajuće vrijednosti, potrebno je dodati argument `na.rm = TRUE` koji će ih zanemariti.

Primjer računanja srednje vrijednosti visina i težina likova iz podatkovnog okvira `starwars`:

```
# s obzirom na to da podatkovni okvir ima nedostajuće vrijednosti u ovim va
rijablama, moramo dodati argument koji ih zanemaruje: na.rm = TRUE.
mean(starwars$height, na.rm=TRUE)

## [1] 174.358

mean(starwars$mass, na.rm=TRUE)

## [1] 97.31186
```

Primjer: Koja je srednja vrijednost mase ljudi, a koja svih ostalih bića iz dijela populacije Star Wars čije su mjere poznate?

```
mean(subset(starwars$mass, starwars$species != "Human"), na.rm = TRUE)

## [1] 107.5611

mean(subset(starwars$mass, starwars$species == "Human"), na.rm = TRUE)

## [1] 82.78182
```

Možemo zaključiti da su u Star Wars sagi svemirci prosječno teži od ljudi.

Medijan

Medijan je vrijednost srednjega podatka koja podatke poredane po veličini dijeli na dva jednako brojna dijela. Ako je broj podataka neparan, medijan je vrijednost srednjega podatka, a ako je broj podataka paran, medijan je srednja vrijednost dvaju srednjih podataka.

```
#neparan broj elemenata skupa
X <- c(1, 2, 3, 4, 5)
median(X)
```

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

```
#paran broj elemenata skupa
Y <- c(1, 2, 3, 4, 5, 6)
median(Y)
```

```
## [1] 3.5
```

Primjer izračuna medijana na skupu starwars:

```
median(starwars$birth_year, na.rm = TRUE)
```

```
## [1] 52
```

U filmovima Star Wars polovica likova starija je od 52 godine.

Mod

Mod je najčešća vrijednost u skupu podataka. Može se jasno uočiti iz stupčastog dijagrama kao kategorija s najvišom frekvencijom, odnosno visinom stupca.

U osnovnim paketima sustava R ne postoji funkcija za mod, stoga se za potrebe računanja moda može koristiti ova funkcija:

```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
```

```
X <- c(2, 1, 2, 3, 3, 2, 3, 1, 2, 1)
Mode(X)
```

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

Primjer: Koja je najčešća boja očiju u populaciji Star Warsa?

```
Mode(starwars$eye_color)
```

```
## [1] "brown"
```

Prisjetimo se da se mjere deskriptivne statistike mogu jednostavno prikazati dijagramom pravokutnika (engl. *box and whiskers plot*) i funkcijom `summary()`.

```
boxplot(starwars$height, main = "Visine likova iz Star Warsa", ylab="Visina u cm")
```


```
summary(starwars$height)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    66.0   167.0   180.0   174.4   191.0   264.0       6
```

Karakteristike mjera centralne tendencije

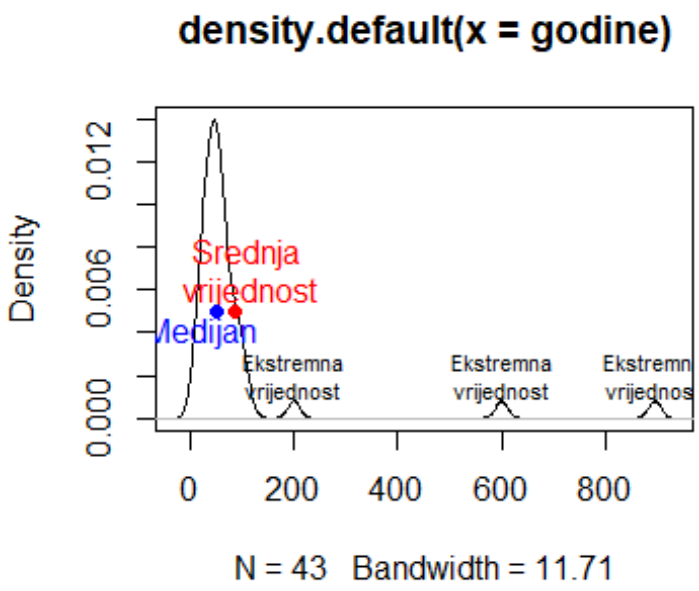
Srednja vrijednost je osjetljiva na ekstremne vrijednosti varijable (engl. *outlier*). Ako varijabla ima ekstremne vrijednosti na desnom/lijevom repu, one će pomaknuti srednju vrijednost udesno/ulijevo. Medijan nije osjetljiv na ekstremne vrijednosti (engl. *outlier*), stoga je u nekim slučajevima bolji pokazatelj centralne tendencije od srednje vrijednosti.

Primjer: Pogledajmo `summary` za godinu rođenja populacije iz Star Warsa, te prikažimo grafički njenu razdiobu.

```
summary(starwars$birth_year)
```

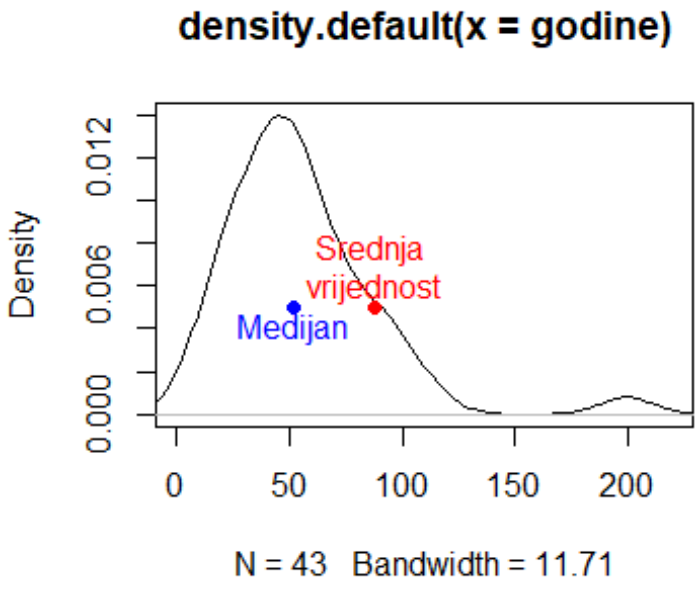
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    8.00   35.00   52.00   87.57   72.00  896.00      44
```

```
godine <-  subset(starwars$birth_year, !is.na(starwars$birth_year))
plot(density(godine))

points(mean(godine), 0.005, col="red", pch=16)
text(mean(godine)+30, 0.007, labels="Srednja \nvrijednost", col="red")

points(median(godine), 0.005, col="blue", pch=16)
text(median(godine)-30, 0.004, labels="Medijan", col="blue")

text(c(200, 600, 896), 0.002, labels="Ekstremna\nvrijednost", cex=0.7)
```



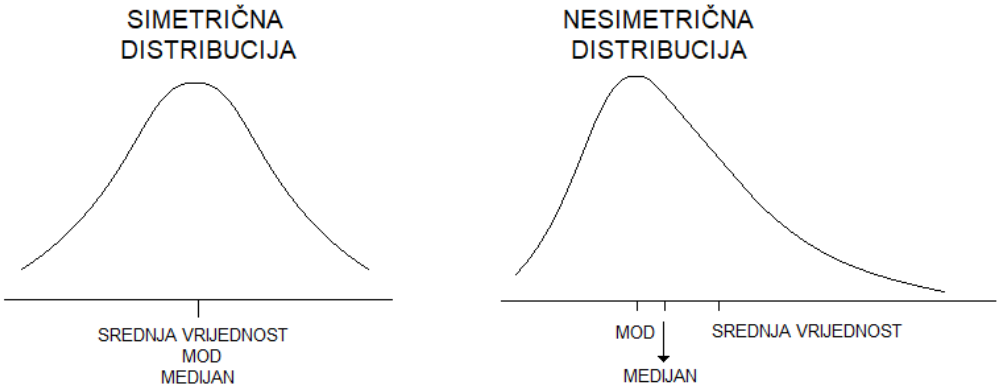
Ili pogledom izbliza:

```
#zumirani prikaz
plot(density(godine), xlim=c(0,220))

points(mean(godine), 0.005, col="red", pch=16)
text(mean(godine), 0.007, labels="Srednja \nvrijednost", col="red")

points(median(godine), 0.005, col="blue", pch=16)
text(median(godine), 0.004, labels="Medijan", col="blue")
```



Troje pojedinaca iz Star Warsa stari su između 200 i 900 godina, i to je pomaknulo srednju vrijednost u desno, na 87 godina, što je veće i od trećeg kvartila (Q3 = 72). S obzirom na to da je 75 % populacije mlađe od 72 godine, medijan 52 je bolji pokazatelj centralne tendencije starosti populacije nego srednja vrijednost.

Ukratko, utjecaj simetričnosti razdiobe na mjere centralne tendencije podataka vidi se jasno u donjem prikazu:



9.2.2. Mjere varijabilnosti podataka

Devijacija

Devijacija je odstupanje pojedinačne vrijednosti varijable od srednje vrijednosti varijable. Zbroj svih devijacija varijable jednak je nuli.

U osnovnim paketima sustavu R nema funkcije za računanje devijacije jer se devijacija sama po sebi rijetko koristi, no zato se može jednostavno dobiti sljedećom računicom:

```
brojevi <- 1:10
devijacija <- brojevi - mean(brojevi)
devijacija
```

```
##  [1] -4.5 -3.5 -2.5 -1.5 -0.5  0.5  1.5  2.5  3.5  4.5
```

```
sum(devijacija)
```

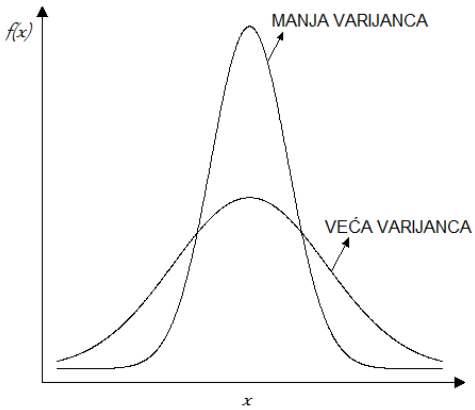
```
## [1] 0
```

Varijanca

Varijanca je prosječna suma kvadriranih devijacija, odnosno suma kvadriranih odstupanja od srednje vrijednosti varijable.

Varijanca populacije : $\sigma^2 = \frac{\Sigma(X_i - \overline{X})^2}{n} = \frac{SS}{n}$

Varijanca uzorka: $s^2 = \frac{\Sigma(X_i - \overline{X})^2}{n-1} = \frac{SS}{n-1}$



U R-u varijanca se izračunava funkcijom `var()`, i njome se konkretno dobiva varijanca uzorka. Za procjenu varijance populacije rezultat funkcije `var()` potrebno je pomnožiti s $\frac{n-1}{n}$.

Ukoliko skup podataka ima nedostajuće vrijednosti, potrebno je dodati argument `na.rm = TRUE` koji će ih zanemariti.

```
var(brojevi)
```

```
## [1] 9.166667
```

Primjer: Izračunajte varijancu visine i težine populacije Star Warsa kojoj su visina i težina poznate.

```
var(starwars$height, na.rm = TRUE)
```

```
## [1] 1208.983
```

```
var(starwars$mass, na.rm = TRUE)
```

```
## [1] 28715.73
```

Standardna devijacija

Standardna devijacija računa se kao drugi korijen iz varijance varijable. Ova mjera raspršenja vrijednosti varijable od prosjeka izražena je u istim jedinicama mjerenja kao i varijabla pa ju je lakše interpretirati nego varijancu koja je izražena u kvadriranim jedinicama mjerenja. Funkcija unutar sustava R kojom računamo standardnu devijaciju je `sd()` ili `sqrt(var())`.

```
sd(brojevi)
```

```
## [1] 3.02765
```

Primjer: Izračunajte standardnu devijaciju visine i težine populacije iz Star Warsa.

```
sd(starwars$height, na.rm = TRUE)
```

```
## [1] 34.77043
```

```
sd(starwars$mass, na.rm = TRUE)
```

```
## [1] 169.4572
```

Raspon

Raspon podataka je razlika između najveće i najmanje vrijednosti.

U sustavu R, funkcija `range()` vraća vektor s minimalnom i maksimalnom vrijednosti varijable.

```
range(brojevi)
## [1]  1 10

range(starwars$height, na.rm=TRUE)
## [1]  66 264

range(starwars$mass, na.rm=TRUE)
## [1]   15 1358
```

```
range(starwars$birth_year, na.rm=TRUE)
## [1]   8 896

#Za duljinu raspona postavimo funkciju range unutar funkcije diff koja raču
na razlike između elemenata vektora.
diff(range(starwars$mass, na.rm=TRUE))

## [1] 1343

diff(range(starwars$birth_year, na.rm=TRUE))

## [1] 888
```

Kvartili

Kvartili dijele podatke u četiri jednako brojna skupa.

U sustavu R kvartili su vidljivi u rezultatu funkcije `summary()`, a mogu se dobiti i naredbom `quantile()`.

```
quantile(brojevi)

##    0%   25%   50%   75%  100%
##  1.00  3.25  5.50  7.75 10.00
```

Unutarnje granice u ovoj podijeli (25 %, 50 % i 75 %) nazivaju se prvi kvartil (Q_1), drugi kvartil (Q_2) i treći kvartil (Q_3). Drugi kvartil je ujedno i medijan.

```
quantile(starwars$birth_year, na.rm=TRUE)

##   0%  25%  50%  75% 100%
##    8   35   52   72  896
```

U funkciji `quantile()` razredi se ne moraju uvijek dijeliti na kvartile (0 %, 25 %, 50 %, 75 %, 100 %) nego se mogu proizvoljno podesiti argumentom `probs`:

```
quantile(brojevi, probs=0.1)

## 10%
## 1.9

quantile(brojevi, probs= c(0.33, 0.66))

##  33%  66%
## 3.97 6.94
```

Takve vrijednosti se onda nazivaju **kvantili**.

Primjer: Pronađi kvantil koji dijeli Star Wars populaciju na najmlađih 10 %.

```
quantile(starwars$birth_year, 0.1, na.rm=T)

##  10%
## 21.2
```

Svi pojedinci mlađi od 21.2 godine spadaju u najmlađih 10 % populacije.

Interkvartilni raspon

Interkvartilni raspon IQR je razlika između trećeg kvartila (ispod kojeg leži 75 % vrijednosti varijable) i prvoga kvartila (ispod kojeg leži 25 % vrijednosti varijable).

$IQR = Q3 - Q1$

Unutar IQR-a nalazi se 50 % vrijednosti varijable. Unutar sustava R intekvartilni raspon podataka možemo dobiti korištenjem funkcije `IQR()`.

```
quantile(brojevi, probs = c(0.25, 0.75))
```

```
##  25%  75%
## 3.25 7.75
```

```
IQR(brojevi)
```

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

Primjer: Koliko iznosi interkvartilni raspon godina rođenja populacije iz Star Warsa?

```
IQR(starwars$birth_year, na.rm = TRUE)
```

```
## [1] 37
```

Kovarijanca

Kovarijanca je mjera zajedničke varijabilnosti dviju varijabli. Ako rast veličine jedne varijable uglavnom odgovara i rastu veličine druge varijable, onda se kaže da im je kovarijanca pozitivna. Ako rast veličine jedne varijabla uglavnom odgovara padu vrijednosti druge varijable, kovarijanca im je negativna. Jakost ili magnitudu kovarijance je teže za interpretirati jer ovisi o magnitudama varijabli, zato se češće radi s normiranom kovarijancom, tj. korelacijom.

Kovarijanca se računa sljedećom formulom:

$$cov(X,Y) = \frac{1}{n}\sum(x_i - \overline{X})(y_i - \overline{Y})$$

Iz ove formule vidi se da je kovarijanca varijable same sa sobom $cov(X,X)$ zapravo varijanca.

U sustavu R kovarijanca se računa funkcijom `cov()`. Ispitajmo kovarijancu za sve četiri numeričke varijable podatkovnog okvira `iris`.

```
cov(iris[1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

Dobili smo rezultat u obliku simetrične matrice 4 × 4, gdje se na dijagonali nalaze varijance varijabli.

Primjer: Izračunajmo kovarijancu visine i težine pojedinaca iz Star Warsa kojima su poznate obje te mjere.

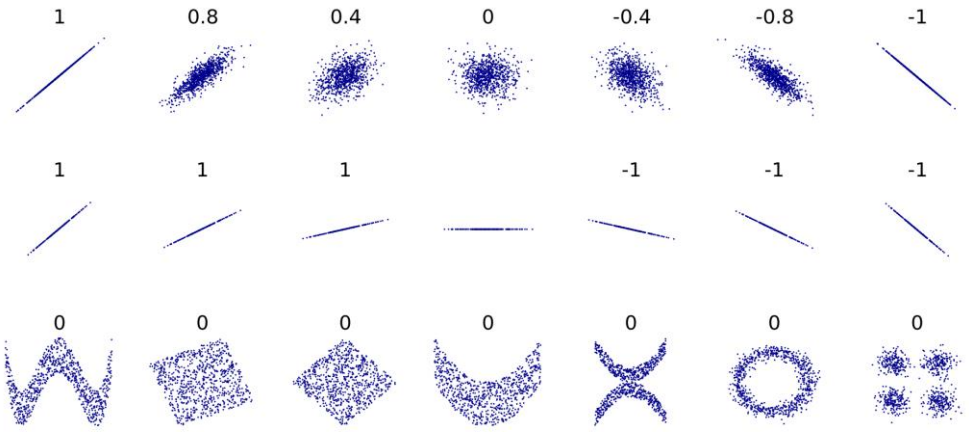
```
poznate_mjere <- subset(starwars, !is.na(height) & !is.na(mass))

cov(poznate_mjere$height, poznate_mjere$mass)
```

```
## [1] 806.0636
```

Iz rezultata bi se moglo naslutiti da je kovarijanca, odnosno zajednička varijabilnost pozitivna i jaka, odnosno da što je pojedinac iz Star Warsa viši, da je i teži. No, s obzirom na to da veličina kovarijance ovisi o veličini podataka, ovu tvrdnju je bolje provjeriti korelacijom.

Korelacija

Korelacija ili koeficijent korelacije je broj između $[-1,1]$ koji pokazuje jakost linearne veze između dviju varijabli. Što je taj broj bliži pozitivnoj ili negativnoj jedinici, linearna veza između varijabli je jača. Korelacija bliža nuli ukazuje na manjak linearne veze između varijabli.



Predznak korelacije isti je kao predznak linearne veze između varijabli. Na primjer, negativna korelacija interpretira se kao povezanost varijabli na način da kada vrijednosti jedne varijable rastu, vrijednosti druge varijable linearno opadaju. Korelacija 0 ne znači da varijable nisu povezane na neki način, nego da nisu povezane linearno.

Računa se po formuli: $cor(X,Y) = \frac{cov(X,Y)}{\sqrt{cov(X,X)*cov(Y,Y)}}$

U sustavu R računa se funkcijom `cor()`. Pogledajmo korelaciju između varijabli podatkovnog okvira `iris`.

```
cor(iris[1:4])
```

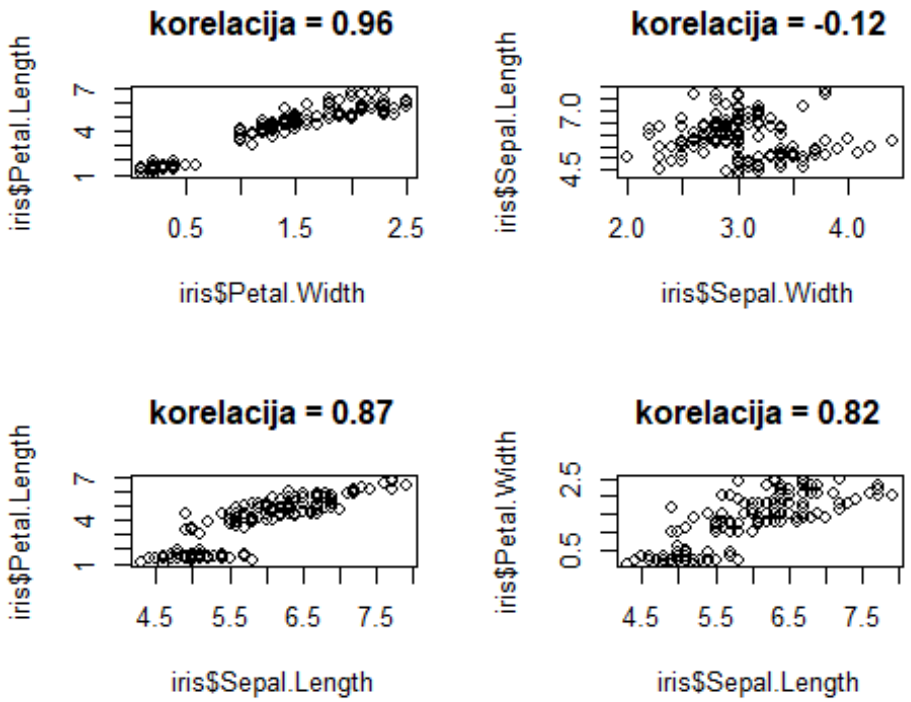
```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Korelacija varijabli iz podatkovnog okvira iris ukazuje na to da postoji jaka linearna veza između duljina i širina latica, no da je vrlo slaba linearna veza između duljina i širina

čašičnoga listića. Također, postoji snažna korelacija između duljine čašičnoga listića i duljine i širine latica. Što je čašični listić dulji, latice cvijeta iris su i dulje i šire.

Te tvrdnje mogu se provjeriti i grafički. Kod jake linearne korelacije, točke su grupirane oko (zamišljenog) pravca. Kod nelinearne korelacije, točke su grupirane oko neke druge krivulje ili su nepravilno raspršene. Što su točke bliže pravcu, korelacija je bliža pozitivnoj ili negativnoj jedinici. Što su točke raspršenije, korelacija je bliža nuli.

```
par(mfrow=c(2, 2))
plot(iris$Petal.Width, iris$Petal.Length, main = "korelacija = 0.96")
plot(iris$Sepal.Width, iris$Sepal.Length, main = "korelacija = -0.12")
plot(iris$Sepal.Length, iris$Petal.Length, main = "korelacija = 0.87")
plot(iris$Sepal.Length, iris$Petal.Width, main = "korelacija = 0.82")
```



Primjer: Postoji li linearna veza između visine i mase pojedinaca iz Star Warsa?

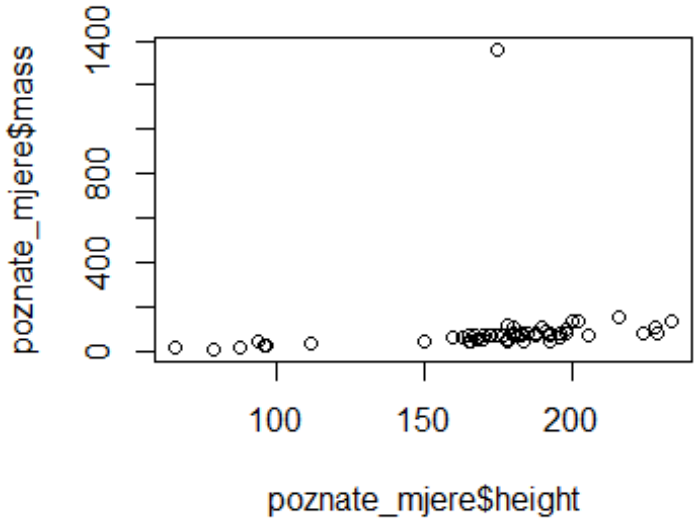
```
cor(poznate_mjere$height, poznate_mjere$mass)
```

```
## [1] 0.1338842
```

Korelacija od 0.13 pokazatelj je dosta male korelacije, i mogli bismo zaključiti da veza između mase i visine pojedinaca iz Star Warsa nije linearna.

Prikažimo tu vezu grafički funkcijom plot.

```
plot(poznate_mjere$height, poznate_mjere$mass)
```



Sada se vidi da postoji jedna ekstremna vrijednost koja očito utječe na rezultat, pa pokušajmo vidjeti kakva je korelacija bez te točke.

Prvo trebamo identificirati u kojem se retku nalazi pojedinac s najvećom masom:

```
which.max(poznate_mjere$mass)  # vraća indeks retka u kojem se postiže maksimum
```

```
## [1] 15
```

```
poznate_mjere[which.max(poznate_mjere$mass),]  # vraća cijeli redak u kojem se postiže maksimum
```

```
## # A tibble: 1 x 10
##   name  height  mass hair_color skin_color eye_color birth_year gender homeworld
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>  <chr>
## 1 Jabb~    175  1358 <NA>       green-tan~ orange           600 herma~ Nal Hutta
## # ... with 1 more variable: species <chr>
```

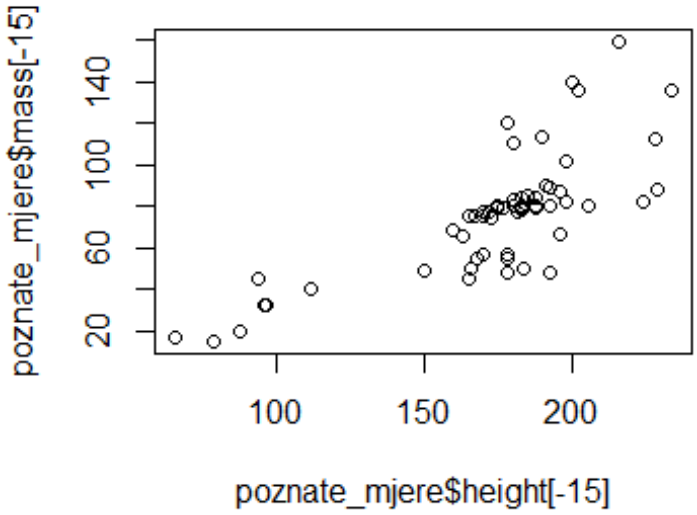
Zatim ponovimo upit za izračun korelacije, ali bez tog retka:

```
cor(poznate_mjere$height[-15], poznate_mjere$mass[-15])
```

```
## [1] 0.7612612
```

Ako izbacimo Jabbu koji ima ekstremno veliku masu s obzirom na visinu, ostatak populacije Star Warsa čije mjere su poznate, ima relativno jasnu, pozitivnu linearnu vezu između visine i mase, odnosno možemo zaključiti da u prosjeku vrijedi da što je pojedinac viši, da je i teži.

```
plot(poznate_mjere$height[-15], poznate_mjere$mass[-15])
```



Dodatne funkcije za uređenje skupa podataka

Funkcija	Opis	Primjer
`cut()`	stvaranje razreda	`cut(starwars$birth_year, breakes= c(100,200,300,1000))`
`rank()`	rangiranje elemenata vektora	`rank(starwars$height)`
`sort()`	sortiranje elemenata vektora	`sort(starwars$height)`
`order()`	vraća indekse u redoslijedu kojim se dobiva sortirani vektor	`order(starwars$height)`

Primjeri za svaku funkciju iz prethodne tablice slijede u nastavku.

- `cut()` - dijeli rang numeričke varijable prema danim granicama (`breaks`) u nove kategorije (`labels`), te svrstava svaku vrijednost varijable prema tim kategorijama. Rezultat se može pohraniti kao novi stupac/varijabla podatkovnog okvira.

```
starwars$starosti <- cut(starwars$birth_year, breaks= c(0,100,200,500,1000)
, labels=c("1.stoljeće", "2.stoljeće","3.-5.stoljeće", "6.-10.stoljeće" ))
```

Stvaranje dodatnih kategorijskih stupaca u kombinaciji s funkcijom `table()` omogućuje sažet prikaz podataka iz neke nove perspektive.

Primjer:

```
#ova tablica nam ne govori mnogo jer sadrži puno kategorija, a u svakoj su
maksimalno dvije jedinke
table(starwars$birth_year)
```

```
##
##    8   15   19   21   22   24   29   31 31.5   33   37   40   41 41.9   44   46
##    1    1    2    1    1    1    1    1    1    1    1    1    1    2    1    1
##   47   48   52   53   54   57   58   62   64   66   67   72   82   91   92  102
##    1    2    2    1    1    1    1    1    1    1    1    2    2    1    2    1
##  112  200  600  896
##    1    1    1    1
```

```
#ova tablica je preglednija
table(starwars$starosti)

##
##     1.stoljeće     2.stoljeće  3.-5.stoljeće 6.-10.stoljeće
##             38              3              0              2
```

Funkcije za sortiranje – `sort()`, `order()`, `rank()`

- Funkcija `sort()` ispermutira vrijednosti elemenata vektora tako da budu uzlazno poredani. Elementi se mogu poredati i silazno argumentom `decreasing = TRUE`.

```
sort(c("B", "C", "A", "E", "D"))

## [1] "A" "B" "C" "D" "E"
```

- Funkcija `order()` vraća indekse elemenata u onom redoslijedu u kojem bi bili poredani uzlazno (silazno ako navedemo argument `decreasing = TRUE`).

```
order(c("B", "C", "A", "E", "D"))

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

- Funkcija `rank()` vraća vektor koji sadrži rang svakoga pojedinog elementa u cjelokupnom vektoru.

```
rank(c("B", "C", "A", "E", "D"))

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

NAPOMENA vezana za `rank()`: U slučaju kada vektor koji rangiramo ima dvije iste vrijednosti, računa se aritmetička sredina njihovoga ranga i pridružuje objema vrijednostima, a prva iduća nakon njih uvećava se za 1. Taj tzv. “tie” može se promijeniti argumentom `ties.method`.

Primjer:

```
y <- c(1, 1, 2, 3)
rank(y)

## [1] 1.5 1.5 3.0 4.0

rank(y, ties.method="first")

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

Zadatak 45

45.1 Pokušajte predvidjeti rezultat sljedećih naredbi, te ih usporedite sa stvarnim rezultatima.

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

order(x)
rank(x)
x[order(x)]
x[rank(x)]
```

45.2 Sortirajte sve članove Star Wars skupine po starosti, od najstarijeg prema najmlađem.

9.2.3. Razdioba frekvencija varijable

Razdioba frekvencija varijable daje uređeni prikaz svih vrijednosti koje je varijabla poprimila u uzlazno sortiranom redoslijedu zajedno s brojem ponavljanja svake vrijednosti varijable. Može biti tablični i grafički, te ovisi o tipu varijable, odnosno o tome je li varijabla diskretna ili kontinuirana.

Diskretne varijable

Za diskretne ili kategorijske varijable razdioba frekvencija definirana je brojem (frekvencijom f) jedinki po svakoj kategoriji/diskretnoj vrijednosti varijable.

Suma svih frekvencija jednaka je veličini uzorka N.

Numerički prikaz razdiobe frekvencija daje tablica frekvencija koja sadrži: frekvencije (f), proporcije, postotke i kumulativne postotke po pojedinačnim vrijednostima varijable.

Relativna frekvencija ili proporcija svake vrijednosti varijable jednaka je $p = f/N$. Postotak se dobije kada se relativne frekvencije pomnože sa 100. Suma svih proporcija iznosi 1. Suma svih postotaka iznosi 100.

Primjer izrade tablice frekvencija za kategorijsku varijablu `eye_color` iz podatkovnog okvira `starwars`:

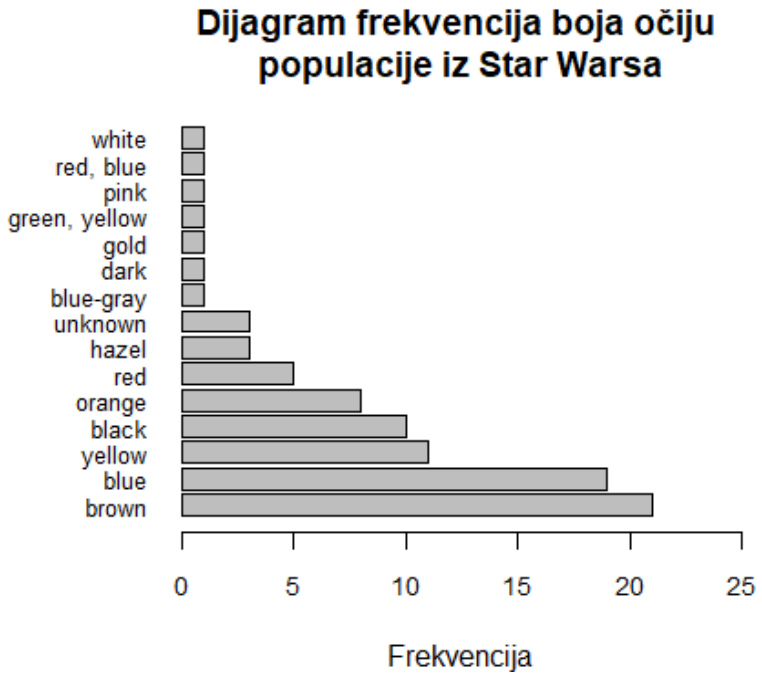
```
tablica_frekvencija <- as.data.frame(table(starwars$eye_color), stringsAsFa
ctors = FALSE)
names(tablica_frekvencija) <- c("Boja_ociju", "Frekvencije")

tablica_frekvencija$Proporcije <- tablica_frekvencija$Frekvencije/sum(tabli
ca_frekvencija$Frekvencije)

tablica_frekvencija$Postotak <- tablica_frekvencija$Proporcije*100

#sortiranje
tablica_frekvencija <-  tablica_frekvencija[order(tablica_frekvencija$Posto
tak, decreasing = TRUE),]

tablica_frekvencija$Kumulativni_postotak <- cumsum(tablica_frekvencija$Post
otak)
```

```
#ispis:
format(tablica_frekvencija, digits=3)
```

```
##       Boja_ociju Frekvencije Proporcije Postotak Kumulativni_postotak
## 4          brown          21     0.2414    24.14                 24.1
## 2           blue          19     0.2184    21.84                 46.0
## 15        yellow          11     0.1264    12.64                 58.6
## 1          black          10     0.1149    11.49                 70.1
## 9         orange           8     0.0920     9.20                 79.3
## 11           red           5     0.0575     5.75                 85.1
## 8          hazel           3     0.0345     3.45                 88.5
## 13       unknown           3     0.0345     3.45                 92.0
## 3      blue-gray           1     0.0115     1.15                 93.1
## 5           dark           1     0.0115     1.15                 94.3
## 6           gold           1     0.0115     1.15                 95.4
## 7  green, yellow           1     0.0115     1.15                 96.6
## 10          pink           1     0.0115     1.15                 97.7
## 12     red, blue           1     0.0115     1.15                 98.9
## 14         white           1     0.0115     1.15                100.0
```

Razdioba frekvencija kategorijskih varijabli grafički se može prikazati pomoću stupčastog dijagrama.

```
par(mar=c(4,7,4,2)) #definira redom, donju, lijevu, gornju i desnu marginu,
bez ovoga se odrežu nazivi na pojedinim vrijednostima osi y

barplot(tablica_frekvencija$Frekvencije, names.arg=tablica_frekvencija$Boja
_ociju,
        las=1, horiz = T,
        main = "Dijagram frekvencija boja očiju\npopulacije iz Star Warsa",
        xlab="Frekvencija", xlim=c(0,25), cex=0.8, cex.axis = 0.9)
```



Dijagram frekvencija može se prikazati i drugim načinima, npr. dijagramom “stabljika-list” (engl. *stem and leaf plot*) i strukturnim krugom (engl. *pie chart*).

Dijagram „stabljika - list” ili S-L dijagram (engl. *stem leaf*) koristi se za prikaz numeričkih varijabli u grafičkom obliku na način da se svaka vrijednost rastavi na stabljiku (prva znamenka ili prvih nekoliko znamenki) i na list (zadnja znamenki ili zadnjih nekoliko znamenki). Takav prikaz pomaže u vizualizaciji razdiobe numeričke varijable.

Na primjer, niz brojeva 28, 19, 22, 30, 25, 31, 18, 27, 30, 24 bismo rastavili tako da prva znamenka ide u stabljiku, a zadnja u list:

Stabljika | list
1|98
2|82574
3|010

Primjer dijagrama stabljika-list nad podacima o visini pojedinaca iz podatkovnog okvira `starwars`:

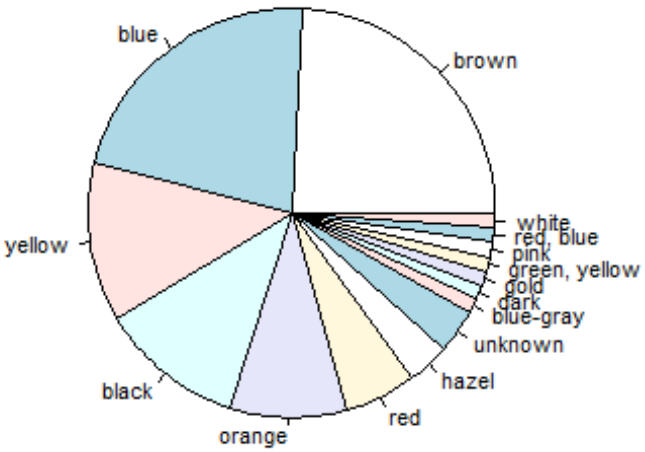
```
stem(starwars$height)
```

```
##   The decimal point is 1 digit(s) to the right of the |
##
##    6 | 69
##    8 | 84667
##   10 | 2
##   12 | 27
##   14 | 007
##   16 | 033555677800001235557888 8
##   18 | 000002333333345888880111333666 88
##   20 | 026636
##   22 | 4894
##   24 |
##   26 | 4
```

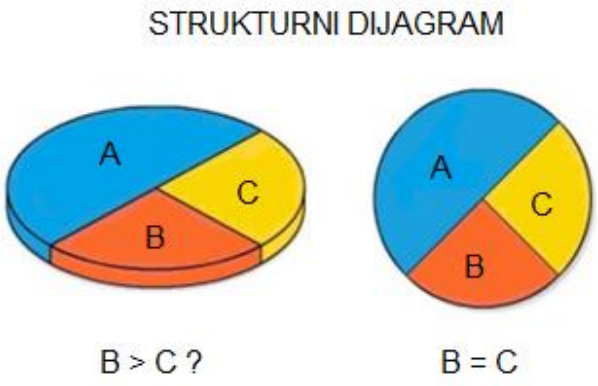
Možemo zaključiti da se visine pojedinaca iz Star Warsa kreću u rasponu od 66 do 264 cm, a naviše je onih koji su visoki između 180 i 200cm.

Strukturni krug (engl. *pie chart*) je kružna grafika za prikaz frekvencija kategorijske varijable. Krug se dijeli na kružne isječke čija veličina predstavlja udjel određene kategorije u populaciji.

```
pie(tablica_frekvencija$Frekvencije, labels = tablica_frekvencija$Boja_ocij
u, cex=0.7, radius = 1.05)
```



Uporaba strukturnih dijagrama se generalno ne preporučuje jer, osim u rijetkim slučajevima, ne pojašnjavaju dodatno informacije (što je primarna svrha grafičkoga prikaza) iz razloga što ljudsko oko i mozak ne percipiraju najjasnije razlike i omjere izražene u obliku kružnih isječaka. Trodimenzionalni strukturni krugovi koji su prikazani pod nekim nagibom još su manje vjerodostojni.



Čak i sama dokumentacija za `pie()` navodi u poglavlju `Notes` da se ne preporučuje korištenje ovakvih dijagrama za prikaz informacija.

Neprekidne varijable

Za računanje tablica frekvencije neprekidne ili kontinuirane varijable, prvo je potrebno odrediti intervale u koje ćemo razvrstati sve vrijednosti. One se odrede tako da se raspon vrijednosti neprekidne varijable, koji je jednak razlici između najveće i najmanje vrijednosti, podijeli na proizvoljan broj intervala koji se dodiruju (ali ne preklapaju).

Numerički prikaz razdiobe frekvencija daje tablica frekvencija koja sadrži: frekvencije, proporcije, postotke i kumulativne postotke po intervalima vrijednosti varijable.

Frekvencija (f) je broj pojavljivanja jedinki u pojedinom intervalu. Relativna frekvencija ili proporcija dobije se formulom $p = f/N$.
Postotak se dobije kada se relativna frekvencija pomnoži sa 100. Suma svih proporcija iznosi 1. Suma svih postotaka iznosi 100.

U sustavu R, intervali se mogu izraditi funkijom `cut()`.

Primjer neprekidne varijable su visine pojedinaca (`heights`) iz Star Warsa.

```
#raspon varijable:
range(starwars$height, na.rm=T)
```

```
## [1]  66 264
```

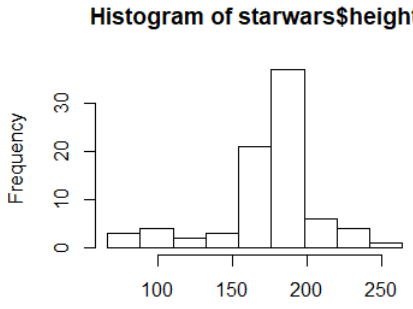
```
#raspon nam treba za argument breaks u funkciji cut(), a kasnije i za breaks u funkciji hist().
starwars$intervali_visina <- cut(starwars$height, breaks = seq(66,264, length.out = 10))

tablica_frekvencija <- as.data.frame(table(starwars$intervali_visina), stringsAsFactors = FALSE)
names(tablica_frekvencija) <- c("Razred_visina", "Frekvencije")

tablica_frekvencija$Proporcije <- tablica_frekvencija$Frekvencije/sum(tablica_frekvencija$Frekvencije)

tablica_frekvencija$Postotak <- tablica_frekvencija$Proporcije*100

tablica_frekvencija$Kumulativni_postotak <- cumsum(tablica_frekvencija$Postotak)

tablica_frekvencija
```

```
##   Razred_visina Frekvencije Proporcije Postotak Kumulativni_postotak
## 1       (66,88]           2     0.0250     2.50                 2.50
## 2      (88,110]           4     0.0500     5.00                 7.50
## 3     (110,132]           2     0.0250     2.50                10.00
## 4     (132,154]           3     0.0375     3.75                13.75
## 5     (154,176]          21     0.2625    26.25                40.00
## 6     (176,198]          37     0.4625    46.25                86.25
## 7     (198,220]           6     0.0750     7.50                93.75
## 8     (220,242]           4     0.0500     5.00                98.75
## 9     (242,264]           1     0.0125     1.25               100.00
```

Grafički prikaz frekvencija neprekidne varijable radi se funkcijom `hist()`, pri čemu se argumentom `breaks` podešava broj intervala.

Primjer histograma nad visinama pojedinaca iz Star Warsa:

```
hist(starwars$height, breaks=seq(66,264, length.out = 10))
```



9.2.4. Sažetak

U sljedećoj tablici navedene su najčešće korištene funkcije deskriptivne statistike:

Funkcija	Opis	Primjer	Rezultat
`min()`	minimum skupa x	`min(starwars$height)`	66
`max()`	maksimum skupa x	`max(starwars$height)`	264
`which.min()`	indeks retka u kojem se postiže minimum	`which.min(starwars$height)`	19
`which.max()`	indeks retka u kojem se postiže maksimum	`which.max(starwars$height)`	54
`sum(x)`	zbroj svih vrijednosti u x	`sum(starwars$mass)`	5741.4
`diff()`	uzastopne razlike	`diff(starwars$height[1:5])`	-5, -71, 106, -52
`mean()`	srednja vrijednost	`mean(starwars$height)`	174.3580247
`median()`	medijan	`median(starwars$height)`	180
`quantile()`	kvantili	`quantile(starwars$height, na.rm=T)`	66, 167, 180, 191, 264
`sd()`	standardna devijacija	`sd(starwars$height)`	34.7704288
`var()`	varijanca	`var(starwars$height)`	1208.982716
`range()`	rang	`range(starwars$height)`	66, 264
`cov()`	kovarijanca dvaju vektora/matrica	`cov(starwars$height, starwars$mass, use="complete.obs")`	806.0635885
`cor()`	korelacija dvaju vektora/matrica	`cor(starwars$height, starwars$mass, use="complete.obs")`	0.1338842
`cumsum()`	kumulativna suma	`cumsum(1:5)`	1, 3, 6, 10, 15
`cumprod()`	kumulativni produkt (faktorijeli)	`cumprod(1:5)`	1, 2, 6, 24, 120

Dodatne funkcije za uređenje skupa podataka:

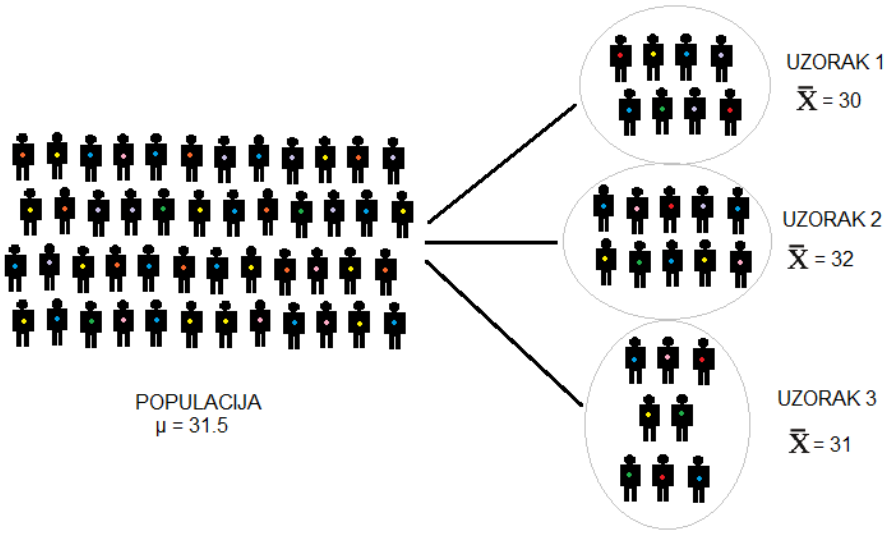
Funkcija	Opis	Primjer
`cut()`	stvaranje razreda	`cut(starwars$birth_year, breakes= c(100,200,300,1000))`
`rank()`	rangiranje elemenata vektora	`rank(starwars$height)`
`sort()`	sortiranje elemenata vektora	`sort(starwars$height)`
`order()`	vraća indekse u redoslijedu kojim se dobiva sortirani vektor	`order(starwars$height)`
`table()`	kontingencijska tablica	`table(starwars$eye_color)`

Neke od grafičkih funkcija često korištenih u deskriptivnoj statistici:

Funkcija	Opis	Primjer
`plot()`	točkasti ili linijski grafikon za prikaz jedne ili dviju varijabli	`plot(iris$Petal.Width, iris$Petal.Length)`
`boxplot()`	dijagram pravokutnika - grafički prikaz statističkih sažetaka	`boxplot(starwars$height)`
`densityplot()`	funkcija gustoće vjerojatnosti iz paketa lattice	`densityplot(poznate_godine$birth_year)`
`barplot()`	stupčasti dijagram - za prikaz razdiobe diskretne varijable	`barplot(Frekvencije ~ Boja_ociju,tablica_frekvencija`
`hist()`	histogram - za prikaz razdiobe kontinuirane varijable	`hist(starwars$height)`

9.3. Inferencijalna statistika

Inferencijalna ili induktivna statistika bavi se izvođenjem zaključaka o populaciji na temelju svojstava uzorka. Kakav će uzorak biti je slučajnost, te se njegova svojstva opisuju slučajnim varijablama.



9.3.1. Slučajna varijabla

Slučajna varijabla je varijabla čije vrijednosti variraju na slučajan način. Može poprimiti skup mogućih različitih vrijednosti, svaku s pridruženom vjerojatnošću, i to određuje njenu razdiobu.

Slučajna varijabla može biti:

- **Diskretna** – poprima konačan broj vrijednosti ili prebrojivo mnogo njih.
  Primjeri: živ/mrtav, bacanje kocke, ocjena u školi, godina rođenja.
- **Neprekidna** – poprima bilo koju vrijednost iz nekog intervala realnih vrijednosti.
  Primjeri: visina, težina, brzina, realni brojevi od 1 do 10.

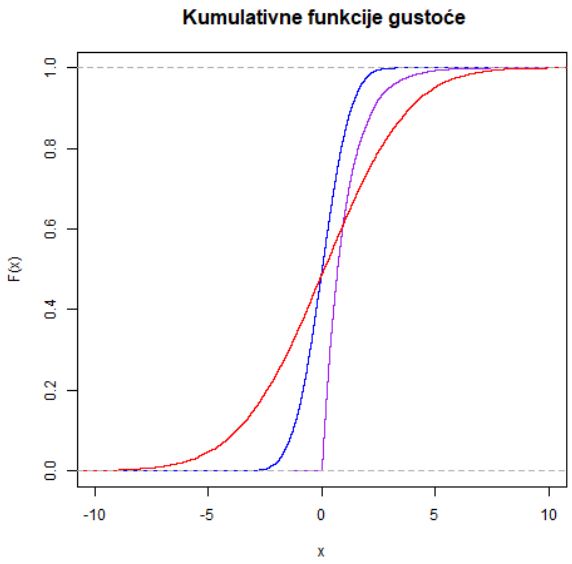
Uz svaku slučajnu varijablu vezane su dvije funkcije: **funkcija razdiobe slučajne varijable** i **funkcija gustoće vjerojatnosti**.

9.3.2. Funkcija razdiobe slučajne varijable

Funkcija razdiobe slučajne varijable X (engl. *cumulative distribution function*, CDF) daje vjerojatnost da slučajna varijabla X poprimi bilo koju vrijednost manju ili jednaku od x. Označava se s $F(x)$:

$$\mathrm{F(x)} = P(X \leq x), \quad -\infty < x < \infty$$

Grafički prikaz takve funkcije uvijek počinje slijeva od vrijednosti 0 na osi y, i završava desno s vrijednosti 1 na osi y.



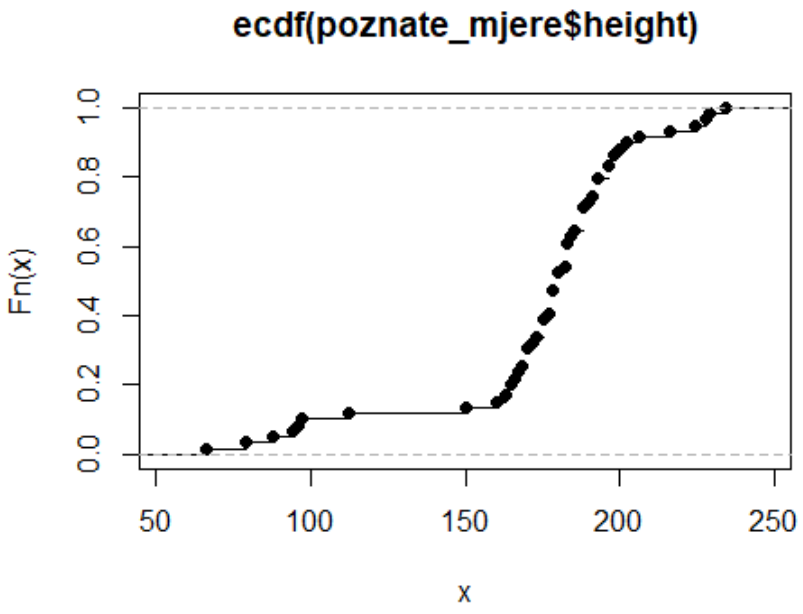
U R-u se koristi funkcija `ecdf()` (*empirical cumulative distribution function*) da bi se dobila funkcija razdiobe neke varijable.

Primjer: Iz dijela populacije Star Wars kojoj su poznate visine, nasumično izabiremo jedan lik. Kolika je vjerojatnost da će biti niži od 1.72 m?

```
CDF_height <- ecdf(poznate_mjere$height)  # izrada funkcije CDF_height

CDF_height(172)   # funkciji CDF_height proslijedimo argument 172

## [1] 0.3220339
```

Grafički prikaz takve funkcije dobije se kada se funkciji `plot()` proslijedi funkcija `ecdf()`. Što je više podataka, graf je glađi.

```
plot(ecdf(poznate_mjere$height))
```



9.3.3. Funkcija gustoće vjerojatnosti

Funkcija gustoće vjerojatnosti diskretne slučajne varijable

Neka diskretna slučajna varijabla X može poprimiti vrijednosti iz skupa $A = (x_1, x_2, \ldots, x_n)$. Tada je funkcija gustoće vjerojatnosti diskretne slučajne varijable X dana izrazom

$$f(x_i) = P(X = x_i), \quad \forall x_i \in A$$
$$f(x) = 0, \quad x \notin A$$

Na primjer, u eksperimentu bacanja simetrične kocke, vjerojatnost da padne bilo koji broj od 1 do 6 je 1/6. Kažemo da je funkcija gustoće vjerojatnosti takvoga događaja:

$$f(x) = 1/6, \quad \text{za } x \in \{1,2,3,4,5,6\}$$
$$f(x) = 0, \quad \text{inače.}$$

Funkcija gustoće vjerojatnosti neprekidne slučajne varijable

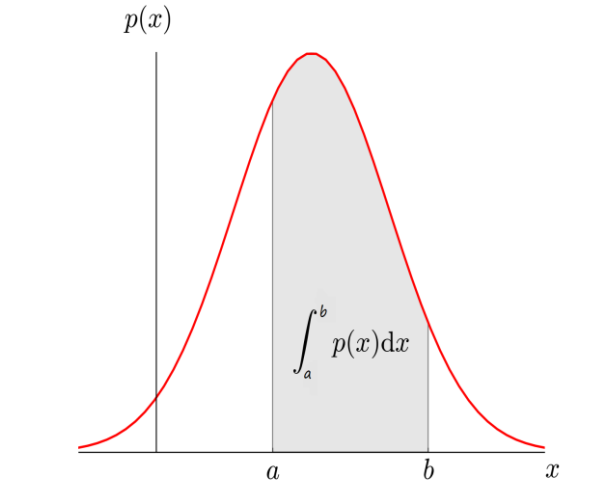
Neka je X neprekidna slučajna varijabla koja može poprimiti vrijednosti na intervalu realnih brojeva $[a, b]$. S obzirom na to da X može poprimiti neprebrojivo mnogo vrijednosti, vjerojatnost da poprimi neku određenu vrijednost je nula, za svaku vrijednost.

Radi toga, kod neprekidnih slučajnih varijabli zapravo se gledaju intervali, odnosno pita se kolika je vjerojatnost da slučajna varijabla X poprimi vrijednost između $[x, x + \epsilon]$, gdje je $\epsilon > 0$, makar taj interval bio infinitezimalnoga raspona.

Vjerojatnost da neprekidna slučajna varijabla X poprimi neku vrijednosti između (k, l) računa se po formuli:

$$P(k \leq X \leq l) = \int_k^l f(x)dx$$

Ta vjerojatnost jednaka je površini ispod funkcije gustoće vjerojatnosti na intervalu (k, l).



Iz ovog vidimo da vrijedi i sljedeća jednakost:

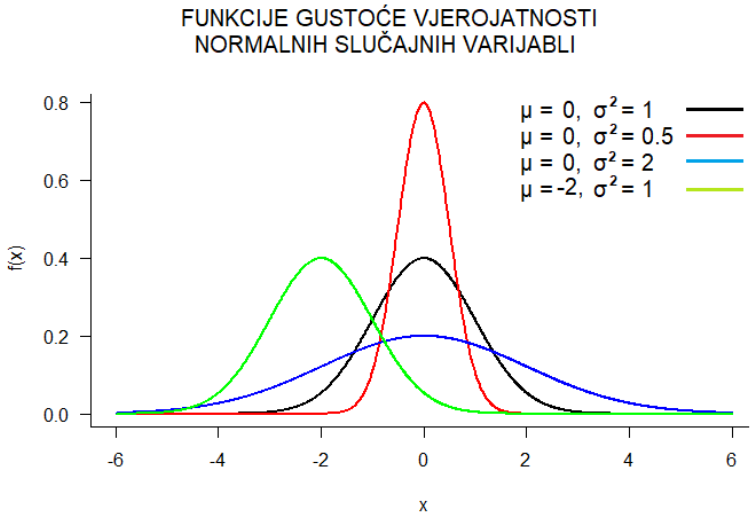
$$\int_k^l f(x)dx = F(l) - F(k)$$

9.4. Normalna ili Gaussova razdioba

U klasičnoj statistici ovo je najvažnija od svih teorijskih razdioba. Funkcija gustoće vjerojatnosti slučajne varijable opisana je dvama parametrima, srednjom vrijednosti μ i varijancom σ^2. Kažemo da je slučajna varijabla X normalna i pišemo $X \sim N(\mu, \sigma^2)$, ako joj je funkcija gustoće vjerojatnosti jednaka:

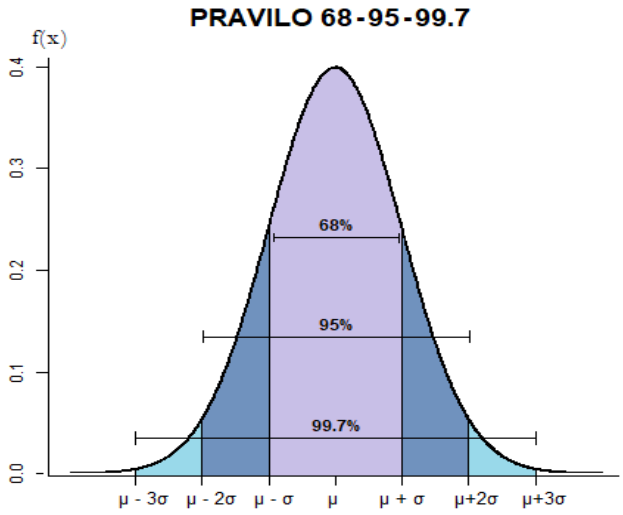
$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad -\infty < x < \infty$$

razdioba ima karakterističan zvonolik oblik oko srednje vrijednosti μ, dok varijanca σ^2 određuje širinu zvonca:



Svojstva normalne razdiobe:

- srednja vrijednost, medijan i mod su jednaki
- standardna devijacija normalne razdiobe je horizontalna udaljenost od sredine do jedne od točaka pregiba
- pravilo 68-95-99.7 - u svakoj normalnoj razdiobi približno 68 % vrijednosti nalazi se u rasponu od ± jedne standardne devijacije od srednje vrijednosti. Približno 95 % vrijednosti nalazi se u rasponu od ±2 standardne devijacije od srednje vrijednosti, te otprilike 99.7 % vrijednosti nalazi se u rasponu od ±3 standardne devijacije od srednje vrijednosti.

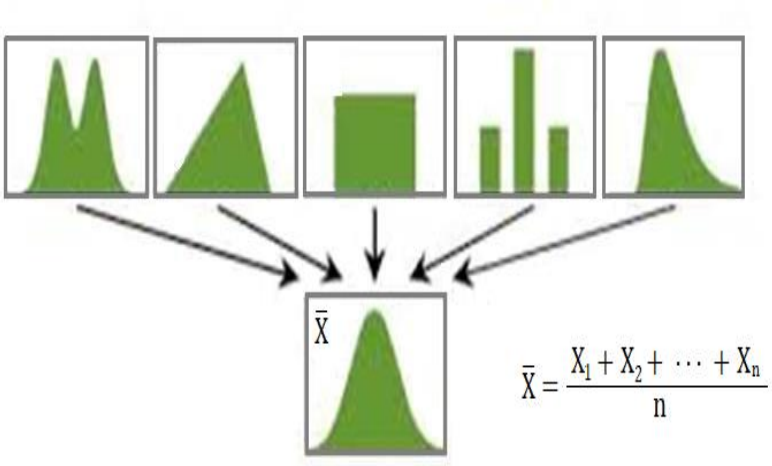


Normalna razdioba koja ima srednju vrijednost 0 i varijancu 1, $X \sim N(0,1)$ zove se standardna normalna razdioba.

Varijable koje opisuju visinu osobe, težinu muškaraca, žena ili novorođenčadi, težine pojedinih komada nekoga voća i povrća, broj cipela muškaraca ili žena, sumu bacanja dviju kocki, krvni tlak, vrijeme putovanja po određenoj ruti, životni vijek baterije, dnevni povrat pojedine dionice/indexa, neki su od primjera normalne razdiobe.

Za dovoljno velik uzorak, mnoge ostale razdiobe teže ka normalnoj (binomna, hipergeometrijska, Poissonova, studentova…).

Varijable koje opisuju sumu nekih drugih nezavisnih i jednako distribuiranih varijabli bez obzira na to koje su razdiobe te varijable, normalno su distribuirane, kao na primjer slučajna varijabla koja je jednaka srednjoj vrijednosti uzorka iz populacije. To svojstvo proizlazi iz tzv. centralnoga graničnog teorema (engl. *central limit theorem*).



9.5. Vjerojatnosne razdiobe u sustavu R

S obzirom na raznolikost i kompleksnost vjerojatnosnih razdioba, olakšavajuća okolnost je da sustav R posjeduje bogatu bazu ugrađenih funkcija vezanih za mnoge funkcije gustoće vjerojatnosti i funkcije razdiobe. One su dio paketa `stats` koji je također osnovni dio sustava R. Naredba `?distributions` ispisat će popis svih razdioba ugrađenih u sustav R, a u datoteci Dodatak u radnom direktoriju nalaze se opisi nekih od tih razdioba.

```
?distributions
```

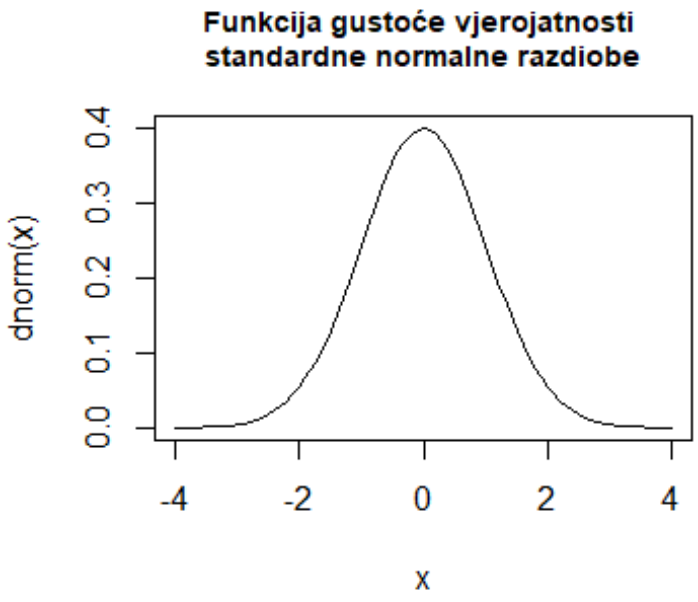
Za svaku od uključenih vjerojatnosnih razdioba postoje četiri osnovne funkcije koje čine sljedeće:

- **funkcija d** računa funkciju gustoće vjerojatnosti neke razdiobe ($f(x) = P(X = x)$). Primjerice, `dnorm`, `dt`, `dbinom`, `dunif`, `dpois` redom su funkcije gustoće vjerojatnosti za normalnu, studentovu, binomnu, uniformnu i Poissonovu razdiobu.

```
dnorm(0) #funkcija gustoće vjerojatnosti standardne normalne razdiobe u točki 0

## [1] 0.3989423

dnorm(1) #funkcija gustoće vjerojatnosti standardne normalne razdiobe u točki 1

## [1] 0.2419707
```

Grafički prikaz `dnorm()` funkcije može se napraviti pomoću funkcije `plot()`.

```
x <- seq(-4,4,0.1)
plot(x, dnorm(x), main="Funkcija gustoće vjerojatnosti \nstandardne normalne razdiobe", cex.main=0.9, type="l")
```



- **funkcija p** računa kumulativnu vjerojatnost do vrijednosti x ($F(x) = P[X \leq x]$). Primjerice, `pnorm`, `pt`, `pbeta`, `pbinom`, `pcauchy`, `punif`, `ppois`...

```
pnorm(0)
```

```
## [1] 0.5
```

```
pnorm(3)
```

```
## [1] 0.9986501
```

Možemo je shvatiti kao površinu ispod grafa d-funkcije na području od $< -inf, x]$. U gornjem primjeru rezultati su se mogli predvidjeti imajući na umu da je standardna normalna razdioba simetrična oko točke 0 i da je točka 3 udaljena tri standardne devijacije od srednje vrijednosti (pravilo 68-95-99.7).

Argumentom lower.tail određuje se s koje strane točke x se zbrajaju vrijednosti $f(x_i)$, preciznije `lower.tail = TRUE` računa $P(X \leq x)$, dok `lower.tail = FALSE` računa `P(X > x)`.

```
pnorm(1)
```

```
## [1] 0.8413447
```

```
1 - pnorm(1, lower.tail = FALSE)
```

```
## [1] 0.8413447
```

Primjer: Provjerimo da pravilo 68 - 95 - 99.7 zaista vrijedi.

```
pnorm(1) - pnorm(-1)
```

```
## [1] 0.6826895
```

```
pnorm(2) - pnorm(-2)
```

```
## [1] 0.9544997
```

```
pnorm(3) - pnorm(-3)
```

```
## [1] 0.9973002
```

- **funkcija `q(P)`** vraća kvantil, odnosno točku na osi x u kojoj je postignuta kumulativna vjerojatnost od P. Drugim riječima, to je inverz funkcije `p`. Primjeri: `qnorm`, `qt`, `qbeta`, `qbinom`...

```
qnorm(0.5)
```

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

```
qnorm(0.9986501)
```

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

Ova funkcija također prima argument `lower.tail` pomoću koje se određuje s koje strane je početak mjerenja.

```
qnorm(0.9)
```

```
## [1] 1.281552
```

```
qnorm(0.1, lower.tail = FALSE)
```

```
## [1] 1.281552
```

- **funkcija `r(n)`** generira slučajnu varijablu navedene razdiobe u obliku vektora duljine n. Primjeri: `rnorm`, `rt`, `rbeta`, `rbinom`…

```
rnorm(5)
```

```
## [1]  1.2352448  0.2390965  0.5126357 -0.2844614  3.1475021
```

```
rnorm(4, mean = 10, sd = 2)
```

```
## [1] 12.939366 12.618181 11.496131  6.160685
```

Zadatak 46
Izradite vektor `r` koji se sastoji od pet slučajnih brojeva iz standardne uniformne razdiobe (između 0 i 1).

```
set.seed(1)
```

Primjer:
Visina žena iz neke populacije normalno je distribuirana sa sredinom 160 cm i standardnom devijacijom 8 cm. Nađite vjerojatnost da je slučajno izabrana žena iz te populacije viša od 172 cm.

```
1-pnorm(172, 160, 8)
```

```
## [1] 0.0668072
```

Koliki je postotak populacije manji od 150 cm?

```
pnorm(150, 160, 8)
```

```
## [1] 0.1056498
```

Odgovor: 10.56 % ove populacije niže je od 150 cm.

Koliko žena treba biti minimalno visoka da bi spadala u 5 % najviše populacije?

```
qnorm(0.95, 160, 8)
```

```
## [1] 173.1588
```

Zadatak 47
47.1 Trajanje baterije na mobitelu je normalno distribuirano sa srednjom vrijednosti od 30 sati i standardnom devijacijom od pet sati. Izračunajte vjerojatnost da baterija izdrži 40 sati.

47.2 Nađite broj sati Q za koji vrijedi da je šansa da baterija izdrži manje od Q sati jednaka 10 %. Hint: $P(trajanje\ \ baterije < Q) = 0.1$.

Zadatak 48
Vrijeme potrebno za sastavljanje jednoga računala je normalno distribuirano sa srednjom vrijednosti 50 minuta i standardnom devijacijom od 10 minuta. Kolika je vjerojatnost da računalo bude sastavljeno između 45 i 60 minuta?

9.6. Testiranje hipoteza

Statistička hipoteza je bilo kakva pretpostavka o razdiobi nekog statističkog obilježja.

Na primjer:

- X ima normalnu razdiobu sa sredinom $\mu = 50$,
- $\lambda = 10$,
- $\sigma \leq 5$,
- $X \sim N(10{,}2)$

Osnovna hipoteza zove se **nul-hipoteza** (H_0), i uz nju se postavlja njoj **alternativna hipoteza** (H_1). Na primjer:

$$H_0\colon \mu = 50$$
$$H_1\colon \mu < 50$$

Testiranje hipoteze je korištenje statistike da bi se na osnovi realizacije slučajnog uzorka odlučilo odbacujeli se ili ne odbacuje H_0. (Ne tvrdi se da je jedna hipoteza točna, već da se s određenom vjerojatnošću nul-hipoteza odbacuje ili ne odbacuje).

Hipoteza je jednostavna ako jednoznačno određuje razdiobu testne statistike (npr. $\mu = 10$), u suprotnom je složena (npr. $\mu \leq 10$, $\mu > 10\ldots$).

U pravilu, za nul-hipotezu uzimaju se jednostavne hipoteze.

S obzirom na to da sve odluke bazirane na uzorcima iz populacije nisu 100 % pouzdane, ni zaključak statističkog testa nije 100 % pouzdan te se može dogoditi da zaključak testa bude pogrešan. Ako se dogodi da se H_0 odbaci kada je ona istinita, dogodila se **pogreška prve vrste**. Ako se dogodi da se H_0 ne odbaci kada ona nije istinita, dogodila se **pogreška druge vrste**.

Pogreške se mogu kontrolirati (jedna ili druga), ali smanjenjem pogreške jedne vrste, povećava se pogreška druge vrste.

Vjerojatnost pogreške prve vrste označava se s α i još se naziva **razina značajnosti**, a postavlja je sam/a analitičar/ka. Obično je na razini $\alpha = 0.05$.

Smanjenjem α smanjujemo vjerojatnost pogreške prve vrste, a povećanjem α smanjujemo vjerojatnost pogreške druge vrste (koja se označava s ß).

Koja pogreška je bitnija ovisi o tome kako se postavila nul-hipoteza. Dobra praksa je postaviti nul-hipotezu za koju se može umanjiti pogreška prve vrste jer se ona kontrolira direktno, dok se ß mora izračunati.

Primjer 1
U jednom laboratoriju testira se dijagnosticiranje neke bolesti. Moguće su dvije vrste pogreške:

- Pacijent ima bolest, a rezultati testa pokazuju da je nema -> pacijent se dalje ne testira, a bolestan je, što može biti pogubno.
- Pacijent nema bolest, a rezultati testa pokazuju da je ima -> pacijent ide dalje na liječenje i nova testiranja, što može dovesti do otkrivanja pogreške u testiranju i spasiti pacijenta.

U ovom slučaju prva opcija je opasnija, te nju želimo minimizirati. Ako postavimo nul-hipotezu na:

H_0: pacijent ima bolest
H_1: pacijent nema bolest

Pogreška prve vrste je da odbacimo H_0 kada je ona istinita, odnosno da kažemo da pacijent nema bolest kada je zapravo ima. Razina značajnosti α, odnosno vjerojatnost pogreške prve vrste ovdje može biti 0.05, ali je ima smisla i smanjiti.

Primjer 2
Tvornica koja proizvodi alarme za požar želi testirati njihovu ispravnost. Kako je najbolje postaviti nul-hipotezu s obzirom na pogreške prve i druge vrste.

Rješenje: U ovom slučaju je bolje da se alarm upali kada nema požara, nego da se alarm ne upali kad ima požara.

H_0: alarm nije ispravan
H_1: alarm je ispravan

Pogreška prve vrste bila bi kada bismo odbacili H_0 dok je ona istinita, dakle ako bismo rekli da je alarm ispravan kada zapravo nije. Pogreška druge vrste je da kažemo da alarm nije ispravan kada zapravo jest ispravan.

9.6.1. Pristup testiranju hipoteze

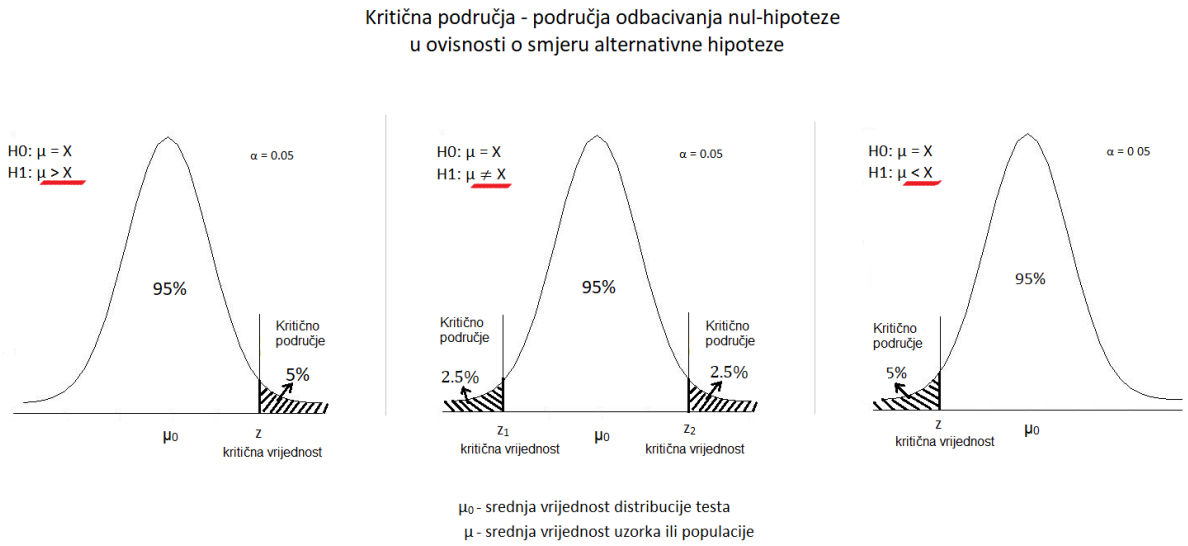
Sada kada znamo kako se postavljaju hipoteze, idemo vidjeti na kojem principu radi njihovo testiranje.

1. način: metoda kritičnoga područja

   Kod testiranja hipoteze potrebna su dva broja - testna statistika i kritična vrijednost.

   - **Testna statistika** se izračuna na temelju danog uzorka, pretpostavke o razdiobi i razini značajnosti α koju analitičar sam postavi.
   - **Kritične vrijednosti** (jedna ili dvije) dobiju se iz statističke tablice određene razdiobe (R ih također posjeduje) i ovisi o razini značajnosti α. One određuju koji interval/intervali realnih brojeva spadaju u kritično područje, odnosno područje u kojem se odbacuje nul-hipoteza.

   Razlikujemo tri slučaja, ovisno o smjeru alternativne hipoteze ($>$, $<$, $\neq$):



Ako testna statistika spada u kritično područje, onda nemamo dovoljno dokaza da prihvatimo nul-hipotezu, odnosno onda odbacujemo nul-hipotezu.

2.način: metoda p-vrijednosti

p-vrijednost je vjerojatnost da se dobila baš ta testna statistika pod uvjetom da je nul-hipoteza točna. Ako je p-vrijednost premala, zaključujemo da su šanse dobivanja takve testne statistike premale da bi H_0 zaista bila točna, te tada odbacujemo nul-hipotezu. No, što je premala vrijednost za p-vrijednost? Odgovor je: sve manje od α.

p-vrijednost se uspoređuje s razinom značajnosti α. Ako je p-vrijednost manja od α, odbacujemo nul-hipotezu, a ako je veća, nemamo dovoljno dokaza da odbacimo nul-hipotezu na toj razini značajnosti.

Za računanje p-vrijednosti koristimo odgovarajuću funkciju `p()`, no opet razlikujemo tri slučaja, ovisno o smjeru alternativne hipoteze.

Pretpostavimo da je razdioba testne statistike normalna, tako da koristimo funkciju pnorm().

1. slučaj

H_0: $\mu = X$
H_1: $\mu > X$

```
=> p = pnorm(testna_statistika, lower.tail = FALSE)
```

2. slučaj

H_0: $\mu = X$
H_1: $\mu < X$

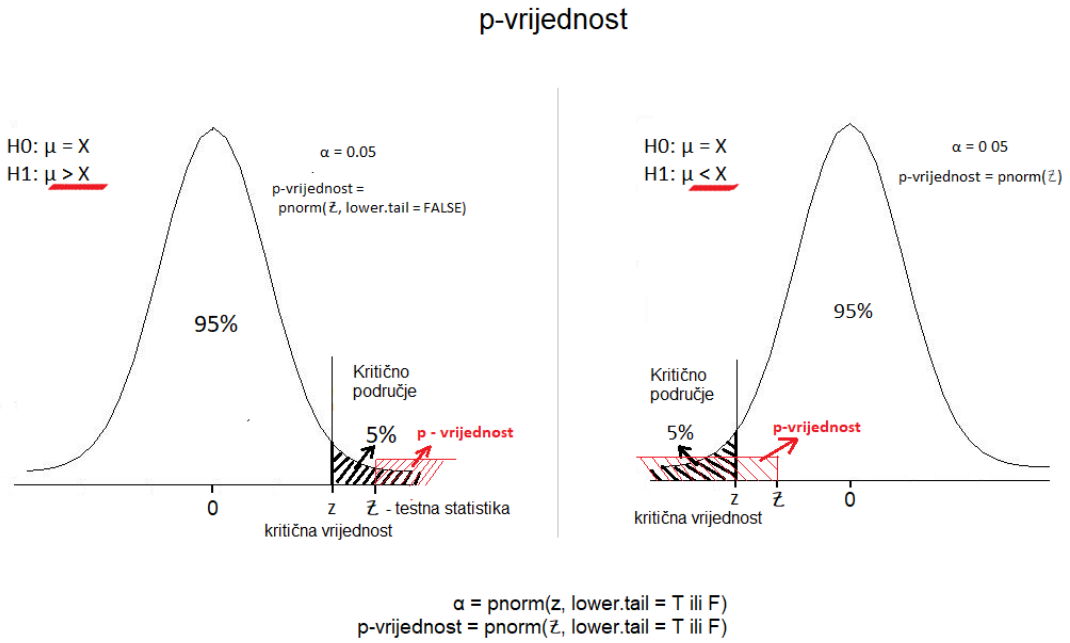
```
=> p = pnorm(testna_statistika)
```

3. slučaj

H_0: $\mu = X$
H_1: $\mu \neq X$

```
=> p = 2*min(pnorm(testna_statistika), pnorm(testna_statistika,
lower.tail = FALSE))
```

U slučaju dvostranoga testa, p-vrijednost je manja od brojeva iz prvih dvaju slučajeva pomnožena brojem dva.



Kao što se vidi, za odluku o odbacivanju ili neodbacivanju nul-hippteze, svejedno je uspoređuje li se testna statistika s kritičnom vrijednosti, ili p-vrijednost s razinom značajnosti α.

Da bi se izračunale testna statistika, kritične vrijednosti i p-vrijednost, potrebno je znati distribuciju testne statistike.

9.6.2. Z-test

Z-test koristimo kada pretpostavljamo da je razdioba testne statistike normalna ili kada može biti aproksimirana normalnoj te kada je poznata standardna devijacija populacije.

Često se koristi u situacijama kada treba odrediti pripada li neki uzorak danoj populaciji.

Na primjer:

H_0: srednja vrijednost uzorka jednaka je srednjoj vrijednosti populacije
H_1: srednje vrijednosti se razlikuju.

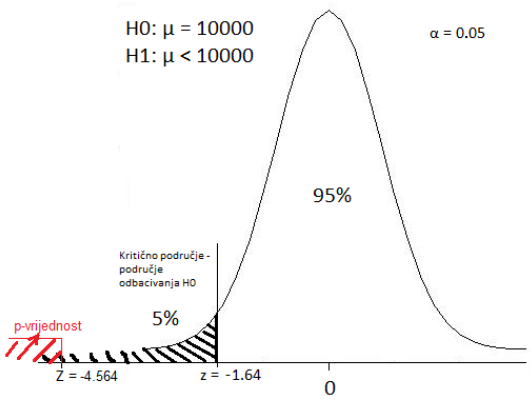
Formula za test statistiku:

$$Z = \frac{\overline{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

Primjer:
Proizvođač na deklaraciji žarulje tvrdi da je vijek trajanja žarulja minimalno 10.000 sati rada. Na uzorku od 30 žarulja pronađeno je da su prosječno radile 9.900 sati. Pretpostavimo populacijsku standardnu devijaciju od 120 sati. Možemo li odbaciti tvrdnju proizvođača na razini značajnosti 0.05?

$H_0:~ \mu = 10.000 $ $H_1:~ \mu < 10.000 $

```
#testna statistika:
Z <- (9900 - 10000)/(120/sqrt(30))
Z

## [1] -4.564355

#kritična vrijednost
z <- qnorm(0.05)
z

## [1] -1.644854
```



S obzirom na to da je test statistika Z manja od kritične vrijednosti z čime Z spada u kritično područje, odbacujemo H_0 na razini značajnosti 5 %.

Drugi način, pomoću p-vrijednosti:

```
p = pnorm(Z)
p

## [1] 2.505166e-06
```

P-vrijednost je manja od razine značajnosti 0.05 pa odbacujemo H_0. Tvrdnju proizvođača možemo odbaciti.

Zadatak 49
Mjerili smo mase kraljevskih pingvina na jednoj koloniji na Antarktici u prethodnoj sezoni gniježđenja. Prosječna masa je iznosila 15,4 kg. Ove godine uzorkovali smo 35 pingvina nasumično u istoj koloniji i srednja vrijednost je iznosila 14,6 kg. Poznata je standardna devijacija mase u populaciji od 2.5 kg. Možemo li odbaciti hipotezu da se srednja masa pingvina ne razlikuje od prošlogodišnje na razini značajnosti od 5 %?

9.6.3. Pouzdani interval

Procjenjivanje populacijskoga parametra na temelju slučajnog uzorka nije 100 % pouzdano, stoga se uz njega procjenjuje i interval u kojem se s visokom vjerojatnošću nalazi pravi populacijski parametar.

Širina pouzdanog intervala (engl. *confidence interval*) ovisi o varijabilnosti podataka slučajnog uzorka, o veličini slučajnog uzorka te o razini pouzdanosti $(1-\alpha)$. Pouzdani interval je uži što je uzorak veći, što je varijabilnost manja i što je razina pouzdanosti manja.

Razina pouzdanosti od 95 % ne znači da postoji vjerojatnost od 95 % da se pravi parametar nalazi u p.i. (pouzdanom intervalu) nego da je postojala 5 % vjerojatnost da imamo uzorak koji ne daje p.i. koji sadrži parametar.

Kada izračunamo procjenu parametra za, na primjer, populacijsku sredinu $\overline{X}$, evo kako možemo konstruirati pouzdani interval za populacijsku sredinu μ.

Pretpostavimo da je slučajni uzorak dobiven iz normalne razdiobe. Razlikujemo dva slučaja, ovisno o tome je li standardna devijacija populacije σ poznata ili ne:

Ako je σ poznata, pouzdani interval je:

$$\overline{X} \pm z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}$$

Ako je σ nepoznata, pouzdani interval je:

$$\overline{X} \pm t_{n-1,\frac{\alpha}{2}} \frac{s}{\sqrt{n}}$$

gdje je s standardna devijacija uzorka. z i t su kvantili normalne, odnosno studentove t-razdiobe za danu vjerojatnost $\alpha/2$ (i n-1 stupnjeva slobode).

Primjer:
Prisjetimo se primjera s proizvođačem žarulja koji je tvrdio da je vijek trajanja žarulja minimalno 10000 sati rada, no na uzorku od 30 žarulja pronađeno je da su prosječno radile 9900 sati. Standardna devijacija je zadana i iznosi 120 sati. Odbačena je tvrdnja proizvođača da je vijek trajanja žarulja 10000 sati, no sada pronađimo pouzdani interval za pravi vijek trajanja žarulja.

$\overline{X} = 9900$
$\sigma = 120$

S obzirom na to da je σ poznata, koristit ćemo kvantile normalne razdiobe u $(1-\alpha/2)$

```
X_bar = 9900
z = qnorm(0.975)
sigma = 120
n = 30

L <- X_bar - z*sigma/sqrt(n)
D <- X_bar + z*sigma/sqrt(n)

cat("pouzdani interval = <", L, ", ", D, ">")
## pouzdani interval = < 9857.059 ,  9942.941 >
```

Zadatak 50

Izabrano je deset bivših studenata nekoga sveučilišta koji su nedavno diplomirali i anketirani su o visini mjesečnih primanja. Njihovi odgovori dani su u vektoru Place. Procijenite srednju vrijednost primanja nedavno diplomiranih studenata toga sveučilišta i nađite 90 % pouzdani interval za srednju vrijednost.

```
Place <- c(5617, 7066, 9594, 6726, 7178, 7898, 5033, 7151, 6514, 4000)
```

Pretpostavimo da su plaće nedavno diplomiranih studenata normalno distribuirane.

NAPOMENA: U osnovnim paketima sustava R ne postoji funkcija za računanje pouzdanog intervala, no postoji funkcija `CI()` iz paketa `Rmisc` koja računa pouzdani interval za dani vektor i nivo značajnosti.

10. DODATAK

10.1. Komunikacijske liste u sustavu R

Unutar sustava R postoje četiri glavne komunikacijske liste (engl. Mailing Lists) na kojima se nalaze informacije o sustavu:

- R-announce (https://stat.ethz.ch/mailman/listinfo/r-announce) – na ovoj se listi objavljuju velike novosti u sustavu, kao što je izdavanje nove verzije sustava R i promjene koje donosi.
- R-packages (https://stat.ethz.ch/mailman/listinfo/r-packages) – ovdje se objavljuju novosti o nadogradnji postojećih i dostupnosti novih paketa – uglavnom se odnosi na spremište CRAN.
- R-help (https://stat.ethz.ch/mailman/listinfo/r-help) – ovo je glavna obavjesna lista za sustav R, a namijenjena je raspravi o problemima i rješenjima pomoću sustava R, najavama koje nisu objavljene na prethodnim dvjema listama, obavijestima o dokumentaciji za sustav R, usporedbi i kompatibilnosti s jezikom S-plus te širenju dobrih primjera. R-announce i R-packages prosljeđuju sve obavijesti na R-help.
- R-devel (https://stat.ethz.ch/mailman/listinfo/r-devel) Ova je lista namijenjena pitanjima i raspravama vezanim za razvoj kôda programskoga jezika R.

Unutar sustava došlo je do stvaranja zajednica korisnika sustava R s posebnim interesima (engl. *Special Interest Group*, SIG). Tako je moguće predbilježiti se za primanje obavijesti vezanih za određeno područje, na primjer u grupi koja koristi i razvija alate za analize prostornih podataka R-SIG-geo na mrežnim stranicama.

10.2. Rješavanje problema uz pomoć zajednice

Ako ste početnik i ne razumijete poruke koje Vam sustav javlja kao pogrešku, najbolji i najjednostavniji način jest kopiranje teksta pogreške i unošenje u internetski pretraživač (na primjer *Google*). Na isti način moguće je pronaći i velik broj dokumenata i znanstvenih radova s temom koja nas zanima.

Ako na opisane načine niste uspjeli riješiti svoj problem, možete postaviti upit na neki od specijaliziranih portala (npr. stackoverflow), pri čemu treba paziti na sljedeća pravila:

- dati primjer koji će drugi korisnici moći reproducirati
- navesti koji ste rezultat očekivali, odnosno pokušali dobiti
- navesti što je prikazano umjesto očekivanog rezultata
- navesti verziju sustava R i paketa korištenih u analizi
- navesti naziv i inačicu operacijskoga sustava koji koristite
- osmisliti naslov poruke koji će na jasan način predstaviti problem.
- osmisliti naslov poruke koji će na jasan način predstaviti problem.

10.3. Specijalizirane konferencije

R-project aktivno podupire dvije konferencije koje redovno organizira R zajednica:

- useR! - konferencija korisnika sustava R i
- DSC (*Directions in Statistical Computing*) - konferencija koja okuplja programere statističkoga softvera i istraživača u statističkom računalstvu koji su usredotočeni na sustav R, ali nije isključivo posvećena sustavu R. Cilj konferencije je osigurati platformu za razmjenu ideja o zbivanjima u statističkom računalstvu.

Popis ostalih konferencije o sustavu R dostupan je na http://www.r-project.org/conferences/.

10.4. Literatura o sustavu R

Unutar sustava R postoji velika količina literature slobodno dostupne korisnicima. Osnovna grupa suradnika za razvoj sustava R (*R Core Team*) zadužena je za održavanje i sadržaj mrežnih stranica s priručnicima.

Priručnici su dostupni pozivom funkcije `help.start()` u kojoj se može pronaći priručnik An Introduction to R.

```
help.start()
```

Ostala korisna literatura:

- Wickham, H., Grolemund, G. (2016) *R for Data Science*. Sebastopol: O'Reilly Media, Inc. (dostupna i online na https://r4ds.had.co.nz/)
- *Bookdown*. URL: https://bookdown.org/ - (sadrži aktualne knjige dostupne online (otvorenoga pristupa))
- Gentelman, R., Hornik K., Parmigiani, G. (2006 — 2009) *Use R!* Springer Berlin: Springer.
- *Books related to R*. URL: http://www.r-project.org/doc/bib/R-books.html
- *The R Journal*. URL: http://journal.r-project.org/current.html
- *The R Manuals*. URL: http://cran.rproject.org/manuals.html
- *Contributed documentation*. URL: http://cran.r-project.org/other-docs.html (popis literature na drugim jezicima)

Na hrvatskom jeziku moguće je pronaći dva dokumenta:

- Kasum, D., Legović, T. (2004) *Uvod u korištenje R-a* (prijevod *An introduction to R* autorskog tima Venables, W.N., Smith, D.M.).
- *Osnovne naredbe u R-u*. URL:http://cran.rproject.org/doc/contrib/Kasum-QuickRefCard_ver.1.2.pdf