Uvod u R i RStudio

Kostelić, Katarina; Etinger, Darko

Authored book / Autorska knjiga

Publication status / Verzija rada: **Published version / Objavljena verzija rada (izdavačev PDF)**

Publication year / Godina izdavanja: **2024**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:595409

https://doi.org/10.17605/OSF.IO/DNFM8

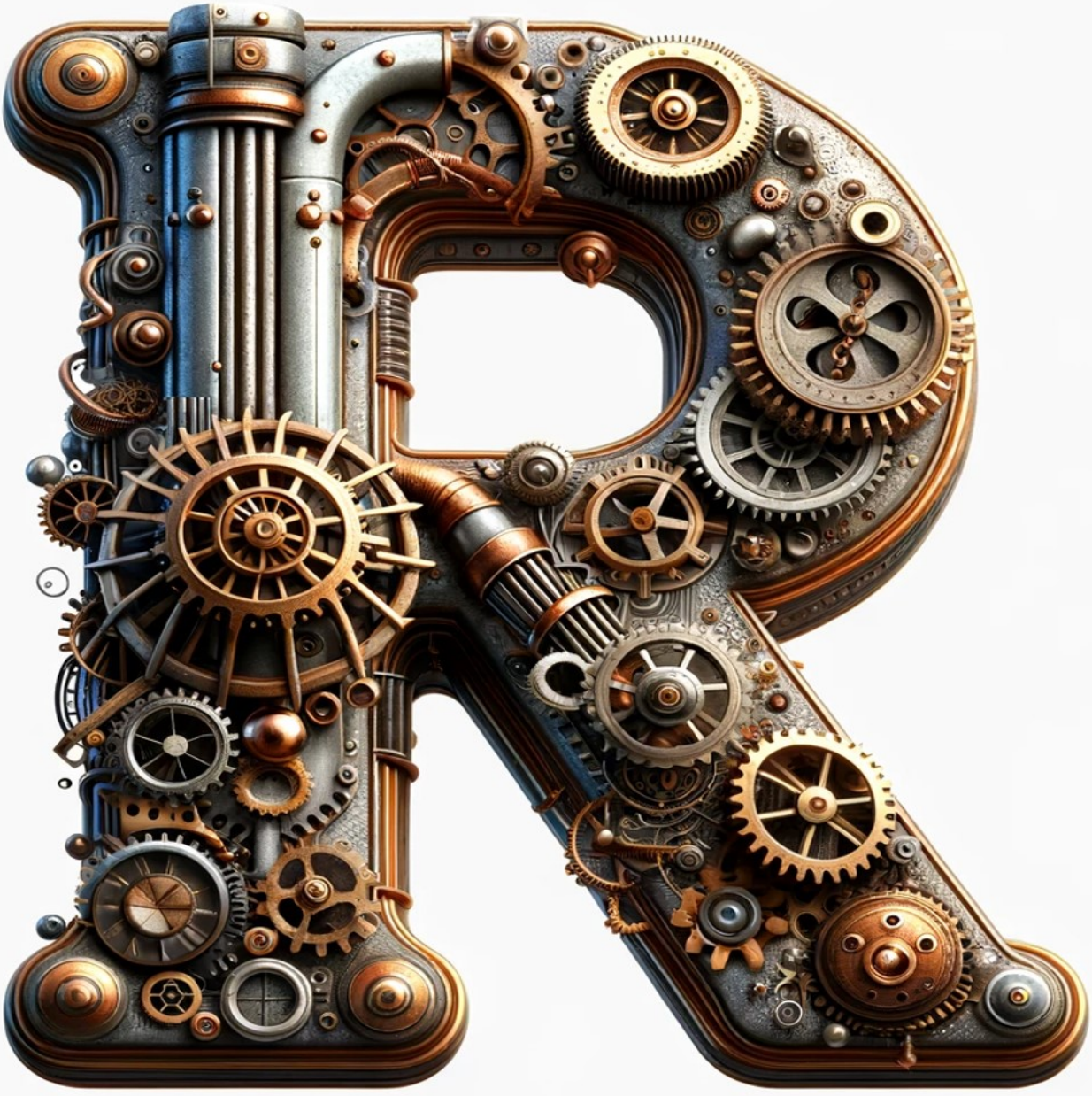
Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-10**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Uvod u R i R Studio



> Katarina Kostelić
> Darko Etinger

Sveučilište Jurja Dobrile u Puli

FIPU Fakultet informatike u Puli

Izdavač i sjedište

Sveučilište Jurja Dobrile u Puli
Zagrebačka 30, Pula

Za izdavača

prof. dr. sc. Marinko Škare

Autori

Katarina Kostelić
Darko Etinger

Recenzenti:

Siniša Sovilj
Siniša Miličić
Ivan Lorencin
Marko Turk

Lektura:

Nitor usluge d.o.o.
Petračićeva 4, 10000 Zagreb

Grafičko oblikovanje i prijelom

RMarkdown, gitbook, DALL · E

ISBN 978-953-377-069-7

DOI 10.17605/OSF.IO/DNFM8

Broj i datum odluke Odbora za izdavačku djelatnost:

Sveučilišni priručnik objavljen je na temelju Odluke Odbora za izdavačku djelatnost Sveučilišta Jurja Dobrile u Puli, KLASA: 611-04/24-01/35, URBROJ: 143-01-15-24-1 od 16. prosinca 2024. godine.

Predgovor

Dobrodošli na stranice priručnika **Uvod u R i RStudio**!

Ovaj priručnik je kreiran s ciljem pružanja pristupačnog načina savladavanja i razumijevanja programskog jezika R i razvojnog okruženja RStudio - važnih i korisnih alata u svijetu analize podataka i primjene kvantitativnih metoda.

Što možete očekivati:

- Jasno objašnjenje osnova: bez obzira na vaše predznanje programiranja, ovaj priručnik će vam pomoći da razumijete osnovne koncepte i načela rada u R-u
- Praktični primjeri: pronaći ćete brojne primjere i vježbe koje će vam pomoći da primijenite naučeno u stvarnim situacijama
- Korak-po-korak: detaljne upute će vas provesti kroz instalaciju, konfiguraciju i korištenje R-a i RStudija
- Savjeti i trikovi: naučite male trikove koji će učiniti vaš rad u R-u učinkovitijim i ugodnijim

Kako koristiti ovaj priručnik:

- Poglavlje po poglavlje: priručnik je strukturiran tako da možete pratiti poglavlja redoslijedom kako biste postupno gradili svoje znanje
- Vježbajte redovito: koristite priložene primjere i vježbe kako biste praktično primijenili ono što ste naučili

Želimo vam uspješno učenje i nadamo se da ćete uživati u svakom koraku svog putovanja kroz R i R Studio!

Katarina Kostelić

Darko Etinger

Sadržaj

1 R i RStudio

R je programski jezik i okruženje za statističku obradu i vizualizaciju podataka. Često se upotrebljava zbog toga što sadrži već ugrađene pakete za statističku analizu podataka. R sadrži (William N. Venables, Smith, and Team 2009):

- napredne alate za upravljanje podacima i njihovu pohranu,
- niz operatora specijaliziranih za izvođenje matematičkih operacija nad nizovima i matricama,
- sveobuhvatnu i povezanu skupinu alata za srednje naprednu analizu podataka,
- grafičke opcije za analizu i prikaz podataka, bilo izravno na računalu ili u tiskanom obliku,
- dobro razvijen, jednostavan za korištenje i efikasan programski jezik, koji obuhvaća uvjetne izjave, petlje, funkcije koje definira korisnik te opcije za unos i ispis podataka. Također, moguće je nadograđivati pakete, ali i samostalno kreirati vlastite funkcije i objekte, pa čak i pakete.

U nastavku ovog poglavlja, upoznat ćete se s osnovnim mogućnostima i prednostima upotrebe R-a i RStudija u praksi, za koje vjerujemo da će motivirati daljnje čitanje. Poglavlje se nastavlja na pobliže objašnjavanje što treba naučito o R-u i RStudiju, kroz potpoglavlja o razumijevanju R-a, statističkoj obradi i grafičkim prikazima u R-u te naprednim tehnikama i rješavanju problema. Posljednja potpoglavlja odnose se na izvore dodatne pomoći i resursa te općenite savjete.

1.1 Upotreba R-a i RStudija u praksi

R i RStudio koriste se u raznim industrijama za analizu i upravljanje podacima te vizualizaciju. Ovi alati nude sveobuhvatan skup funkcija koje zadovoljavaju potrebe znanstvenika, analitičara i menadžera. Budući da je R programski jezik otvorenog koda, omiljen je zbog svoje opsežne biblioteke statističkih i grafičkih tehnika, dok RStudio pruža integrirano razvojno okruženje koje poboljšava korisničko iskustvo.

1.1.1 Upravljanje podacima i manipulacija

U praksi, R se koristi za upravljanje strukturama podataka i manipuliranje skupovima podataka. To uključuje spajanje i kombiniranje podatala te izdvajanje podskupova podataka, kao i rukovanje varijablama datuma i vremena (Horton and Kleinman 2015).

Paket tidyverse u R-u posebno je popularan za upravljanje podacima, omogućujući korisnicima učinkovito čišćenje i transformaciju podataka (Dayal 2015).

1.1.2 Analiza podataka

R pruža široku lepezu statističkih funkcija, uključujući distribucije vjerojatnosti, generiranje slučajnih brojeva i matrične operacije. Podržava uobičajene statističke postupke kao što su deskriptivna statistika, bivarijatna statistika i tablice kontingencije (Horton and Kleinman 2015). Napredne statističke metode kao što su linearna regresija, ANOVA, generalizirani linearni modeli i analiza preživljavanja također su implementirane u R, što ga čini moćnim alatom za složenu analizu podataka (Horton and Kleinman 2015).

Osim toga, dodatnu širinu stvara dostupnost metoda poput Bayezijanske statistike, strojnog učenja, sentiment analize te metoda poput linearnog i nelinearnog programiranja, dinamičkog programiranja i višekriterijskog odlučivanja. Iako je R u početku bio korišten gotovo isključivo za statističku analizu te upravljanje podacima (čišćenje i transformaciju), danas omogućuje gotovo svaki tip kvantitativne i kvalitativne analize.

1.1.3 Vizualizacija

R nudi robusne grafičke mogućnosti, omogućujući stvaranje jednovarijantnih, dvovarijantnih i multivarijantnih grafikona. Također su mogući grafikoni posebne namjene i interaktivne vizualizacije, čime se poboljšava interpretabilnost podataka (Horton and Kleinman 2015). RStudio olakšava stvaranje i upravljanje tim vizualizacijama, pružajući korisničko sučelje za spremanje i konfiguriranje grafičkih izlaza (Horton and Kleinman 2015). Takva korisnička sučelja nazivaju se *engl. dashboards*.

1.1.4 Reprodukcija istraživanja i simulacija

R i RStudio podržavaju ponovljivu analizu kroz značajke skriptiranja i upravljanja projektima. To osigurava da se analize mogu lako replicirati i dijeliti (Hair et al. 2021). Izračuni performansi modela temeljeni na simulaciji i generiranje podataka također su uobičajene primjene koje praktičarima omogućuju učinkovito modeliranje i predviđanje ishoda (Horton and Kleinman 2015).

1.1.5 Praktične primjene u različitim sektorima

R i RStudio naširoko se koriste u raznim industrijama za statističke analize, operacijska istraživanja, rudarenje podataka i aplikacije znanosti o podacima. Profesionalci i stručnjaci koriste R i RStudio kako bi poboljšali procese donošenja odluka vođene podacima. U nastavku su neki specifični načini na koje se ti alati koriste u različitim industrijama:

- R se intenzivno koristi u financijama za zadatke poput upravljanja rizikom, optimizacije portfelja i financijskog modeliranja. Njegova sposobnost rukovanja složenim statističkim izračunima i vizualizacijama čini ga vrijednim alatom u ovom sektoru (Allen 2017).
- U znanosti o odlučivanju, R se koristi za analizu podataka i modeliranje, čime se pružaju uvidi u velike skupine podataka što pruža potporu procesima strateškog odlučivanja (Allen 2017).
- R je moćan alat za rudarenje podataka koji se koristi u industrijama kao što su osiguranje i medicina, pa čak i u javnom sektoru. Podržava različite tehnike rudarenja podataka, pomažući stručnjacima dobivanje vrijednih uvida iz podataka (Zhao and Cen 2013).
- Studije slučaja iz stvarnog svijeta pokazuju primjenu R-a u rješavanju problema specifičnih za različite industrije, pokazujući njegovu prilagodljivost i učinkovitost u različitim okruženjima (Zhao and Cen 2013).
- R je kamen temeljac u podatkovnoj znanosti, kombinirajući tradicionalnu statistiku sa strojnim učenjem za analizu velikih podataka. Koristi se u bioinformatici, optimizaciji opskrbnog lanca, zdravstvu i marketingu, ali i šire (Miller 2017).
- RStudio poboljšava mogućnosti R-a pružajući integrirano razvojno okruženje koje pojednostavljuje upravljanje podacima, izgradnju modela i prezentaciju rezultata (Van der Loo 2012).
- RStudio je popularno integrirano razvojno okruženje, IDE (*engl. Integrated Development Environment*) koje olakšava razvoj i primjenu R-a, olakšavajući analitičarima provođenje statističke analize i izvještavanja. Podržava replikativna ili ponovljiva istraživanja, kao i stvaranje prilagođenih paketa (Van der Loo 2012).
- Funkcionalnost IDE-a olakšava upravljanje i istraživanje skupova podataka, što je ključno za provođenje sveobuhvatnih projekata znanosti o podacima (Miller 2017).

Knjiga **Data Mining Applications with R** (Dayal 2015) je izvrsna kolekcija praktičnih primjena R-a. Evo nekliko zanimljivih slučajeva.

- Na primjer, analiza podataka električne mreže objašnjava kako se R i Hadoop zajedno koriste za analizu ogromnih skupova podataka sa senzora u električnoj mreži. S ovom postavkom, analitičari mogu brzo uočiti probleme ili događaje u elektroenergetskom sustavu čišćenjem i sortiranjem podataka, što pomaže u otkrivanju problema izravno iz stvarnih podataka umjesto da se oslanja samo na predviđanja.
- Na antropološkoj konferenciji, R je korišten za analizu podataka preuzetih s Twittera. Proučavajući tweetove, istraživači su mogli uočiti glavne teme o kojima se raspravljalo, identificirati članove zajednice i vidjeti kako su se ljudi povezali. Ovaj slučaj pokazuje kako R pomaže u pronalaženju popularnih tema i identificira online zajednice sa zajedničkim interesima.

- R se može primijeniti i na digitalne knjižnice s ciljem organiziranja i pronalaženja uzoraka u velikim zbirkama istraživačkih radova. To uključuje prepoznavanje glavnih tema, grupiranje sličnih tema i pronalaženje poveznica između autora ili radova. Pomaže da velike biblioteke informacija budu lakše pretražive i upravljive.

- Primjer *Recommender Systems* pokazuje kako se R može koristiti za kreiranje sustava preporuka, poput onih koji se koriste na stranicama za kupovinu, za predlaganje proizvoda. Za pronalaženje najboljih preporuka za korisnike koriste se različite metode, uspoređujući tehnike na temelju čimbenika kao što su točnost i iznenađenje. Pomaže programerima odabrati pravi pristup za izradu personaliziranih prijedloga.

- U sportu, R se koristi za analizu učinka igrača i tima. Prikupljanjem podataka o rezultatima, statistikama igrača i ishodima utakmica, analiza provedena uz pomoć R-a može pomoći trenerima i analitičarima da donesu bolje odluke o strategiji, treningu, pa čak i odabiru igrača.

- Poduzeća koriste R za predviđanje buduće prodaje proučavanjem povijesnih podataka o prodaji. U konkretnom primjeru, analizira se kako osiguravajuće društvo može koristiti podatke o svojim trenutnim klijentima za modeliranje ciljanih profila kupaca usmjerenih na unakrsnu prodaju polica kampera, kao i ciljanje novih kupaca. Postojeći podaci o kupcima se pregledavaju i prethodno obrađuju, nakon čega slijedi preliminarna analiza (kao što je linearna regresijska analiza) kako bi se steklo razumijevanje prirode podataka. Razmotrene su različite metode modeliranja prije nego što je odlučeno da se razviju modeli klasifikatora profila ciljnih kupaca. Prikazani su modeli izvedeni s četiri različite metode klasifikatora s usporednim rezultatima njihove sposobnosti objašnjavanja i predviđanja, kao i računalne učinkovitosti. Naposljetku, kritički se ocjenjuju rezultati data mininga i rezultirajući modeli, kao i moguća poboljšanja u procesu izgradnje modela, uz zaključke kako rezultati mogu pružiti smjernice za ciljanu marketinšku strategiju.

- Preferencije kupaca često su temelj za podršku strateškom planiranju i donošenju odluka u poduzećima. Jedan od načina analiziranja preferencija je modeliranje višekriterijskih odluka (MCDM) kupaca. Međutim, to zahtijeva istovremeno razmatranje više kriterija, što je za mnoge tradicionalne metode nedostatno. Autori primjenjuju tehniku agregacije (*Choquet Integral*) i predstavljaju novorazvijeni okvir, nazvan Rfmtool, za analizu preferencija. Pomoću studije slučaja singapurske hotelske industrije, demonstrira se kako se ovaj skup alata može koristiti za otkrivanje preferencija koje utječu na hotelske odabire putnika. Očekuje se da će predstavljena tehnika i razvijeni alati koristiti istraživačima i menadžerima diljem svijeta u izvođenju učinkovitijeg poslovnog upravljanja.

- Jedna se studija fokusira na korištenje R-a za analizu i optimizaciju DNS (*engl. Domain Name System*) prometa. Kako DNS promet brzo raste svake godine, uz dodatne sigurnosne protokole (DNSSEC), poslužiteljima je potrebno više resursa poput CPU snage i memorije. Kako bi unaprijedili taj proces, autori koriste R za rudarenje podataka, kao što su klasteriranje i smanjenje dimenzionalnosti, za grupiranje i upravljanje prometom učinkovitije na temelju naziva domena (FQDN) umjesto IP adresa. R-ove funkcije i paketi podržavaju metode poput K-means klasteriranja, linearnog programiranja i heurističkih pristupa koje omogućuju modeliranje i unaprijeđenje procesa kroz uravnoteženje opterećenja poslužitelja. Testirajući ih na DNS prometu u stvarnom svijetu, modelom je postignuto značajno smanjenje resursa CPU-a (oko 30%) s minimalnim razlikama u opterećenju između poslužitelja, čime su DNS platforme radile glatko i učinkovitije.

Iako je najistaknutija primjena u znanosti o podacima i poslovnoj analitici, primjeri s preferencijama potrošača i optimizacijom DNS prometa pokazuju kako se programski jezik R koristi i u operacijskim istraživanjima. R pruža osnovne i napredne alate za rješavanje problema optimizacije i stvaranje modela koji rješavaju složena pitanja, kao što su optimizacije sustava, procesa i odluka. Integracija R-a u operacijska istraživanja olakšava analizu i modeliranje sustava, čineći ga vrijednim resursom za studente, stručnjake i istraživače u tom području (González-Pérez, López, and Sampedro 2014).

Dakle, R i RStudio su vrlo učinkoviti u mnogim industrijama, ali nisu jedini dostupan analitički alat, odnosno programski jezik. Na primjer, Python, nudi slične mogućnosti i ima vlastite prednosti, poput jednostavnosti integracije s web aplikacijama i širim bibliotekama strojnog učenja. Bez obzira na to, snažna statistička osnova R-a i opsežan ekosustav paketa i dalje ga čine preferiranim izborom za mnoge stručnjake koji se bave analizom podataka, modeliranjem i optimizacijom.

R i RStudio su moćni alati za analizu podataka, ali zahtijevaju određenu razinu znanja programiranja. Preciznost potrebna u sintaksi naredbi i rukovanju objektima može predstavljati izazov za početnike. Međutim, dostupnost opsežne dokumentacije i podrške zajednice pomaže u ublažavanju ovih izazova, čineći R i RStudio dostupnima široj publici (Dayal 2015).

1.2 Razumijevanje R-a i osnove rada u R-u

Osnove rada u R-u podrazumijevaju učenje instalacije, postavljanja radnog okruženja (poput RStudio) i osnovnih koncepta kao što su varijable, tipovi podataka i osnovne funkcije. Nakon toga, potrebno je upoznati se sa strukturama podataka u R-u, uključujući vektore, liste, matrice i podatkovne okvire (*engl. data frames*). Također, važno je naučiti kako učitavati, pregledavati, čistiti i transformirati podatke.

1.3 Statistika i grafički prikaz u R-u

Nakon savladavanja osnova, korisno je upoznati se s osnovama statističke analize u R-u, uključujući opisnu statistiku, testiranje hipoteza i linearnu regresiju. Osim toga, izuzetno je korisno naučiti kako stvoriti i prilagoditi grafičke prikaze u R-u, kreirajući različite vrste grafikona.

1.4 Napredne tehnike i rješavanje problema

Nakon savladavanja osnova koje nudi ovaj priručnik, studenti će imati dostatni temelj za daljnje razvijanje vještine programiranja u R-u, uključujući upotrebu petlji, uvjetnih izjava i funkcija. Naprednije tehnike obuhvaćat će i upravljanje složenim podacima, dok će se korisnici u radu naići i na potrebe za optimizacijom koda i rješavanja programskih grešaka.

1.5 Gdje pronaći pomoć i resurse

- Stack Overflow: Za specifična pitanja i probleme.
- R-bloggers: Za širok spektar blogova i članaka o R-u.
- CRAN: Za službenu dokumentaciju R-a i paketa.
- Knjige i časopisi: R for Data Science (Wickham, Çetinkaya-Rundel, and Grolemund 2023) i The R Journal za dublje razumijevanje.

Postoji više inačica uvoda u R i/ili RStudio, na primjer:

- An Introduction to R (William N. Venables et al. 2009) - službeni priručnik za rad u R-u, dostupan putem CRAN-a
- Uvod u korištenje R-a (Kumbatović, Kasum, and Legović 2004) - prijevod službenog priručnika na hrvatski jezik
- An Introduction to R (Douglas et al. 2022) - priručnik/ udžbenik
- Intro to R - kernel putem kaggle.com

Nadalje, postoje i brojni online tečajevi za učenje R jezika, najčešće u kontekstu statističke analize, znanosti o podacima ili poslovne analitike. Među platformama koje nude takve programe su datacamp, codeacademy, coursera, udemy i druge.

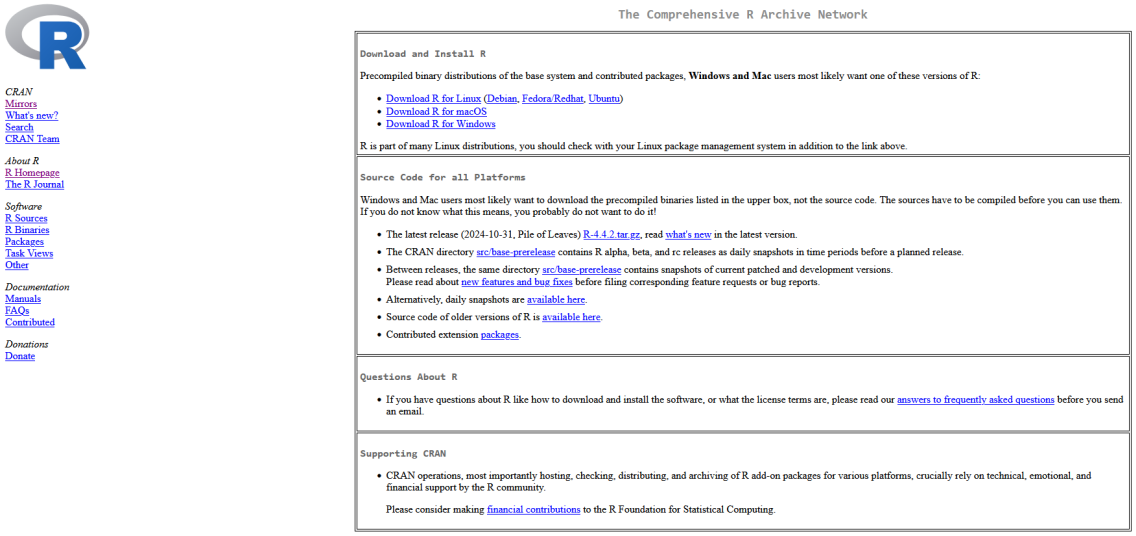
1.6 Općeniti savjeti

- Vježbajte kroz male projekte ili analize stvarnih podataka.
- Budite strpljivi i ne bojte se grešaka; one su dio učenja.
- Koristite kombinaciju različitih resursa za najbolje rezultate.
- Ostanite u tijeku s najnovijim trendovima i paketima u R ekosustavu.

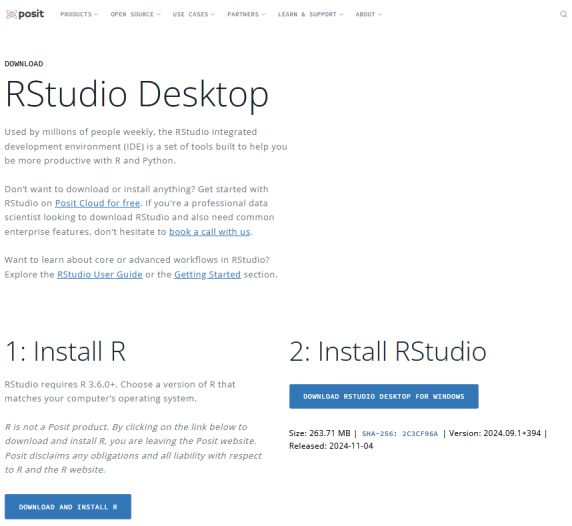
2 Osnovni pojmovi i prvi koraci

Ovo poglavlje služi kao temeljni uvod u rad s R-om, pružajući sve potrebne informacije za početak rada, s posebnim naglaskom na praktične aspekte i osnovnu organizaciju rada. Kroz strukturiran pristup, poglavlje omogućava postupno upoznavanje s alatima i konceptima potrebnim za učinkovito korištenje R-a u analizi podataka.

Potrebno je prvo instalirati R na računalo, a potom RStudio. Najbolje je R preuzeti sa službene stranice The Comprehensive R Archive Network. Instalacija RStudija može se provesti na dva načina – preuzimanjem datoteke s Posit-a ili iz R-a (potreban je paket devtools). Prvi je način puno jednostavniji i češće korišten. Nakon preuzimanja instalacijskih datoteka, slijedite upute za instalaciju. Primjer kako postupak instalacije može izgledati dostupan je na GitHubu.



Slika 1: Preuzimanje putem CRAN-a



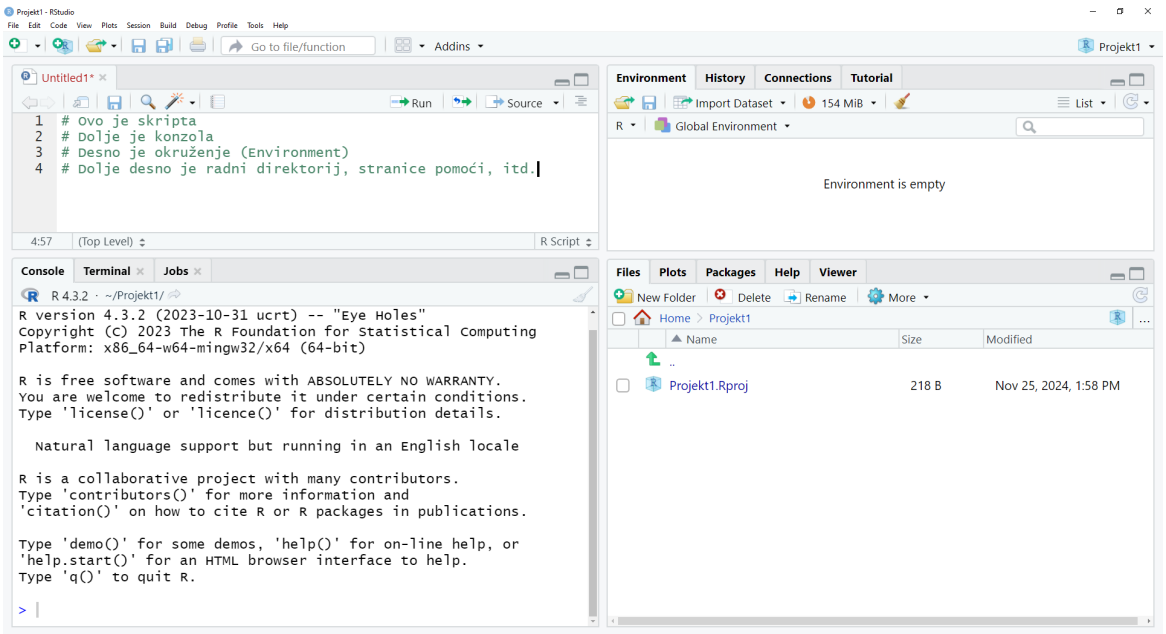
Slika 2: Preuzimanje putem posit-a

Digresija: Posit je novi naziv za tvrtku koja je prethodno bila poznata kao RStudio. Promjena imena odražava širu misiju tvrtke: pružanje podrške alatima ne samo za R, već i za Python i druge jezike korištene u analitici podataka. Posit Connect (ranije poznat kao RStudio Connect) je alat koji omogućuje distribuciju, dijeljenje i hosting analitičkog sadržaja razvijenog pomoću R-a i Python-a, kao što su analitičke aplikacije, izvještaji, nadzorne ploče (dashboards) i API-ji.

Jedna od prednosti R-a jest što se radi o besplatnom, open source softverskom okruženju. Zbog toga je njegova upotreba raširena, a brojni primjeri upotrebe pojedinih paketa mogu se naći u pratećoj dokumentaciji tih paketa (dostupnoj online i preko Help opcije), online forumima te GitHubu. Pritom je **CRAN** (the Comprehensve R Archive Network) osnovna mrežna stranica koja funkcionira kao repozitorij sustava R, paketa i dokumentacije.

U ovom uvodu, prikazat će se osnovni pojmovi i funkcije, osnovne strukture podataka, kreiranje i definiranje tih struktura te odabiranje, dodavanje i brisanje elemenata iz različitih vrsta struktura podataka, uz osnove rada s tim strukturama podataka. Navedene su osnove koje se uobičajeno koriste tijekom pripreme podataka za analizu.

RStudio je najčešće korištena platforma za rad pri korištenju R-a. Ova platforma ne samo da omogućava uređivanje i pisanje skripti, nego također pruža i raznovrsne korisne alate (Irizarry 2019). Za početak, upoznat ćete se s osnovama rada u RStudiju.



Slika 3: RStudio korisničko sučelje

- Gore lijevo: Script/ Source
- Gore desno: Environment
- Dolje lijevo: Console
- Dolje desno: Files/ Plots/ Packages/Help/ Viewer

Ako je prethodno otvoren novi **R script** (skripta), gore lijevo prikazuju se zadane naredbe. Alternativno, u tom prozoru prikazuju se korištene datoteke i objekti uz naredbu `view()` ili klik na datoteke/objekte u prozoru Environment.

2.1 Okruženje ili environment

Environment predstavlja radno okruženje i tu se nalaze svi korišteni objekti (učitani podatkovni okviri (*engl. data frame*), funkcije, varijable, itd.) u pojedinoj sesiji. Okruženje u R-u je zapravo kolekcija parova ime/vrijednost (gdje su imena varijabli, a vrijednosti su objekti pridruženi tim imenima). Neki objekti mogu biti nevidljivi u standardnom prikazu okruženja, kao što su zatvoreni (*engl. locked*) objekti ili oni kreirani unutar specifičnih struktura.

Globalno okruženje je vrhovna razina u kojoj radite. Lokalna okruženja se stvaraju unutar funkcija i izolirana su od globalnog okruženja. Klikom na **Global Environment** dobiva se pregled svih aktivnih paketa u sesiji. U tom prozoru moguće je pratiti i utrošenu memoriju, ukloniti sve objekte iz okruženja, pregledati povijest izvršenih naredbi ili pregledati veze (ako su uspostavljane tijekom sesije). Također, dostupni su i linkovi na tutorijale o RStudiju.

Možete koristiti funkcije kao što su `ls()` ili `objects()` za prikaz svih objekata unutar trenutnog okruženja, odnosno `rm()` za uklanjanje objekata iz okruženja kako biste oslobodili memoriju i izbjegli zabunu s imenima. Preporučuje se korištenje RStudio projekata za odvajanje različitih radnih okruženja. Svaki projekt u RStudiju ima svoje okruženje. Također, periodično provjeravajte upotrebu memorije pomoću `memory.size()` i `memory.limit()` na Windowsu ili `gc()` na svim platformama.

Organizirajte kod u zasebne skripte prema zadatku ili funkcionalnosti kako biste izbjegli nered u globalnom okruženju.

2.2 Konzola

Console ili konzola prikazuje izvršene naredbe. Naredbe se mogu upisivati i izravno u konzolu. Druga, preferirana mogućnost, je upisivati naredbe u skriptu, pri čemu će se one prikazati u konzoli tek nakon što se te naredbe izvrše (Run ili Ctrl+Enter). Novu skriptu možete kreirati tako da u gornjem desnom uglu, ispod izbornika Files, kliknete na zeleni krug s bijelim plusom i iz padajućeg izbornika odaberete **R script**. Alternativno, možete koristiti prečicu Ctrl+Shift+N.

2.3 Datoteke i radni direktorij

U donjem desnom prozoru na početku svake sesije bit će prikazane datoteke (**Files**). Moguće ih je učitati izravno, klikom na datoteku te odabirom Import data. Nakon što su učitane, datoteke će biti pripisane objektu (najčešće je to podatkovni okvir) koji će se pojaviti u prozoru Environment. Prikazane datoteke (*engl. Files*) su one koje se nalaze u radnom direktoriju, pri čemu je zadana (*engl. default*) mapa Dokumenti (*engl. Documents*). Provjera za utvrđivanje koja se mapa koristi kao radni direktorij vrši se putem upisa `getwd()` u konzolu ili skriptu. Ukoliko se radni direktorij želi promijeniti, to je moguće učiniti pomoću `setwd()` i upisa puta do mape radnog direktorija.

```
> getwd()
> setwd("C:/Users/Documents/Intro")
```

2.4 Projekti

Pri radu u R-u jedan od jednostavnijih pristupa je kreiranje dokumenta R projekta (**R project**) na početku rada na pojedinom predmetu (File-> New Project->New Directory->New Project te upis

naziva novog projekta; alternativno, u gornjem desnom uglu kliknete na dodavanje novog projekta). Na taj način kreira se novo radno okruženje. Ako radite na više predmeta istovremeno i želite svaki od njih sačuvati kako biste se kasnije mogli vratiti radu ili pregledu svakog od njih, korištenje ove opcije će to olakšati. Naime, pojedinom projektu pristupate putem opcije *Open project*, a ukoliko ste sve pohranili prije posljednjeg zatvaranja, možete nastaviti rad na tom predmetu/projektu, bez da morate počinjati ispočetka svaki sljedeći put. Osim toga, to je korisno i zato jer ćete ponekad htjeti koristiti dijelove koda/ naredbi koje ste kreirali ranije, a ova opcija to olakšava.

Ipak, treba imati na umu da je glavni izvor temeljem kojeg rekreirate radno okruženje zapravo skripta. Čak i ako okruženje nije sačuvano, pomoću skripte možete ponovo instalirati pakete i kreirati objekte.

2.5 Skripte

Skripta se može spremiti u obliku zasebnog dokumenta, **R script**. Ako želite upisati komentare, kao podsjetnik čemu pojedina naredba služi ili objašnjenje skupa naredbi, ispred tih komentara potrebno je upisati #. Tako spremljena datoteka može se ponovo koristiti, dorađivati i/ili ispravljati.

```
> # Ovo je komentar.
> # Ako su ljestve ispred teksta, R će ga ignorirati.
> # Da ispred teksta nisu ljestve, R bi javljao grešku u kodu.
```

Osim skripte, moguće je koristiti R dokument **R Markdown**. R Markdown je izvrsna opcija ako se R koristi pri sastavljanju izvješća ili dokumenata, jer podržava pisanje i uređivanje (css, latex) tekstova u kombinaciji s izvođenjem koda i izlazom dokumenta u različitim formatima (html, pdf, docx, epub, i dr.). Nova bilježnica ili R Markdown naći će se kao opcija u padajućem izborniku ispod R script. R Markdown može sadržavati kombinaciju koda, teksta i rezultata, a izlazne datoteke kreiraju se klikom na Knit u alatnoj traci prozora Source. Osim navedene kombinacije upotrebe teksta, koda i rezultata, u R Markdownu, naredbe se, u pravilu, pišu i izvršavaju u blokovima (ili *engl. chunk*, klik na ikonu +c za dodavanje bloka koda ili prečicom Ctrl+Alt+I). Po pitanju naglašavanja teksta, # ispred teksta na početku reda označit će tekst kao naslov poglavlja; tekst između *dviju zvjezdica* bit će označen ukošeno (Italics), a tekst između **dvostrukih zvjezdica** podebljano (Bold).

Skripta može sadržavati jednu ili niz povezanih naredbi, a naredba ili skup naredbi pokreću se klikom na **Run** ili prečicom **Ctrl+Enter**. S obzirom da je R izuzetno osjetljiv na korištenu sintaksu (velika i mala slova, tipfelere, otvaranje i zatvaranje zagrada i dr.), korištenje skripte često predstavlja uštedu vremena. Prvo, nakon upisane naredbe uz pripadajući broj retka uz naredbu prikazat će se crveni kružić s ili bez x-a ukoliko ta naredba sadrži grešku. Drugo, čak ako se naredba primijeni, a obavijest o grešci prikaže u konzoli, neće biti potrebno pretipkavanje čitave naredbe, nego samo korekcija u skripti.

Korištenje skripti u R-u je ključno za učinkovito i organizirano programiranje. Evo nekoliko savjeta kako ih pravilno koristiti:

- Organizirajte skriptu tako da svaki dio koda obavlja određenu funkciju. Ovo uključuje učitavanje podataka, transformaciju podataka, analizu podataka, i vizualizaciju.
- Redovito komentirajte svoj kod kako bi bio jasniji vama i drugima. Komentari bi trebali objasniti što određeni dio koda radi i zašto.

- Koristite smislena i opisna imena za varijable. Izbjegavajte kratka ili nejasna imena, kao što su x, df, ili temp, osim u slučaju kratkotrajnih privremenih varijabli.
- Ako određeni dio koda koristite više puta, razmislite o pretvaranju tog dijela u funkciju. To olakšava održavanje i testiranje koda.
- Umjesto da izravno upisujete putanje do datoteka ili ključne parametre, koristite varijable. To olakšava promjene i čini kod prenosivijim.
- Koristite RStudio projekte za organizaciju skripti, podataka i povezanih datoteka. To pomaže u očuvanju putanja i olakšava suradnju s drugima.
- Koristite sustave kontrole verzija poput Git-a za praćenje promjena u skriptama. To omogućava lakše vraćanje na prijašnje verzije i suradnju.
- Redovito testirajte svoj kod kako biste osigurali da radi kako je očekivano. Možete koristiti pakete poput testthat za automatsko testiranje u R-u.
- Pišite kod imajući na umu da biste ga mogli ponovo koristiti. Ako mislite da ćete kod ponovno koristiti, napišite ga tako da bude lako prilagodljiv novim situacijama i da vam bude jasan i čitak čak i ako prođe puno vremena prije nego ga ponovo upotrijebite.
- Za složenije skripte ili pakete, napišite dokumentaciju koja objašnjava kako se skripta koristi, njezine funkcije i primjere.
- Redovito ažurirajte skripte kako bi se pratile promjene u R-u i povezanim paketima.

2.6 Plots, Viewer i Help

Prikaz donjeg desnog prozora možete promijeniti odabirom Plots, Packages, Help ili Viewer. **Plots** prikazuje kreirane grafičke prikaze. **Help** nudi prikaz dokumentacije vezane uz pakete i/ili naredbe. Taj je prikaz izuzetno koristan, osobito pri korištenju novih naredbi ili pri podsjećanju kako naredbu treba zadati (ili kako točno naredba glasi) jer daje uvid u argumente koje pojedina naredba sadržava, uz mogućnosti i primjere upisa te rezultat/output s obzirom na zadane argumente. **Viewer** će prikazati lokalni sadržaj vezan uz korištenje mrežnih resursa (npr. izrada interaktivnog html grafičkog prikaza, iteracije pri simulacijama uz distribuciju upotrebe jezgri i radne memorije i sl.).

2.7 Paketi

Packages prikazuje popis dostupnih paketa. Paketi se mogu pozvati iz skripte ili konzole s naredbom `library()` ili klikom uz željeni paket na popisu Packages. Pri radu u R-u, u pravilu, koriste se paketi. **Paketi** predstavljaju grupirane funkcionalnosti R-a. Svaki paket sastoji se od kompiliranog koda i skupa funkcija (ili skupina funkcija). Primjena pojedinog paketa najčešće je vezana za točno određeni, definirani format podataka. Zato je prateća dokumentacija izuzetno važna i treba steći naviku iščitavanja prateće dokumentacije.

Odabirom **Packages** u donjem desnom prozoru, prikazat će se popis svih osnovnih paketa koji su već instalirani. Ako želite koristiti neki od paketa koji se ne nalaze na tom popisu, potrebno ih je prvo instalirati. To se radi naredbom `install.packages()`, npr. `install.packages("tidyverse")` (Wickham et al. 2019). Da bi se paket aktivirao, potrebno ga je pozvati pomoću `library()`, npr. `library(tidyverse)` ili klikom na kućicu pored paketa nakon što se naziv pojavi na popisu. Prateću dokumentaciju moguće je provjeriti na dva načina. Prvi je način kliknuti na **Help** i u polje za pretragu utipkati naziv paketa. Drugi način je upotreba naredbe `library(help="tidyverse")`. Postoje tisuće dostupnih paketa i njihov broj raste.

Grupirane pakete po temama (područjima primjene/metodama) moguće je naći na CRANE-u, The Comprehensive R Archive Networku (Wickham and Bryan 2023). Osim toga, neki su paketi deponirani na GitHubu i drugim repozitorijima, ali se tad najčešće radi o paketima u razvojnim fazama.

Ukratko, R je 'GNU S,' slobodno dostupan jezik i okruženje za statističko računanje i grafiku koje pruža širok spektar modeliranja, statističkih i grafičkih tehnika: linearno i nelinearno modeliranje, statistički testovi, analiza vremenskih serija, klasifikacija, klasteriranje itd.(Ripley (2001)). Naznačeni osnovni pojmovi detaljnije će se razložiti u nastavku.

2.8 Najčešće početničke greške

1. **Nerazumijevanje osnovnih koncepta**: Mnogi početnici preskaču osnove R-a, kao što su projekti, skripte, bilježnice, tipovi podataka, strukture podataka (vektori, liste, matrice, podatkovni okviri) i osnovne funkcije. Razumijevanje ovih temelja ključno je za efikasno korištenje R-a.

2. **Ignoriranje vektorskih operacija**: R je dizajniran s naglaskom na vektorske operacije, što omogućava obradu cijelih nizova podataka bez potrebe za petljama. Neiskorištavanje ovih mogućnosti čini kod sporijim i manje efikasnim.

3. **Prekomjerna upotreba petlji**: Iako su petlje (for, while) ponekad neophodne, u mnogim slučajevima one se mogu izbjeći koristeći vektorske operacije ili primjenom funkcija iz obitelji apply (npr. apply, lapply, sapply).

4. **Zanemarivanje postojećih funkcija i paketa**: R ima veliku zajednicu i gotovo da nema statističke metode ili algoritma koji već nije implementiran. Ponovno kreiranje funkcija može dovesti do gubitka vremena i veće vjerojatnosti pogreške.

5. **Nepravilno upravljanje radnim direktorijem**: Početnici često zanemaruju postavljanje radnog direktorija ili nepravilno koriste relativne i apsolutne putanje, što može dovesti do grešaka u učitavanju i spremanju podataka.

6. **Zanemarivanje vizualizacije podataka**: Vizualizacija je ključna za razumijevanje podataka. Zanemarivanje paketa poput ggplot2 za izradu grafikona često je propuštena prilika za bolje razumijevanje i prezentaciju rezultata.

7. **Ignoriranje upozorenja i poruka o greškama**: Poruke o greškama i upozorenja postoje s razlogom. Ignoriranje ovih poruka može dovesti do ozbiljnih problema u kodu koje može biti teško otkriti kasnije.

8. **Nepravilno upravljanje verzijama**: Ako se ne koriste sustavi za upravljanje verzijama poput Git-a, praćenje promjena i suradnja s drugima mogu biti otežane.

9. **Zanemarivanje važnosti ponavljanja analize**: Ako se ne dokumentiraju koraci analize, kasnije ponavljanje te analize, kao i provjera rezultata, mogu biti otežani.

Pitanja za ponavljanje

1. Koje su glavne prednosti korištenja R-a kao programskog jezika za statističku analizu?
2. Koji su osnovni koraci za instalaciju R-a i RStudija?
3. Što predstavlja CRAN, i zašto je važan za rad u R-u?
4. Kako u RStudiju možete otvoriti novu skriptu, a kako novu R Markdown datoteku?
5. Što se nalazi u prozoru Environment u RStudiju, i koja je njegova funkcija?
6. Koja je razlika između globalnog i lokalnog okruženja u R-u?
7. Koja je svrha radnog direktorija, i kako ga možete postaviti u R-u?
8. Koje su osnovne funkcije za rad s objektima u radnom okruženju (npr., prikaz, uklanjanje)?
9. Kako funkcionira korištenje paketa u R-u? Navedite primjer instalacije i aktivacije paketa.
10. Što su najčešće početničke greške pri korištenju R-a i RStudija?

3 Instalacija i aktivacija paketa

Ovo poglavlje se bavi instalacijom i aktivacijom paketa u R programskom jeziku. Objašnjava se kako paketi čine ključni dio R ekosustava, a detaljno se opisuje proces instalacije paketa kroz različite izvore poput CRAN-a i GitHuba. Poglavlje također pokriva praktične aspekte rada s paketima, uključujući njihovo učitavanje, ažuriranje te rješavanje najčešćih problema i pogrešaka koje se mogu javiti tijekom instalacije i korištenja paketa.

Paketi u R-u obično sadrže skup funkcija i kompiliranog koda, ali mogu uključivati i podatke te dokumentaciju koja pomaže korisnicima da ih učinkovito koriste. Standardna instalacija R-a dolazi s osnovnim paketima, ali postoji tisuće dodatnih paketa koji omogućuju specijalizirane analize, vizualizacije, povezivanje s bazama podataka i mnoge druge funkcionalnosti. Drugim riječima, paketi predstavljaju ključni dio ekosustava i omogućavaju pristup širokom rasponu funkcionalnosti, od osnovne statističke analize do napredne vizualizacije podataka i strojnog učenja.

3.1 Instalacija paketa

Paketi se obično instaliraju pomoću funkcije `install.packages("ime_paketa")`. Većina paketa dostupna je na CRAN-u (Comprehensive R Archive Network), ali paketi se mogu instalirati i s drugih izvora kao što su GitHub, Bioconductor itd.

CRAN pruža listu dostupnih paketa, često s opisima i uputama. Za specijalizirane pakete, Bioconductor i GitHub su također popularni izvori.

Za svaki paket postoji dokumentacija koja objašnjava njegovu upotrebu. Dostupna je pomoću `help(package="naziv_paketa")` ili `"?ime_funkcije"`. U nedoumicama, korisne su stranice kao što je Stack Overflow ili na primjer Towards Data Science.

Najčešće korišteni paketi u R-u pokrivaju širok spektar funkcionalnosti, uključujući analizu podataka, vizualizaciju, manipulaciju podataka, statističku analizu i izradu izvještaja. Neki od najpopularnijih paketa su:

- dplyr: Ovaj paket je koristan za manipulaciju podacima. Omogućuje lako i brzo sortiranje, filtriranje, grupiranje i sumiranje podataka (Yarberry and Yarberry 2021).
- ggplot2: Paket za vizualizaciju podataka koji omogućuje izradu složenih grafova i dijagrama koristeći gramatiku grafičkog dizajna (Wickham and Wickham 2016).
- tidyr: Koristi se za 'čišćenje' podataka, pomaže u preuređivanju podataka u čist i uredan format (Wickham and Wickham 2017).
- readr: Dio „tidyverse" skupa paketa, koristi se za brzo i efikasno učitavanje datoteka kao što su CSV, TSV i FWV (Wickham et al. 2023).
- tibble: Moderna verzija podatkovnog okvira u R-u, dio „tidyverse" skupa, olakšava rad s tabelarnim podacima (Müller et al. 2023).
- lubridate: Ovaj paket olakšava rad s datumima i vremenima u R-u, omogućujući jednostavnu manipulaciju i izračunavanje vremenskih intervala (Spinu, Grolemund, and Wickham 2021).
- stringr: Dio „tidyverse" skupa, koristi se za manipulaciju i obradu stringova (tekstualnih podataka) (Wickham 2019).

- linprog: koristi Simplex algoritam za rješavanje problema linearnog programiranja (Henningsen 2022)
- lpsolve:svestrani alat za linearno programiranje koji primarno koristi revidirani Simplex algoritam, ali uključuje i dodatne napredne tehnike te je optimiziran za brzinu Konis, Schwendinger, and Hornik (2023).
- nloptr: alat za rješavanje problema nelinearne optimizacije (Ypma et al. 2024).
- purrr: Omogućava funkcionalno programiranje u R-u, što pomaže u pisanju čistog i efikasnog koda (Wickham, Henry, and RStudio 2023).
- caret: Koristi se za strojno učenje, omogućuje jednostavnu izradu prediktivnih modela i njihovu validaciju (Kuhn et al. 2007).
- shiny: Omogućava izradu interaktivnih web aplikacija u R-u bez potrebe za temeljitim poznavanjem HTML-a, CSS-a ili JavaScripta (Chang et al. 2023).

Instalacija paketa općenito funkcionira na sljedeći način:

```
> install.packages("Naziv_paketa")
```

Instalacija paketa iz izvora koji nisu standardni CRAN repozitoriji, kao što je GitHub, zahtijeva korištenje alata kao što je paket devtools. Paket devtools omogućuje instalaciju paketa izravno iz izvora koda, što je posebno korisno za razvojne verzije paketa ili pakete koji još nisu dostupni na CRAN-u. Malo je vjerojatno da će ovo biti potrebno u ranijim fazama učenja R-a, no ako naiđete na takvu situaciju, možete pristupiti problemu na sljedeći način.

Na primjer, ako želite instalirati paket dostupan na GitHub-u, morate prvo instalirati paket devtools (Wickham et al. 2022) ako ga već nemate. Tada možete koristiti funkciju `install_github()` iz devtools da instalirate paket izravno s GitHuba. Trebat ćete navesti put do repozitorija na GitHubu, koji obično uključuje korisničko ime vlasnika repozitorija i naziv paketa. Primjer:

```
> install.packages("devtools")
> library(devtools)
>
> #ili
>
> devtools::install_github("korisnickoime/nazivpaketa")
>
> #Na primjer, ako želite instalirati razvojnu verziju paketa ggplot2:
> devtools::install_github("tidyverse/ggplot2")
```

Prilikom instalacije paketa s GitHuba, važno je biti svjestan da ti paketi možda nisu prošli sve standardne CRAN testove i provjere, pa stoga mogu biti nestabilni ili nespojivi s drugim paketima.

3.2 Učitavanje paketa

Nakon instalacije, paket se učitava u radnu sesiju pomoću `library(ime_paketa)` ili `require(ime_paketa)`. `library()` će dati grešku ako paket nije pronađen, dok `require()` daje upozorenje i vraća FALSE, što je korisno u uvjetnom programiranju.

```
> library(Naziv_paketa)
```

3.3 Ažuriranje paketa

Paketi se redovito ažuriraju, a korisnici mogu ažurirati svoje pakete pomoću funkcije `update.packages()`. Paketi često ovise o drugim paketima. Instaliranje i ažuriranje jednog paketa može povući i instalaciju ovisnih paketa. Pa tako, ako ažurirate jedan paket, često će biti nužno instalirati i povezane pakete.

```
> update.packages("Naziv_paketa")
```

3.4 Najčešće pogreške

Pri učitavanju paketa u R, korisnici mogu naići na nekoliko uobičajenih pogrešaka. Ovdje su neke od najčešćih grešaka koje se događaju prilikom učitavanja paketa i kako ih izbjeći:

- Nepostojanje paketa: Greška se javlja ako pozvani paket nije instaliran. R koristi pakete koji su instalirani u sustavu, pa ako pokušate učitati paket koji nije instaliran, dobit ćete poruku o grešci. Rješenje je jednostavno instalirati paket koristeći `install.packages("Naziv_paketa")` prije njegovog pozivanja s `library(Naziv_paketa)`.
- Pogrešan naziv paketa: Često se dogodi da korisnici pogrešno upišu naziv paketa, bilo zbog tipfelera, korištenja krivih velikih ili malih slova, ili zbog pogrešnog naziva. R je osjetljiv na veličinu slova, pa je važno točno upisati naziv paketa.
- Sukobi između paketa: Neki paketi mogu imati funkcije s istim imenima, što može dovesti do sukoba. Ako dva paketa koja su učitana u istoj sesiji imaju funkcije istih imena, R će koristiti funkciju iz paketa koji je posljednji učitan. Da biste izbjegli ovu vrstu problema, možete koristiti sintaksu `paket::funkcija()` kako biste jasno naveli koju funkciju želite koristiti.
- Zastarjele verzije paketa: Ponekad paketi koje želite koristiti zbog zastarjelosti nisu kompatibilni s vašom verzijom R-a ili s drugim paketima koje koristite. Uvijek je dobro redovito ažurirati R i sve instalirane pakete pomoću `update.packages()` kako biste smanjili ovaj rizik.
- Problem s ovisnostima: Neki paketi zahtijevaju da određeni drugi paketi budu prethodno instalirani. Ako ove ovisnosti nisu zadovoljene, instalacija ili učitavanje paketa može rezultirati greškom. Rješenje je pažljivo pročitati upute za instalaciju i osigurati da su sve potrebne ovisnosti instalirane.
- Korištenje paketa iz nestandardnih izvora: Instalacija paketa iz izvora kao što su GitHub zahtijeva dodatne korake, poput korištenja paketa devtools. Važno je slijediti upute i biti svjestan da paketi iz ovih izvora mogu biti u razvojnoj fazi i potencijalno nestabilni.

- Problemi s pravima pristupa: Prilikom instalacije paketa, osobito na višekorisničkim ili zaštićenim sustavima, može doći do grešaka zbog nedostatnih prava za instalaciju u globalne direktorije. U takvim slučajevima, korisnici mogu koristiti osobne biblioteke ili zatražiti pomoć od administratora sustava.

Pitanja za ponavljanje

1. Što su paketi u R-u, i zašto su važni za rad u ovom programskom jeziku?
2. Koja je standardna funkcija za instalaciju paketa iz CRAN repozitorija?
3. Kako se instaliraju paketi koji nisu dostupni na CRAN-u, na primjer, s GitHub-a?
4. Objasnite razliku između funkcija `library()` i `require()` prilikom učitavanja paketa.
5. Koja je svrha funkcije `update.packages()`, i kada je važno ažurirati pakete?
6. Koji su neki od najčešće korištenih paketa u R-u, i za koje vrste analiza ili zadataka se koriste?
7. Kako biste riješili problem ako dobijete grešku da paket nije pronađen prilikom učitavanja?
8. Što učiniti ako različiti paketi u R-u sadrže funkcije s istim imenima? Kako možete jasno naznačiti funkciju iz određenog paketa?
9. Zašto može doći do problema s pravima pristupa prilikom instalacije paketa, i kako se taj problem može riješiti?
10. Koje korake možete poduzeti ako instalirani paket ne radi zbog nekompatibilnosti s drugim paketima ili verzijom R-a?

4 Identifikatori i vrijednosti

4.1 Osnovni tipovi podataka

Poznavanje osnovnih tipova podataka predstavlja temeljnu vještinu za rad u R-u iz nekoliko razloga. Prvo, tip podataka određuje kako će R pohraniti i rukovati unesenim podacima, što direktno utječe na točnost i efikasnost analize. Ako, na primjer, numerički podatak greškom označimo kao tekst (*engl. character*), nećemo moći provoditi matematičke operacije nad njim.

Razumijevanje tipova podataka važno je za ispravno učitavanje podataka, gdje R mora znati treba li neki stupac tretirati kao brojeve, tekst ili datume. Također, ovo znanje omogućuje otkrivanje i ispravljanje grešaka.

Mnoge statističke funkcije zahtijevaju specifične tipove podataka, a manipulacija podacima često uključuje njihove transformacije. Bez razumijevanja osnovnih tipova podataka, mogli bismo doći do pogrešnih zaključaka ili bi analiza mogla biti značajno otežana zbog neočekivanog ponašanja funkcija uslijed unosa pogrešnog tipa podataka.

	Naziv	Primjer	Vrsta
1.	double	1, 2.33, 50.99, 3e10	realni brojevi
2.	integer	5l, -2, 5387l	cjeli brojevi
3.	character	slova, oznake, 50 %, pa}	znakovni tip
4.	logical	TRUE, FALSE, T, F	logički
5.	complex	1+5i, 5-2i	kompleksni brojevi
6.	raw	as.raw(11)	sirovi

Ova tablica prikazuje šest osnovnih tipova podataka u R-u, njihove primjere i kratke opise. Tip *double* koristi se za realne brojeve i najčešći je numerički tip u R-u, omogućujući precizne decimalne kalkulacije. *Integer* predstavlja cijele brojeve. Tip *character* obuhvaća sve tekstualne podatke, uključujući slova, oznake i brojeve koji se tretiraju kao tekst. *Logical* tip je posebno važan za uvjetno izvršavanje koda i pohranu *true/false* vrijednosti, dok *complex* tip omogućuje rad s kompleksnim brojevima koji su važni u određenim matematičkim i inženjerskim primjenama. *Raw* tip se rjeđe koristi i služi za pohranu sirovih (binarnih) podataka.

Provjera vrste podataka može se izvršiti pomoću `typeof()`, ili funkcijama **is** (`is.integer()`, `is.character()`, itd.) `typeof()` naredba povratno daje naziv tipa podataka (double, integer, itd.), dok je naredba koja sadrži **is** logička i povratno daje TRUE ili FALSE.

Tip podataka moguće je promijeniti funkcijom koja sadrži **as** (`as.integer()`, `as.double()`, `as.character()`, itd.). Ipak, preporučuje se ne koristiti promjenu tipa podataka prije nego se izvrši detaljni uvid u podatke. Naime, može se dogoditi (i događa se) da niz podataka sadrži pogrešno zapisane vrijednosti, na primjer, pojavljuje se „o“ na mjestu gdje bi trebala biti „0“, stoga je tip podataka identificiran kao character umjesto integer li double. Sama promjena tipa podataka neće riješiti ovaj problem, nego podatke treba provjeriti i urediti prije takve promjene.

Pitanja za ponavljanje

1. Zašto je važno poznavati osnovne tipove podataka u R-u?
2. Koji tip podataka koristimo za prikaz realnih brojeva, i kako se on zapisuje u R-u?
3. Što se događa ako podatak koji bi trebao biti broj unesemo kao tekst (character)?
4. Kako provjeriti tip podataka varijable u R-u?
5. Što radi funkcija `as.double()` i kada biste ju koristili?

4.2 Nazivi

Pri kreiranju podataka različitih struktura, potrebno im je dodijeliti nazive. Ponekad će to biti slova, na primjer, *a*, *b*, *x*, *y* i slično. No, ako se bavite s većim brojem varijabli, onda će im biti potrebno dodijeliti nazive vezane uz pojam. Pritom se preporučuje:

- izbjegavati dijakritike (č, ć, đ, dž, š i ž)
- naziv ne može sadržavati razmak
- umjesto razmaka, u nazivu ne koristiti crticu/povlaku (-), nego donju povlaku (_), na primjer: visina_cm
- alternativno, naziv se može zapisati kao jedna riječ, na primjer, VisinaCm ili visinacm
- druga alternativa je koristiti točke: visina.cm

Nadalje, R je osjetljiv na velika i mala slova. To znači da Visina_cm $\neq$ visina.cm. To ujedno znači da Visina.cm i visina.cm mogu biti dvije zasebne, različite varijable. No, važno je obratiti pozornost na to pri pozivanju varijabli, osobito u slučaju dobivanja poruka upozorenja kao što su *'Error: object 'Visina.cm' not found'* ili *'#> Error in eval(expr, envir, enclos): object 'Visina.cm' not found'*.

Također, poželjno je izbjegavati nazive koje su vezani uz već postojeće funkcije u R-u. Postoji nekoliko naziva koje novi korisnici R-a često pokušavaju koristiti za svoje varijable ili objekte, ne znajući da su to već postojeće funkcije u R-u. Najčešće se radi o intuitivnim imenima poput c, t, data, mean, summary, max, min, sum, fun, df, table, array, matrix, list ili vector. Ovi nazivi su zapravo *rezervirani* jer predstavljaju važne funkcije u R-u - `c()` je funkcija za kreiranje vektora, `t()` za transponiranje matrice, `mean()` za računanje prosjeka, i tako dalje. Kad pokušamo koristiti ova imena za vlastite objekte, to može dovesti do zbunjujućih grešaka i nepredvidljivog ponašanja programa. Zato je važno od početka usvojiti naviku davanja jedinstvenih, opisnih imena svojim objektima koja ne interferiraju s ugrađenim funkcijama R-a. Na primjer, umjesto *data* možemo koristiti *podaci* ili neki specifičniji naziv poput *anketa_podaci*, a umjesto *mean* možemo koristiti *prosjek*.

Pitanja za ponavljanje

1. Što bi bilo preporučljivo koristiti umjesto razmaka u nazivima varijabli?
2. Objasnite što znači da je R „osjetljiv na velika i mala slova" u kontekstu naziva varijabli.
3. Kako biste nazvali varijablu koja predstavlja težinu u kilogramima, a da se naziv pridržava preporučenih smjernica?
4. Zašto je važno izbjegavati nazive poput c, t, mean, i sum za varijable?
5. Kako možete provjeriti je li naziv koji želite koristiti već zauzet kao funkcija u R-u?

4.3 Kreiranje varijabli

Iako će se najčešće koristiti varijable iz učitanog podatkovnog okvira, korisno je znati da se varijable mogu i izraditi pridruživanjem vrijednosti. Na primjer, ako želimo kreirati varijablu a koja sadrži vrijednost 1, to je moguće napraviti na sljedeći način:

```
> a <- 1
> a
```

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

Iako je kreiranje varijabli korisno, podatke je češće nužno i važnije definirati s obzirom na njihovu strukturu. Sjetite se i napomene kod programskih paketa, često će programski paketi iziskivati točno određen tip i strukturu podataka.

No, varijable će češće sadržavati više opažanja, to jest podataka. Onda to može izgledati ovako:

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

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

Kreiranje varijabli je temeljna operacija u R-u koja omogućava pohranu podataka za kasniju upotrebu. Međutim, postoji nekoliko uobičajenih grešaka koje korisnici čine prilikom kreiranja varijabli:

- Upotreba nedozvoljenih znakova u imenima varijabli: R ne dopušta upotrebu određenih znakova u imenima varijabli, kao što su razmaci i znakovi interpunkcije (npr. !, ?, #, %, &). Osim toga, prvi znak ne može biti broj. Greška u korištenju ovih znakova može dovesti do sintaktičkih grešaka.
- Prepisivanje ugrađenih funkcija: Ako korisnik dodjeljuje varijabli naziv koji je već rezerviran za neku od R-ovih ugrađenih funkcija (npr. c, mean, sum), to može dovesti do neočekivanih rezultata ili grešaka u izvođenju koda. Uvijek je dobro provjeriti koristi li se ime koje je već rezervirano. To možete učiniti tako što ćete naziv započeti pisati u skripti ili *chunk-u* koda. Čim počnete pisati, RStudio nagađa koju funkciju želite upisati i nudi imena u lebdećem izborniku. Ako se u tom izborniku ponudi naziv kojeg ste namjeravali dati varijabli, odaberite drugi naziv.
- Varijable koje nisu prethodno inicijalizirane: Pristup varijablama koje nisu prethodno inicijalizirane ili definirane rezultira greškom (pozivanje varijable koja ne postoji). Pokušaj pristupa varijabli koja nije definirana generira poruku o grešci.
- Nejasna namjena varijabli zbog lošeg imenovanja: Iako ne generira tehničku pogrešku, neprecizno imenovanje varijabli (npr. korištenje imena poput x, temp, ili var1) može otežati razumijevanje koda. Takav će se pristup koristiti samo za pomoćne izračune koje korisnik nema namjeru ponavljati. Dobra je praksa koristiti opisne nazive koji jasno odražavaju namjenu varijable.

- Neispravno pretpostavljena tipizacija: Ponekad korisnici nenamjerno kreiraju pogrešan tip podatka varijable. Na primjer, pridruživanjem brojevnih vrijednosti navodnicima (`a <- "1"`) varijabla *a* postaje znakovni niz umjesto numeričke vrijednosti. To može dovesti do naknadnih grešaka u matematičkim operacijama i analizi podataka.

- Upotreba neodgovarajućeg pridruživanja: Iako su `<-` i `=` u većini slučajeva zamjenjivi pri pridruživanju vrijednosti, postoje situacije gdje je poželjno koristiti jedno umjesto drugog. Na primjer, `<-` se tradicionalno koristi za pridruživanje varijabli u globalnom prostoru imena, dok se = u pravilu koristi unutar funkcija za specifikaciju argumenata.

- Pogreška u vektorizaciji: Pri pridruživanju više vrijednosti varijabli bez korištenja funkcije `c()` može doći do greške. Ispravan način za kreiranje vektora je upotreba funkcije `c()`, kao što je prikazano u primjeru. To nas ujedno vodi do sljedeće teme i kreiranja vektora.

Pitanja za ponavljanje

1. Što će se dogoditi ako pokušate kreirati varijablu koristeći naziv koji sadrži razmak (npr. `moj podatak <- 5`)?
2. Ako napišete `a <- "10"`, koji je tip podataka varijable a, i što će se dogoditi ako pokušate izračunati `a + 5`?
3. Ako varijabla nije prethodno inicijalizirana, na što će R upozoriti prilikom njezinog pozivanja?
4. Što je bolje koristiti za pridruživanje varijabli, = ili <-, i zašto? Kada biste koristili jedno u odnosu na drugo?
5. Ako napišete `b <- c(3, 5, 7)`, što predstavlja ova varijabla b i kako biste dobili vrijednost na drugoj poziciji?
6. Koji znakovi nisu dozvoljeni pri imenovanju varijabli u R-u? Možete li koristiti znak # ili & u nazivu?
7. Što će se dogoditi ako koristite `c <- c(1, 2, 3)` za kreiranje varijable? Možete li predložiti drugačiji naziv za ovu varijablu kako bi se izbjegla zamjena?
8. Koja je razlika između varijable koja sadrži vrijednost `a <- 1` i varijable koja sadrži više vrijednosti, kao što je `b <- c(1, 2, 3)`?
9. Ako koristite `a <- c(1, 2, 3, 4, 5)`, kako biste ažurirali varijablu a tako da joj dodate broj 6 na kraj?
10. Kako biste provjerili postoji li naziv varijable koji je već rezerviran za funkciju u R-u?

5 Osnovni operatori i funkcije

5.1 Operatori u R-u

Ovo poglavlje predstavlja različite vrste operatora koji se koriste u R programskom jeziku, uključujući aritmetičke, logičke, usporedne, operatore za dodjelu, kontrolu tijeka, specifične operatore te operatore za funkcije. Kroz pregled svakog tipa operatora, prikazat će se prikladne oznake za manipulaciju podacima, izvođenje matematičkih operacija i kontrolu tijeka programa.

- Operatori za dodjelu:
- '<-' ili '=': standardni operatori za dodjelu
- '«-': globalna dodjela
- '->': dodjela s desna na lijevo

Primjeri:

```
> a <- 40
> b <- 10
> print(a)
```

```
## [1] 40
```

```
> print(b)
```

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

```
> a = 50
> b = 20
> print(a)
```

```
## [1] 50
```

```
> print(b)
```

```
## [1] 20
```

```
> moja_funkcija <- function() {
+   a <<- 60
+   b <<- 30
+ }
>
> moja_funkcija()
> print(a)
```

```
## [1] 60
```

```
> print(b)
```

```
## [1] 30
```

```
> 70 -> a
> 80 -> b
> print(a)
```

```
## [1] 70
```

```
> print(b)
```

```
## [1] 80
```

```
> a <- 40
> b <- 10
> print(a)
```

```
## [1] 40
```

```
> print(b)
```

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

- Aritmetički operatori:
- '+' (plus): zbrajanje
- '-' (minus): oduzimanje
- '*' (puta): množenje
- '/' (podijeljeno s): dijeljenje
- '^' ili '**': potenciranje
- '%%': moduo (ostatak pri dijeljenju)
- '%/%': cjelobrojno dijeljenje

Primjeri:

```
> 40 + 10
```

```
## [1] 50
```

```
> 40 - 10
```

```
## [1] 30
```

```
> 40 * 10
```

```
## [1] 400
```

```
> 40 / 10
```

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

```
> 40 ^ 10
```

```
## [1] 1.048576e+16
```

```
> 40 %% 10
```

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

```
> 40 %/% 10
```

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

- Logički operatori:
- '&' ili '&&': logički AND
- '|' ili '||': logički OR
- '!': logički NOT
- '!=': logički NOT EQUAL to
- Ovi operatori se koriste za usporedbu i testiranje uvjeta

Primjeri:

```
> result_and <- (a > 30) & (b < 20)
> print(result_and)
```

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

```
> result_and_short <- (a > 30) && (b < 5)
> print(result_and_short)
```

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

```
> result_or <- (a > 50) | (b < 20)
> print(result_or)
```

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

```
> result_or_short <- (a > 50) || (b < 5)
> print(result_or_short)
```

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

```
> result_not <- !(a < 50)
> print(result_not)
```

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

```
> result_not_equal <- (a != b)
> print(result_not_equal)
```

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

- Operatori za usporedbu:
- '<:' manje od
- '<=:' manje ili jednako
- '>:' veće od
- '>=:' veće ili jednako
- '==:' jednakost
- '!=:' nejednakost

Primjeri:

```
> result_less_than <- a < b
> print(result_less_than)
```

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

```
> result_less_equal <- a <= b
> print(result_less_equal)
```

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

```
> result_greater_than <- a > b
> print(result_greater_than)
```

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

```
> result_greater_equal <- a >= b
> print(result_greater_equal)
```

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

```
> result_equal <- a == b
> print(result_equal)
```

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

```
> result_not_equal <- a != b
> print(result_not_equal)
```

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

- Operatori za kontrolu tijeka:
- '%in%': operator pripadnosti, koristi se za provjeru nalazi li se element u skupu

Primjer:

```
> moj_set <- c(10, 20, 30, 40, 50)
> result_a_in_set <- a %in% moj_set
> print(result_a_in_set)
```

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

```
> result_b_in_set <- b %in% moj_set
> print(result_b_in_set)
```

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

```
> result_60_in_set <- 60 %in% moj_set
> print(result_60_in_set)
```

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

- Specifični operatori:
- '%*%': matrično množenje
- '%/%': cjelobrojno dijeljenje
- '%o%': vanjski produkt
- '%x%': Kroneckerov produkt
- ':': operator za stvaranje sekvenci

Primjeri:

```
> matrix1 <- matrix(c(a, b, b, a), nrow = 2)
> matrix2 <- matrix(c(b, a, a, b), nrow = 2)
> result_matrix_mult <- matrix1 %*% matrix2
> print(result_matrix_mult)
```

```
##      [,1] [,2]
## [1,]  800 1700
## [2,] 1700  800
```

```
> result_int_div <- a %/% b
> print(result_int_div)
```

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

```
> result_outer_product <- a %o% c(b, a)
> print(result_outer_product)
```

```
##      [,1] [,2]
## [1,]  400 1600
```

```
> result_kronecker_product <- a %x% matrix(c(b, a), nrow = 1)
> print(result_kronecker_product)
```

```
##      [,1] [,2]
## [1,]  400 1600
```

```
> result_sequence <- b:a
> print(result_sequence)
```

```
##  [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
## [26] 35 36 37 38 39 40
```

- Operatori za funkcije:
- '%>%': operator pipe, popularan u dplyr i tidyverse paketima, koristi se za lančanje naredbi
- '|>': alternativni zapis operatora pipe, koristi se u kontekstu „onda/ then".

Primjeri:

```
> library(dplyr)
>
> result_pipe <- a %>%
+   `*`(2) %>%   # množenje s 2
+   `+`(b) %>%   # pribrajanje b
+   `^`(2)       # kvadriranje rezultata
> print(result_pipe)  # Rezultat operacije ((a * 2) + b)^2
```

```
## [1] 8100
```

```
> divide_by_2 <- function(x) x / 2
> add_b <- function(x) x + b
>
> result_pipe_then <- a |>
+   divide_by_2() |>   # Poziv funkcije dijeljenja s 2
+   add_b()            # Poziv funkcije pribrajanja b
>
> print(result_pipe_then)  # Rezultat (a / 2) + b
```

```
## [1] 30
```

Ovi operatori su ključni u svakodnevnom radu s R-om, omogućujući manipulaciju podacima, izvođenje matematičkih operacija i kontrolu tijeka programa. Razumijevanje i pravilna upotreba ovih operatora su osnova za efikasno programiranje u R-u.

Pitanja za ponavljanje

1. Ako je `a <- 50` i `b <- 20`, što će biti rezultat izraza `a %% b`?
2. Ako definirate `c <- 3` i koristite naredbu `c <- c + 2`, koja će biti nova vrijednost varijable c?
3. Ako imate vektor `x <- c(5, 10, 15)` i naredbu `x %in% c(10, 15, 20)`, što će biti rezultat?
4. Što će naredba `40 %/% 6` vratiti, i koja je razlika u odnosu na `40 / 6`?
5. Ako napišete `a <- 5`; `b <- 3`; `a > b & b < 5`, hoće li rezultat biti TRUE ili FALSE?
6. Ako koristite operator %>% u R-u, objasnite što radi sljedeći izraz: `c(1, 2, 3) %>% sum() %>% sqrt()`.
7. Što će izraz `5 ^ 2` vratiti? Koji operator u R-u koristite za potenciranje?
8. Ako koristite operator ! na logičkoj vrijednosti TRUE, koji će biti rezultat?
9. Ako imate naredbu `seq(1, 10, by = 2)`, koju će sekvencu brojeva vratiti?
10. Ako koristite `a <- 20`; `b <- 4`, što će naredba `a / b == 5` vratiti kao rezultat?

5.2 Osnovne funkcije

Ovo poglavlje predstavlja osnovne funkcije u R programskom jeziku. Fokus je na četiri osnovne funkcije, a svaka od njih je detaljno objašnjena s opisom parametara i praktičnim primjerima primjene.

R ima veliku bazu funkcija. Pri pozivanju funkcije, koristi se naziv funkcije i potom se upisuju zagrade. U zagradi će se naći argumenti ili parametri funkcije. Svaka funkcija ima drugačije definirane argumente ili parametre. Pri susretu s novom funkcijom, najbolje je koristiti **Help** u donjem desnom prozoru u RStudiju, koji nudi pretraživanje dokumentacije i opis funkcija.

Na primjer, za nekoliko češće korištenih funkcija, ovdje je navedeno prvih nekoliko funkcija koje će se koristiti u nastavku:

c {base}

- `c()`
- ovo je generička funkcija za kombiniranje elemenata
- zadano (*engl. default*):

  `c(..., recursive = FALSE, use.names = TRUE)`

  gdje

  - '...' označava elemente koji će biti povezani;
  - recursive je logički parametar kojem se mogu pridružiti vrijednosti TRUE ili FALSE, pri čemu FALSE postavka ne mijenja zapis zadanih elemenata, a TRUE rezultira rekurzivnim preuzimanjem elemenata kroz popis i kombiniranjem elemenata u vektor
  - use.names je logički parametar kojem se mogu pridružiti vrijednosti TRUE ili FALSE, pri čemu TRUE označava da nazivi trebaju biti sačuvani

- primjeri (navedeni i na stranici Help pri pretrazi za c()):

  - `x <- 1:4`
    `c(list(A = c(B = 1)), recursive = TRUE)`
    `c(list(A = c(B = 1, C = 2), B = c(E = 7)), recursive = TRUE)`

```
> x <- 1:4
> c(list(A = c(B = 1)), recursive = TRUE)
> c(list(A = c(B = 1, C = 2), B = c(E = 7)), recursive = TRUE)
```

Na primjer: žele se analizirati godišnje temperature za nekoliko gradova i trebate kreirati vektor s tim temperaturama.

Kako koristiti: koristi se funkcija `c()` za stvaranje vektora koji sadrži prosječne godišnje temperature za svaki grad.

```
> temperature <- c(16, 18, 20, 15, 17)
```

Dakle, elementi 16, 18, 20, 15, 17 su kombinirani i pridruženi vektoru, odnosno pohranjeni pod nazvom temperature. U vektoru temperature, 16 se nalazi na prvom mjestu, 18 na drugom, 20 na trećem, itd. Njihov je poredak određen zapisanim redosljedom.

print {base}

- print()
- ovo je generička funkcija za ispis argumenata
- automatski prepoznaje tip argumenta koji se ispisuje te će sukladno tome nuditi različite parametre koji se mogu podesiti
- za ispis faktora:
  - `print(x, quote = FALSE, max.levels = NULL, width = getOption("width"), ...)`
- za ispis tablice:
  - ```
    print(x, digits = getOption("digits"), quote = FALSE,
    na.print = "", zero.print = "0",
    right = is.numeric(x) || is.complex(x),
    justify = "none", ...)
    ```
- za ispis funkcije:
  - `print(x, useSource = TRUE, ...)`

  gdje
  - *x* označava objekt koji se ispisuje
  - *'...'* označava dodatne argumente koji se pridružuju funkciji
  - *quote* je logički parametar koji označava trebaju li se znakovi ispisivati koristeći navodne znakove
  - *max.levels* - cjelobrojna vrijednost, označava koliko razina faktora treba ispisati
  - *digits* - označava najmanji broj značajnih znamenki
  - *na.print* - treba li ispisati NA vrijednosti
  - *zero.print* - koliko nula treba ispisati
  - *right* - poravnanje ispisa

```
> print("Dobar dan!")
> print(x)
```

Na primjer: nakon analize podataka, želite ispisati rezultate na ekran kako biste ih mogli pregledati ili podijeliti s kolegama.

Kako koristiti: upotrijebite `print()` za ispisivanje rezultata, na primjer, ispisa ranije kreiranih prosječnih temperatura.
```

```
> print(temperature)
```

```
## [1] 16 18 20 15 17
```

seq {base}

- `seq()`
- generiranje pravilnih sekvenci brojeva
  - `seq.int(from, to, by, length.out, along.with, ...)`
    `seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),`
    `length.out = NULL, along.with = NULL, ...)`
  - *from* - označava početnu vrijednost u nizu
  - *to* - označava posljednju vrijednost u nizu
  - *by* - označava povećanje u nizu brojeva
- na primjer, seq(1,15,3) rezultirat će nizom brojeva: 1, 4, 7, 10, 13

```
> seq(1, 15, 3)
```

Na primjer: planirate niz eksperimenata koji se trebaju odvijati u redovitim intervalima tijekom godine, i trebate generirati sekvencu datuma za svaki eksperiment.

Kako koristiti: upotrijebite `seq()` za stvaranje niza ili sekvence brojeva koji predstavljaju, na primjer, svaki deseti dan u godini.

```
> dani <- seq(from = 1, to = 365, by = 10)
> dani
```

```
##  [1]   1  11  21  31  41  51  61  71  81  91 101 111 121 131 141 151 161 171 181
## [20] 191 201 211 221 231 241 251 261 271 281 291 301 311 321 331 341 351 361
```

rep {base}

- `rep()`
- replikacija/ ponavljanje vrijednosti određeni broj puta
  - `rep(x, ...)`

  `rep.int(x, times)`

  `rep_len(x, length.out)`

  gdje

- x je vektor bilo koje vrste ili faktor
- *times* je cjelobrojna vrijednost koja naznačava koliko se puta x ponavlja
- *length.out* - je nenegativna cjelobrojna vrijednost koja označava željenu duljinu vektora
- *each* - nenegativna cjelobrojna vrijednost koja označava koliko se puta *svaki* element naveden u x-u ponavlja

```
> rep(c(0, 1), 8)
```

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

Na primjer: pretpostavimo da razvijate algoritam za obradu nizova brojeva i želite testirati kako se vaš algoritam ponaša s ponavljajućim uzorcima. Potreban vam je jednostavan niz koji se sastoji od uzorka 0, 1, 1 koji se ponavlja nekoliko puta.

Kako koristiti: možete koristiti `rep()` za kreiranje željenog niza koji ponavlja uzorak `c(0, 1, 1)` pet puta. Tako dobiveni vektor sadrži ukupno 15 elemenata jer je početni uzorak `c(0, 1, 1)` koji se sastoji od tri elementa ponovljen pet puta.

```
> sequence <- rep(c(0, 1, 1), 5)
> sequence
```

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

Pitanja za ponavljanje

1. Koja je svrha funkcije `c()` u R-u, i kako se koristi za stvaranje vektora?
2. Što radi argument recursive u funkciji `c()`?
3. Kada biste koristili funkciju `print()` i koja je njena osnovna svrha?
4. Koji parametar u funkciji `print()` omogućava prikazivanje vrijednosti bez navodnika?
5. Kako funkcionira funkcija `seq()` i čemu služe argumenti from, to, i by?
6. Koji je rezultat naredbe `seq(5, 25, by = 5)`?
7. Kada biste koristili funkciju `rep()` i što čini argument times?
8. Ako želite generirati sekvencu koja ponavlja niz `c("A", "B")` deset puta, kako biste to postigli koristeći rep()?
9. Koji je rezultat naredbe `rep(c(1, 5, 10), 3)`?
10. Što biste trebali učiniti ako želite saznati više o parametrima funkcije u R-u, kao što su `print()` ili `seq()`?

Ovih nekoliko primjera upotrijebit će se već u sljedećem poglavlju. Toplo vam preporučujemo i potičemo vas na samostalno korištenje *Help* stranica dostupnih putem RStudija.

6 Jednodimenzionalni tip podataka

6.1 Kreiranje vektora

U ovom poglavlju objašnjava se kreiranje vektora. Poglavlje pokriva različite načine kreiranja vektora, od ručnog unosa elemenata do korištenja ugrađenih funkcija za generiranje sekvenci, ponavljanja vrijednosti i kreiranja slučajnih brojeva temeljenih na teorijskim distribucijama.

Vektor je uređeni niz koji nastaje pridruživanjem elemenata. *Uređeni niz* znači da je redno mjesto elementa fiksno i da se pozivom određenog rednog mjesta element može identificirati. Nadalje, važno je znati da elementi moraju biti *istog tipa*. Ako elementi nisu istog tipa, tada se radi o listi (više o tome kasnije), a ne vektoru.

Jedna od ključnih karakteristika vektora u R-u je da svi elementi unutar vektora moraju biti istog tipa. Ako se pokuša kombinirati elemente različitih tipova, R će implicitno promijeniti tip podataka elemenata u vektoru kako bi osigurao homogenost. Na primjer, kombiniranje numeričkih i znakovnih tipova rezultirat će vektorom znakovnog tipa, što može dovesti do neočekivanih rezultata ili grešaka u kasnijim analizama.

U R-u, vektor je jednodimenzionalni tip podataka, ali zbog fleksibilnosti R-ovog sustava, različita tumačenja mogu stvoriti konfuziju. Dakle, vektor u R-u je jednostavan niz elemenata iste vrste (npr. numerički, logički, karakterni) i nema dodatne dimenzije osim dužine. Svaki element ima indeks (poziciju) koji se koristi za pristup elementima. Ipak, R omogućuje pridruživanje atributa dimenzija (`dim()`) vektoru, što ga pretvara u n-dimenzionalni niz, ali tad vektor postaje *matrica* ili *array*. U strogo tehničkom smislu, vektor u R-u je jednodimenzionalan. Dodavanjem dimenzija (`dim()` atributa) vektor u R-u se preoblikuje i postaje matrica (za 2 dimenzije) ili polje (*array*, za 3 ili više dimenzije), ali njegova osnovna struktura u memoriji ostaje vektor. Dodavanje atributa dimenzije zapravo mijenja način na koji se interpretira i pristupa elementima. No, time ćemo se baviti u poglavlju o višedimenzionalnim tipovima podataka.

6.1.1 Kreiranje vektora koristeći izravan upis elemenata

Elementi se pridružuju vektoru koristeći funkciju combine, `c()`. Primjer kreiranja vektora:

```
> a <- c(1, 2, 3, 4, 5)
> b <- c(5, 10, 15, 20, 25)
> c <- c("Pula", "Rijeka", "Zagreb", "Osijek", "Split")
> d <- c("Iva", "Ana", "Maja", "Marko", "Ivan")
> e <- c(TRUE, TRUE, TRUE, FALSE, FALSE)
```

Vektori koji predstavljaju kvalitativne varijable ili kategorijske podatke (kao što su vektori c ili e) u R-u često se koriste kao faktori kada je potrebno analizirati ili modelirati kategorijske varijable. Međutim, sami faktori rijetko su korisni; obično se koriste u kombinaciji s drugim numeričkim ili logičkim podacima kako bi se omogućila dublja analiza ili vizualizacija povezanosti između različitih vrsta podataka. Zbog toga ćemo se faktorima detaljnije baviti tek na kraju sljedećeg poglavlja, nakon što savladamo višedimenzionalne tipove podataka.

Ispis vektora vršimo upisom naziva vektora ili pomoću funkcije `print()`.

```
> print(a)
```

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

No, ispis je moguć i samo pozivom naziva vektora:

```
> b
```

```
## [1]  5 10 15 20 25
```

ili

```
> c
```

```
## [1] "Pula"   "Rijeka" "Zagreb" "Osijek" "Split"
```

Također, možemo opaziti da neće biti razlike u načinu ispisa s obzirom na to koristi li se `print()` ili samo naziv.

```
> print(d)
```

```
## [1] "Iva"   "Ana"   "Maja"  "Marko" "Ivan"
```

```
> e
```

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

6.1.2 Vektor indeksa

Vektor koji sadrži redoslijedne cijele brojeve kao elemente naziva se i vektor indeksa, zbog toga što vrijednosti elemenata odgovaraju rednim mjestima. Takvi se vektori ponekad koriste kao pomoćna radnja pri izračunima.

```
> f <- 1:100
> f
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

6.1.3 Kreiranje vektora cjelobrojnih vrijednosti u zadanom intervalu

Ponekad je potrebno kreirati duži vektor, pa čak i uz neke specifične karakteristike te bi ručni upis bio prezahtjevan. Tada su sljedeće mogućnosti vrlo korisne.

```
> f <- 100:200
> f
```

```
##   [1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
##  [19] 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
##  [37] 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
##  [55] 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
##  [73] 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
##  [91] 190 191 192 193 194 195 196 197 198 199 200
```

Prethodnom naredbom kreira se vektor koji sadrži sve cjelobrojne vrijednosti između 100 i 200, uključujući 100 i 200. Na takav način mogu se kreirati vektori koji će sadržavati cijele brojeve u bilo kojem rasponu.

6.1.4 Kreiranje sekvencijskih vektora

```
> g<-seq(1, 200, by=3)
> g
```

```
##  [1]   1   4   7  10  13  16  19  22  25  28  31  34  37  40  43  46  49  52  55
## [20]  58  61  64  67  70  73  76  79  82  85  88  91  94  97 100 103 106 109 112
## [39] 115 118 121 124 127 130 133 136 139 142 145 148 151 154 157 160 163 166 169
## [58] 172 175 178 181 184 187 190 193 196 199
```

Prethodnom funkcijom generiran je vektor g, kojemu je prva vrijednost 1, a svaka sljedeća vrijednost je treća po redu, do 200. Promjenom parametara može se kontrolirati izlaz. Na primjer, `seq(5,10,2)` rezultirat će vektorom koji će sadržavati vrijednosti 5, 7 i 9.

6.1.5 Kreiranje vektora s ponovljenim vrijednostima

```
> h<-rep(c(1, 5, 10), times=2)
> h
```

```
## [1]  1  5 10  1  5 10
```

Prethodnom funkcijom kreiran je vektor temeljem replikacije zadanog vektora `c(1, 5, 10)`, na način da se ponavlja dva puta. Umjesto `times=2`, moguće je koristiti, na primjer, `each=2`, pri čemu se svaka vrijednost ponavlja dva puta (1, 1, 5, 5, 10, 10), ili npr. `length.out=7`, pri čemu

kreira vektor duljine 7 na način da se niz vrijednosti ponavlja dok se ne dosegne sedam elemenata. Isprobajte sami.

U situacijama u kojima je potrebno kreirati vektor koji sadrži elemente koji se ponavljaju, to se može učiniti koristeći naredbu `rep()`. Naredba `rep()` ima dva argumenta. Prvi argument je broj ili znak koji se ponavlja, a drugi označava koliko će se puta taj broj ili znak ponoviti.

```
> m <- c(rep(2, 4), rep(0, 2), rep(5, 3))
> m
```

```
## [1] 2 2 2 2 0 0 5 5 5
```

Ovo može biti korisno, na primjer, u linearnom programiranju gdje je potrebno definirati vektor smjera ograničenja u linearnom programu.

```
> m <- c(rep("<=", 3), rep(">=", 4), "=")
> m
```

```
## [1] "<=" "<=" "<=" ">=" ">=" ">=" ">=" "="
```

6.1.6 Kreiranje vektora koristeći teorijske distribucije vjerojatnosti/ generator slučajnih brojeva

U nekim analizama, bit će potrebno kreirati vektore temeljem teorijskih distribucija vjerojatnosti. U paketu **stats** koji je dio osnovnog R okruženja, za kreiranje vektora vezanih uz kreiranje slučajnih brojeva uz pomoć teorijskih distribucija vjerojatnosti. `rnorm()` je generator slučajnih brojeva baziran na normalnoj distribuciji i jedan je od najčešće korištenih generatora slučajnih brojeva.

Na primjer, ako se želi kreirati vektor koji će sadržavati 100 elemenata koji se ravnaju prema normalnoj distribuciji s prosjekom 55 i standardnom devijacijom 5, to je moguće učiniti na sljedeći način:

```
> norm <- rnorm(1000, 55, 5)
> glimpse(norm)
```

```
##  num [1:1000] 53.2 58.7 47.6 52.6 54.8 ...
```

```
> mean(norm)
```

```
## [1] 54.62767
```

```
> sd(norm)
```

```
## [1] 4.977531
```

Nalik ovom primjeru, moguće je kreirati vektore koji se ravnaju i prema drugim teorijskim distribucijama. Među najčešće korištenima su `runif()`, koji funkcionira kao generator slučajnih brojeva baziran na uniformnoj distribuciji, `rbinom()` na binomnoj distribuciji, `rpois()` na Poissonovoj distribuciji, `rchisq()` na Hi-kvadrat distribuciji, `rf()` na F-distribuciji, `rbeta()` na beta distribuciji, `rgamma()` na gamma distribuciji. Pri korištenju funkcija kao što su `runif()` ili `rnorm()`, važno je razumjeti distribuciju iz koje se uzorkuje.

U situacijama u kojima vektor odražava vjerojatnosti, često će se koristiti uniformna distribucija pri kreiranju vektora. Pritom će biti potrebno definirati minimum i maksimum raspona u kojem se generiraju vrijednosti, a ako se žele generirati vrijednosti koje predstavljaju vjerojatnosti, onda će biti $min = 0$ i $max = 1$. Osim toga, potrebno je odrediti koliko je brojeva potrebno kreirati.

```
> unif <-runif(50, 0, 1)
> unif
```

```
##  [1] 0.62234688 0.25175609 0.17974466 0.18142932 0.08232822 0.26832344
##  [7] 0.37248125 0.51255263 0.58447847 0.59491200 0.15436787 0.23821250
## [13] 0.77413045 0.31407772 0.68766259 0.55507119 0.36478423 0.31379010
## [19] 0.70174296 0.85974615 0.86279269 0.84729329 0.95296711 0.94531155
## [25] 0.59009455 0.26384831 0.22138744 0.56896259 0.42840116 0.88557444
## [31] 0.37123330 0.27719774 0.49834838 0.20283707 0.70882792 0.15983155
## [37] 0.22161969 0.48022264 0.65865197 0.25103358 0.48658351 0.15863513
## [43] 0.81580084 0.40352108 0.48231828 0.63021093 0.26561176 0.37655649
## [49] 0.44098212 0.24049263
```

6.2 Rad s vektorima

U ovom poglavlju objašnjavaju se osnovne operacije s vektorima u R-u, uključujući spajanje, sortiranje i manipulaciju elemenata, kao i izračun osnovnih statističkih pokazatelja. Također se uvode koncepti poput dodavanja atributa, recikliranja elemenata i logičkih upita, omogućujući korisnicima fleksibilnu manipulaciju podacima. Na kraju, poglavlje daje praktične primjere za usporedbu i obradu podataka unutar vektora, što je ključno za efikasnu analizu.

6.2.1 Spajanje vektora

Vektori se mogu spojiti, a rezultat će predstavljati jedan niz podataka, pri čemu redoslijed vektora uvjetuje redoslijed podataka u nizu. Za spajanje vektora također se koristi `c()`.

```
> i <- c(a, b)
> i
```

```
##  [1]  1  2  3  4  5  5 10 15 20 25
```

```
> j <- c(b, a)
> j
```

```
##  [1]  5 10 15 20 25  1  2  3  4  5
```

Obratite pozornost na to da vektori i i j nisu jednaki. Također, vektori se mogu sastojati od elemenata tipa character, na primjer:

```
> k <- c(c, d)
> k
```

```
##  [1] "Pula"   "Rijeka" "Zagreb" "Osijek" "Split"  "Iva"    "Ana"    "Maja"
##  [9] "Marko"  "Ivan"
```

6.2.2 Sortiranje elemenata vektora

U specifičnim situacijama u kojima nam je važno znati na kojim se pozicijama nalaze elementi s obzirom na svoju vrijednost, može pomoći funkcija `order()`. Ova funkcija neće posložiti elemente redom (sjetite se da je vektor uređeni niz podataka), nego će vratiti **niz rednih mjesta** koje bi elementi trebali zauzeti ako bi bili poredani po veličini.

```
> ord <- order(j)
> ord
```

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

Ako je elemente vektora potrebno sortirati, koristi se funkcija `sort()`. Funkcija sort učinit će ono što njezin naziv govori, sortirat će podatke. Vodite računa o tome da, ako upišete npr. `sort(j)`, elementi vektora j bit će ispisani redoslijedno, ali samo ispisani - neće biti sačuvani. Ako želite sačuvati sortirane elemente vektora, onda morate te elemente pripisati novom objektu, tj. vektoru (u ovom slučaju to je ord). Sličan se pristup koristi svugdje u ovoj lekciji. Ipak, vidjet ćete naknadno, neće biti potrebno uvijek koristiti ovaj pristup, a ponekad je i nepoželjno zbog kreiranja velikog broja varijabli s kojima se kasnije teško snaći.

```
> j_sort <- sort(j)
> j_sort
```

```
##  [1]  1  2  3  4  5  5 10 15 20 25
```

6.2.3 Tip vektora

Za ispitivanje **tipa vektora** također se može koristiti `typeof()`. U slučaju kombinacije elemenata različitih tipova, tip vektora preuzet će tzv. „jači" tip podataka, pri čemu je odnos tipova podataka:

$$logical \rightarrow integer \rightarrow double \rightarrow character$$

Što to znači? Na primjer, ako niz podataka sadrži makar jednu character vrijednost, tada će tip tog niza podataka poprimiti tip character.

```
> typeof(a)
```

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

```
> typeof(k)
```

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

Vektor a je tip **double**, a vektor k je **character**. Njihovim spajanjem dobiva se vektor koji sadrži brojeve i znakove, ali će takav vektor (y) biti identificiran kao tip **character**, a ne *double*.

```
> y <- c(a, k)
> typeof(y)
```

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

6.2.4 Dodavanje atributa

Pod **dodavanjem atributa** najčešće se misli na dodavanje naziva stupcima i recima (zaglavlja i predstupci u tablicama).

Na primjer, ako je zadan vektor

```
> l <- c("Poslovna analitika", 6, 2, 1, 1, "T")
> length(l)
```

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

elementima vektora mogu se pridružiti nazivi:

```
> names(l) <- c("kolegij", "ECTS", "Sati predavanja",
+               "Sati vježbi", "Sati seminara", "Obavezan")
```

provjerite kako sad izgleda vektor l.

```
> l
```

```
##              kolegij                ECTS      Sati predavanja
## "Poslovna analitika"                 "6"                   "2"
##          Sati vježbi       Sati seminara             Obavezan
##                  "1"                 "1"                   "T"
```

Ako se osim naziva dodaju bilo kakvi dodatni atributi, tada više neće biti prepoznat kao vektor (ako provjeravate npr. funkcijom `is.vector()`). Ipak, moguće je dodavati i dodatne atribute, ako za to postoji opravdani razlog i ako za daljnju analizu nije nužno da niz bude prepoznat kao vektor. Tada se to čini funkcijom `attr()`.

```
> attr(l, "studij") <- "Informatički menadžment"
> l
```

```
##              kolegij                ECTS      Sati predavanja
## "Poslovna analitika"                 "6"                   "2"
##          Sati vježbi       Sati seminara             Obavezan
##                  "1"                 "1"                   "T"
## attr(,"studij")
## [1] "Informatički menadžment"
```

6.2.5 Duljina vektora

Duljina vektora ispituje se pomoću `length()`. Ako ispitate duljinu vektora *l* prije i nakon dodavanja naziva, vidjet ćete da se duljina vektora nije promijenila.

```
> length(l)
```

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

6.2.6 Odabir elemenata vektora

Također, ponekad je potrebno odabrati pojedini element vektora, uobičajeno se to čini s uglatim zagradama, `[ ]`. Na primjer:

```
> a[1]
```

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

selektira element koji se nalazi na prvom mjestu u vektoru *a*, ili na primjer,

```
> i[7]
```

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

selektira element koji se nalazi na 7 mjestu u vektoru *i*.

Pokušaj pristupa elementu vektora koji ne postoji, kao što je `i[100]` za vektor *i*, rezultirat će ispisom NA (*engl. Not Available*) umjesto greške.

```
> i[100]
```

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

Moguće je odabrati i nekoliko elemenata istovremeno, na primjer:

```
> j[c(2, 5, 8)]
```

```
## [1] 10 25  3
```

odnosno elemente na drugom, petom i osmom mjestu u vektoru *j*. Nadalje,

```
> j[3:7]
```

```
## [1] 15 20 25  1  2
```

odabire sve elemente koji se nalaze na mjestima od trećeg do sedmog, uključujući elemente na trećem i sedmom mjestu.

Osim toga, može se koristiti i funkcija `seq()`, na primjer:

```
> j[seq(1, 10, by=2)]
```

```
## [1]  5 15 25  2  4
```

Izraz `j[seq(1, 10, by=2)]` odabire elemente iz vektora *j* koristeći sekvencu indeksa generiranu funkcijom `seq(1, 10, by=2)`. Ova funkcija stvara sekvencu brojeva od 1 do 10 u koracima od 2 (tj. 1, 3, 5, 7, 9), pa izraz dohvaća elemente *j* s tim indeksima, birajući svaki drugi element unutar prvih 10 elemenata vektora.

6.2.7 Zamjena i brisanje elemenata vektora

Osim provjere, to jest, odabira pojedinih elemenata vektora, ponekad je potrebno **zamijeniti** određene **vrijednosti**. Na primjer, neka je zadan vektor m,

```
> m <- c(2, 20, 3, 5, "o", 6, "k", 10, 35, 22)
> typeof(m)
```

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

Vektor m je trenutno tipa character, možete provjeriti i s `is.character(m)`. Ako je za daljnju obradu potrebno da vektor bude definiran kao integer ili double, može se uočiti da elementi „o" i „k" priječe u tome da vektor bude tako identificiran. Može se pretpostaviti da je umjesto „o" trebala biti zapisana nula, pa se može izvršiti **zamjena elementa**:

```
> m[5] <- 0
> m
```

```
##  [1] "2"  "20" "3"  "5"  "0"  "6"  "k"  "10" "35" "22"
```

Za element „k" ne može se pretpostaviti koji je broj trebao zamijeniti, pa je jedna od mogućnosti **brisanje elementa**:

```
> m <- m[-7]
> m
```

```
## [1] "2"  "20" "3"  "5"  "0"  "6"  "10" "35" "22"
```

Drugi način brisanja elementa je `m[[7]] <- NULL`. Obratite pozornost da se nakon ove operacije mijenja duljina vektora, `length(m)`.

6.2.8 Operacije na vektorima

Na vektorima je moguće vršiti određene **operacije**, na primjer:

```
> #vektor a zabilježen je kao objekt i nalazi se u gornjem desnom prozoru,
> #Environment. Tamo možete provjeriti koje elemente sadrži vektor a,
> #kako bi popratili sljedeći niz operacija.
>
> a * 3
```

```
## [1]  3  6  9 12 15
```

Zbrajanje elementa 5 s vektorom a rezultirat će uvećanjem svakog elementa vektora a za 5.

```
> a + 5
```

```
## [1]  6  7  8  9 10
```

Kvadriranje vektora rezultirat će vektorom koji sadrži kvadrate elemenata vektora a.

```
> a^2
```

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

Moguće je izvršiti i kombinaciju operacija. Na primjer, $a \cdot 2 + 3$ rezultirat će vektorom u kojem je svaki element vektora a pomnožen s dva i uvećan za tri.

```
> a * 2 + 3
```

```
## [1]  5  7  9 11 13
```

R će poštivati prioritete računskih operacija, pa će tako u sljedećem primjeru prvo riješiti zagradu, a potom rezultat pomnožiti sa svakim elementom vektora a.

```
> a * (2 + 3)
```

```
## [1]  5 10 15 20 25
```

Moguće je vektore množiti međusobno. No tad treba voditi računa o tome da će se izmnožiti samo elementi na pripadajućim pozicijama. Dakle, prvi element u vektoru a s prvim elementom vektora b, pa drugi element vektora a s drugim elementom vektora b, itd.

```
> a * b
```

```
## [1]   5  20  45  80 125
```

Na sličan će se način vektori zbrajati. Dakle, zbrajanje vektora je moguće, a rezultat će biti novi vektor koji sadrži elemente zbroja vrijednosti elemenata a i b na odgovarajućim pozicijama.

```
> a + b
```

```
## [1]  6 12 18 24 30
```

6.2.9 Izračun pokazatelja deskriptivne statistike vektora

Za vektore je moguće izračunati i neke osnovne statističke podatke, koristeći `fivenum()` (Tuckeyevih pet brojeva), `sum()`, `min()`, `max()`, `mean()`.

Tuckeyevih pet brojeva su minimum, prvi kvartil, medijan, treći kvartil i maksimum.

```
> fivenum(j)
```

```
## [1]  1  3  5 15 25
```

Minimum se može utvrditi i pozivom naredbe `min()`.

```
> min(j)
```

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

Maksimalna vrijednost u vektoru može se utvrditi i pozivom naredbe `max()`.

```
> max(j)
```

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

Naredba `mean()` utvrđuje aritmetičku sredinu vektora.

```
> mean(j)
```

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

Naredba `median()` utvrđuje medijalnu vrijednost u nizu (medijan).

```
> median(j)
```

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

Također, možemo utvrditi i total tako što ćemo zbrojiti sve vrijednosti u nizu. Drugim riječima, vektore možemo tretirati kao varijable.

```
> sum(j)
```

```
## [1] 90
```

Naredba `summary()` uz Tuckeyevih pet brojeva uključuje i prosjek.

```
> summary(j)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.25    5.00    9.00   13.75   25.00
```

Također, za ranije kreirani vektor *norm* možemo provjeriti ravna li se prema pretpostavljenoj distribuciji $N(55, 5)$.

```
> mean(norm)
```

```
## [1] 54.62767
```

```
> sd(norm)
```

```
## [1] 4.977531
```

Važno je napomenuti da funkcije kao što su `mean()`, `sum()` i `median()` neće ispravno raditi na vektorima koji sadrže znakovne tipove ili vrijednosti koje nedostaju (*NA*). Kako bi se to izbjeglo, korisno je provjeriti vektor prije primjene ovih funkcija ili koristiti argumente poput `na.rm = TRUE` za ignoriranje NA vrijednosti.

```
> typeof(j)
```

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

```
> summary(is.na(j)) # sažetak provjere svih vrijednosti u vektoru sadrže li NA
```

```
##    Mode   FALSE
## logical      10
```

```
> mean(j, na.rm = TRUE)
```

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

Funkcija `typeof(j)` vraća tip podataka elemenata unutar vektora *j*. U ovom slučaju, ispis *[1] "double"* ukazuje na to da vektor *j* sadrži elemente tipa double, što znači da su to cijeli ili decimalni brojevi.

Naredba `summary(is.na(j))` koristi se za sažimanje rezultata provjere o prisutnosti NA vrijednosti unutar vektora *j*. Funkcija `is.na(j)` vraća logički vektor iste duljine kao *j*, gdje su vrijednosti *TRUE* za elemente koji su *NA* i *FALSE* za one koji nisu *NA*. `summary()` zatim daje brojčani sažetak ovog logičkog vektora, pri čemu se prikazuje koliko ima *FALSE* i koliko *TRUE* vrijednosti. U ovom slučaju, svi elementi vektora *j* su valjani, što znači da nema vrijednosti koje nedostaju (*FALSE*).

`mean(j, na.rm = TRUE)` izračunava aritmetičku sredinu elemenata unutar vektora j. Argument `na.rm = TRUE` uvjetuje ignoriranje NA vrijednosti prilikom izračuna. To je važno samo ako vektor sadrži NA vrijednosti. Međutim, s obzirom da u vektoru *j* nema NA vrijednosti, ovaj argument tehnički nije bio potreban. Rezultat *[1] 9* označava da prosjek elemenata vektora *j* iznosi 9.

Ukratko, ove naredbe pokazuju kako provjeriti tip podataka vektora, identificirati i sumarizirati prisutnost nedostajućih podataka te sigurno izračunati osnovne statističke mjere poput prosjeka, uzimajući u obzir moguću prisutnost NA vrijednosti.

6.2.10 Recikliranje vektora

Nadalje, moguće je vršiti neke računske operacije čak i *ako vektori nisu jednakih duljina*, na primjer:

```
> j + a
```

```
##  [1]  6 12 18 24 30  2  4  6  8 10
```

```
> j * a
```

```
##  [1]   5  20  45  80 125   1   4   9  16  25
```

```
> j / a
```

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

a proces kojim se to postiže je tzv. **recikliranje**, pri čemu se redoslijedno ponavljaju elementi kraćeg vektora dok se ne izvrši računska operacija s posljednjim elementom duljeg vektora.

6.2.11 Usporedba vektora

Nadalje, vektore je moguće **uspoređivati**, pri čemu će izlaz predstavljati niz elemenata logičkog tipa, na primjer:

```
> a > j
```

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

```
> j != a
```

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

```
> a == j
```

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

6.2.12 Logički upiti

Na vektorima je moguće vršiti i **logičke upite** te na taj način odabirati elemente. Na primjer,

```
> d[e]
```

```
## [1] "Iva"  "Ana"  "Maja"
```

```
> b > 3
```

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

```
> b[b > 3]
```

```
## [1]  5 10 15 20 25
```

```
> n <- c(TRUE, FALSE, FALSE)
> i[n]
```

```
## [1]  1  4 10 25
```

ovaj posljednji primjer, osim logičkog upita, koristi i princip recikliranja, a rezultat je svaki treći element vektora *i*.

Prije nego nastavite, obratite pozornost na Environment. Svi kreirani objekti, nalaze se na popisu, uz navođenje tipa, duljine i ispisa prvih nekoliko elemenata.

Pitanja za ponavljanje

1. Koji je rezultat sljedeće naredbe: `c <- c(3, 5, "text", TRUE)`? Objasnite zašto dolazi do takvog rezultata.
2. Kako biste kreirali vektor v koji sadrži brojeve od 20 do 50, s razmakom od 5 cijelih brojeva između svakih dviju vrijednosti?
3. Navedite razliku između `rep(x, times=3)` i `rep(x, each=3)`. Koji bi bio rezultat za `x <- c(1, 2, 3)`?
4. Ako definirate `a <- c(10, 20, 30, 40, 50)` i `b <- c(5, 15, 25, 35, 45)`, koji je rezultat `a + b`?
5. Koristite funkciju `rnorm()` da generirate 10 slučajnih brojeva iz normalne distribucije s prosjekom 50 i standardnom devijacijom 10. Napišite kod.
6. Kreirajte vektor koji sadrži sekvencu brojeva od 1 do 100, a svaki peti broj mora biti 0.
7. Koji je rezultat sljedeće naredbe: `rep(c(2, 4), each=3, times=2)`? Objasnite što se dešava u naredbi.
8. Upotrijebite funkciju `runif()` za generiranje 5 brojeva između 0 i 1. Koje su glavne karakteristike takve distribucije?
9. Zašto se preporučuje da svi elementi vektora budu istog tipa? Što će se dogoditi ako `c(1, "string", TRUE)` definirate kao vektor?
10. Objasnite rezultat funkcije `seq(5, 15, by=2)`. Koliko elemenata se kreira u ovom slučaju?
11. Što je rezultat sljedeće naredbe: `c(5, 10, 15) + c(2, 4)`? Objasnite kako se odvija proces recikliranja elemenata.
12. Ako su `a <- c(3, 5, 7, 9)` i `b <- c(2, 4, 6, 8)`, koji je rezultat `a > b`? Koje vrijednosti sadrži izlaz?
13. Koji je rezultat naredbe `sort(c(8, 3, 5, 1, 9), decreasing = TRUE)`? Kako biste pohranili rezultat u novi vektor?
14. Ako je `x <- c(20, NA, 15, NA, 10)`, kako biste izračunali prosječnu vrijednost elemenata vektora ignorirajući NA vrijednosti?
15. Definirajte vektor `y <- c("R", "Studio", "Analysis")` i dodajte naziv svakom elementu koristeći `names()`. Kako biste pristupili vrijednosti s nazivom "Studio?"
16. Koji je rezultat `order(c(3, 8, 2, 7))`? Što znači povratna vrijednost ove funkcije?
17. Ako je `z <- c(4, 9, 12, 5, 7)`, koji elementi će biti odabrani naredbom `z[z > 6]`? Što ova sintaksa postiže?
18. Kreirajte vektor `v <- c("A", "B", "C", "A", "D", "B")`. Koristite `unique()` za pronalaženje jedinstvenih vrijednosti u v. Koji rezultat očekujete?
19. Ako su `p <- c(1, 2, 3, 4, 5)` i `q <- c(10, 20, 30)`, koji je rezultat `p * q`? Objasnite što se događa zbog recikliranja elemenata.

20. Kako biste uklonili treći element iz vektora `k <- c(2, 4, 6, 8, 10)`? Koji će biti novi sadržaj vektora nakon te operacije?

6.3 Kreiranje faktora

U ovom poglavlju objašnjava se kreiranje i primjena faktora, koji se u R-u u pravilu koriste za reprezentaciju kvalitativnih varijabli (opisnih i stupnjevitih), mjerenih na nominalnoj ili ordinalnoj razini. Poglavlje pokriva načine kreiranja faktora, definiranje razina s poretkom, te postupke za pretvorbu faktora u druge tipove podataka, što je često korisno pri radu s podacima dobivenim putem anketa. Dodatno, ukazat će se na česte greške, poput neadekvatne pretvorbe faktora u numeričke vrijednosti, kako bismo mogli izbjeći nesporazume pri analizi i interpretaciji podataka.

Faktori predstavljaju **kvalitativne varijable** (opisne i stupnjevite, mjerene na nominalnoj i ordinalnoj ljestvici), a sastoje se od određenog broja razina (*engl. level*; poprimaju konačan broj unaprijed definiranih vrijednosti). Razine ili leveli mogu se uraditi na način da postoji poredak. U nekim izvorima naići ćete da se faktor tretira kao tip podataka, ali je pravilnije o njima razmišljati kao o vrsti varijabli. Kreiraju se pomoću funkcije `factor()`. Elemente je moguće kombinirati putem `c()`, a moguće je i postojeće vektore pretvoriti u faktore.

```
> Faktor1 <- factor(c("Plava", "Crvena", "Zelena", "Crna"))
> Faktor1
```

```
## [1] Plava  Crvena Zelena Crna
## Levels: Crna Crvena Plava Zelena
```

```
> Faktor2 <- factor(d)
> Faktor2
```

```
## [1] Iva   Ana   Maja  Marko Ivan
## Levels: Ana Iva Ivan Maja Marko
```

U navedena dva primjera radi se o nominalnim varijablama, kod kojih poredak nije moguće utvrditi. Za stupnjevite varijable, kao što je, na primjer, završena razina obrazovanja, možemo utvrditi poredak, a isti može biti relevantan pri daljnjim analizama. U takvom slučaju, faktor se može zadati na sljedeći način:

```
> z <- c("Osnovna skola", "Srednja skola", "Fakultet")
> Faktor3 <- factor(z, levels = z, ordered = TRUE)
> Faktor3
```

```
## [1] Osnovna skola Srednja skola Fakultet
## Levels: Osnovna skola < Srednja skola < Fakultet
```

Ako se pri zadavanju vektora doda `ordered=TRUE` bez specifikacije razina, tada će elementi biti poredani po abecedi ili po veličini.

```
> Faktor4 <- factor(a, ordered = TRUE)
> Faktor4
```

```
## [1] 1 2 3 4 5
## Levels: 1 < 2 < 3 < 4 < 5
```

Ako faktor želimo pretvoriti natrag u numerički tip varijable, to je moguće učiniti na sljedeći način:

```
> num <- as.numeric(Faktor4)
> num
```

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

Isto bi funkcioniralo i da je upotrijebljena funkcija `as.double()`, odnosno za cjelobrojne vrijednosti `as.integer()`, ili `as.character()` za znakovne elemente. Provjera:

```
> typeof(num)
```

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

Još nekoliko primjera.

```
> chr <- as.character(Faktor3)
> chr
```

```
## [1] "Osnovna skola" "Srednja skola" "Fakultet"
```

```
> #obratite pozornost na ovaj primjer
> Faktor6 <- factor(h)
> Faktor6
```

```
## [1] 1  5  10 1  5  10
## Levels: 1 5 10
```

```
> num2 <- as.numeric(Faktor6)
> num2
```

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

U ovom primjeru može se uočiti da je pretvorbom u numeričku varijablu zapravo sačuvan niz rednih mjesta razina Faktora 6, a ne vrijednosti opažanja. To znači da je potrebno napraviti dodatnu promjenu, prvo u znakovni tip, a onda u numerički tip varijable.

```
> num2 <- as.numeric(as.character(Faktor6))
> num2
```

```
## [1]  1  5 10  1  5 10
```

Iz ovog primjera možemo uočiti da je izuzetno važno provjeriti rezultat nakon svake izvršene promjene, osobito u početku dok još niste u potpunosti sigurni čemu služi koja funkcija.

Moguće je i pojedinu varijablu iz postojećeg podatkovnog okvira pretvoriti u faktor, no o tome više u poglavlju posvećenom podatkovnim okvirima.

Dodavanje razina postojećem faktoru:

```
> levels(Faktor3) <- c(levels(Faktor3), "Doktorat")
> Faktor3
```

```
## [1] Osnovna skola Srednja skola Fakultet
## Levels: Osnovna skola < Srednja skola < Fakultet < Doktorat
```

Faktori su izuzetno korisni za analizu odgovora anketnih podataka, gdje su najčešće odgovori na pitanja o demografskim karakteristikama zapisani kao opisne ili stupnjevite varijable. Osim toga, mnogi statistički modeli u R-u zahtijevaju faktore za kategoričke prediktorne varijable kako bi ispravno interpretirali kategorije. Pri vizualizaciji podataka, faktori omogućuju kontrolu nad redosljedom prikaza kategorija, što može biti ključno za jasnoću grafikona.

6.4 Najčešće greške pri kreiranju i radu s faktorima

- **Neadekvatno upravljanje razinama**: Jedna od najčešćih grešaka nastaje pri dodavanju novih elemenata u faktor, pri čemu ti elementi nisu prepoznati kao postojeće razine. R će automatski dodijeliti NA novim elementima koji nisu definirani kao razine faktora.
- **Pretvorba u numeričke vrijednosti**: Direktna pretvorba faktora u numeričke vrijednosti (`as.numeric(faktor)`) može rezultirati vrijednostima koje predstavljaju oznake razina, a ne pridružene stvarne numeričke vrijednosti. Ispravan način je prvo pretvoriti faktor u znakovni niz, a zatim u numeričke vrijednosti.
- **Zanemarivanje poretka za uređene faktore**: Pri radu s uređenim faktorima, važno je pravilno postaviti i koristiti poredak, jer to utječe na analizu i vizualizaciju podataka.

Pitanja za ponavljanje

1. Što su faktori u R-u i koja je njihova osnovna svrha u analizi podataka, posebno kod kvalitativnih varijabli?

2. Kada bi bilo korisno postaviti argument `ordered = TRUE` prilikom kreiranja faktora i što se time postiže?

3. Koja je razlika između nominalnih i ordinalnih faktora i kako se postavlja razlika prilikom kreiranja faktora?

4. U primjeru `z <- c("Osnovna skola", "Srednja skola", "Fakultet"); Faktor3 <- factor(z, levels = z, ordered = TRUE)`, što će biti rezultat i redoslijed razina u faktoru Faktor3?

5. Ako faktor pretvorimo u numerički tip pomoću `as.numeric(faktor)`, koje greške mogu nastati i kako ih izbjeći?

6. Što će vratiti naredba `levels(Faktor1)` ako je Faktor1 definiran kao `Faktor1 <- factor(c("Plava", "Crvena", "Zelena", "Crna"))`?

7. Kako se koristi funkcija `levels()` za dodavanje novih razina faktoru i kako biste dodali razinu "Doktorat" faktoru Faktor3?

8. Nakon kreiranja faktora Faktor4 kao `factor(a, ordered = TRUE)`, kako možete provjeriti je li poredak postavljen po abecednom redoslijedu ili prema drugom kriteriju?

9. Ako se faktor pretvara u numerički tip podataka, zašto može biti potrebno prvo ga pretvoriti u znakovni niz pomoću `as.character()` i kako izgleda postupak?

10. Ako u faktoru zadanom kao `Faktor6 <- factor(h)` pokušate izvesti naredbu `num2 <- as.numeric(Faktor6)`, što se može dogoditi i kako pravilno izvesti pretvorbu?

7 Višediomenzionalni tipovi podataka

7.1 Kreiranje matrice

U ovom poglavlju objašnjava se kako kreirati i strukturirati matrice u R-u, uključujući različite načine definiranja redova i stupaca, unos elemenata te postavljanje atributa poput naziva redaka i stupaca. Poglavlje također prikazuje kako koristiti vektore za kreiranje matrica i kako prilagoditi strukturu matrice za analizu podataka.

Matrice su **dvodimenzionalni vektori** (dakle, vrijede isti uvjeti kao i za vektore), a mogu se kreirati na više načina. Matricu je uobičajeno definirati koristeći funkciju `matrix()`, u kojoj je potrebno navesti podatke, broj stupaca i broj redaka. Osim toga, može se definirati način slaganja elemenata u matricu, zadano (*engl. default*) je slaganje po stupcu, a može se definirati slaganje po recima unosom **byrow=TRUE**. Podaci se mogu kreirati, a može se koristiti uvezeni skup podataka (*engl. dataset*) (npr. ako se uvezeni dataset zove Analiza, tada je `data=Analiza`, kasnije će biti više riječi o tome).

7.1.1 Kreiranje matrice koristeći izravan upis elemenata

Ukoliko postoje specifični elementi koje je potrebno upisati u matricu točno određenim redom, onda kreiranje matrice može biti napravljeno na sljedeći način:

```
> Mat <- matrix(c(5,  6,  8,
+                 10, 11, 15,
+                 2,  7,  3,
+                 12, 28, 19,
+                 13, 22, 24,
+                 31, 25, 8),
+               ncol = 3, byrow = TRUE)
> Mat
```

```
##      [,1] [,2] [,3]
## [1,]    5    6    8
## [2,]   10   11   15
## [3,]    2    7    3
## [4,]   12   28   19
## [5,]   13   22   24
## [6,]   31   25    8
```

Pri definiranju matrice, koristi se naredba `matrix()`. Prvi parametar funkcije su podaci temeljem kojih se kreira matrica. Ako se podaci unose ručno, potrebno ih je unijeti kao vektor, koristeći funkciju `c()`. Potom je potrebno definirati broj redaka i/ili stupaca. Dovoljno je definirati samo jedno (broj redaka ili broj stupaca), ali se može definirati oboje. Potom je potrebno definirati parametar *byrow*. To je logički parametar koji je unaprijed zadan (*engl. default*) na *FALSE* i kao takav uvjetuje da će podaci biti očitani po stupcima, a ne po recima. Ako je definirano `byrow = TRUE`, tada će podaci biti očitani i zapisani u matricu po recima. Ovaj se pristup češće koristi s obzirom da je ljudima uglavnom prirodnije čitati i upisivati podatke po recima nego po stupcima.

Ako ne bismo koristili postavku `byrow = TRUE`, tada se automatski podrazumijeva upotreba `byrow = FALSE` i očitavanje i zapis podataka u matricu po stupcima. Sljedeći primjer pokazuje upravo takvu situaciju. Obratite pozornost na to da u zapisu niže također koristimo manje pregledan zapis podataka. To znači da pri kreiranju matrica zapravo nije potrebno svaki redak ili stupac buduće matrice pisati u zaseban redak i R će očitati podatke neovisno o zapisu. U pravilu, zapis nalik na prethodni primjer koristimo radi preglednosti koda i lakše provjere.

```
> Mat <- matrix(c(5, 6, 8, 10, 11, 15, 2, 7, 3, 12,
+                 28, 19, 13, 22, 24, 31, 25, 8),
+               ncol = 3)
> Mat
```

```
##      [,1] [,2] [,3]
## [1,]    5    2   13
## [2,]    6    7   22
## [3,]    8    3   24
## [4,]   10   12   31
## [5,]   11   28   25
## [6,]   15   19    8
```

Osim toga, ako su u pitanju cjelobrojni redoslijedni elementi, onda matricu možemo kreirati i na sljedeći način:

```
> A <- matrix(data = 10:18, nrow = 3, ncol = 3)
> A
```

```
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
```

7.1.2 Matrica indeksa

Matrica koja sardži redoslijedne cijele brojeve kao elemente naziva se i matrica indeksa, zbog toga što vrijednosti elemenata odgovaraju rednim mjestima. Takve se matrice ponekad koriste u pomoćnim izračunima.

```
> B <- matrix(data = 1:9, nrow = 3, ncol=3, byrow=TRUE)
> B
```

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

7.1.3 Kreiranje matrice iz vektora

Matrica se može kreirati i temeljem vektora. Kreirajmo malo duži vektor, na primjer,

```
> o <- c(a, j)
> o
```

```
##  [1]  1  2  3  4  5  5 10 15 20 25  1  2  3  4  5
```

Da bi se matrica kreirala pomoću vektora, broj elemenata u vektoru mora odgovarati broju elemenata u matrici koja će se kreirati (ovdje ne funkcionira recikliranje). Zbog toga ćemo prvo provjeriti duljinu vektora.

```
> length(o)
```

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

Da bi vektor pretvorili u matricu, potrebno mu je zadati dvije dimenzije, a dimenzije se zadaju pomoću funkcije `dim()`. U ovom slučaju, duljina vektora je 15, pa možemo kreirati matricu dimenzija, npr. 3×5.

```
> dim(o) <- c(3, 5)
```

Već pri sljedećem ispisu moguće je uočiti rezultat primjene dim(). No, obratite pozornost na to da će matrica kreirana na ovaj način uvijek biti popunjena po stupcima, a ne po recima.

```
> o
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    4   10   25    3
## [2,]    2    5   15    1    4
## [3,]    3    5   20    2    5
```

Ako je potrebno, tako kreiranoj matrici moguće je pridodati atribute, a **nazivi stupaca i redaka** mogu se pridružiti kao što je ranije prikazano.

```
> rownames(o) <- c("Prvi", "Drugi", "Treći")
> colnames(o) <- c("Stupac 1.", "Stupac 2.", "Stupac 3.",
+                  "Stupac 4.", "Stupac 5.")
> print(o)
```

```
##       Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
## Prvi          1         4        10        25         3
## Drugi         2         5        15         1         4
## Treći         3         5        20         2         5
```

7.1.4 Dodjela atributa

Za unos naziva redaka i stupaca mogu se koristiti `colnames()` i `rownames()`.

```
> C <- matrix(10:18, nrow = 3, ncol = 3, byrow = TRUE)
> colnames(C)<-c("A", "B", "C")
> rownames(C)<-c("1.", "2.", "3.")
> C
```

```
##     A  B  C
## 1. 10 11 12
## 2. 13 14 15
## 3. 16 17 18
```

Nazivi stupaca i redaka matrice mogu se unijeti pomoću argumenta **dimnames**. Pritom je prvo potrebno kreirati listu koja će imati dva elementa: vektor naziva redaka i vektor naziva stupaca.

```
> C<-list(c("Crveno", "Žuto", "Zeleno"), c("Lokacija 1", "Lokacija 2"))
> matrix(data=c("DA", "NE", "N/a"), nrow = 3, ncol = 2, dimnames = C)
```

```
##        Lokacija 1 Lokacija 2
## Crveno "DA"       "DA"
## Žuto   "NE"       "NE"
## Zeleno "N/a"      "N/a"
```

Obratite pozornost da je ovime prepisana prethodno kreirana matrica *C*. Treba voditi računa o nazivima vektora i matrica, jer R neće javljati nikakvo upozorenje ako želite promijeniti postojeći vektor ili matricu.

7.2 Rad s matricama

U ovom poglavlju obrađuju se osnovne operacije s matricama u R-u, uključujući provjeru tipa, odabir i zamjenu elemenata te različite oblike filtriranja i spajanja. Također, poglavlje pruža uvid u operacije kao što su transponiranje i množenje matrica te konverziju matrice u vektor ili podatkovni okvir, omogućujući fleksibilnost u analizi podataka. Dodatno, poglavlje naglašava važnost razumijevanja prednosti i ograničenja matrica za različite analitičke zadatke.

7.2.1 Provjera tipa matrice

Je li objekt matrica može se provjeriti koristeći `is.matrix()`.

```
> is.matrix(o)
```

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

Osim toga, moguće je provjeriti radi li se o matrici i koristeći `class()`:

```
> class(o)
```

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

Za uvid u karakteristike elemenata matrice, koristi se `typeof()`.

```
> typeof(o)
```

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

7.2.2 Odabir elemenata matrice

Kao i kod vektora, iz matrica je moguće **odabirati elemente** s obzirom na njihovu poziciju.

```
> A[2, 3]
```

```
## [1] 17
```

Pritom **prvi broj označava redak, a drugi stupac**. Tako je 17 element koji pripada drugom retku i trećem stupcu matrice A. Ispis matrice moguće je odabrati i s obzirom na retke i stupce. Na primjer, ako se želi odabrati/ ispisati samo prvi redak matrice A, tada je to moguće učiniti na sljedeći način:

```
> A[1, ]
```

```
## [1] 10 13 16
```

```
> o[1, ]
```

```
## Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
##         1         4        10        25         3
```

Možete uočiti razliku između ispisa prvog retka matrice A kojoj nisu zadani nazivi stupaca i redaka, u odnosu na matricu o, kojoj su zadani nazivi stupaca i redaka.Ipak, uočite i to da imenovanje stupaca i redaka ne utječe na odabir.

Ako bi se, koristeći isti izraz upisao, npr. `o[2,]`, tada bi rezultat prikazao samo drugi redak. Ako se želi ispisati više redaka, tada je to potrebno definirati npr. ovako:

```
> o[c(1, 2), ]
```

```
##       Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
## Prvi          1         4        10        25         3
## Drugi         2         5        15         1         4
```

Slično vrijedi i za stupce, samo će se u tom slučaju prva pozicija prije zareza ostavljati praznom.

```
> o[ ,1]
```

```
##  Prvi Drugi Treći
##     1     2     3
```

```
> o[ ,c(1, 2)]
```

```
##       Stupac 1. Stupac 2.
## Prvi          1         4
## Drugi         2         5
## Treći         3         5
```

```
> o[ ,c(3, 5)]
```

```
##       Stupac 3. Stupac 5.
## Prvi         10         3
## Drugi        15         4
## Treći        20         5
```

7.2.3 Zamjena elemenata matrica

Kao i kod vektora, elementi matrice mogu se zamijeniti drugim elementom. Na primjer:

```
> o[2, 2] <- 100
> o
```

```
##       Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
## Prvi          1         4        10        25         3
## Drugi         2       100        15         1         4
## Treći         3         5        20         2         5
```

Ako je potrebno, mogu se zamijeniti i svi elementi pojedinog retka ili stupca.

```
> o[ ,3] <- c(15, 20, 25)
> o
```

```
##       Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
## Prvi          1         4        15        25         3
## Drugi         2       100        20         1         4
## Treći         3         5        25         2         5
```

```
> o[3, ] <- c(4, 6, 26, 3, 6)
> o
```

```
##       Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
## Prvi          1         4        15        25         3
## Drugi         2       100        20         1         4
## Treći         4         6        26         3         6
```

Ako je potrebno, pri zamjeni se mogu koristiti i već prikazane operacije na vektorima.

```
> o[3, ]<-c(o[3, ] + 1)
> o
```

```
##       Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
## Prvi          1         4        15        25         3
## Drugi         2       100        20         1         4
## Treći         5         7        27         4         7
```

```
> o[ ,5] <- c(o[ ,5] * 3)
> o
```

```
##       Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
## Prvi          1         4        15        25         9
## Drugi         2       100        20         1        12
## Treći         5         7        27         4        21
```

Generalno, sve operacije u kojima se matrice pojavljuju bit će osjetljive na elemente koji nedostaju (NA), zbog čega je brisanje elemenata matrice (iako moguće) nepoželjno.

7.2.4 Filtriranje matrice - odabir elemenata prema kriteriju

Slično kao kod vektora, mogu se ispisati elementi **prema zadanom kriteriju**, npr. elementi veći od 10.

```
> o[o > 10]
```

```
## [1] 100  15  20  27  25  12  21
```

7.2.5 Kreiranje matrice s ponovljenim vrijednostima i zadavanje dijagonale

Osim toga, moguće su situacije u kojima će se htjeti kreirati specifične matrice. Na primjer matrica koja sadržava sve 0, osim jedinica na dijagonali. To je moguće učiniti na sljedeći način:

```
> D <- matrix(0, nrow = 3, ncol = 3)
> diag(D) <-1
> D
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

Slično tome, moguće je učiniti i obrnuto. Na primjer, ako se želi kreirati matrica koja će sadržavati sve jedinice, osim nula na dijagonali, to je moguće na sljedeći način:

```
> E <-matrix(1, nrow = 3, ncol = 3)
> diag(E)<-0
> E
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    1
## [2,]    1    0    1
## [3,]    1    1    0
```

7.2.6 Spajanje matrica

Kao što je moguće spajati vektore, tako je moguće **spajati** i matrice. Pritom je potrebno paziti na dimenzije matrica koje se spajaju. Matrice koje se podudaraju u broju redaka mogu se spojiti funkcijom `cbind()`, a matrice koje se podudaraju po broju stupaca funkcijom `rbind()`. Ranije kreirane matrice A i B imaju jednak broj stupaca i redaka. Provjerimo koja je razlika ako se primjenjuje spajanje po stupcu i spajanje po retku.

```
> AB <- rbind(A, B)
> AB
```

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

Prethodnom naredbom matrice su spojene po recima (r-row). Kreirana je nova matrica, AB, pri čemu matrica A tvori prva tri retka nove matrice, a matrica B sljedeća tri retka.

Sljedećom naredbom matrice se spajaju po stupcima (c-column). Tako će druga matrica biti nadodana prvoj matrici u obliku dodatna tri stupca.

```
> BA <- cbind(A, B)
> BA
```

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

U sljedećem primjeru kombinirat ćemo ranije naučene funkcije. Recimo da se odabirom svakog sedmog elementa iz vektora f želi kreirati vektor p. Nakon provjere duljine vektora, dodjeljuju mu se odgovarajuće dimenzije. Usporedbom s već kreiranim matricama, uočava se da novokreirana matrica ima jednak broj redaka kao matrica o, te se matrice p i o spajaju funkcijom `cbind()`.

```
> p <- f[seq(1, 101, by=7)]
> p
```

```
##  [1] 100 107 114 121 128 135 142 149 156 163 170 177 184 191 198
```

```
> length(p)
```

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

```
> dim(p) <- c(3, 5)
> po <- cbind(o, p)
> po
```

```
##       Stupac 1. Stupac 2. Stupac 3. Stupac 4. Stupac 5.
## Prvi          1         4        15        25         9 100 121 142 163 184
## Drugi         2       100        20         1        12 107 128 149 170 191
## Treći         5         7        27         4        21 114 135 156 177 198
```

7.2.7 Operacije na matricama

Nadalje, jedna od često potrebnih operacija s matricama je **transponiranje** `t()` (zamjena redaka i stupaca).

```
> t(po)
```

```
##           Prvi Drugi Treći
## Stupac 1.    1     2     5
## Stupac 2.    4   100     7
## Stupac 3.   15    20    27
## Stupac 4.   25     1     4
## Stupac 5.    9    12    21
##            100   107   114
##            121   128   135
##            142   149   156
##            163   170   177
##            184   191   198
```

Matrice se mogu i zbrajati

```
> A + B
```

```
##      [,1] [,2] [,3]
## [1,]   11   15   19
## [2,]   15   19   23
## [3,]   19   23   27
```

Nadalje, matrice se mogu množiti skalarom:

```
> 3 * A
```

```
##      [,1] [,2] [,3]
## [1,]   30   39   48
## [2,]   33   42   51
## [3,]   36   45   54
```

Osim toga, matrice se mogu i množiti:

```
> A * B
```

```
##      [,1] [,2] [,3]
## [1,]   10   26   48
## [2,]   44   70  102
## [3,]   84  120  162
```

Obratite pozornost na to da je ova operacija rezultirala množenjem odgovarajućih elemenata dviju matrica, ali ne i množenjem matrica kako se to učilo na matematici. Da biste to napravili, potrebno je koristiti operator **%*%**:

```
> A %*% B
```

```
##      [,1] [,2] [,3]
## [1,]  174  213  252
## [2,]  186  228  270
## [3,]  198  243  288
```

7.2.8 Konverzija matrice u vektor

Matrice je moguće konvertirati u vektor, ali pritom treba posvetiti osobitu pozornost poretku elemenata. Na primjer, izravna transformacija u vektor rezultirat će očitavanjem elemenata po stupcu. Podsjetimo se prvo kako je definirana matrica A.

```
> A
```

```
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
```

Ako ovu matricu izravno trasformiramo u vektor, dobivamo sljedeći rezultat:

```
> v <- as.vector(A)
> v
```

```
## [1] 10 11 12 13 14 15 16 17 18
```

Uočava se da se u vektoru nalaze elementi matrice iščitani po stupcima. S obzirom da se u praksi matrični zapis često koristi za jednostavniju i bržu manipulaciju podataka tablica, češće ćete htjeti da elementi u vektoru budu iščitani i zapisani po recima u matrici. To se može postići međukorakom, odnosno transponiranjem matrice.

```
> v <- as.vector(t(A))
> v
```

```
## [1] 10 13 16 11 14 17 12 15 18
```

Nadalje, tijekom konverzije matrice u vektor, moguće je promijeniti vrijednosti matrice na način da se one umanje, uvećaju ili množe skalarom. Na primjer, ako se svi elementi matrice žele uvećati za npr. 5, tada će se to učiniti na sljedeći način:

```
> v <- as.vector(5 + t(A))
> v
```

```
## [1] 15 18 21 16 19 22 17 20 23
```

Također, ako se elementi žele uvećati 5 puta, tada je elemente matrice potrebno pomožiti brojem 5:

```
> v <- as.vector(5 * t(A))
> v
```

```
## [1] 50 65 80 55 70 85 60 75 90
```

Također, pri transformaciji u vektor, moguće je i promijeniti predznak vrijednostima elemenata matrice:

```
> v <- as.vector(-t(A))
> v
```

```
## [1] -10 -13 -16 -11 -14 -17 -12 -15 -18
```

Ako se vrijednosti u matrici žele uvećati, umanjiti ili množiti s različitim vrijednostima po stupcima, onda je moguće za to primijeniti vektor:

```
> v <- as.vector(c(1, 2, 3) * t(A))
> v
```

```
## [1] 10 26 48 11 28 51 12 30 54
```

Napomena: pri svakoj provedenoj konverziji podataka, poželjno je ispisati rezultate kako biste provjerili je li konverzija rezultirala željenim promjenama. Ovi su postupci često samo pomoćne radnje u zahtjevnijim analizama i greške u ovim koracima mogu rezultirati ozbiljnim pogrešama u konačnim rezultatima.

7.2.9 Konverzija matrice u podatkovni okvir

Matrice podataka s uključenim nazivima stupaca i redaka često su korisnije u obliku podatkovnog okvira, zato jer brojni paketi koriste upravo takvu strukturu podataka. Matrice se lako mogu spremiti u obliku podatkovnog okvira na sljedeći način:

```
> X <- as.data.frame(po)
> str(X)
```

```
## 'data.frame':    3 obs. of  10 variables:
##  $ Stupac 1.: num  1 2 5
##  $ Stupac 2.: num  4 100 7
##  $ Stupac 3.: num  15 20 27
##  $ Stupac 4.: num  25 1 4
##  $ Stupac 5.: num  9 12 21
##  $ V6       : num  100 107 114
##  $ V7       : num  121 128 135
##  $ V8       : num  142 149 156
##  $ V9       : num  163 170 177
##  $ V10      : num  184 191 198
```

Obratite pozornost na prozor Environment, u kojem se sad pojavio objekt X.

7.2.10 Izračun pokazatelja deskriptivne statistike za matrice

Osim toga, ako se izdvoji pojedini stupac matrice, na njemu se mogu izvršavati naredbe kao i na vektorima. Na primjer, ako se želi utvrditi prosjek vrijednosti prvog stupca matrice A, to se može učiniti na sljedeći način:

```
> mean(A[ ,1])
```

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

Treba obratiti posebnu pozornost na to što se događa ako se ne izdvoji specifični stupac, nego se odabere *mean* za matricu A:

```
> mean(A)
```

```
## [1] 14
```

Naime, izračunata je aritmetička sredina svih elemenata matrice A, što može predstavljati problem ako postoji pojmovna razlika u vrijednostima zapisanim u stupcima matrice (tj., ako bi stupci matrice trebali predstavljati varijable).

7.3 Prednosti i nedostaci rada s matricama te najčešće greške

Nekonzistentnost dimenzija matrica

Jedna od čestih grešaka prilikom rada s matricama je pokušaj izvođenja operacija na matricama koje nemaju kompatibilne dimenzije (na primjer, pokušaj množenja dviju matrica čije dimenzije ne zadovoljavaju pravilo množenja matrica). R će vratiti grešku ako dimenzije nisu usklađene za određenu operaciju.

Neispravna primjena funkcija

Ovdje se misli na pokušaj primjene funkcija koje očekuju vektore (na primjer, `sum()`, `mean()`) izravno na matricama bez specificiranja dimenzija (redaka ili stupaca) na kojima se funkcija treba primijeniti. To može dovesti do neočekivanih rezultata, kao što je izračun prosjeka svih elemenata u matrici umjesto po stupcima ili recima.

Zaboravljanje postavke byrow prilikom kreiranja matrice:

Važno je zadati na koji će se način matrica popunjavati: po stupcima ili recima. Ako zaboravite postaviti `byrow=TRUE`, a namjeravali se popuniti matricu po redovima, elementi će se rasporediti po stupcima, što može rezultirati neželjenim rasporedom podataka. Štoviše, ako ne izvršite provjeru nakon kreiranja matrice, preskakanje ovog argumenta može uzrokovati veće pogreške u kasnijim izračunima.

Korištenje nespojivih tipova podataka

Budući da matrice u R-u mogu sadržavati samo jedan tip podataka, pokušaj kombiniranja brojeva i tekstualnih vrijednosti u istoj matrici rezultirat će konverzijom svih elemenata u tekstualne vrijednosti. To može izazvati probleme pri kasnijoj obradi podataka.

Pogrešno indeksiranje i odabir podataka

Pogreške u indeksiranju, poput pokušaja pristupa elementima koji ne postoje (zbog krive specifikacije indeksa) mogu dovesti do grešaka ili neočekivanih rezultata. Također, bitno je razumjeti razliku između odabira pojedinačnih elemenata, redaka ili stupaca.

Zanemarivanje atributa matrice prilikom transformacija

Prilikom konverzije matrice u vektor ili izvođenja nekih transformacija, moguće je izgubiti imena redaka i stupaca ili druge atribute matrice. Važno je biti svjestan ovih promjena i po potrebi ponovno postaviti atribute.

Matrice u R-u su moćan alat za numeričku analizu i manipulaciju podataka, ali imaju određena ograničenja koja ih čine manje fleksibilnima u odnosu na liste ili podatkovne okvire za određene vrste analiza i obrade podataka. Evo ključnih ograničenja matrica:

Homogenost tipova podataka

Jedno od glavnih ograničenja matrica jest da one mogu sadržavati samo jedan tip podataka. Ako matrica sadrži brojeve i tekstualne podatke, svi će biti prisilno pretvoreni u tekstualni tip, što može dovesti do gubitka numeričkih operacija nad tim podacima. Nasuprot tome, podatkovni okviri i liste mogu sadržavati elemente različitih tipova podataka (npr., numeričke, logičke, znakovne), što ih čini prikladnijima za složene skupove podataka koji se često nalaze u praktičnim primjenama.

Dvodimenzionalna struktura

Matrice su inherentno dvodimenzionalne, što znači da imaju ograničenje na retke i stupce. Ovo ograničenje može biti problematično kada radite s višedimenzionalnim podacima. Liste, s druge strane, mogu sadržavati složene i višedimenzionalne strukture podataka, uključujući i druge liste ili podatkovne okvire, omogućavajući time veću fleksibilnost u organizaciji podataka.

Manipulacija podacima i pristup

Podatkovni okviri u R-u su optimizirani za statističku analizu i manipulaciju podataka, s brojnim funkcijama iz različitih paketa koje olakšavaju rad s podacima. Podatkovni okviri omogućuju jednostavno filtriranje, sortiranje, grupiranje i sumiranje podataka, dok matrice nisu toliko prilagođene za takve operacije.

Pristup i indeksiranje

Iako matrice omogućuju pristup elementima, recima i stupcima putem indeksiranja, podatkovni okviri pružaju dodatnu fleksibilnost omogućujući pristup stupcima po imenu, što može biti intuitivnije i smanjiti mogućnost grešaka prilikom analize podataka.

Veličina i performanse

Za vrlo velike skupove podataka, posebno kada se radi o rijetkim matricama koje sadrže mnogo nula, specijalizirane strukture podataka poput rijetkih matrica (*sparseMatrix* iz paketa *Matrix*) mogu pružiti značajne prednosti u smislu učinkovitosti pohrane i performansa obrade. Podatkovni okviri i liste ne nude izravnu podršku za ovakve optimizacije.

Unatoč svojim ograničenjima, matrice se dalje često koristimo u mnogim scenarijima u R-u, posebice zbog njihove efikasnosti i jednostavnosti kada su u pitanju specifični tipovi analize i manipulacije podacima. Evo nekoliko situacija u kojima su matrice posebno korisne.

Linearna algebra i matematičke operacije

Matrice su idealne za izvođenje operacija linearne algebre, uključujući množenje matrica, transponiranje, izračunavanje determinanti, inverza i svojstvenih vrijednosti. U situacijama gdje su ove operacije ključne, kao što je rješavanje sustava linearnih jednadžbi, matrice su nezaobilazan alat.

Numerička analiza

Matrice se često koriste u numeričkim simulacijama i algoritmima, gdje je potrebno manipulirati velikim količinama numeričkih podataka. Primjeri: numeričko rješavanje diferencijalnih jednadžbi, optimizaciju i simulacije.

Obrada signala i slike

U obradi signala i slika, podaci su često predstavljeni kao dvodimenzionalni nizovi (na primjer, pikseli slike), gdje matrice omogućuju efikasnu obradu i analizu tih podataka, uključujući filtriranje, detekciju rubova, i segmentaciju.

Statistički modeli i multivarijatna analiza

Pri modeliranju statističkih odnosa među višedimenzionalnim podacima, matrice se koriste za predstavljanje podataka i koeficijenata u višedimenzionalnim statističkim modelima, kao što su višestruka linearna regresija, analiza varijance i faktorska analiza.

Grafovi i mreže

U analizi grafova i mreža, matrice susjedstva i incidencije koriste se za predstavljanje veza među čvorovima, što omogućava primjenu algoritama za pretraživanje grafova, pronalaženje najkraćih putova, i druge analize mrežnih struktura.

Računanje s rijetko popunjenim matricama

U situacijama gdje su podaci rijetko raspoređeni (npr., u matricama s mnogo nula), specijalizirane strukture rijetkih matrica omogućuju efikasno pohranjivanje i obradu, čineći matrice prikladnima za rad s velikim skupovima podataka u kojima je većina elemenata nula.

Pitanja za ponavljanje

1. Ako želite kreirati matricu od brojeva 1 do 9 koja će imati 3 reda i 3 stupca, kako biste to postigli koristeći funkciju `matrix()`?

2. Što znači argument `byrow = TRUE` u funkciji `matrix()`? Koja je razlika u rezultatu ako je `byrow = FALSE`?

3. Ako je definirana matrica `Mat <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)`, što će ispisati naredba `Mat[2, 1]`?

4. Napišite naredbu kojom ćete kreirati matricu B koja sadrži brojeve od 1 do 12, raspoređene u 3 stupca i 4 reda. Kako biste osigurali da su elementi zapisani po recima?

5. Ako je kreirana matrica `C <- matrix(10:18, nrow = 3, byrow = TRUE)`, kako biste promijenili nazive stupaca u Stupac1, Stupac2, i Stupac3?

6. Što će ispisati naredba `dim(matrix(1:9, nrow = 3, byrow = TRUE))`? Objasnite što predstavlja povratna vrijednost funkcije `dim()`.

7. Ako kreirate vektor `v <- c(1, 2, 3, 4, 5, 6)` i pokušate ga pretvoriti u matricu dimenzija 2×4 koristeći naredbu `dim(v) <- c(2, 4)`, što će se dogoditi? Zašto?

8. Kreirajte matricu D dimenzija 2×3 koristeći `matrix()` funkciju i pridružite joj nazive redaka „Red1" i „Red2" te nazive stupaca „Stupac1," „Stupac2," i „Stupac3" pomoću funkcija `rownames()` i `colnames()`.

9. Ako želite kreirati matricu koja sadrži niz logičkih vrijednosti (TRUE, FALSE, TRUE, FALSE) dimenzija 2×2, koja će naredba to omogućiti? Kako biste provjerili tip podataka unutar matrice?

10. Objasnite kako biste trebali dodati naziv „Student" retku i naziv „Predmet" stupcu u matrici 3×3 koristeći argument dimnames.

11. Ako kreirate matricu `M <- matrix(1:6, nrow = 2)`, koliko redaka i koliko stupaca ima matrica M, te koji će biti ispis matrice?

12. Koristeći matricu A dimenzija 3×3 definiranu kao `A <- matrix(1:9, ncol = 3, byrow = TRUE)`, što će ispisati naredba `A[3, 2]`?

13. Kako biste dodijelili vrijednost 100 u drugi redak i treći stupac matrice A? Napišite naredbu i provjerite novu vrijednost elementa.

14. Ako je matrica `B <- matrix(c(5, 10, 15, 20), nrow = 2)`, što će ispisati naredba `B[, 1]`? Objasnite što označava zapisana pozicija u indeksiranju.

15. Kako bi izgledao ispis matrice C nakon primjene naredbe `diag(C) <- 0`, ako je `C <- matrix(1, nrow = 3, ncol = 3)`?

16. Imate matrice `X <- matrix(1:6, nrow = 2)` i `Y <- matrix(7:12, nrow = 2)`. Pomoću koje funkcije biste spojili te dvije matrice tako da Y bude nastavak X u stupcima? Napišite naredbu.

17. Ako je matrica A definirana kao `A <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)`, što će ispisati `A %*% t(A)`? Objasnite rezultat ove operacije.

18. Ako imate matricu `M <- matrix(1:4, nrow = 2, ncol = 2)` i želite konvertirati ovu matricu u vektor, koja je naredba za to, te što će biti rezultat?

19. Imate matricu `P <- matrix(10:18, nrow = 3, byrow = TRUE)`. Kako biste pronašli aritmetičku sredinu elemenata drugog stupca? Napišite naredbu i rezultat.

20. Što će ispisati `mean(matrix(1:9, nrow = 3))`? Koja je razlika između ove naredbe i `mean(matrix(1:9, nrow = 3)[, 1])`?

7.4 Kreiranje i rad s listama

U ovom poglavlju objašnjavaju se osnovne funkcionalnosti lista u R-u, kao što su kreiranje, dodavanje i odabir elemenata te spajanje i razdvajanje lista. Liste su fleksibilni objekti koji mogu pohranjivati elemente različitih tipova, uključujući druge strukture poput vektora i matrica, što ih čini korisnima za složene podatkovne strukture. Dodatno, poglavlje naglašava prednosti i uobičajene greške pri radu s listama, omogućujući korisnicima efikasno upravljanje heterogenim podacima u analitičkim zadacima.

Nalik vektorima, liste su jednodimenzionalni objekti, ali mogu sadržavati **elemente različitih tipova** i mogu uključivati **druge objekte** (vektore, matrice, druge liste, funkcije i dr., što znači da elementi unutar liste ne moraju biti jednodimenzionalni). Zadaje se funkcijom `list()`. Na primjer,

```
> lista1 <-list(5, "Pula", a, AB)
> lista1
```

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

Elementi liste naznačeni su *dvostrukim uglatim zagradama* i označavaju poziciju elementa u listi. Dakle, i dalje se radi o **uređenom nizu** i prema poziciji je moguće odabirati elemente.

```
> lista1[[2]]
```

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

7.4.1 Dodavanje atributa

Elementima liste možemo pridružiti **nazive**.

```
> names(lista1)<-c("broj", "grad", "vektor", "matrica")
> lista1
```

```
## $broj
## [1] 5
##
## $grad
## [1] "Pula"
##
## $vektor
## [1] 1 2 3 4 5
##
## $matrica
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
## [4,]    1    2    3
## [5,]    4    5    6
## [6,]    7    8    9
```

Elementima liste moguće je pridodati nazive i prilikom kreiranja liste.

```
> lista2<-list(Drzava="Hrvatska", Grad="Hvar", Selo="Heki",
+              Rijeka="Hudinja", Jezero="Harambašić",
+              Opcina="Hum na Sutli", Planina="Hahlići",
+              Zivotinja=c("Hrcak", "Hobotnica", "Hijena"))
> lista2
```

```
## $Drzava
## [1] "Hrvatska"
##
## $Grad
## [1] "Hvar"
##
## $Selo
## [1] "Heki"
##
## $Rijeka
## [1] "Hudinja"
##
## $Jezero
## [1] "Harambašić"
##
## $Opcina
## [1] "Hum na Sutli"
```

```
##
## $Planina
## [1] "Hahlići"
##
## $Zivotinja
## [1] "Hrcak"     "Hobotnica" "Hijena"
```

7.4.2 Odabir elemenata

Odabire je moguće vršiti putem pozicije elementa i putem naziva.

```
> lista2[1]
```

```
## $Drzava
## [1] "Hrvatska"
```

```
> lista2$Drzava
```

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

```
> lista1$matrica
```

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

Nadalje, možemo odabirati elemente unutar elemenata liste, na primjer:

```
> lista1$matrica[4, 3]
```

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

```
> lista1[[3]][15]
```

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

7.4.3 Dodavanje elemenata

Ako želimo postojeću listu **proširiti za element**, onda je to moguće učiniti pridruživanjem.

```
> lista2$Biljka <- "Hren"
> lista2
```

```
## $Drzava
## [1] "Hrvatska"
##
## $Grad
## [1] "Hvar"
##
## $Selo
## [1] "Heki"
##
## $Rijeka
## [1] "Hudinja"
##
## $Jezero
## [1] "Harambašić"
##
## $Opcina
## [1] "Hum na Sutli"
##
## $Planina
## [1] "Hahlići"
##
## $Zivotinja
## [1] "Hrcak"     "Hobotnica" "Hijena"
##
## $Biljka
## [1] "Hren"
```

Također, možemo dodati element u neki od elemenata liste.

```
> lista2$Biljka[2] <- "Hibiskus"
> lista2
```

```
## $Drzava
## [1] "Hrvatska"
##
## $Grad
## [1] "Hvar"
##
## $Selo
## [1] "Heki"
##
## $Rijeka
## [1] "Hudinja"
```

```
##
## $Jezero
## [1] "Harambašić"
##
## $Opcina
## [1] "Hum na Sutli"
##
## $Planina
## [1] "Hahlići"
##
## $Zivotinja
## [1] "Hrcak"     "Hobotnica" "Hijena"
##
## $Biljka
## [1] "Hren"     "Hibiskus"
```

Ako element koji se pridružuje nije potrebno imenovati, onda se pridruživanje može izvršiti na sljedeći način:

```
> lista1[5] <- "50 %"
> lista1
```

```
## $broj
## [1] 5
##
## $grad
## [1] "Pula"
##
## $vektor
## [1] 1 2 3 4 5
##
## $matrica
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
## [4,]    1    2    3
## [5,]    4    5    6
## [6,]    7    8    9
##
## [[5]]
## [1] "50 %"
```

7.4.4 Izmjena ili brisanje elementa

Ako se želi **izbrisati element**, onda je to moguće na sljedeći način:

```
> lista1[[1]] <- NULL
> lista1
```

```
## $grad
## [1] "Pula"
##
## $vektor
## [1] 1 2 3 4 5
##
## $matrica
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
## [4,]    1    2    3
## [5,]    4    5    6
## [6,]    7    8    9
##
## [[4]]
## [1] "50 %"
```

Izmjena elementa:

```
> lista1[[2]] <- c(6, 7, 8, 9, 10)
> lista1
```

```
## $grad
## [1] "Pula"
##
## $vektor
## [1]  6  7  8  9 10
##
## $matrica
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
## [4,]    1    2    3
## [5,]    4    5    6
## [6,]    7    8    9
##
## [[4]]
## [1] "50 %"
```

7.4.5 Spajanje lista

Nadalje, slično kao što se mogu spajati vektori i matrice, mogu se **spajati** i liste.

```
> lista3 <- list(lista1, lista2)
> lista3
```

```
## [[1]]
## [[1]]$grad
## [1] "Pula"
##
## [[1]]$vektor
## [1]  6  7  8  9 10
##
## [[1]]$matrica
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
## [4,]    1    2    3
## [5,]    4    5    6
## [6,]    7    8    9
##
## [[1]][[4]]
## [1] "50 %"
##
##
## [[2]]
## [[2]]$Drzava
## [1] "Hrvatska"
##
## [[2]]$Grad
## [1] "Hvar"
##
## [[2]]$Selo
## [1] "Heki"
##
## [[2]]$Rijeka
## [1] "Hudinja"
##
## [[2]]$Jezero
## [1] "Harambašić"
##
## [[2]]$Opcina
## [1] "Hum na Sutli"
##
## [[2]]$Planina
## [1] "Hahlići"
##
## [[2]]$Zivotinja
## [1] "Hrcak"     "Hobotnica" "Hijena"
```

```
##
## [[2]]$Biljka
## [1] "Hren"     "Hibiskus"
```

Alternativno, liste se mogu spojiti i kombiniranjem:

```
> lista4 <- c(lista1, lista2)
> lista4
```

```
## $grad
## [1] "Pula"
##
## $vektor
## [1]  6  7  8  9 10
##
## $matrica
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
## [4,]    1    2    3
## [5,]    4    5    6
## [6,]    7    8    9
##
## [[4]]
## [1] "50 %"
##
## $Drzava
## [1] "Hrvatska"
##
## $Grad
## [1] "Hvar"
##
## $Selo
## [1] "Heki"
##
## $Rijeka
## [1] "Hudinja"
##
## $Jezero
## [1] "Harambašić"
##
## $Opcina
## [1] "Hum na Sutli"
##
## $Planina
## [1] "Hahlići"
```

```
##
## $Zivotinja
## [1] "Hrcak"     "Hobotnica" "Hijena"
##
## $Biljka
## [1] "Hren"     "Hibiskus"
```

7.4.6 Razdvajanje lista

Razdvajanje liste vršimo pomoću `unlist()`.

```
> unlist(lista4)
```

```
##           grad        vektor1        vektor2        vektor3        vektor4
##          "Pula"            "6"            "7"            "8"            "9"
##        vektor5       matrica1       matrica2       matrica3       matrica4
##           "10"           "10"           "11"           "12"            "1"
##       matrica5       matrica6       matrica7       matrica8       matrica9
##            "4"            "7"           "13"           "14"           "15"
##      matrica10      matrica11      matrica12      matrica13      matrica14
##            "2"            "5"            "8"           "16"           "17"
##      matrica15      matrica16      matrica17      matrica18
##           "18"            "3"            "6"            "9"         "50 %"
##         Drzava           Grad           Selo         Rijeka         Jezero
##     "Hrvatska"         "Hvar"         "Heki"      "Hudinja"   "Harambašić"
##         Opcina        Planina     Zivotinja1     Zivotinja2     Zivotinja3
## "Hum na Sutli"      "Hahlići"        "Hrcak"    "Hobotnica"       "Hijena"
##        Biljka1        Biljka2
##         "Hren"     "Hibiskus"
```

Funkcija `unlist()` pretvara listu u vektor, pa možete uočiti da su svi elementi matrice zasebno odvojeni. Isprobajte primjenu `unlist()` na *lista3* i usporedite rezultate.

Ako su listi već pridodani **atributi** (nazivi), oni se mogu provjeriti/ ispisati ponoću `attributes()` ili `attr()`.

```
> attributes(lista2)
```

```
## $names
## [1] "Drzava"    "Grad"      "Selo"      "Rijeka"    "Jezero"    "Opcina"
## [7] "Planina"   "Zivotinja" "Biljka"
```

```
> attr(lista1, "names")
```

```
## [1] "grad"    "vektor"  "matrica" ""
```

7.5 Prednosti i nedostaci rada s listama te najčešće greške

Uobičajene greške pri radu s listama

- **Neispravan pristup elementima**: greške u pristupu elementima liste mogu se dogoditi ako koristite jednostruke uglate zagrade (`[ ]`) umjesto dvostrukih (`[[ ]]`) za pristup elementima liste. Jednostruke zagrade vraćaju podlistu, dok dvostruke vraćaju element.
- **Zaboravljanje heterogenosti podataka**: budući da liste mogu sadržavati različite tipove podataka, nemojte zaboraviti na ovu činjenicu pri manipulaciji ili analizi podataka unutar liste, jer to može dovesti do neočekivanih rezultata ili grešaka.
- **Zadavanje (nepotrebno) složene strukture podataka**: Ponekad korisnici nepotrebno kompliciraju strukturu podataka koristeći liste kada bi jednostavnija struktura podataka, kao što je vektor ili matrica, bila adekvatnija i efikasnija za određeni zadatak.

Prednosti korištenja lista

- **Fleksibilnost**: liste u R-u mogu sadržavati elemente različitih tipova, uključujući brojeve, znakove, vektore, matrice, pa čak i druge liste. Ova fleksibilnost omogućava da u jednom objektu pohranimo kompleksne skupove podataka.
- **Strukturiranje složenih podataka**: Liste su idealne za organiziranje složenih i hijerarhijskih podataka. Primjerice, možete imati listu koja sadrži podatke o različitim geografskim regijama, gdje svaki element liste predstavlja određenu regiju s vlastitom podlistom podataka relevantnih za tu regiju.
- **Spremanje funkcija i njihovih rezultata**: Liste se mogu koristiti za spremanje niza funkcija ili čak rezultata izvršenja funkcija, što omogućava jednostavan pristup i analizu tih podataka kasnije.

Kad koristiti liste

- **Za rad s heterogenim podacima**: liste su idealan izbor kada trebate raditi s nizom objekta različitih tipova podataka koje trebate držati zajedno.
- **Za složene strukture podataka**: ako vaši podaci imaju složenu ili hijerarhijsku strukturu, liste omogućuju efikasno grupiranje i organizaciju tih podataka.
- **Kada funkcije vraćaju više od jednog rezultata**: liste su korisne za spremanje rezultata funkcija koje vraćaju više od jednog objekta, omogućavajući vam da pohranite sve rezultate zajedno i pristupate im po potrebi.

Pitanja za ponavljanje

1. Kreirali ste listu `moja_lista <- list(ime = "Ivan", godine = 25, grad = "Zagreb")`. Koja će naredba ispisati vrijednost elementa „godine" iz ove liste?

2. Ako imate listu `lista_a <- list(1, "tekst", c(TRUE, FALSE))`, kako biste pristupili drugom elementu liste, a kako biste dobili drugi element unutar trećeg elementa liste?

3. Kako biste dodali novi element naziva "država" s vrijednošću "Hrvatska" postojećoj listi moja_lista iz prvog pitanja?

4. Kako bi izgledao rezultat naredbe `unlist(list(5, "grad", TRUE))` i koji tip će imati dobiveni objekt?

5. Ako imate listu `moji_podaci <- list(broj = 100, naziv = "Projekt", podaci = c(1, 2, 3))`, što će ispisati naredba `moji_podaci$podaci[2]`?

6. Kreirali ste listu `nova_lista <- list(ime = "Ana", godine = 30, grad = "Split")`. Napišite naredbu koja će promijeniti vrijednost elementa „godine" na 31.

7. Ako imate dvije liste `lista1 <- list(a = 1, b = 2)` i `lista2 <- list(c = 3, d = 4)`, koja će naredba spojiti ove dvije liste u jednu listu?

8. Što će ispisati `lista2$Selo` za listu *lista2* definiranu kao `lista2 <- list(Selo = "Heki", Rijeka = "Hudinja")`? Objasnite rezultat.

9. Što će se dogoditi ako pokušate pristupiti elementu `lista1[[5]]`, a lista *lista1* ima samo tri elementa? Kakav će rezultat R vratiti?

10. U listi `lista3 <- list(A = matrix(1:4, 2, 2), B = c("Jabuka", "Kruška"))`, kako biste ispisali element u drugom stupcu i prvom retku matrice pohranjene pod nazivom „A"?

7.6 Kreiranje podatkovnih okvira (data frame)

U ovom poglavlju objašnjavaju se različiti načini kreiranja podatkovnih okvira u R-u. Kroz primjere, poglavlje prikazuje četiri pristupa za konstruiranje podatkovnog okvira, ilustrirajući fleksibilnost i različite stilove kodiranja u R-u. Osim toga, naglašava važnost prilagođavanja tipa podataka unutar stupaca kako bi se izbjegle moguće greške prilikom analize.

Podatkovni okviri (data frame ili df) su *uređeni dvodimenzionalni skupovi podataka* koji se mogu sastojati od *elemenata različitih tipova*, uz uvjet da se *unutar pojedinog stupca nalaze elementi istog tipa*. Primjer:

```
##    Drzava  Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 1. Croatia 57668                     2980
## 2. France  211998                    48069
## 3. China   158606                    149720
## 4. Italy   93228.6                   61194.6
## 5. Austria 30816                     11043
## 6. Ireland 10926                     8643
```

Podaci se odnose na zadnje dostupne podatke iz 2018. godine dostupne na mrežnim stranicama Svjetske Banke, DT.INT.ARVL i ST.INT.DPRT izraženi u tisućama. Recimo da se ovi podaci žele *zapisati u obliku podatkovnog okvira*. Za to postoji više načina.

Prvi način:

```
> y <- c("Croatia", 57668.00,     2980.00,
+        "France",     211998.00,    48069.00,
+        "China",      158606.00,  149720.00,
+        "Italy",      93228.60,   61194.60,
+        "Austria",  30816.00,   11043.00,
+        "Ireland",  10926.00,     8643.00)
>
> Tablica2 <- matrix(y, nrow = 6, ncol = 3, byrow = TRUE)
> colnames(Tablica2) <- c("Drzava", "Broj_turistickih_dolazaka",
+                         "Broj_turistickih_putovanja_(odlazni)")
> rownames(Tablica2) <- c("1.", "2.", "3.", "4.", "5.", "6.")
> Turisti1<-as.data.frame(Tablica2)
>
> #s obzirom da su podaci uneseni mješovito s obzirom na tip,
> # ako se broj turističkih dolazaka i odlazaka ne definiraju kao
> # double ili integer, preuzet će tip chr
>
> Turisti1$`Broj_turistickih_dolazaka` <-
+   as.double(Turisti1$`Broj_turistickih_dolazaka`)
> Turisti1$`Broj_turistickih_putovanja_(odlazni)` <-
+   as.double(Turisti1$`Broj_turistickih_putovanja_(odlazni)`)
> Turisti1
```

```
##     Drzava Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 1. Croatia                   57668.0                                2980.0
## 2.  France                  211998.0                                48069.0
## 3.   China                  158606.0                               149720.0
## 4.   Italy                   93228.6                                61194.6
## 5. Austria                   30816.0                                11043.0
## 6. Ireland                   10926.0                                 8643.0
```

Drugi način:

```
> Drzava <- c("Croatia", "France", "China", "Italy", "Austria", "Ireland")
> Br_t_dol <- c(57668, 211998, 158606, 93228,  30816, 10926)
> Br_t_odl <- c(2980, 48069, 14972, 61194, 11043, 8643)
> Turisti2 <- data.frame(Drzava, Br_t_dol, Br_t_odl)
> Turisti2
```

```
##    Drzava Br_t_dol Br_t_odl
## 1 Croatia    57668     2980
## 2  France   211998    48069
## 3   China   158606    14972
## 4   Italy    93228    61194
## 5 Austria    30816    11043
## 6 Ireland    10926     8643
```

Treći način:

```
> D1 <- c("Croatia",    57668.00,   2980.00)
> D2 <- c("France", 211998.00,  48069.00)
> D3 <- c("China",  158606.00,  149720.00)
> D4 <- c("Italy",  93228.00,    61194.60)
> D5 <- c("Austria",    30816.00, 11043.00)
> D6 <- c("Ireland",    10926.00,   8643.00)
> Tur <- rbind(D1, D2, D3, D4, D5, D6)
> colnames(Tur) <- c("Drzava",  "Broj_turistickih_dolazaka",
+                    "Broj_turistickih_putovanja_(odlazni)")
> rownames(Tur) <- c("1.", "2.", "3.", "4.", "5.", "6.")
> Turisti3 <- as.data.frame(Tur)
>
> #s obzirom da su podaci uneseni mješovito s obzirom na tip,
> # ako se broj turističkih dolazaka i odlazaka ne definiraju kao
> # double ili integer, preuzet će tip chr
>
> Turisti3$`Broj_turistickih_dolazaka` <-
+   as.double(Turisti3$`Broj_turistickih_dolazaka`)
> Turisti3$`Broj_turistickih_putovanja_(odlazni)` <-
+   as.double(Turisti3$`Broj_turistickih_putovanja_(odlazni)`)
> Turisti3
```

```
##     Drzava Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 1. Croatia                     57668                                2980.0
## 2.  France                    211998                               48069.0
## 3.   China                    158606                              149720.0
## 4.   Italy                     93228                               61194.6
## 5. Austria                     30816                               11043.0
## 6. Ireland                     10926                                8643.0
```

Četvrti način:

```
> Turisti4 <- data.frame(Drzava=c("Croatia", "France", "China",
+                                 "Italy", "Austria", "Ireland"),
+                        Br_t_dol=c(57668, 211998, 158606,
+                                   93228, 30816, 10926),
+                        Br_t_odl=c(2980, 48069, 14972,
+                                   61194, 11043, 8643))
> Turisti4
```

```
##    Drzava Br_t_dol Br_t_odl
## 1 Croatia    57668     2980
## 2  France   211998    48069
## 3   China   158606    14972
## 4   Italy    93228    61194
## 5 Austria    30816    11043
## 6 Ireland    10926     8643
```

Može se uočiti da su drugi i četvrti način jednostavniji i brži za primjenu. Poanta ovih primjera je da uočite da je do rješenja moguće doći na različite načine. Dakako, neki od načina bit će elegantniji od drugih, ali važno je uočiti da ne postoji isključivo „jedan način na koji se nešto radi". Igrajte se, isprobavajte i otkrijte svoj stil.

Već je pri unosu podataka u podatkovni okvir na prvi i treći način primijenjen jedan oblik **odabira**. Prije nego se pozabavimo odabirima, pogledajmo što čini podatkovni okvir.

7.7 Rad s podatkovnim okvirima

U ovom poglavlju objašnjava se rad s podatkovnim okvirima, uključujući pregled i odabir elemenata, dodavanje i brisanje podataka te filtriranje i sortiranje prema specifičnim kriterijima. Poglavlje također obuhvaća kreiranje podskupova podataka, osnovne statističke analize, kao i prikaz podataka u obliku tablica. Na kraju, razmatraju se prednosti i nedostaci podatkovnih okvira, kao i najčešće greške pri radu s njima.

7.7.1 Uvid u podatkovni okvir

Ako bismo htjeli provjeriti **što podatkovni okvir sadrži**, to možemo učiniti koristeći `str()`, `head()` ili `tail()`.

```
> str(Turisti1)
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ Drzava                              : chr  "Croatia" "France" "China" "Italy" ...
##  $ Broj_turistickih_dolazaka           : num  57668 211998 158606 93229 30816 ...
##  $ Broj_turistickih_putovanja_(odlazni): num  2980 48069 149720 61195 11043 ...
```

Funkcija `str()` daje uvid u data frame (ili *df*), tj. podatkovni okvir. Ovaj podatkovni okvir sastoji se od 6 opažanja za svaku od 3 varijable. Navedene su varijable, tip varijabli i ispis prvih nekoliko elemenata (slično zapisu u prozoru Environment). Dakle, podatkovni okvir je uređen varijablama i brojem opažanja. Varijable su zapisane u stupcima, a opažanja po recima.

`head()` će standardno rezultirati prikazom prvih 6 redaka df-a. Ako bismo uz naziv df-a dodali i broj, na primjer 3, tada bi ispis sadržavao prva tri retka.

```
> head(Turisti1, 3)
```

```
##     Drzava Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 1. Croatia                     57668                                  2980
## 2.  France                    211998                                 48069
## 3.   China                    158606                                149720
```

Slična je situacija s naredbom `tail()`, samo što standardno ispisuje posljednjih 6 redaka. Ako se uz naziv podatkovnog okvira doda broj, tada će ispisati onoliko redaka koliko je definirano brojem.

```
> tail(Turisti1, 2)
```

```
##     Drzava Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 5. Austria                     30816                                 11043
## 6. Ireland                     10926                                  8643
```

Nadalje, može se koristiti i `glimpse()` iz paketa dplyr.

```
> library(dplyr)
> glimpse(Turisti1)
```

```
## Rows: 6
## Columns: 3
## $ Drzava                                  <chr> "Croatia", "France", "China", "~
## $ Broj_turistickih_dolazaka               <dbl> 57668.0, 211998.0, 158606.0, 93~
## $ `Broj_turistickih_putovanja_(odlazni)` <dbl> 2980.0, 48069.0, 149720.0, 6119~
```

Ako se podatkovni okvir pretražuje prema elementima, npr. `[1,2]`, tada će odabrani element biti prvo opažanje pripisano drugoj varijabli (prvi redak, drugi stupac). Osim toga, češće se ispituje pojedina varijabla. Promotrimo to kroz nekoliko načina **odabiranja elemenata** iz podatkovnog okvira.

7.7.2 Odabir elemenata

Također, treba voditi računa da, ako se odabir ne spremi pod novim nazivom, „odabir" će se prikazati samo u ispisu (bez kreiranja novog vektora, matrice ili df-a). Najjednostavniji je način upis znaka dolara (`$`) nakon naziva podatkovnog okvira. Po upisu znaka dolara, pokazat će se padajući izbornik s nazivima varijabli, pri čemu se može samo kliknuti na odabranu varijablu.

```
> Turisti1$Drzava
```

```
## [1] "Croatia" "France"  "China"   "Italy"   "Austria" "Ireland"
```

Slično kao kod vektora, ako je već odabran jedan stupac (varijabla) iz df-a, tada će se upisom broja u uglate zagrade izolirati taj element. Na primjer, `Turisti1$Drzava[3]`, rezultira ispisom 3. elementa varijable Država iz podatkovnog okvira Turisti1.

```
> Turisti1$Drzava[3]
```

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

No, s obzirom da je podatkovni okvir dvodimenzionalan (ima retke i stupce), element se može odabrati i brojčanim određivanjem retka i stupca. Na primjer, `Turisti1[1,2]` će rezultirati ispisom elementa koji se nalazi u prvom retku i drugom stupcu.

```
> Turisti1[1, 2]
```

```
## [1] 57668
```

Obratite pozornost na to da `Turisti1[2,1]` rezultira odabirom elementa koji se nalazi u drugom retku prvog stupca. Odnosno, `Turisti1[1,2]` $\neq$ `Turisti1[2,1]`.

```
> Turisti1[2, 1]
```

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

Sljedeći je način odabira elementa upisom naziva varijable u uglate zagrade uz naziv podatkovnog okvira. Pritom treba paziti da naziv bude unutar navodnih znakova.

```
> Turisti1["Drzava"]
```

```
##     Drzava
## 1. Croatia
## 2.  France
## 3.   China
## 4.   Italy
## 5. Austria
## 6. Ireland
```

Ako se od moguće dvije dimenzije odabere samo jedna, onda će rezultat biti prikaz retka ili stupca. Tako, na primjer, `Turisti1[2,]` rezultira prikazom drugog retka.

```
> Turisti1[2, ]
```

```
##    Drzava Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 2. France                   211998                                 48069
```

Nasuprot tome, `Turisti1[,2]` rezultirat će prikazom drugog stupca.

```
> Turisti1[ ,2]
```

```
## [1]  57668.0 211998.0 158606.0  93228.6  30816.0  10926.0
```

7.7.3 Dodavanje elemenata

```
> USA <-data.frame(Drzava = "United States",
+                  Br_t_dol = 169324.92,
+                  Br_t_odl = 157873.00)
> Turisti2<-rbind(Turisti2, USA)
> Turisti2
```

```
##          Drzava Br_t_dol Br_t_odl
## 1       Croatia  57668.0     2980
## 2        France 211998.0    48069
## 3         China 158606.0    14972
```

```
## 4         Italy  93228.0    61194
## 5       Austria  30816.0    11043
## 6       Ireland  10926.0     8643
## 7 United States 169324.9   157873
```

Recimo da se za potrebe daljnje analize želi izvršiti subset europskih država. Iako je nama lako odrediti koje države pripadaju Europi, a koje drugim kontinentima, R-u je potrebno dodati podatke koji će sadržavati potrebne karakteristike. Jedan od načina je pridodavanje kontinenta svakom opažanju, a drugi je dodavanje elemenata logičkog tipa u obliku zasebne varijable. Ovdje će se iskoristiti drugi pristup. Sličan pristup može se, općenito primijeniti pri ručnom upisu i **dodavanju varijable** postojećem podatkovnom okviru.

```
> EU <- c(TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE)
> Turisti2 <- cbind(Turisti2, EU)
> Turisti2
```

```
##          Drzava Br_t_dol Br_t_odl    EU
## 1       Croatia  57668.0     2980  TRUE
## 2        France 211998.0    48069  TRUE
## 3         China 158606.0    14972 FALSE
## 4         Italy  93228.0    61194  TRUE
## 5       Austria  30816.0    11043  TRUE
## 6       Ireland  10926.0     8643  TRUE
## 7 United States 169324.9   157873 FALSE
```

7.7.4 Podskup podatkovnog okvira prema kriteriju (filtriranje)

Ponekad je za daljnju analizu potreban **podskup podataka**, što se postiže funkcijom `subset()`. Na primjer, recimo da nas zanimaju samo one države koje su zabilježile više od 30 000 dolazaka turista.

```
> Tur_sub <- subset(Turisti1, Turisti1$`Broj_turistickih_dolazaka`>30000)
> Tur_sub
```

```
##     Drzava Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 1. Croatia                   57668.0                                2980.0
## 2.  France                  211998.0                               48069.0
## 3.   China                  158606.0                              149720.0
## 4.   Italy                   93228.6                               61194.6
## 5. Austria                   30816.0                               11043.0
```

Ili, na primjer, ako nas zanimaju sve države osim Hrvatske, onda će to izgledati ovako:

```
> Tur_noCro <- subset(Turisti1, Drzava != "Croatia")
> Tur_noCro
```

```
##     Drzava Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 2.  France                  211998.0                                48069.0
## 3.   China                  158606.0                               149720.0
## 4.   Italy                   93228.6                                61194.6
## 5. Austria                   30816.0                                11043.0
## 6. Ireland                   10926.0                                 8643.0
```

Subset je moguće napraviti prema bilo kojem kriteriju kojeg je moguće zadati temeljem postojećih podataka. U nekim situacijama postojeći podaci neće biti dovoljni za potrebe analize i to će iziskivati dodavanje elemenata i/ili varijabli.

Možete stvoriti podskup temeljen na kriteriju članstva država u EU, pri čemu se koristi stupac EU kao logički vektor, na sljedeći način:

```
> Tur_eu <- subset(Turisti2, EU==TRUE)
> Tur_eu
```

```
##    Drzava Br_t_dol Br_t_odl   EU
## 1 Croatia    57668     2980 TRUE
## 2  France   211998    48069 TRUE
## 4   Italy    93228    61194 TRUE
## 5 Austria    30816    11043 TRUE
## 6 Ireland    10926     8643 TRUE
```

Iako je dobiven podskup prema traženom kriteriju, može se uočiti da su nazivi redaka ostali isti kao ranije.

7.7.5 Sortiranje elemenata

Ako se elementi u podatkovnom okviru žele redoslijedno urediti prema vrijednostima pojedine varijable, tada se provodi **sortiranje**, koje se najčešće vrši pomoću funkcije `order()` i **uglatih zagrada**.

```
> Turisti2[order(Br_t_dol), ]
```

```
##    Drzava Br_t_dol Br_t_odl    EU
## 6 Ireland    10926     8643  TRUE
## 5 Austria    30816    11043  TRUE
## 1 Croatia    57668     2980  TRUE
## 4   Italy    93228    61194  TRUE
## 3   China   158606    14972 FALSE
## 2  France   211998    48069  TRUE
```

Prethodni je ispis rezultirao poredanim prikazom podataka iz podatkovnog okvira Turisti2.

Ako bismo htjeli provjeriti **kako podatkovni okvir izgleda** nakon primjene ove funkcije, to možemo učiniti koristeći `str()`, `head()` ili `tail()`.

```
> str(Turisti2)
```

```
## 'data.frame':    7 obs. of  4 variables:
##  $ Drzava  : chr  "Croatia" "France" "China" "Italy" ...
##  $ Br_t_dol: num  57668 211998 158606 93228 30816 ...
##  $ Br_t_odl: num  2980 48069 14972 61194 11043 ...
##  $ EU      : logi  TRUE TRUE FALSE TRUE TRUE TRUE ...
```

Dok str daje uvid u dimenzije podatkovnog skupa, uz popis varijabli, njihovih vrsta te ispisa prvih nekoliko opažanja, head daje ispis prvih 6 redaka podatkovnog skupa.

```
> head(Turisti2)
```

```
##    Drzava Br_t_dol Br_t_odl    EU
## 1 Croatia    57668     2980  TRUE
## 2  France   211998    48069  TRUE
## 3   China   158606    14972 FALSE
## 4   Italy    93228    61194  TRUE
## 5 Austria    30816    11043  TRUE
## 6 Ireland    10926     8643  TRUE
```

funkcija `head()` zadano (*engl. default*) prikazuje prvih 6 redaka podatkovnog okvira, a taj se broj može promijeniti, na primjer:

```
> head(Turisti2, 3)
```

```
##    Drzava Br_t_dol Br_t_odl    EU
## 1 Croatia    57668     2980  TRUE
## 2  France   211998    48069  TRUE
## 3   China   158606    14972 FALSE
```

Funkcija `tail()` prikazuje posljednjih 6 redaka podatkovnog okvira, a može se podesiti i drugačiji broj redaka ovisno o potrebama.

```
> tail(Turisti2)
```

```
##          Drzava Br_t_dol Br_t_odl    EU
## 2        France 211998.0    48069  TRUE
## 3         China 158606.0    14972 FALSE
## 4         Italy  93228.0    61194  TRUE
## 5       Austria  30816.0    11043  TRUE
## 6       Ireland  10926.0     8643  TRUE
## 7 United States 169324.9   157873 FALSE
```

Ono što se može uočiti iz **pregleda podatkovnog okvira** jest da nije došlo do promjene u poretku. Da bi promjena u poretku bila evidentirana, potrebno je kreirati novi podatkovni okvir.

```
> Tur_ord <- Turisti2[order(Br_t_dol), ]
> Tur_ord
```

```
##    Drzava Br_t_dol Br_t_odl    EU
## 6 Ireland    10926     8643  TRUE
## 5 Austria    30816    11043  TRUE
## 1 Croatia    57668     2980  TRUE
## 4   Italy    93228    61194  TRUE
## 3   China   158606    14972 FALSE
## 2  France   211998    48069  TRUE
```

Funkcija `sort()` ne primjenjuje se na podatkovnom okviru, nego na vektorima, tako da ovdje nije od koristi.

```
> Turisti2[order(Drzava), ]
```

```
##    Drzava Br_t_dol Br_t_odl    EU
## 5 Austria    30816    11043  TRUE
## 3   China   158606    14972 FALSE
## 1 Croatia    57668     2980  TRUE
## 2  France   211998    48069  TRUE
## 6 Ireland    10926     8643  TRUE
## 4   Italy    93228    61194  TRUE
```

Obratite pozornost na to da je `order()` moguće primijeniti i na elementima tipa character. Nadalje, funkcija `order()` će uvijek sortirati vrijednosti silazno, ako drugačije nije navedeno. Redoslijed prema kriteriju uzlaznog povećavanja vrijednosti elemenata odlaznih turista moguće je zadati:

```
> Turisti2[order(Br_t_odl, decreasing = TRUE), ]
```

```
##    Drzava Br_t_dol Br_t_odl    EU
## 4   Italy    93228    61194  TRUE
## 2  France   211998    48069  TRUE
## 3   China   158606    14972 FALSE
## 5 Austria    30816    11043  TRUE
## 6 Ireland    10926     8643  TRUE
## 1 Croatia    57668     2980  TRUE
```

7.7.6 Brisanje elemenata

Nadalje, tijekom pripreme podataka za analizu može se pojaviti potreba za **brisanje**m elemenata, opažanja ili varijabli. **Brisanje elementa** zapravo se vrši putem zamjene elementa s oznakom za nedostupnu vrijednost (Not Available/Missing Values), koja predstavlja logičku konstantu duljine 1, a R ju tumači kao indikator nepostojeće vrijednosti.

```
> Turisti2[5, 4] <- NA
> Turisti2
```

```
##          Drzava Br_t_dol Br_t_odl    EU
## 1       Croatia  57668.0     2980  TRUE
## 2        France 211998.0    48069  TRUE
## 3         China 158606.0    14972 FALSE
## 4         Italy  93228.0    61194  TRUE
## 5       Austria  30816.0    11043    NA
## 6       Ireland  10926.0     8643  TRUE
## 7 United States 169324.9   157873 FALSE
```

Brisanje stupca:

```
> Turisti2$EU <- NULL
> Turisti2
```

```
##          Drzava Br_t_dol Br_t_odl
## 1       Croatia  57668.0     2980
## 2        France 211998.0    48069
## 3         China 158606.0    14972
## 4         Italy  93228.0    61194
## 5       Austria  30816.0    11043
## 6       Ireland  10926.0     8643
## 7 United States 169324.9   157873
```

Napomena: Izmjene vezane za dodavanje i brisanje elemenata izravno utječu na postojeći dataset, bez potrebe za kreiranjem novog objekta.

Drugi način odnosi se na prikaz bez pojedinog stupca i tehnički taj stupac neće biti obrisan, samo neće biti prikazan.

```
> Turisti2[-2]
```

```
##          Drzava Br_t_odl
## 1       Croatia     2980
## 2        France    48069
## 3         China    14972
## 4         Italy    61194
## 5       Austria    11043
## 6       Ireland     8643
## 7 United States   157873
```

Da bi se stupac izbrisao koristeći ovaj pristup, potrebno je kreirati novi objekt:

```
> Tur_odl <- Turisti2[-2]
```

Na sličan način, mogu se brisati reci ili samo ukloniti iz prikaza.

```
> Tur0 <- Turisti1[-1, ]
> Tur0
```

```
##     Drzava Broj_turistickih_dolazaka Broj_turistickih_putovanja_(odlazni)
## 2.  France                  211998.0                                48069.0
## 3.   China                  158606.0                               149720.0
## 4.   Italy                   93228.6                                61194.6
## 5. Austria                   30816.0                                11043.0
## 6. Ireland                   10926.0                                 8643.0
```

Također, moguće je ukloniti više opažanja istovremeno, navodeći pripadajuće retke na prvom mjestu unutar uglatih zagrada, koristeći funkciju combine.

```
> Tur_noEU <- Turisti2[-c(1, 2, 4, 5, 6), ]
> Tur_noEU
```

```
##          Drzava Br_t_dol Br_t_odl
## 3         China 158606.0    14972
## 7 United States 169324.9   157873
```

Osim toga, moguće je izvršiti i kombinaciju pretraživanja po varijablama i zamjenu elementa, na primjer:

```
> #Za Br_t_dol Francuske, želi se zamijeniti postojeća vrijednost s NA
> Turisti2$Br_t_dol[Turisti2$Drzava == "France"] <- NA
> Turisti2
```

```
##          Drzava Br_t_dol Br_t_odl
## 1       Croatia  57668.0     2980
## 2        France       NA    48069
## 3         China 158606.0    14972
## 4         Italy  93228.0    61194
## 5       Austria  30816.0    11043
## 6       Ireland  10926.0     8643
## 7 United States 169324.9   157873
```

Iako u ovako jednostavnim primjerima, ovakav pristup djeluje suvišno, u zahtjevnijim skupovima podataka koji broje tisuće elemenata, ovaj je pristup često učinkovitiji.

7.7.7 Učitavanje podatkovnog okvira sadržanog u R-u

Postoje neki podatkovni okviri koji su već uključeni u R, a brojni su uključeni u različite pakete. Jedan od češće korištenih podatkovnih okvira za vježbu i primjere je *iris*. Podatkovni okvir *iris* sadrži opažanja o duljini i širini latica za tri vrste cvijeća iris (Iris setosa, versicolor i virginica). U spomenutom podatkovnom okviru postoji 150 opažanja (redaka) i 5 varijabli (stupaca), s nazivima: *Sepal.Length*, *Sepal.Width*, *Petal.Length*, *Petal.Width* i *Species*.

Podatkovni okvir koji je sadržan u R-u ili nekom od već instaliranih paketa može se pozvati koristeći naredbu `data()`, na primjer, na sljedeći način:

```
> data("iris")
```

Prvo, nakon što upišete data i otvorite zagradu, nudi se izbor dostupnih podatkovnih okvira. Drugo, ako vas zanimaju detalji o pojedinom podatkovnom okviru, potrebno je naziv upisati u tražilicu u prozoru Help (donji desni prozor).

Provjerimo kako skup podataka izgleda.

```
> 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 ...
```

```
> head(iris, 5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

Ako ne želimo stalno koristiti operator $ u kombinaciji s nazivom podatkovnog okvira, korisna je funkcija `attach()`. Ta će funkcija pridružiti podatkovni okvir R-ovom putu traženja, što omogućuje izravno pozivanje varijabli putem imena.

```
> attach(iris)
```

7.7.8 Izračun pokazatelja deskriptivne statistike

Izračunajmo nekoliko jednostavnih pokazatelja.

```
> mean(Sepal.Length)
```

```
## [1] 5.843333
```

```
> fivenum(Sepal.Width)
```

```
## [1] 2.0 2.8 3.0 3.3 4.4
```

```
> sd(Sepal.Length)
```

```
## [1] 0.8280661
```

Kombinacija statističkih pokazatelja, s naglaskom na mjere središnje tendencije, može se dobiti naredbom `summary()`.

```
> summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Ako želimo izračunati uobičajene pokazatelje deskriptivne statistike, to možemo učiniti na način da izračunamo svaki od pokazatelja, a rezultate pohranimo u listu rezultata.

```
> mean_Sepal.Length <- mean(Sepal.Length)
> median_Sepal.Length <- median(Sepal.Length)
> mode_Sepal.Length <- as.numeric(names(sort(table(Sepal.Length),
+                                            decreasing = TRUE))[1])
>
> # Kvartili
> quartiles <- quantile(Sepal.Length)
>
> # Mjere disperzije
```

```
> range_Sepal.Length <- diff(range(Sepal.Length))
> iqr_Sepal.Length <- IQR(Sepal.Length)
> mad_Sepal.Length <- mad(Sepal.Length)
> var_Sepal.Length <- var(Sepal.Length)
> sd_Sepal.Length <- sd(Sepal.Length)
> cv_Sepal.Length <- sd_Sepal.Length / mean_Sepal.Length * 100
>
> # Mjere oblika distribucije
> library(e1071)
> skewness_Sepal.Length <- skewness(Sepal.Length)
> kurtosis_Sepal.Length <- kurtosis(Sepal.Length)
>
> # Prikaz rezultata
> mjere_deskriptivne_statistike_Sepal.Length <- list(
+   Prosjek = mean_Sepal.Length,
+   Medijan = median_Sepal.Length,
+   Mod = mode_Sepal.Length,
+   PrviKvartil = quartiles[2],
+   TreciKvartil = quartiles[4],
+   Raspon = range_Sepal.Length,
+   IQR = iqr_Sepal.Length,
+   MAD = mad_Sepal.Length,
+   Varijanca = var_Sepal.Length,
+   StandardnaDevijacija = sd_Sepal.Length,
+   KoeficijentVarijacije = cv_Sepal.Length,
+   Asimetrija = skewness_Sepal.Length,
+   Zaobljenost = kurtosis_Sepal.Length
+ )
>
> mjere_deskriptivne_statistike_Sepal.Length
```

```
## $Prosjek
## [1] 5.843333
##
## $Medijan
## [1] 5.8
##
## $Mod
## [1] 5
##
## $PrviKvartil
## 25%
## 5.1
##
## $TreciKvartil
## 75%
## 6.4
```

```
##
## $Raspon
## [1] 3.6
##
## $IQR
## [1] 1.3
##
## $MAD
## [1] 1.03782
##
## $Varijanca
## [1] 0.6856935
##
## $StandardnaDevijacija
## [1] 0.8280661
##
## $KoeficijentVarijacije
## [1] 14.17113
##
## $Asimetrija
## [1] 0.3086407
##
## $Zaobljenost
## [1] -0.6058125
```

Ovdje korištene naredbe mogu se primjenjivati samo na vektorima. To znači da bismo koristeći ovakav pristup morali računati sve ove pokazatelje zasebno, za svaku varijablu u podatkovnom okviru.

Alternativno, postoje dva paketa koja se najčešće koriste za opisivanje podataka. Oba paketa koja će se ovdje prikazati mogu se koristiti za izračun pokazatelja deskriptivne statistike pojedine varijable (zapisane u vektoru) ili više varijabli zajedno (zapisanih u podatkovnom okviru).

Prvi je paket `psych`, a izračunati pokazatelji deskriptivne statistike dohvaćaju se naredbom `describe()`. Za više mogućnosti podešavanja, pročitajte stranice pomoći paketa.

```
> library(psych)
> pokazatelji_iris <- describe(iris, IQR = TRUE)
> knitr::kable(pokazatelji_iris[,1:10], align = "c", digits = 2, format = "latex")
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range
Sepal.Length	1	150	5.84	0.83	5.80	5.81	1.04	4.3	7.9	3.6
Sepal.Width	2	150	3.06	0.44	3.00	3.04	0.44	2.0	4.4	2.4
Petal.Length	3	150	3.76	1.77	4.35	3.76	1.85	1.0	6.9	5.9
Petal.Width	4	150	1.20	0.76	1.30	1.18	1.04	0.1	2.5	2.4
Species*	5	150	2.00	0.82	2.00	2.00	1.48	1.0	3.0	2.0

```
> knitr::kable(pokazatelji_iris[,11:14], align = "c", digits = 2, format = "latex")
```

	skew	kurtosis	se	IQR
Sepal.Length	0.31	-0.61	0.07	1.3
Sepal.Width	0.31	0.14	0.04	0.5
Petal.Length	-0.27	-1.42	0.14	3.5
Petal.Width	-0.10	-1.36	0.06	1.5
Species*	0.00	-1.52	0.07	2.0

Drugi paket odnosi se na `summarytools`, koji sadrži naredbu `descr()`. Ova naredba sadrži brojne mogućnosti podešavanja, uključujući i podešavanja outputa. Ovdje su uključeni argumenti `stats = "all"` i `style = "rmarkdown"` radi ilustracije. Više mogućnosti naći ćete na stranicama pomoći paketa, odnosno ove naredbe.

```
> library(summarytools)
> descr(iris, stats = "all", style = "rmarkdown", size = 150)
```

```
## ### Descriptive Statistics
## #### iris
## **N:** 150
##
## |            | Petal.Length | Petal.Width | Sepal.Length | Sepal.Width |
## |----------------:|-------------:|------------:|-------------:|------------:|
## |        **Mean** |         3.76 |        1.20 |         5.84 |        3.06 |
## |     **Std.Dev** |         1.77 |        0.76 |         0.83 |        0.44 |
## |         **Min** |         1.00 |        0.10 |         4.30 |        2.00 |
## |          **Q1** |         1.60 |        0.30 |         5.10 |        2.80 |
## |      **Median** |         4.35 |        1.30 |         5.80 |        3.00 |
## |          **Q3** |         5.10 |        1.80 |         6.40 |        3.30 |
## |         **Max** |         6.90 |        2.50 |         7.90 |        4.40 |
## |         **MAD** |         1.85 |        1.04 |         1.04 |        0.44 |
## |         **IQR** |         3.50 |        1.50 |         1.30 |        0.50 |
## |          **CV** |         0.47 |        0.64 |         0.14 |        0.14 |
## |    **Skewness** |        -0.27 |       -0.10 |         0.31 |        0.31 |
## | **SE.Skewness** |         0.20 |        0.20 |         0.20 |        0.20 |
## |    **Kurtosis** |        -1.42 |       -1.36 |        -0.61 |        0.14 |
## |     **N.Valid** |       150.00 |      150.00 |       150.00 |      150.00 |
## |   **Pct.Valid** |       100.00 |      100.00 |       100.00 |      100.00 |
```

7.7.9 Pretvaranje varijable u faktor unutar postojećeg podatkovnog okvira

Kako bi se ilustrirala pretvorba varijable u faktor, koristit će se podatkovni okvir `mtcars`, dostupan u R-u. S obzirom da je unaprijed instaliran s R-om i dostupan u osnovnoj instalaciji, može se odmah koristiti bez dodatnih preuzimanja ili instalacija. `mtcars` predstavlja tehničke karakteristike 32 automobila iz 1973. i 1974. godine, koji su bili uključeni u američki časopis Motor Trend.

```
> data(mtcars)
> head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
> str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Ovdje su *cyl* (broj cilindara) i *gear* (broj brzina) numeričke varijable, ali se i cjelobrojne kvantitativne varijable mogu tretirati kao kategorije za potrebe analize. Pritom traba imati u vidu da je to prikladan/ praktičan odabir samo ako te varijabli sadrže mali broj modaliteta. Nadalje, varijabla *vs* označava tip motora, a varijabla *am* razlikuje automatski i ručni prijenos.

```
> mtcars$cyl <- factor(mtcars$cyl, levels = c(4, 6, 8), ordered = TRUE,
+                      labels = c("4_cilindra", "6_cilindara", "8_cilindara"))
> mtcars$gear <- factor(mtcars$gear, levels = c(3, 4, 5), ordered = TRUE,
+                       labels = c("3_brzine", "4_brzine", "5_brzina"))
> mtcars$vs <- factor(mtcars$vs,
+                     labels = c("V_motor", "Linijski_motor"))
> mtcars$am <- factor(mtcars$am,
+                     labels = c("Automatski_prijenos", "Ručni_prijenos"))
```

Ako želimo sad provjeriti kako podatkovni okvir izgleda, možemo koristiti `str()`.

```
> str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
```

```
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : Ord.factor w/ 3 levels "4_cilindra"<"6_cilindara"<..: 2 2 1 2 3 2 3 1 1 2 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : Factor w/ 2 levels "V_motor","Linijski_motor": 1 1 2 2 1 2 1 2 2 2 ...
##  $ am  : Factor w/ 2 levels "Automatski_prijenos",..: 2 2 2 1 1 1 1 1 1 1 ...
##  $ gear: Ord.factor w/ 3 levels "3_brzine"<"4_brzine"<..: 2 2 2 1 1 1 1 2 2 2 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

S obzirom da se faktori tretiraju kao kvalitativne varijable, možemo kreirati tablicu kontingencije. Ta tablica bilježi apsolutne frekvencije svake kombinacije modaliteta promatranih dviju varijabli.

```
> table(mtcars$am, mtcars$gear)
```

```
##
##                      3_brzine 4_brzine 5_brzina
##   Automatski_prijenos       15        4        0
##   Ručni_prijenos             0        8        5
```

No, faktori su češće korisni za ispitivanje karakteristika podskupina, a to se može napraviti putem izračuna pokazatelja deskriptivne statistike za svaku skupinu. U datasetu mtcars, varijabla *mpg* (*engl. miles per gallon* ili milje po galonu) mjeri efikasnost potrošnje goriva automobila. Veća vrijednost označava bolju ekonomičnost goriva. Varijabla *hp* (*engl. horsepower* ili konjska snaga) mjeri snagu motora automobila te veće vrijednosti označavaju snažnije motore.

```
> library(psych)
> deskr_stat <- describeBy(mtcars[, c("mpg", "hp")], group = mtcars$am)
> deskr_stat$Automatski_prijenos
```

```
##     vars  n   mean    sd median trimmed   mad  min   max range  skew kurtosis
## mpg    1 19  17.15  3.83   17.3   17.12  3.11 10.4  24.4    14  0.01    -0.80
## hp     2 19 160.26 53.91  175.0  161.06 77.10 62.0 245.0   183 -0.01    -1.21
##        se
## mpg  0.88
## hp  12.37
```

```
> deskr_stat$Rucni_prijenos
```

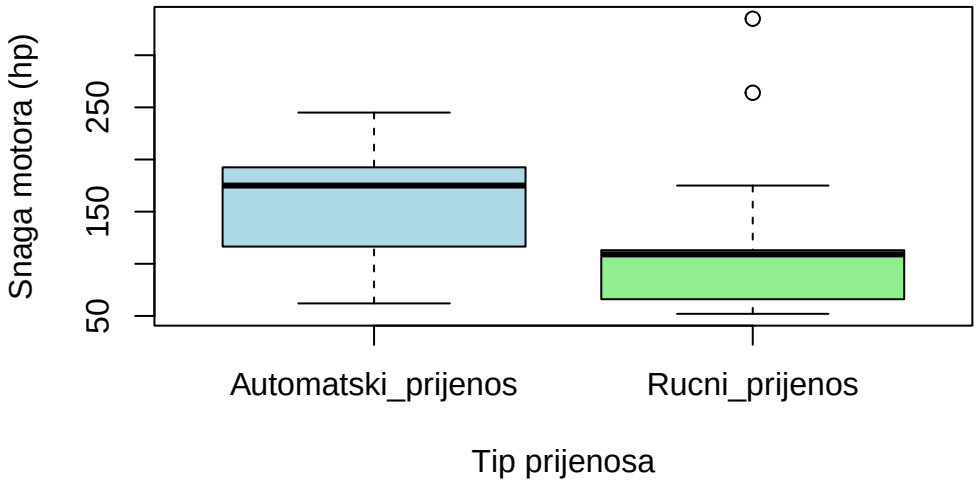
```
## NULL
```

Ovdje je upotrijebljena naredba `describeBy()` iz paketa *psych*, koja računa statističke pokazatelje za svaku grupu određenu faktorom.Ova naredba pohranjuje rezultate u listu, spremljenu pod

nazivom *deskr_stat*. Ovdje se pozivaju sažeci `Automatski_prijenos` i `Rucni_prijenos` odvojeno, radi preglednijeg ispisa.

Nadalje, faktori mogu biti korisni i za kreiranje grafičkih prikaza temeljem kojih možemo dobiti detaljnije uvide. Recimo da nas zanimaju uvidi u distribuciju konjskih snaga motora s obzirom na to imaju li ručni ili automatski prijenos.

```
> boxplot(hp ~ am, data = mtcars,
+         xlab = "Tip prijenosa",
+         ylab = "Snaga motora (hp)",
+         col = c("lightblue", "lightgreen"))
```



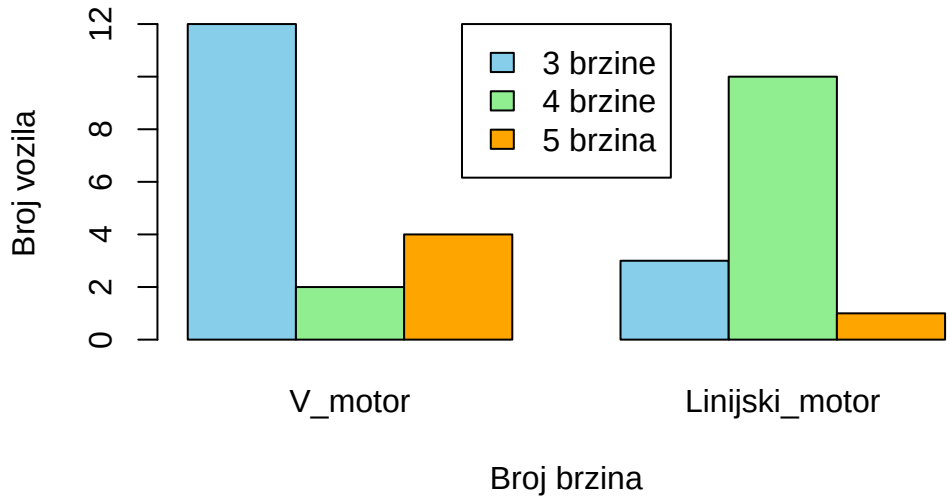
Korištenje faktora omogućuje podjelu podataka na temelju kategorija, što donosi nekoliko važnih prednosti iz perspektive analize podataka. Box-plotovi na grafikonu jasno pokazuju distribuciju snage motora (hp) između vozila s automatskim i ručnim prijenosom - dakle, omogućuje vizualnu usporedbu dviju grupa, odnosno distribucija dviju grupa. Ovdje možemo vidjeti da vozila s ručnim prijenosom imaju nižu medijalnu vrijednost snage motora, dok vozila s automatskim prijenosom pokazuju viši medijan i širi interkvartil.

Osim toga, u ovom slučaju faktorizacija pomaže u identifikaciji varijabilnosti. Možemo uočiti da su vrijednosti snage motora za vozila s ručnim prijenosom grupirane bliže medijanu, što ukazuje na manju varijabilnost. Nasuprot tome, vozila s automatskim prijenosom imaju širi raspon varijacija, što ukazuje na veću raznolikost u snazi motora.

Nadalje, iz perspektive upotrebe R-a, faktori pojednostavljuju proces grupiranja i daljnju manipulaciju podacima. Ako je varijabla *am* postavljena kao faktor, podaci se lako mogu filtrirati, grupirati i analizirati pomoću funkcija dostupnih u paketima *tapply*, *dplyr* ili *ggplot2*.

Evo još jednog primjera, recimo da želimo kreirati stupčasti dijagram broja brzina prijenosa (*gear*) s obzirom na to kakav je motor (*vs*).

```
> freq_table <- table(mtcars$gear, mtcars$vs)
>
> barplot(
+   freq_table,
+   beside = TRUE,                # Prikazuje stupce jedan pored drugog
+   col = c("skyblue", "lightgreen", "orange"),
+   legend.text = c("3 brzine", "4 brzine", "5 brzina"),
+   args.legend = list(x = "top"),
+   xlab = "Broj brzina",
+   ylab = "Broj vozila"
+ )
```



U ovom primjeru, kreiran je stupčasti dijagram koji usporedno prikazuje broj vozila s *V_motorom* i *linijskim_motorom* prema broju brzina. Možemo odmah primijetiti razlike u popularnosti pojedinih kombinacija (npr. *v_motor* s 3 brzine je češći od *linijskog_motora* s 3 brzine).

Ovi primjeri ilustrirali su upotrebu faktora pri analizi podataka u sklopu podatkovnih okvira. Dodatne primjere ilustracija i analize podataka možete naći u poglavljima "Jednostavne vizualizacije" i "Primjer upravljanja podacima."

7.7.10 Pretvaranje podatkovnog okvira u tablicu

Podatkovni okvir je najčešće korištena struktura pri analizi podataka. A sad, promotrimo mogućnosti kreiranja tablice. Iako se čitav podatkovni okvir može pretvoriti u tablicu (koristeći `as.table()`), to nije korisno ni preporučljivo. Češće će biti korisno izolirati elemente koji se žele prikazati u tablici.

```
> table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

```
> table(Sepal.Length, Species)
```

```
##              Species
## Sepal.Length setosa versicolor virginica
##          4.3      1          0         0
##          4.4      3          0         0
##          4.5      1          0         0
##          4.6      4          0         0
##          4.7      2          0         0
##          4.8      5          0         0
##          4.9      4          1         1
##          5        8          2         0
##          5.1      8          1         0
##          5.2      3          1         0
##          5.3      1          0         0
##          5.4      5          1         0
##          5.5      2          5         0
##          5.6      0          5         1
##          5.7      2          5         1
##          5.8      1          3         3
##          5.9      0          2         1
##          6        0          4         2
##          6.1      0          4         2
##          6.2      0          2         2
##          6.3      0          3         6
##          6.4      0          2         5
##          6.5      0          1         4
##          6.6      0          2         0
##          6.7      0          3         5
##          6.8      0          1         2
##          6.9      0          1         3
##          7        0          1         0
##          7.1      0          0         1
##          7.2      0          0         3
##          7.3      0          0         1
##          7.4      0          0         1
##          7.6      0          0         1
##          7.7      0          0         4
##          7.9      0          0         1
```

Iskoristimo prethodno kreiran podatkovni okvir za drugačiji prikaz.

```
> table(Turisti2$Drzava)
```

```
##
##       Austria         China       Croatia        France       Ireland
##             1             1             1             1             1
##         Italy United States
##             1             1
```

```
> table(Turisti2$Drzava, Turisti2$Br_t_dol)
```

```
##
##                10926 30816 57668 93228 158606 169324.92
##   Austria          0     1     0     0      0         0
##   China            0     0     0     0      1         0
##   Croatia          0     0     1     0      0         0
##   France           0     0     0     0      0         0
##   Ireland          1     0     0     0      0         0
##   Italy            0     0     0     1      0         0
##   United States    0     0     0     0      0         1
```

Dakle, kroz ovih nekoliko primjera možete uočiti da funkcija `table()`* koristi zadane varijable kao zaglavlja i predstupce, dok elementi tablice predstavljaju apsolutne frekvencije. Iako u slučaju *Turisti2* ne dobivamo informativnu tablicu, u nekim će situacijama ovaj pristup biti vrlo koristan, osobito za kvalitativne varijable.

7.8 Prednosti i nedostaci rada s podatkovnim okvirima te najčešće greške

Prednosti podatkovnih okvira

Svestranost: podatkovni okviri podržavaju stupce različitih tipova podataka, što ih čini idealnim za rad s heterogenim podacima tipičnim za većinu stvarnih analitičkih zadataka.

Jednostavnost upotrebe: Sintaksa za manipulaciju podatkovnim okvirima je intuitivna, a R nudi bogat skup funkcija za njihovu obradu, učitavanje, čišćenje, transformaciju, agregaciju i vizualizaciju.

Integracija s alatima za analizu podataka: mnogi paketi u R-u dizajnirani su upravo za rad s podatkovnim okvirima, uključujući popularne pakete za analizu podataka poput dplyr, tidyr i ggplot2.

Nedostaci podatkovnih okvira

Performanse i memorija

Podatkovni okviri nisu uvijek najefikasniji način za pohranu i obradu podataka, posebno kad je riječ o vrlo velikim skupovima podataka. U takvim slučajevima, alternativne strukture podataka kao što su tibbles (moderniji oblik podatkovnih okvira) ili data.table mogu pružiti bolje performanse.

Ograničenja dvodimenzionalnosti

Podatkovni okviri su prvenstveno dvodimenzionalne strukture. To ih može ograničiti u radu s višedimenzionalnim skupovima podataka.

Uobičajene greške

Pogrešan tip podataka u stupcima: ponekad, prilikom učitavanja ili transformacije podataka, stupci podatkovnog okvira mogu biti promijenjeni u pogrešan tip podataka, što može utjecati na analizu.

Neispravno indeksiranje i pristup elementima: razlika između pristupa podacima pomoću `[ ]`, `[[ ]]`, i `$` može biti zbunjujuća. To dalje može dovesti do neispravnog odabira ili promjena podataka.

Neadekvatno upravljanje vrijednostima koje nedostaju: Vrijednosti koje nedostaju (NA) mogu uzrokovati probleme prilikom izvođenja statističkih analiza ili vizualizacije podataka.

Najčešća upotreba podatkovnih okvira

Eksploracijska analiza podataka: Podatkovni okviri su osnovni alat za istraživanje, obradu i sažimanje podataka.

Statistička analiza: Većina statističkih modela u R-u, poput linearnih i logističkih regresija, prirodno radi s podatkovnim okvirima.

Vizualizacija podataka: Alati za vizualizaciju, kao što je ggplot2, dizajnirani su za rad s podatkovnim okvirima, olakšavajući izradu složenih grafikona.

Učitavanje i manipulacija podacima: Podatkovni okviri su često korišten format za učitavanje, čišćenje i pripremu podataka prije dublje analize.

Pitanja za ponavljanje

1. Što su podatkovni okviri u R-u i koja je njihova osnovna struktura?
2. Koja je glavna razlika između podataka u stupcima i podataka u redovima unutar podatkovnog okvira?
3. Kada je korisno koristiti `as.double()` funkciju prilikom rada s podatkovnim okvirima?
4. Koje su prednosti korištenja `data.frame()` funkcije za kreiranje podatkovnog okvira u usporedbi s direktnim unosom podataka kao matrice?
5. Kako bi trebalo pristupiti kreiranju podatkovnog okvira kada su podaci različitih tipova, poput numeričkih vrijednosti i tekstualnih oznaka?
6. U prvom primjeru, nakon što se kreira Turisti1 kao data.frame, zašto se koristi `as.double()` na stupcima Broj_turistickih_dolazaka i Broj_turistickih_putovanja_(odlazni)? Objasnite funkciju ovog dijela koda:

```
Turisti1$Broj_turistickih_dolazaka<-as.double(Turisti1$Broj_turistickih_dolazaka)
```

7. U drugom primjeru, pogledajte kod za kreiranje Turisti2. Što će prikazati sljedeća naredba: `Turisti2[1, "Drzava"]`?
8. U trećem primjeru, objasnite svrhu funkcije `rbind()` u sljedećem dijelu koda: `Tur<-rbind(D1, D2, D3, D4, D5, D6)`.
9. U četvrtom primjeru, što će vratiti sljedeća naredba: `Turisti4$Br_t_dol[3]`?
10. Kako možete promijeniti naziv stupca *Br_t_dol* u *Broj_turista_dolasci* u podatkovnom okviru Turisti4?
11. Koji je rezultat naredbe `str(Turisti1)` ako Turisti1 sadrži tri varijable s različitim tipovima podataka? Objasnite što svaka linija ispisa predstavlja.
12. Kako biste dohvatili prvi redak i treći stupac podatkovnog okvira Turisti1?
13. Kako možete dodati novi redak u podatkovni okvir Turisti2 koristeći podatke za zemlju „Japan" s dolascima od 120000 i odlascima od 110000?
14. Ako želite izdvojiti sve države s više od 50,000 dolazaka iz Turisti1, koju biste naredbu kreirali?
15. Koji je izlaz naredbe `Turisti1[order(Turisti1$Br_t_dol, decreasing = TRUE),]`? Što se događa ako izostavite `decreasing = TRUE`?
16. Kako možete dodati logički stupac naziva EU u podatkovni okvir Turisti2, gdje će vrijednost biti TRUE za države članice EU, a FALSE za ostale?
17. Napišite naredbu koja vraća samo prvi i treći stupac podatkovnog okvira Turisti1.
18. Što se događa kada koristite `attach(iris)`? Kako se pristupa varijabli Species nakon što se koristi `attach()`?
19. Ako želite obrisati stupac EU iz Turisti2, koji ćete kod koristiti?
20. Koji je rezultat `table(iris$Species)` i što ova tablica prikazuje?

7.9 Kreiranje tablica

U ovom poglavlju objašnjava se kreiranje tablica, uključujući postupke za formatiranje i ispis tablica pomoću funkcija te razmatranje tibblea kao moderne alternative podatkovnim okvirima. Poglavlje naglašava važnost čitljivosti i prilagodbe tablica, kao i upotrebu tibblea za jednostavnije manipulacije i bolju integraciju s paketima iz tidyverse ekosustava. Osim toga, daje se pregled uobičajenih grešaka pri formatiranju i pružaju se smjernice za najbolje prakse u kreiranju tablica, čime se olakšava predstavljanje podataka na jasan i estetski prihvatljiv način.

Tablica s početka priče, koja prikazuje osnovne tipove podataka kreirana je kao matrica i ispisana koristeći paket knitr (Xie and al 2023) i naredbu `kable()`. Sad će se prikazati točno kako.

```
> library(knitr)
> x <- c("double",  "1, 2.33, 50.99, 3e10", "realni brojevi",
+                 "integer",    "5l, -2, 5387l",    "cijeli brojevi",
+                 "character",  "slova, oznake, 50 %, pa}", "znakovni tip",
+                 "logical",    "TRUE, FALSE, T, F",    "logički",
+                 "complex",    "1+5i, 5-2i", "kompleksni brojevi",
+                 "raw",    "as.raw(11)",   "sirovi")
>
> Tablica <- matrix(x, nrow = 6, ncol = 3, byrow = TRUE)
> colnames(Tablica) <- c("Naziv",   "Primjer", "Vrsta")
> rownames(Tablica) <- c("1.", "2.", "3.", "4.", "5.", "6.")
> T<-as.table(Tablica)
> knitr::kable(T)
```

	Naziv	Primjer	Vrsta
1.	double	1, 2.33, 50.99, 3e10	realni brojevi
2.	integer	5l, -2, 5387l	cijeli brojevi
3.	character	slova, oznake, 50 %, pa}	znakovni tip
4.	logical	TRUE, FALSE, T, F	logički
5.	complex	1+5i, 5-2i	kompleksni brojevi
6.	raw	as.raw(11)	sirovi

U prvom koraku, kreiran je vektor koji sadrži sve podatke. Potom je vektor konvertiran u matricu odgovarajućih dimenzija. Potom su kreirani zaglavlje i predstupac, odnosno stupcima i recima su pridruženi nazivi. Tad je matrica spremljena kao tablica. Pozivom `knitr::kable()` ispisuje se tablica koja je pregledna i obogaćena izmjenično obojanim recima za lakše čitanje.

Usporedimo to s ispisom tablice bez posljednje naredbe, to jest bez upotrebe knitr paketa. Rezultat je ispis tablice, ali u usporedbi s prvom tablicom djeluje manje pregledno.

```
> x <- c("double",  "1, 2.33, 50.99, 3e10", "realni brojevi",
+                 "integer",    "5l, -2, 5387l",    "cjeli brojevi",
+                 "character",  "slova, oznake, 50 %, pa}", "znakovni tip",
+                 "logical",    "TRUE, FALSE, T, F",    "logički",
+                 "complex",    "1+5i, 5-2i", "kompleksni brojevi",
```

```
+                 "raw",    "as.raw(11)",   "sirovi")
>
> Tablica <- matrix(x, nrow = 6, ncol = 3, byrow = TRUE)
> colnames(Tablica) <- c("Naziv",   "Primjer", "Vrsta")
> rownames(Tablica) <- c("1.", "2.", "3.", "4.", "5.", "6.")
> T<-as.table(Tablica)
> T
```

```
##    Naziv     Primjer                  Vrsta
## 1. double    1, 2.33, 50.99, 3e10     realni brojevi
## 2. integer   5l, -2, 53871            cjeli brojevi
## 3. character slova, oznake, 50 %, pa} znakovni tip
## 4. logical   TRUE, FALSE, T, F        logički
## 5. complex   1+5i, 5-2i               kompleksni brojevi
## 6. raw       as.raw(11)               sirovi
```

Nadalje, maleni podatkovni okvir može se ispisati kao tablica, ako se takva tablica namjerava predstaviti na primjer, u izvješću. Ranije je tako kreiran podatkovni okvir Tur_eu, kojeg možemo ispisati u obliku tablice:

```
> knitr::kable(Tur_eu)
```

	Drzava	Br_t_dol	Br_t_odl	EU
1	Croatia	57668	2980	TRUE
2	France	211998	48069	TRUE
4	Italy	93228	61194	TRUE
5	Austria	30816	11043	TRUE
6	Ireland	10926	8643	TRUE

Možete uočiti da je u ovom slučaju preskočen korak konverzije i koristi se samo tablični ispis. Također, ako se želi prilagoditi ispis u obliku tablice, tada se nazivi država mogu staviti u predstupac u kojem se trenutno nalaze samo redni brojevi.

```
> #prvi stupac dodijelimo novom vektoru
> predst <- Tur_eu[ , 1]
>
> #Kreiramo dataftame koristeći stupce broja odlazaka i dolazaka turista
> tabl <-Tur_eu[ , c(2:3)]
>
> #pridružujemo novi vektor nazivima redaka tabl
> rownames(tabl) <- predst
>
> #ispisujemo tabl
> knitr::kable(tabl)
```

	Br_t_dol	Br_t_odl
Croatia	57668	2980
France	211998	48069
Italy	93228	61194
Austria	30816	11043
Ireland	10926	8643

Za razliku od tabličnog ispisa, funkcija `table()` u R-u služi za kreiranje tablica frekvencija, koje prikazuju koliko se puta pojavljuje svaka jedinstvena vrijednost unutar jednog vektora ili kombinacije više vektora. Ovo je osnovna naredba za analizu kvalitativnih podataka i vrlo je korisna za brze uvide u distribuciju podataka. Kad se koristi s jednim vektorom, `table()` broji pojavljivanja svake jedinstvene vrijednosti. Za dva vektora, stvara tablicu kontingence koja prikazuje učestalosti kombinacija tih varijabli.

Kako bi se ilustrirala upotreba tablica, koristit će se podaci o nekretninama, Real_estate prikupljeni sa stranice Zillow, koja procjenjuje cijene nekretnina u SAD-u. Ovdje će se koristiti dio podataka vezan za Saratogu, pripremljen za analizu Zillow Internal, ožujak 2013., Dick De Veaux, 7. listopada 2015..

```
> nekretnine <-
+   read.delim("http://sites.williams.edu/rdeveaux/files/2014/09/Saratoga.txt")
> glimpse(nekretnine)
```

```
## Rows: 1,728
## Columns: 16
## $ Price         <int> 132500, 181115, 109000, 155000, 86060, 120000, 153000, 1~
## $ Lot.Size      <dbl> 0.09, 0.92, 0.19, 0.41, 0.11, 0.68, 0.40, 1.21, 0.83, 1.~
## $ Waterfront    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Age           <int> 42, 0, 133, 13, 0, 31, 33, 23, 36, 4, 123, 1, 13, 153, 9~
## $ Land.Value    <int> 50000, 22300, 7300, 18700, 15000, 14000, 23300, 14600, 2~
## $ New.Construct <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Central.Air   <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
## $ Fuel.Type     <int> 3, 2, 2, 2, 2, 2, 4, 4, 3, 2, 4, 2, 3, 2, 4, 2, 4, 2, 4,~
## $ Heat.Type     <int> 4, 3, 3, 2, 2, 2, 3, 2, 4, 2, 2, 2, 4, 3, 2, 3, 2, 2, 3,~
## $ Sewer.Type    <int> 2, 2, 3, 2, 3, 2, 2, 2, 2, 1, 2, 2, 2, 3, 2, 3, 2, 1, 2,~
## $ Living.Area   <int> 906, 1953, 1944, 1944, 840, 1152, 2752, 1662, 1632, 1416~
## $ Pct.College   <int> 35, 51, 51, 51, 51, 22, 51, 35, 51, 44, 51, 51, 41, 57, ~
## $ Bedrooms      <int> 2, 3, 4, 3, 2, 4, 4, 4, 3, 3, 7, 3, 2, 3, 3, 3, 3, 4, 2,~
## $ Fireplaces    <int> 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
## $ Bathrooms     <dbl> 1.0, 2.5, 1.0, 1.5, 1.0, 1.0, 1.5, 1.5, 1.5, 1.5, 1.0, 2~
## $ Rooms         <int> 5, 6, 8, 5, 3, 8, 8, 9, 8, 6, 12, 6, 4, 5, 8, 4, 7, 12, ~
```

Ovaj podatkovni okvir sadrži 16 varijabli, od kojih svaka predstavlja različite aspekte podataka o nekretninama. Pri kreiranju tablice naredbom `table()`, možemo koristiti jednu ili dvije varijable. Za potrebe kreiranja tablice, prvo ćemo iskoristiti dvije varijable: Waterfront (binarna varijabla, odgovara na pitanje nalazi li se nekretnina uz obalu) i Fireplaces (broj kamina). Iako je broj kamina

kvantitativna cjelobrojna varijabla, poprima samo četiri vrijednosti, što omogućuje jednostavan prikaz putem tablice.

```
> nekretnine$Waterfront <- factor(nekretnine$Waterfront,
+                                 levels = c(0, 1),
+                                 labels = c("Ne", "Da"))
> nekretnine$Fireplaces <- factor(nekretnine$Fireplaces,
+                                 levels = c(0, 1, 2, 3, 4))
> table("Uz obalu" = nekretnine$Waterfront,
+       "Broj kamina" = nekretnine$Fireplaces)
```

```
##         Broj kamina
## Uz obalu   0   1   2   3   4
##       Ne 733 934  42   2   2
##       Da   7   8   0   0   0
```

Odabrane varijable pretvorili smo u faktore. Možete uočiti kako je i broj kamina (koji sadrži mali broj cjelobrojnih vrijednosti) pretvoren u faktor za potrebe kreiranja ove tablice.

Naredbom `table()` kreirana je tablica kontingencije. Lako možemo uočiti da postoji relativno malo nekretnina uz obalu te one sadrže jedan ili niti jedan kamin. Na primjer, od svih promatranih nekretnina, 42 nekretnine nisu uz obalu i imaju po dva kamina.

```
> #table(nekretnine)
```

Naredba `table(nekretnine)`, očekivano bi rezultirala greškom - jer, kao što je ranije navedeno, `table()` podržava analizu jedne ili dviju varijabli.

Ako naredbu

table()

primijenimo na jednu varijablu, rezultat će biti apsolutne frekvencije za svaki modalitet obilježja.

```
> table(nekretnine$Fireplaces)
```

```
##
##   0   1   2   3   4
## 740 942  42   2   2
```

Iako se `table()` koristi za analizu i sažimanje podataka jedne ili dviju varijabli, pretvorba u tablicu putem `as.table()` može se koristiti i za prezentaciju izračunatih pokazatelja u izvještajima ili dokumentima. Rezultati analiza prikazani u obliku tablica (frekvencije, odnosi među varijablama i sl.) postaju jasniji i čitljiviji, što je korisno kao podloga za interpretaciju i dijeljenje rezultata s drugima.

```
> pokazatelji_nekretnine <- as.table(summary(nekretnine[,1:5]))
> kable(pokazatelji_nekretnine)
```

Price	Lot.Size	Waterfront	Age	Land.Value
Min. : 5000	Min. : 0.0000	Ne:1713	Min. : 0.00	Min. : 200
1st Qu.:145000	1st Qu.: 0.1700	Da: 15	1st Qu.: 13.00	1st Qu.: 15100
Median :189900	Median : 0.3700	NA	Median : 19.00	Median : 25000
Mean :211967	Mean : 0.5002	NA	Mean : 27.92	Mean : 34557
3rd Qu.:259000	3rd Qu.: 0.5400	NA	3rd Qu.: 34.00	3rd Qu.: 40200
Max. :775000	Max. :12.2000	NA	Max. :225.00	Max. :412600

Za rad s tablicama, funkcija `as.table()` može pretvoriti rezultate analiza ili matricu u tablični format. Ovo je korisno za prikaz sažetaka podataka kada želite naglasiti učestalosti ili odnose između varijabli. Međutim, `table()` i `as.table()` nisu prikladni za konverziju cijelih data frameova u tablični format.

Iako se ne radi o tablicama, nego o oblicima podatkovnih okvira, **tibble** se često spominje kao alternativa tablicama. Tibble je moderna verzija podatkovnog okvira u R-u, koja je dio tidyverse paketa. Tibble je dizajniran da bude jednostavniji i učinkovitiji za upotrebu u analizi podataka, posebice u kontekstu tidyverse filozofije i funkcija.

Jedan od glavnih razloga za korištenje tibble u kontekstu tablica je to što ispisuje podatke na čitljiviji način u konzoli. Naime, prikazuju samo prvi dio velikih skupova podataka i bolje upravljaju širinom stupaca.Nadalje, za razliku od tradicionalnih podatkovnih okvira, tibble ne pretvara znakove u faktore automatski. To je korisno jer sprječava neočekivane probleme koji proizlaze iz neželjenih tipova podataka.

Ostale prednosti tibblea se odnose na kreiranje podskupova (ne rezultiraju jednodimenzionalnim strukturama, kao što su vektori, što je čest slučaj s klasičnim podatkovnim okvirima). Osim toga, tibble omogućuju imena stupaca koja nisu valjana u standardnim R podatkovnim okvirima, kao što su prazni znakovi ili simboli. Moguće ih je dalje obrađivati i vizualizirati s drugim tidyverse paketima kao što su dplyr, ggplot2, tidyr i druge, što olakšava obradu podataka i vizualizaciju.

```
> library(tibble)
```

```
##
## Attaching package: 'tibble'

## The following object is masked from 'package:summarytools':
##
##     view
```

```
> tibi <- tibble(
+   Drzava = Tur_eu[,1],
+   Dolasci = Tur_eu[,2],
+   Odlasci = Tur_eu[,3]
```

```
+ )
>
> tibi
```

```
## # A tibble: 5 x 3
##   Drzava  Dolasci Odlasci
##   <chr>     <dbl>   <dbl>
## 1 Croatia   57668    2980
## 2 France   211998   48069
## 3 Italy     93228   61194
## 4 Austria   30816   11043
## 5 Ireland   10926    8643
```

7.10 Prednosti i nedostaci tablica te najčešće greške

Prednosti kreiranja tablica u R-u

- **Poboljšana čitljivost** (upotreba funkcija kao što je `kable()` iz knitr paketa omogućuje nam stvaranje tablica koje su čitljive i estetski ugodne, što je posebno korisno za izvještavanje i dijeljenje rezultata),
- **Fleksibilnost i prilagodba** (R nudi širok spektar mogućnosti za prilagodbu tablica, uključujući stiliziranje, formatiranje i dodavanje zaglavlja, što omogućava precizno prilagođavanje izgleda tablica potrebama korisnika),
- **Integracija s drugim R paketima** (tablice kreirane u R-u lako se integriraju s drugim paketima za analizu i vizualizaciju podataka, omogućujući relativno glatku suradnju između različitih koraka analitičkog procesa).

Uobičajene greške pri kreiranju tablica

- **Neadekvatno formatiranje** (zanemarivanje detalja formatiranja, poput širine stupaca ili poravnanja teksta, može rezultirati tablicama koje su teške za čitanje),
- **Pogrešno rukovanje velikim skupovima podataka** (pokušaj prikaza prevelikog broja redaka ili stupaca bez odgovarajućeg prilagođavanja može dovesti do nepreglednih tablica koje ne stanu na stranicu ili zaslon),
- **Zanemarivanje prednosti tibblea** (neiskorištavanje prednosti tibblea može rezultirati manje efikasnim radom s podacima, posebno kad je riječ o velikim skupovima podataka ili kad je potrebna integracija s tidyverse ekosustavom).
- **Zabuna oko razlike između tablice i podatkovnog okvira** (u alatima kao što su MS Excel, Google Sheets ili Libre Calc, često se tablice poistovjećuju s podatkovnim okvirima - iako i u tim alatima postoje razlike, zamjena najčešće neće rezultirati greškama. No, taj pristup ne smijete prenijeti na rad u R-u. U R-u ni na koji način nije opravdano čitav *df* spremiti kao tablicu, jer će to bitno otežati daljnju analizu ili rezultirati greškama.)

Najbolje prakse za kreiranje tablica

- **Koristite *kable()* za osnovne potrebe** (za jednostavne potrebe prikazivanja podataka, funkcija `kable()` iz knitr paketa je često dovoljna i pruža dobru ravnotežu između jednostavnosti i fleksibilnosti).
- **Isprobajte paket *gt* za složenije tablice** (za složenije potrebe, poput stvaranja visoko prilagođenih tablica za publikacije, paket *gt* nudi napredne mogućnosti stiliziranja i prilagodbe),
- **Prilagodite izgled tablice korisničkim potrebama** (uvijek prilagodite izgled tablice tako da odgovara kontekstu u kojem će biti korištena, bilo da je to seminarski rad, esej, izvještaj ili prezentacija),
- **Razmotrite korištenje *tibble*a za rad s podacima** (tibble, modernija verzija podatkovnog okvira, nudi bolje performanse i veću fleksibilnost pri radu s podacima, što može olakšati kreiranje i manipulaciju tablicama).

Pitanja za ponavljanje

1. Koja je glavna svrha funkcije `kable()` iz paketa knitr pri kreiranju tablica, i kako doprinosi čitljivosti podataka?
2. Objasnite postupak kreiranja osnovne tablice u R-u koristeći vektor, matricu i funkciju `kable()` za prikaz.
3. Koje su prednosti korištenja tibble objekta u usporedbi s klasičnim podatkovnim okvirom (df) pri radu s podacima?
4. Kako bi izgledao postupak pretvaranja postojećeg podatkovnog okvira u tibble koristeći paket tibble?
5. Koje su uobičajene greške koje se javljaju pri kreiranju tablica, posebno kada je riječ o formatiranju velikih skupova podataka?
6. Kako se može prilagoditi ispis tablice tako da se nazivi država iz stupca prebace u predstupac (nazive redaka) za bolju preglednost?
7. Što treba učiniti da se pri kreiranju tablice automatski izbjegne pretvorba tekstualnih podataka u faktore?
8. Kako se mogu kreirati tibble tablice s imenima stupaca koji sadrže posebne znakove ili prazne znakove? U čemu tibble omogućava više fleksibilnosti u imenovanju stupaca?
9. Za koje je slučajeve prikladnije koristiti napredne pakete kao što je gt pri kreiranju tablica, a kada je dovoljno koristiti osnovne funkcije poput `kable()`?
10. Koje su najbolje prakse za prikaz tablica s velikim brojem redaka i stupaca u izvještajima i prezentacijama?

8 Uvoz podataka

U ovom poglavlju objašnjavaju se različiti načini uvoza podataka u R, uključujući učitavanje lokalnih datoteka poput .csv i .xlsx, dohvaćanje podataka s internetskih izvora te povezivanje s bazama podataka putem SQL upita. Poglavlje pokriva ključne funkcije i specijalizirane biblioteke poput readxl i DBI. Na kraju, opisane su uobičajene greške pri uvozu podataka te najbolje prakse za osiguravanje pravilnog i sigurnog rada s podacima, uz savjete za optimizaciju procesa uvoza.

8.1 Uvoz podataka pohranjenih na računalu

Postoji nekoliko različitih načina učitavanja podataka u R. Ako podatke imate na računalu i ako su u .csv formatu, onda ih je moguće učitati pomoću naredbe `read.csv()`. Podaci prethodno moraju biti pripremljeni kao podatkovni okvir (varijable u stupcima, a opažanja u recima).

```
> data <- read.csv("data.csv")
```

Ako se podatkovni okvir ne nalazi u direktoriju projekta, tad je potrebno navesti put do datoteke, na primjer:

```
> data <- read.csv("C:/User/Dokumenti/datoteka.csv")
```

Za učitavanje običnih tekstualnih datoteka, može se koristiti read.table ili readLines.

```
> data <- read.table("putanja/do/datoteke.txt", header = TRUE, sep = "\t")
```

Za učitavanje MS Excel datoteka, potreban je odgovarajući paket za provođenje te naredbe. readxl ili openxlsx su paketi koji sadrže naredbe za učitavanje Excel datoteke.

```
> #instaliranje paketa
> install.packages("openxlsx")
>
> #pozivanje paketa (aktivacija)
> library(openxlsx)
>
> #naredba
> data <- read.xlsx("data.xlsx")
```

8.2 Uvoz podataka iz R paketa

Da biste učitali podatke iz paketa koji sadrži podatke u R, prvo morate instalirati i učitati paket, a zatim učitati podatke koji su dio tog paketa. Većina paketa koji sadrže podatke omogućuju izravan pristup tim podacima nakon što se paket učita.

```
> install.packages("ggplot2")
> library(ggplot2)
> data(mpg)
```

Nakon ovog koraka, skup podataka mpg je dostupan za analizu i vizualizaciju u vašem R radnom prostoru. Također, popularni su podaci `data(mtcars)`, `data("iris")` i `data("USArrests")`, dostupni u baznoj verziji R-a. Oni dolaze u paketu datasets koji je automatski učitan kada pokrenete R. Dakle, nije potrebno instalirati ili posebno učitati ovaj paket da biste pristupili ovim skupovima podataka.

8.3 Dohvaćanje i uvoz podataka iz online izvora

Ako želite općenito dohvatiti podatke s nekog URL-a u R-u, postupak će ovisiti o formatu tih podataka. Najčešće korišteni formati za online podatke su CSV, XML i HTML tablice. Za .csv to možete učiniti na sljedeći način:

```
> url <- "http://example.com/data.csv"
> podaci <- read.csv(url)
```

Isto se postiže i na ovaj način:

```
> podaci <- read.csv("http://example.com/data.csv")
```

Za XML podatke, potreban je paket xml2:

```
> install.packages("xml2")
> library(xml2)
>
> url <- "http://example.com/data.xml"
> podaci <- read_xml(url)
```

Za dohvaćanje tablica iz HTML stranica koristi se paket rvest:

```
> install.packages("rvest")
> library(rvest)
>
> url <- "http://example.com"
> podaci <- url %>%
+   read_html() %>%
+   html_table()
```

U ovom primjeru, `html_table()` će pokušati dohvatiti sve tablice s navedene mrežne stranice. Možete dodati dodatne parametre ili funkcije za izdvajanje određene tablice.

Prije upotrebe bilo kojeg od ovih paketa, važno je provjeriti dopušta li URL s kojeg dohvaćate podatke takav pristup i je li zaštićen autorskim pravima ili drugim ograničenjima. Također, neki URL-ovi mogu zahtijevati posebne parametre za pristup (npr. API ključeve ili ovlaštenja).

Za učitavanje podataka iz specifičnih online izvora koji sadrže baze podataka (npr. World Bank), možete koristiti pakete u R-u koji su specijalizirani za povezivanje i dohvaćanje podataka iz tih izvora. Jedan od takvih paketa je wbstats, koji omogućuje pristup upravo podacima Svjetske banke.

```
> install.packages("wbstats")
> library(wbstats)
```

Prije nego što dohvatite podatke, možete istražiti koji su indikatori dostupni. wbstats paket pruža funkcije za pretragu indikatora.

```
> wb_indicators <- wbsearch("labor market")
```

Nakon što identificirate željene indikatore, možete dohvatiti podatke koristeći wb funkciju. Morate navesti indikatore i, po potrebi, druge parametre poput zemalja, vremenskih razdoblja i sl.

```
> podaci <- wb(indicator = "ID_indikatora", country = c("HR", "SI"),
+              startdate = 2010, enddate = 2020)
```

Ovdje ID_indikatora predstavlja identifikator indikatora koji želite dohvatiti, country je lista zemalja (kodova zemalja), a startdate i enddate definiraju vremenski raspon za koji želite podatke.

8.4 Uvoz podataka iz baza podataka

Učitavanje podataka iz online SQL baza podataka u R zahtijeva nekoliko koraka i odgovarajuće pakete za uspostavljanje veze s bazom podataka. Prvo, trebate instalirati i učitati odgovarajuće pakete. Na primjer, za MySQL možete koristiti paket RMySQL, a za PostgreSQL RPostgreSQL. Drugi korak je uspostavljanje veze s bazom podataka. Trebat će vam podaci za pristup bazi podataka kao što su ime hosta (servera), korisničko ime, lozinka i ime baze podataka. Nakon uspostavljanja veze, možete izvršiti SQL upite da biste dohvatili podatke.

```
> install.packages("DBI")
> install.packages("RMySQL")
> library(DBI)
> library(RMySQL)
>
> veza <- dbConnect(MySQL(),
+                   dbname = "ime_baze_podataka",
+                   host = "host_baze_podataka",
+                   username = "korisnicko_ime",
+                   password = "lozinka")
>
> upit <- "SELECT * FROM neka_tablica"
> podaci <- dbGetQuery(veza, upit)
```

Nakon dohvaćanja podataka, važno je prekinuti vezu s bazom podataka (ako ne kreirate interaktivnu aplikaciju). Kada radite s bazama podataka, posebnu pažnju treba posvetiti sigurnosti, posebno kod rukovanja korisničkim imenima i lozinkama. Koristite sigurne metode za pohranu i pristup ovim podacima.

```
> dbDisconnect(veza)
```

Za druge vrste baza podataka (npr. SQL Server, Oracle) postupak je sličan, ali će se koristiti različiti paketi (npr. ROracle, RODBC za Oracle, RSQLServer za SQL Server, itd.).

8.5 Uobičajene greške pri uvozu podataka

- **Nepravilno navođenje puta do datoteke** (jedna od čestih grešaka je pogrešno navođenje putanja do datoteka, što može dovesti do grešaka pri učitavanju)
- **Zanemarivanje opcija za učitavanje** (neiskorištavanje opcija poput stringsAsFactors (za `read.csv()` i `read.table()`) može dovesti do neželjenih tipova podataka, posebno kada su u pitanju tekstualni podaci, što će produljiti vrijeme prilagodbe podataka)
- **Sigurnosna pitanja pri radu s bazama podataka** (neadekvatno rukovanje pristupnim podacima za baze podataka može predstavljati sigurnosni rizik)

Najbolje prakse za uvoz podataka

- **Korištenje readr paketa za CSV datoteke**: paket *readr* nudi funkcije kao što su `read_csv()`, koje su često brže i bolje rukuju tipovima podataka od osnovnih R funkcija poput `read.csv()`.
- **Upotreba paketa *haven* za rad s datotekama SPSS, Stata, i SAS**: ako radite s podacima iz ovih statističkih programa, *haven* omogućuje efikasno učitavanje takvih datoteka.
- **Pazite na enkodiranje znakova**: kod učitavanja podataka iz različitih jezičnih područja, važno je paziti na pravilno enkodiranje znakova.
- **Provjerite i očistite podatke nakon učitavanja**: uvijek provjerite strukturu učitanih podataka (`str()`, `head()`, itd.) i po potrebi izvršite početno čišćenje podataka (npr. obrada vrijednosti koje nedostaju), kao što je objašnjeno u poglavlju o radu s podatkovnim okvirima.

Pitanja za ponavljanje

1. Koja je osnovna naredba za učitavanje .csv datoteka u R i kako navodimo put do datoteke koja se nalazi u drugom direktoriju?
2. Kako se koristi funkcija `read.table()` za učitavanje tekstualnih datoteka i što određuju argumenti poput `header = TRUE` i `sep = "\t"`?
3. Koje pakete treba koristiti za učitavanje podataka iz Excel datoteka i kako aktiviramo te pakete nakon instalacije?
4. Na koji način učitavamo podatke koji su već sadržani u paketu ggplot2, poput skupa podataka mpg?
5. Objasnite postupak učitavanja .csv datoteke s internetskog URL-a i navedite primjer takve naredbe.
6. Kako dohvaćamo podatke u XML formatu i koji paket je potreban za tu funkcionalnost?
7. Ako koristimo SQL bazu podataka za učitavanje podataka u R, koji su osnovni koraci za povezivanje i dohvaćanje podataka pomoću SQL upita?
8. Zašto je važno pravilno rukovati pristupnim podacima za baze podataka, kao što su korisničko ime i lozinka, i kako se uspostavlja i prekida veza s bazom podataka?
9. Koje su uobičajene greške koje mogu nastati pri uvozu podataka u R, posebice kod rada s putanjama do datoteka i tipovima podataka?
10. Koje su najbolje prakse kod učitavanja podataka, a koje uključuju korištenje paketa readr i haven?

9 Elementi kontrole toka

Ovo poglavlje uvodi osnovne elemente kontrole toka u R-u, kao što su for i while petlje te uvjetne naredbe if, ifelse i switch. Ovi alati omogućuju nam uvjetno izvršavanje koda, ponavljanje radnji i fleksibilnu manipulaciju podacima temeljem zadatih uvjeta. Prikazani primjeri i primjene pokazuju kako kontrola toka može poboljšati učinkovitost i prilagodljivost analitičkog procesa u R-u.

U R-u, „for", „if" i „ifelse" su temeljni elementi kontrole toka. Svaki od ovih elemenata ima svoju specifičnu ulogu i sintaksu:

For petlja:

- „For" se koristi za izvođenje ponovljenih radnji unutar petlje.
- Sintaksa:

  ```
  for (varijabla in skup) {
  # izvrši neku radnju
  }
  ```

- Ovdje varijabla prolazi kroz svaki element u skup.

If uvjet:

- „If" se koristi za ispitivanje uvjeta te izvršavanje koda ako je uvjet istinit (TRUE).
- Sintaksa:

  ```
  if (uvjet) {
  # izvrši neku radnju
  }
  ```

- Ovdje se radnja izvršava samo ako je uvjet istinit.

Ifelse funkcija:

- „Ifelse" nije petlja, već vektorizirana funkcija koja omogućava ispitivanje uvjeta i vraća jednu vrijednost ako je uvjet istinit, a drugu vrijednost ako nije.
- Sintaksa:

  ```
  ifelse(uvjet, vrijednost_za_true, vrijednost_za_false)
  ```

- Ovdje funkcija vraća vrijednost_za_true ako je uvjet istinit, inače vraća vrijednost_za_false.

Svaki od ovih elemenata igra važnu ulogu u strukturiranju i kontroli toka programa u R-u. „For" petlja se koristi za iteracije, „if" uvjet za kontrolu toka na osnovi logičkih uvjeta, dok se „ifelse" često koristi za vektorizirane operacije koje zahtijevaju uvjetno izvršavanje.

9.1 for

Na primjer, recimo da želimo dobiti uvide u podatke o olujama. Podaci storms iz paketa dplyr sadrže informacije o putanjama oluja. To su podaci iz baze podataka Atlantskog uragana NOAA (Nacionalne uprave za oceane i atmosferu SAD), koji uključuju položaje i karakteristike oluja od 1975. do 2021. godine. Oluje od 1979. godine nadalje mjerene su svakih šest sati tijekom života oluje, dok su podaci za ranije godine ponekad nepotpuni.

Format podataka

Podaci su formatirani kao tibble (moderniji oblik podatkovnih okvira u R-u) s 19,066 opažanja i 13 varijabli:

- name: ime oluje.
- year, month, day: datum izvještaja. Godina, mjesec i dan su odvojene varijable koje zajedno čine datum.
- hour: sat izvještaja (u UTC vremenu).
- lat, long: lokacija središta oluje. Geografska širina i dužina.
- status: klasifikacija oluje (tropska depresija, tropska oluja ili uragan).
- category: kategorija uragana prema Saffir-Simpsonovoj skali, izračunata na temelju brzine vjetra.
  - NA: Nije uragan
  - 1: 64+ čvorova
  - 2: 83+ čvorova
  - 3: 96+ čvorova
  - 4: 113+ čvorova
  - 5: 137+ čvorova
- wind: maksimalna održavana brzina vjetra oluje (u čvorovima).
- pressure: zračni tlak u središtu oluje (u milibarima).
- tropicalstorm_force_diameter: promjer (u nautičkim miljama) područja koje doživljava vjetrove snage tropske oluje (34 čvora ili više). Dostupno samo od 2004. godine.
- hurricane_force_diameter: promjer (u nautičkim miljama) područja koje doživljava vjetrove snage uragana (64 čvora ili više). Dostupno samo od 2004. godine.

Na ovim podacima možemo isprobati nekolicinu primjera upotrebe kontrole toka. Prvo moramo učitati podatke.

```
> library(dplyr)
> data(storms)
> glimpse(storms)
```

```
## Rows: 19,537
## Columns: 13
## $ name                          <chr> "Amy", "Amy", "Amy", "Amy", "Amy", "Amy",~
## $ year                          <dbl> 1975, 1975, 1975, 1975, 1975, 1975, 1975,~
## $ month                         <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,~
## $ day                           <int> 27, 27, 27, 27, 28, 28, 28, 28, 29, 29, 2~
## $ hour                          <dbl> 0, 6, 12, 18, 0, 6, 12, 18, 0, 6, 12, 18,~
## $ lat                           <dbl> 27.5, 28.5, 29.5, 30.5, 31.5, 32.4, 33.3,~
## $ long                          <dbl> -79.0, -79.0, -79.0, -79.0, -78.8, -78.7,~
## $ status                        <fct> tropical depression, tropical depression,~
## $ category                      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ wind                          <int> 25, 25, 25, 25, 25, 25, 25, 30, 35, 40, 4~
## $ pressure                      <int> 1013, 1013, 1013, 1013, 1012, 1012, 1011,~
## $ tropicalstorm_force_diameter <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ hurricane_force_diameter     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

for petlje se obično koriste za iterativno izvršavanje koda nad elementima nekog skupa, kao što su redovi ili stupci u datasetu. Evo nekoliko primjera kako možete koristiti for petlju s datasetom storms:

- **Ispisivanje vrijednosti**

Možete koristiti for petlju da prođete kroz redove ili stupce dataset-a i ispišete određene vrijednosti. Na primjer, ispisati nazive svih oluja:

```
> for (i in 1:nrow(storms)) {
+   print(unique(storms$name[i]))
+ }
```

Ova petlja prolazi kroz svaki red u datasetu storms i ispisuje ime oluje. Ovdje je to zapisano kao komentar, jer bi u suprotnom rezultiralo ispisom 19066 redaka s nazivima oluja.

- **Izračunavanje i pohrana rezultata**

Možete izračunati nešto za svaki red i pohraniti rezultate u novi vektor. Na primjer, izračunati maksimalnu brzinu vjetra za svaku godinu:

```
> # izdvajanje jedinstvenih godina
> unique_years <- unique(storms$year)
>
> # kreiranje praznog vektora u koji će se pohraniti najveće brzine vjetra,
> # duljine vektora jedinstvenih godina
> max_wind_speed_per_year <- numeric(length(unique_years))
>
> for (i in 1:length(unique_years)) {
+   # i in 1:length(unique_years) funkcionira kao brojač
```

```
+   # i redom prolazi kroz godine
+   year_data <- storms[storms$year == unique_years[i], ]
+   # privremeni vektor (temp) u koji se pohranjuju podaci za promatranu godinu
+   max_wind_speed_per_year[i] <- max(year_data$wind, na.rm = TRUE)
+   # utvrđivanje maksimalne vrijednosti za godinu i pohrana u
+   # vektor max_wind_speed_per_year
+ }
>
> summary(max_wind_speed_per_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    80.0   115.0   125.0   127.7   145.0   165.0
```

```
> # vektor max_wind_speed_per_year i dalje ima puno (47) opažanja,
> # pa će summary pružiti sažeti uvid u rezultate
```

Ukratko, ova petlja prolazi kroz svaku jedinstvenu godinu u datasetu i računa maksimalnu brzinu vjetra za tu godinu.

Na sličan način, može nas zanimati koliko je dugo svaka oluja trajala prema satima u kojima je zabilježena. Stoga, prethodni kod prilagođavamo s obzirom na predmet interesa.

```
> unique_names <- unique(storms$name) # izdvajanje jedinstvenih naziva
> max_hour <- numeric(length(unique_names)) # kreiranje praznog vektora
> min_hour <- numeric(length(unique_names)) # kreiranje praznog vektora
> duration_hour <- numeric(length(unique_names)) # kreiranje praznog vektora
>
> for (i in 1:length(unique_names)) {
+   names_data <- storms[storms$name == unique_names[i], ]
+   max_hour[i] <- max(names_data$hour, na.rm = TRUE)
+   # utvrđivanje maksimalne vrijednosti
+   min_hour[i] <- min(names_data$hour, na.rm = TRUE)
+   # utvrđivanje minimalne vrijednosti
+   duration_hour[i] <- max_hour[i]-min_hour[i]
+ }
>
> storms_duration <- cbind.data.frame(unique_names, min_hour,
+                                     max_hour, duration_hour)
> summary(storms_duration$duration_hour)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   18.00   18.00   18.98   20.00   23.00
```

Napomene za korištenje for petlji u R-u

- Iako su intuitivne i lako razumljive, for petlje u R-u su često manje efikasne od vektoriziranih operacija i funkcija poput `apply()`. Razmislite o korištenju vektoriziranih operacija gdje je to moguće za bolje performanse.

- Prilikom rada s većim skupovima podataka, for petlje mogu biti sporije. U takvim slučajevima, paketi poput data.table ili dplyr mogu pružiti brže alternative.

9.2 apply, sapply, lapply

Da biste zamijenili for petlju apply funkcijom u R-u, trebate uzeti u obzir vrstu operacije koju izvodite. **apply** funkcije (koje uključuju apply, lapply, sapply, vapply itd.) omogućuju vektorizirane operacije na podatkovnim okvirima i listama.

- **Ispisivanje vrijednosti**

Za ispisivanje imena oluja iz skupa podataka storms, možete koristiti lapply ili sapply. Međutim, ova operacija neće biti efikasnija s apply funkcijom jer je ispisivanje inherentno iterativni proces. Ali, za demonstraciju:

```
> sapply(storms$name, print)
```

- **Izračunavanje maksimalne brzine vjetra po godini**

Ovo je složeniji primjer gdje apply funkcije uistinu mogu biti korisnije. Koristeći sapply ili lapply, možemo izračunati maksimalnu brzinu vjetra za svaku godinu:

```
> unique_years <- unique(storms$year)
> max_wind_speed_per_year <- sapply(unique_years, function(yr) {
+   max(storms$wind[storms$year == yr], na.rm = TRUE)
+ })
>
> summary(max_wind_speed_per_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    80.0   115.0   125.0   127.7   145.0   165.0
```

Ovdje sapply prolazi kroz svaku jedinstvenu godinu i primjenjuje funkciju koja računa maksimalnu brzinu vjetra za tu godinu. Rezultat je numerički vektor maksimalnih brzina vjetra za svaku godinu.

Napomene za korištenje apply funkcija:

- apply funkcije su obično efikasnije od for petlji za vektorizirane operacije.
- lapply vraća listu, dok sapply pokušava pojednostaviti rezultat u vektor ili matricu ako je to moguće.
- Kod složenijih operacija ili kada su performanse kritične, alternativni pristupi kao što su funkcije iz paketa data.table mogu biti još efikasniji.

9.3 if

Primjena „if" uvjeta na skupu podataka poput storms iz paketa dplyr može se koristiti za različite analize i manipulacije podacima. Evo nekoliko primjera kako možete koristiti „if" uvjete sa skupom podataka storms:

- **Filtriranje i analiza**

Možete koristiti „if" uvjete za filtriranje oluja na temelju određenih kriterija, kao što su kategorija, brzina vjetra ili godina. Na primjer, za ispitivanje je li neka oluja bila kategorije 4 ili veća. Prisjetimo se opisa podataka:

```
+ NA: Nije uragan
+ 1: 64+ čvorova
+ 2: 83+ čvorova
+ 3: 96+ čvorova
+ 4: 113+ čvorova
+ 5: 137+ čvorova
```

Dakle, kategorija 4 utvrđuje se temeljem jačine vjetra 113 čvorova ili više od toga. U ovom slučaju ne možemo izravno koristiti varijablu storms$category kao uvjet, zbog toga što sadrži NA. S obzirom da je *if* logički uvjet, ne može izvršiti provjeru vrijednosti koje nedostaju.

```
> for (i in 1:nrow(storms)) {
+   if (storms$wind[i] >= 113) {
+     print(paste(storms$name[i], "je bio uragan kategorije", storms$category[i]))
+   }
+ }
```

Ispis bi rezultirao podužim popisom, pa je zbog toga ovdje izuzet. No, možete sami isprobati ispis.

- **Izmjena u skupu podataka temeljem uvjeta**

Možete dodati novi stupac u podatkovni skup na osnovi nekog uvjeta. Na primjer, označavanje oluje kao „Snažne" ili „Slabe" na temelju brzine vjetra (proizvoljno odabranih 100 čvorova):

```
> storms$intensity <- NA # kreiramo prazan stupac u skupu podataka
>
> for (i in 1:nrow(storms)) {
+   if (storms$wind[i] >= 100) {
+     storms$intensity[i] <- "Snažna"
+   } else {
+     storms$intensity[i] <- "Slaba"
+   }
+ }
>
> table(storms$intensity)
```

```
##
##  Slaba Snažna
##  18273   1264
```

- **Statistička analiza s uvjetima**

Ovdje ćemo koristiti „if" uvjete za izračunavanje statističkih podataka, kao što su prosječne vrijednosti, ali samo za određene oluje. Na primjer, izračunajmo prosječni tlak za oluje kategorije 5:

```
> kat_5_tlak <- numeric()
>
> for (i in 1:nrow(storms)) {
+   if (storms$wind[i] >= 137) {
+     kat_5_tlak <- c(kat_5_tlak, storms$pressure[i])
+   }
+ }
> print(paste("Oluje kategorije 5, prosječno dosežu ",
+             mean(kat_5_tlak, na.rm = TRUE),
+             "milibara tlaka zraka u središtu"))
```

```
## [1] "Oluje kategorije 5, prosječno dosežu  918.172413793103 milibara tlaka zraka u središtu"
```

Napomena: dok „if" uvjeti mogu biti korisni za iterativno pregledavanje i manipulaciju redaka podatkovnog okvira, vektorizirane funkcije u R-u često nude efikasnija i čišća rješenja za ovakve operacije. Na primjer, dplyr funkcije kao što su `filter()`, `mutate()` i `summarise()` mogu obaviti slične zadatke bez potrebe za eksplicitnom petljom.

9.4 ifelse

Za korištenje *if* uvjeta bez *for* petlje na skupu podataka storms, možete se osloniti na vektorizirane funkcije poput **ifelse**. *ifelse* je vrlo koristan za primjenu uvjeta na cijeli vektor ili stupac u podatkovnom okviru.

- **Klasifikacija**

Pretpostavimo da želite klasificirati oluje kao „Snažne" ili „Slabe" (kao u ranijem primjeru) na temelju njihove maksimalne brzine vjetra. U ovom slučaju, možete koristiti ifelse da dodate novi stupac u podatkovni okvir:

```
> storms$intensity <- NA
>
> storms <- storms %>%
+   mutate(intensity = ifelse(wind >= 100, "Snažna oluja", "Slaba oluja"))
>
> table(storms$intensity)
```

```
##
##  Slaba oluja Snažna oluja
##        18273         1264
```

ifelse testira uvjet $wind \geq 100$ za svaki red u skupu podataka storms. Ako je uvjet istinit, dodjeljuje vrijednost „Snažna oluja" novom stupcu intensity. Ako uvjet nije istinit, dodjeljuje „Slaba oluja".

- **Filtriranje podataka**

Također, možete koristiti kombinaciju **ifelse** i **filter** za filtriranje podataka na temelju određenog uvjeta. Na primjer, izdvajanje podataka samo za uragane kategorije 5:

```
> category_5_storms <- filter(storms, wind >= 137)
>
> glimpse(category_5_storms)
```

```
## Rows: 116
## Columns: 14
## $ name                          <chr> "Anita", "Anita", "David", "David", "Davi~
## $ year                          <dbl> 1977, 1977, 1979, 1979, 1979, 1979, 1979,~
## $ month                         <dbl> 9, 9, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,~
## $ day                           <int> 2, 2, 30, 30, 30, 31, 31, 31, 31, 5, 5, 5~
## $ hour                          <dbl> 0, 6, 6, 12, 18, 0, 6, 12, 18, 0, 6, 12, ~
## $ lat                           <dbl> 24.6, 24.2, 16.0, 16.3, 16.6, 16.8, 17.0,~
## $ long                          <dbl> -96.2, -97.1, -64.2, -65.2, -66.2, -67.3,~
## $ status                        <fct> hurricane, hurricane, hurricane, hurrican~
## $ category                      <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,~
## $ wind                          <int> 140, 150, 140, 145, 150, 145, 145, 145, 1~
## $ pressure                      <int> 931, 926, 925, 924, 924, 927, 928, 927, 9~
## $ tropicalstorm_force_diameter <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ hurricane_force_diameter      <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ intensity                     <chr> "Snažna oluja", "Snažna oluja", "Snažna o~
```

Ovdje filter funkcija iz dplyr paketa omogućuje filtriranje redaka u podatkovnom okviru storms, s obzirom na uvjet da je vrijednost stupca wind veća ili jednaka 137.

Iako se može činiti da se skup podataka i podatkovni okvir koriste naizmjenično kao sinonimi, to nije tako. Ponovimo kratko njihova značenja. Skup podataka (*engl. dataset*) je pojam koji se koristi za označavanje bilo kojeg skupa prikupljenih podataka, neovisno o vrsti i načinu pohrane. To mogu biti skupovi pojedinačnih opažanja, dokumenata, slika, tablica, baza podataka itd. Nadalje, skupovi podataka mogu se pohraniti u različitim formatima kao što su csv datoteke, Excel tablice, SQL baze podataka i slično. Podatkovni okvir (*engl. dataframe*) je specifičan pojam koji se najčešće koristi u kontekstu programskih jezika za statističku analizu i manipulaciju podacima, poput R i Python (posebno uz biblioteku pandas). Podatkovni okvir predstavlja tabličnu strukturu podataka s redovima i stupcima, gdje reci predstavljaju pojedinačne zapise (primjere, mjerenja, itd.), a stupci predstavljaju varijable (atribute, značajke). Na primjer, ako bismo na raspolaganju imali podatkovni okvir rezultata anketiranja, pojedini redak sadržavao bi odgovore jednog ispitanika,

dok bi pojedini stupac sadržavao odgovore na pojedino pitanje. Ova struktura omogućava lakšu manipulaciju, analizu i vizualizaciju podataka. Dakle, podatkovni okvir je ujedno i skup podataka, ali neće svaki skup podataka ujedno biti i podatkovni okvir.

9.5 Razlika između ifelse i kombinecije if i else

Razlike između ifelse funkcije i kombinacije if i else izraza u R-u leže u načinu kako oni pristupaju uvjetnom izvršavanju koda, njihovoj vektorizaciji, i u situacijama u kojima se obično koriste.

ifelse je vektorizirana funkcija koja omogućava provjeru uvjeta nad cijelim vektorom elemenata odjednom i vraća vektor rezultata temeljem tog uvjeta. Sintaksa:

```
> ifelse(test, da, ne)
```

- test: logički vektor koji sadrži uvjet koji se provjerava za svaki element,
- da: vrijednost koja se vraća ako je uvjet istinit,
- ne: vrijednost koja se vraća ako je uvjet neistinit.

Primjer: Zamislimo da imamo vektor temperatura i želimo označiti sve temperature iznad 25 stupnjeva kao „Toplo", a manje ili jednake 25 kao „Ugodno".

```
> temperature <- c(22, 28, 25, 30, 18)
> vrijeme <- ifelse(temperature > 25, "Toplo", "Ugodno")
> print(vrijeme)
```

```
## [1] "Ugodno" "Toplo"  "Ugodno" "Toplo"  "Ugodno"
```

if i **else** izrazi koriste se za uvjetno izvršavanje koda temeljem ispunjenja nekog uvjeta. Za razliku od ifelse, if i else se ne mogu direktno primijeniti na vektore, već evaluiraju samo prvi element logičkog vektora ako im se takav prosljedi. Sintaksa:

```
> if (uvjet) {
+   # izvršava se ako je uvjet istinit
+ } else {
+   # izvršava se ako uvjet nije istinit
+ }
```

Primjer: Uzmimo isti scenarij s temperaturama, ali primijenimo if i else za provjeru pojedinog elementa.

```
> temperatura <- 28
> if (temperatura > 25) {
+   vrijeme <- "Toplo"
+ } else {
+   vrijeme <- "Ugodno"
+ }
> print(vrijeme)
```

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

Za vektoriziranu provjeru, morali bismo koristiti petlju ili neku vrstu aplikacijske funkcije (npr., lapply) ili kombinaciju s for petljom.

Ključne razlike

1. Vektorizacija: ifelse može obraditi cijele vektore odjednom, automatski primjenjujući uvjet na svaki element i vraćajući vektor rezultata. Kombinacija if i else može evaluirati samo pojedinačne elemente ili se mora koristiti unutar petlje za iteraciju kroz vektore.
2. Performanse: ifelse je obično brži za vektorizirane operacije jer je specifično dizajniran za takve zadatke. if i else mogu biti korisni za kontrolu toka unutar funkcija ili skripti gdje evaluacija pojedinačnih uvjeta ima smisla.
3. Složenost: Za složenije uvjetne strukture s više grana, if i else pružaju veću fleksibilnost i čitljivost. ifelse je praktičan za jednostavne, brze provjere i zamjene u vektorima.

Ovisno o zadatku, odabrat ćete ifelse za vektorizirane operacije i jednostavne zamjene, ili if i else za složeniju logiku kontrole toka gdje je potrebna evaluacija pojedinačnih uvjeta ili gdje se koriste složenije strukture odlučivanja.

9.6 switch

switch izraz omogućava jednostavno grananje izvršavanja koda temeljem vrijednosti nekog izraza. Ovo je posebno korisno kada imate višestruke uvjete koje želite testirati protiv jedne varijable. switch može smanjiti potrebu za složenim if-else if lancima, čineći kod čišćim i lakšim za čitanje. Sintaksa:

```
switch(izraz, slučaj1 = vrijednost1, slučaj2 = vrijednost2, ...,
slučajN = vrijednostN)
```

gdje je *izraz* varijabla čiju vrijednost uspoređujete, a *slučajX* su moguće vrijednosti te varijable. Ako se vrijednost izraza podudara s nekim od slučajeva, izvršit će se zadani kod za prepoznati slučaj.

Primjer: Zamislimo aplikaciju za praćenje vremena putem koje želimo korisnicima prikazati poruku temeljenu na trenutnoj sezoni.

```
> trenutna_sezona <- "ljeto"
>
> poruka <- switch(trenutna_sezona,
+                  proljece = "Vrijeme je za sadnju cvijeća!",
+                  ljeto = "Vrijeme je za plažu i sunčanje!",
+                  jesen = "Vrijeme je za berbu i promatranje boja lišća!",
+                  zima = "Vrijeme je za skijanje i uživanje u snijegu!",
+                  "Nepoznata sezona")
>                  # Zadnji slučaj je „Nepoznata sezona" ako se izraz
>                  # ne podudara ni s jednim slučajem
```

```
>
> print(poruka)
```

```
## [1] "Vrijeme je za plažu i sunčanje!"
```

Ovaj kod postavlja varijablu *trenutna_sezona* na „ljeto" i koristi *switch* za odabir odgovarajuće poruke. Budući da je *trenutna_sezona* postavljena na „ljeto", ispisat će se: „Vrijeme je za plažu i sunčanje!".

Prednosti korištenja *switch*a:

- Smanjuje potrebu za dugim if-else if lancima, čineći kod preglednijim i lakšim za održavanje.
- Olakšava upravljanje višestrukim grananjima izvršavanja temeljenim na jednoj varijabli.
- Pomaže u očuvanju performansi koda u scenarijima s mnogo uvjeta.

U praksi, switch se često koristi kada imate unaprijed zadan skup mogućih vrijednosti za varijablu i želite izvršiti različite radnje za svaku od tih vrijednosti. To ga čini idealnim alatom za situacije poput obrade korisničkih unosa, odabira konfiguracijskih opcija, upravljanja stanjima u aplikaciji i slično.

9.7 while

while petlja se koristi za ponavljanje izvođenja bloka koda dok god je određeni uvjet zadovoljen. Za razliku od *for* petlje, koja prolazi kroz fiksni skup elemenata, while petlja može se koristiti kada unaprijed ne znamo koliko puta će biti potrebno izvršiti radnju. Sintaksa:

```
> while (uvjet) {
+   # kod koji se izvršava dok je uvjet istinit
+ }
```

U ovom kontekstu, uvjet je logički izraz. Ako je uvjet istinit (TRUE), kod unutar petlje se izvršava. Nakon svake provedbe, uvjet se ponovno provjerava. Petlja se nastavlja sve dok je uvjet istinit. Ako uvjet postane neistinit (FALSE), izvršavanje petlje se prekida.

Primjer: Pretpostavimo da želimo izračunati faktorijel broja. Faktorijel broja n (označeno s n!) je umnožak svih pozitivnih cijelih brojeva manjih ili jednakih n. Na primjer, $5! = 5 \times 4 \times 3 \times 2 \times 1$

```
> broj <- 5
> faktorijel <- 1  # Početna vrijednost faktorijela
>
> while (broj > 0) {
+   faktorijel <- faktorijel * broj
+   # Množi trenutnu vrijednost faktorijela s trenutnim brojem
+   broj <- broj - 1  # Smanjuje broj za 1
+ }
>
> print(faktorijel)
```

```
## [1] 120
```

U ovom primjeru, while petlja se koristi za izračun faktorijala broja 5. Petlja počinje s brojem 5 i množi trenutnu vrijednost faktorijala s trenutnim brojem sve dok broj ne postane 0.

Prednosti korištenja while petlje:

- Fleksibilnost u izvođenju koda bez prethodnog znanja o broju iteracija.
- Korisna je za situacije kad izvršavanje ovisi o vanjskim uvjetima ili promjenjivim stanjima.

Bitno je pripaziti da uvjet u while petlji u nekom trenutku postane neistinit, inače može dovesti do beskonačne petlje koja može „zamrznuti" ili ozbiljno usporiti izvršavanje programa. Uvijek osigurajte da postoji mehanizam za prekid petlje kako bi se izbjegle potencijalne beskonačne iteracije.

Pitanja za ponavljanje

1. Koja je svrha for petlje u R-u, i kako biste je koristili za ispisivanje imena svih oluja iz skupa podataka storms?

2. Kako koristiti if uvjet za provjeru je li brzina vjetra veća od 100 čvorova i dodati oznaku „Snažna oluja" ili „Slaba oluja" za svaku oluju u storms podatkovnom okviru?

3. Koja je razlika između if uvjeta i ifelse funkcije u R-u, te kada biste koristili jednu, a kada drugu?

4. Objasnite kako ifelse može pojednostaviti klasifikaciju podataka u podatkovnom okviru. Prikažite primjer klasifikacije oluja u storms podatkovnom okviru na temelju brzine vjetra.

5. Što radi apply funkcija i u kojim situacijama je ona efikasnija od for petlje?

6. Kako biste primijenili switch izraz za odabir različitih poruka koje se prikazuju ovisno o sezoni („proljeće", „ljeto", „jesen", „zima")?

7. Objasnite kada biste koristili while petlju umjesto for petlje i navedite primjer za izračun faktorijela broja pomoću while petlje.

8. Kako koristiti lapply funkciju za primjenu operacije na svakom elementu liste u R-u? Dajte primjer u kojem lapply računa kvadrate brojeva u listi.

9. Koje su najčešće greške pri korištenju while petlje, i kako ih izbjeći?

10. Zašto je važno osigurati kraj petlje (npr. u while petlji), i kako biste mogli provjeriti ili izbjeći beskonačnu petlju u R-u?

10 Kreiranje funkcija

Kreiranje vlastitih funkcija u R-u omogućava nam da modulariziramo kod i grupiramo ponavljajuće zadatke u ponovno upotrebljive blokove. Na taj način smanjujemo potrebu za ponavljanjem koda, poboljšavamo čitljivost te omogućavamo brže i jednostavnije promjene u budućnosti. Ovo poglavlje pokriva osnovne primjere, kao što su jednostavne funkcije s jednim argumentom, funkcije s više argumenata i funkcije koje vraćaju više izlaznih vrijednosti koristeći složene strukture podataka, poput lista. Također, upoznat ćemo se s funkcijama koje omogućuju postavljanje zadane vrijednosti za argumente, što korisnicima daje dodatnu fleksibilnost pri pozivanju funkcije bez potrebe za unosom svakog argumenta.

U R-u, funkcije se kreiraju pomoću ključne riječi **function**. Osnovna sintaksa za kreiranje funkcije je sljedeća:

```
ime_funkcije <- function(argumenti) {

# tijelo funkcije

return(vrijednost)

}
```

Gdje je *ime_funkcije* ime koje dajete funkciji, *argumenti* su ulazni parametri funkcije, a *vrijednost* je ono što funkcija vraća. `return()` neće uvijek biti potreban jer R automatski vraća posljednju liniju iz tijela funkcije, ali ako je output strukturiran, treba koristiti tu naredbu.

Evo jednostavnog primjera funkcije koja izračunava kvadrat broja:

```
> kvadrat <- function(x) {
+   rezultat <- x^2
+   return(rezultat)
+ }
```

U ovom primjeru, funkcija kvadrat prima jedan argument x i vraća njegov kvadrat. Možemo isprobati kako radi, tako što ćemo pozvati definiranu funkciju `kvadrat()` i unijeti argument, npr. 5.

```
> kvadrat(5)
```

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

Evo još jednog primjera, funkcije koja računa prosječnu vrijednost niza brojeva:

```
> prosjek <- function(brojevi) {
+   suma <- sum(brojevi)
+   broj_elemenata <- length(brojevi)
+   return(suma / broj_elemenata)
+ }
```

Primjer upotrebe:

```
> vektor <- rnorm(100, 35, 12)
>
> prosjek(vektor)
```

```
## [1] 36.38612
```

Kreiranje funkcija na ovaj način omogućuje vam da zapakirate ponavljajuće zadatke u ponovno upotrebljive blokove koda, što čini vaše programiranje u R-u efikasnijim i urednijim.

10.1 Funkcije s više argumenata

Funkcije mogu imati više argumenata koji omogućuju veću fleksibilnost i prilagodbu njihovog ponašanja.

```
> zbroj_i_umnozak <- function(x, y) {
+   zbroj <- x + y
+   umnozak <- x * y
+   return(list("Zbroj" = zbroj, "Umnozak" = umnozak))
+ }
```

Kreirana je funkcija `zbroj_i_umnozak()` kao primjer funkcije koja prima dva argumenta (x i y), izračunava njihov zbroj i umnožak, te vraća obje vrijednosti kao listu. Postupak kreiranja:

- Funkcija počinje s ključnom riječi *function* nakon koje slijede argumenti funkcije unutar zagrada (x, y). Ovi argumenti su varijable koje funkcija očekuje kada se pozove.
- Unutar tijela funkcije, prvo se izračunava zbroj dvaju brojeva pohranjenih u varijablama x i y, a zatim se izračunava njihov umnožak. Rezultati se pohranjuju u varijable zbroj i umnožak.
- Potom funkcija vraća listu koja sadrži oba rezultata: zbroj i umnožak. Lista je složeni objekt u R-u koji može sadržavati više elemenata različitih tipova. Ovdje, elementi liste imaju nazive „zbroj" i „umnozak" kojima se može pristupiti po imenu.

```
> rezultat <- zbroj_i_umnozak(4, 5)
> print(rezultat)
```

```
## $Zbroj
## [1] 9
##
## $Umnozak
## [1] 20
```

Pozivanje funkcije odvija se na sljedeći način:

- Funkcija se može pozvati s dva brojna argumenta, npr., `zbroj_i_umnozak(4, 5)`. Ovaj poziv funkcije izračunat će zbroj (4 + 5 = 9) i umnožak (4 * 5 = 20) brojeva 4 i 5, te će vratiti listu s rezultatima.
- Korištenjem `print(rezultat)`, gdje je rezultat varijabla u koju je pohranjen povrat funkcije, može se prikazati izlaz funkcije u konzoli.

10.2 Zadane vrijednosti argumenata

Funkcije mogu imati zadane vrijednosti za neke ili sve argumente, što korisnicima omogućava da ih pozovu bez potrebe za specificiranjem svih argumenata. To znači da kada pozivate funkciju, nije potrebno specificirati vrijednost za svaki argument ako je za njega već postavljena defaultna vrijednost. Ovo je korisno za postavljanje standardnih opcija ili vrijednosti koje su najčešće korištene, omogućujući korisniku da preskoči unos tih vrijednosti pri svakom pozivu funkcije.

```
> pozdrav <- function(ime = "Svijete") {
+   poruka <- paste("Pozdrav,", ime, "!")
+   return(poruka)
+ }
>
> pozdrav()
```

```
## [1] "Pozdrav, Svijete !"
```

Ako funkciju pozovemo bez argumenata, ispisuju se zadane vrijednosti. Ova funkcija ispisuje pozdravnu poruku, koristeći zadano ime „Svijete" ako korisnik ne specificira drugačije. No, ako upišemo argument, na primjer, „Ana", onda će upravo taj argument zamijeniti zadani argument.

```
> pozdrav("Ana")
```

```
## [1] "Pozdrav, Ana !"
```

Ako u argument upišemo vektor, funkcija će ponoviti radnju za svaki element vektora.

```
> pozdrav(c("Ana", "Marija", "Lidija"))
```

```
## [1] "Pozdrav, Ana !"    "Pozdrav, Marija !" "Pozdrav, Lidija !"
```

10.3 Funkcije koje vraćaju više vrijednosti

Funkcije mogu vratiti više vrijednosti pomoću listi ili drugih složenih struktura podataka.

```
> pokazatelji <- function(brojevi) {
+   prosjek <- mean(brojevi)
+   medijan <- median(brojevi)
+   maksimum <- max(brojevi)
+   minimum <- min(brojevi)
+   return(list("Prosjek" = prosjek, "Medijan" = medijan,
+               "Maksimum" = maksimum, "Minimum" = minimum))
+ }
```

Pri kreiranju funkcije koja treba vratiti više vrijednosti, uobičajeni pristup je te vrijednosti pohraniti u listu. Lista može sadržavati elemente različitih tipova i dužina, što je čini idealnom za grupiranje više vrijednosti koje funkcija treba vratiti. U navedenom primjeru, funkcija *pokazatelji* prihvaća vektor *brojevi* kao ulazni argument i izračunava nekoliko pokazatelja deskriptivne statistike (prosjek, medijan, maksimum, minimum). Svaki izračunati pokazatelj (prosjek, medijan, maksimum, minimum) kreira zasebnu varijablu. Na kraju, ove se varijable spajaju u listu koristeći funkciju `list()`, gdje se svakoj vrijednosti dodjeljuje ime (npr., „Prosjek", „Medijan", itd.). Ova imena koriste se za identifikaciju pojedinih elemenata liste kada se lista vrati kao rezultat funkcije.

```
> vektor <- c(1, 2, 3, 4, 5, 100)
> rezultat <- pokazatelji(vektor)
> print(rezultat)
```

```
## $Prosjek
## [1] 19.16667
##
## $Medijan
## [1] 3.5
##
## $Maksimum
## [1] 100
##
## $Minimum
## [1] 1
```

Ova funkcija izračunava i vraća prosjek, medijan, maksimum, i minimum niza brojeva. Ovi primjeri ilustriraju kako R omogućuje razvijanje funkcija koje su moćne, fleksibilne i prilagodljive potrebama korisnika.

Kada se funkcija pozove, može se pristupiti pojedinim elementima vraćene liste koristeći oznaku `$` i imena elemenata.

```
> rezultat$Prosjek
```

```
## [1] 19.16667
```

Korištenjem lista na ovaj način, funkcije u R-u mogu efikasno upravljati i vraćati više izlaznih vrijednosti, čineći kod fleksibilnijim i čitljivijim.

Pitanja za ponavljanje

1. Kako se definira funkcija u R-u, i koja je svrha ključne riječi function pri kreiranju funkcije?
2. Koja je razlika između korištenja `return()` unutar tijela funkcije i izostavljanja te naredbe u R-u?
3. Napišite jednostavnu funkciju koja izračunava kub broja. Kako biste pozvali tu funkciju za broj 3?
4. Kako možete kreirati funkciju koja prima dva argumenta i vraća njihov zbroj i umnožak kao listu?
5. Što su zadane vrijednosti argumenata u funkciji i kako one poboljšavaju fleksibilnost poziva funkcije? Navedite primjer.
6. Koje su prednosti korištenja lista kao povrata iz funkcije kada je potrebno vratiti više vrijednosti?
7. Kako možete definirati funkciju koja ima zadanu vrijednost za argument, ali omogućava korisniku da tu vrijednost promijeni pri pozivu funkcije?
8. Kreirajte funkciju koja prima niz brojeva i vraća listu koja sadrži srednju vrijednost, standardnu devijaciju i raspon.
9. Što će se dogoditi ako pozovete funkciju pozdrav s vektorom imena `c("Ana", "Ivan")`? Objasnite kako R rukuje vektorima unutar funkcija.
10. Kako biste u povratnoj vrijednosti iz funkcije pristupili specifičnom elementu liste, primjerice medijanu iz prethodne funkcije koja vraća deskriptivne statističke pokazatelje?

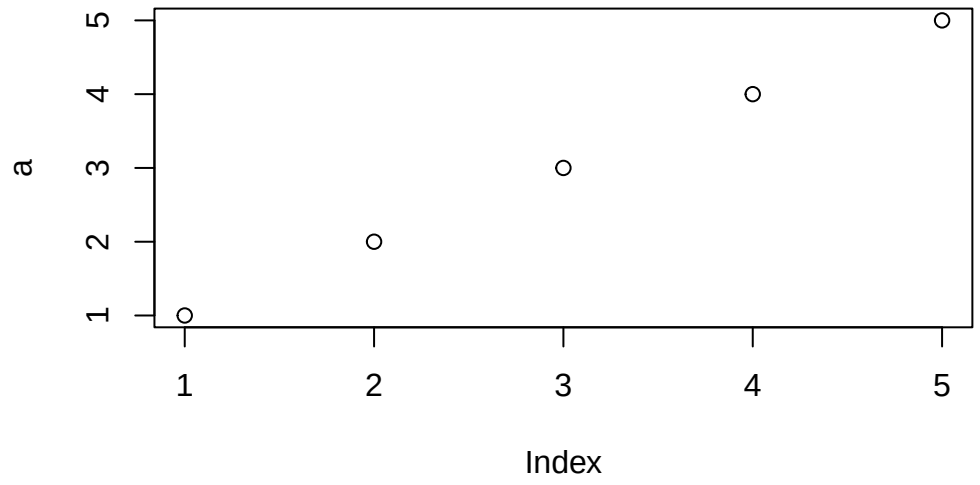
11 Jednostavne vizualizacije

U ovom poglavlju obrađuju se osnovne metode vizualizacije podataka u R-u, koristeći funkcije poput `plot()`, `barplot()`, `hist()`, i `pie()` te paket ggplot2 za napredniju vizualizaciju. Različiti tipovi grafova prikazani su kroz primjere, uključujući dijagrame rasipanja, linijske dijagrame, histogram, torta dijagram, i mozaik grafikon, pri čemu je naglasak stavljen na prilagodbu parametara kao što su boje, nazivi osi i stilovi linija. Na kraju poglavlja, izlažu se se prednosti i nedostaci vizualizacija u R-u, uobičajene pogreške koje treba izbjegavati te preporuke za najbolje prakse, uključujući jednostavnost i upotrebu kontrasta za bolju preglednost.

11.1 Primjeri najčešće korištenih vizualizacija

Osnovna naredba za crtanje grafova u R-u je `plot()`. Naredba će rezultirati različitim prikazom ovisno o tome što se crta. Na primjer, ako želimo nacrtati ranije kreirani vektor a, onda će to izgledati ovako:

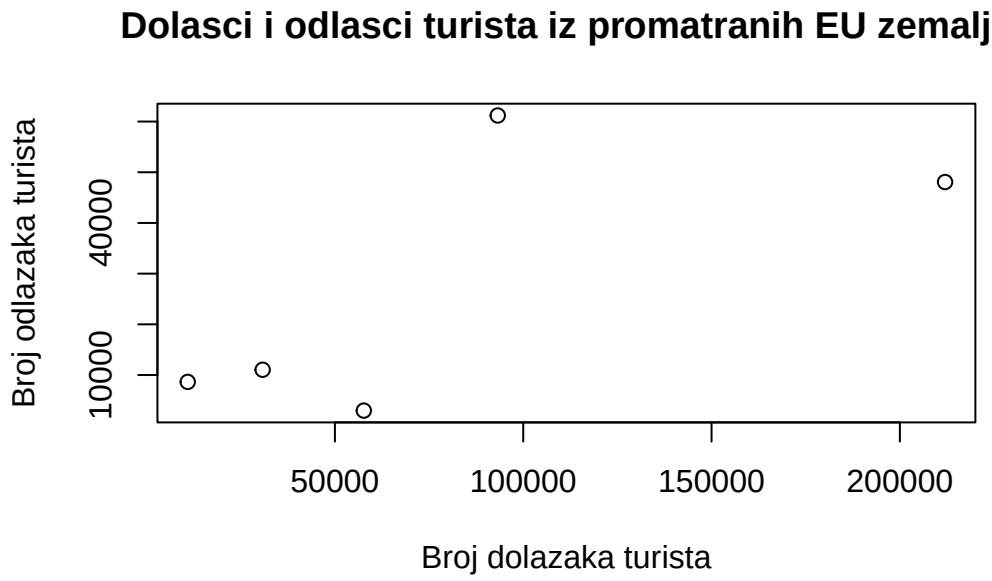
```
> plot(a)
```



Na x-osi naći će se redni broj opažanja, a na y-osi bit će vrijednosti opažanja. Bez dodatnih parametara, ovo je vrlo neinformativan graf.

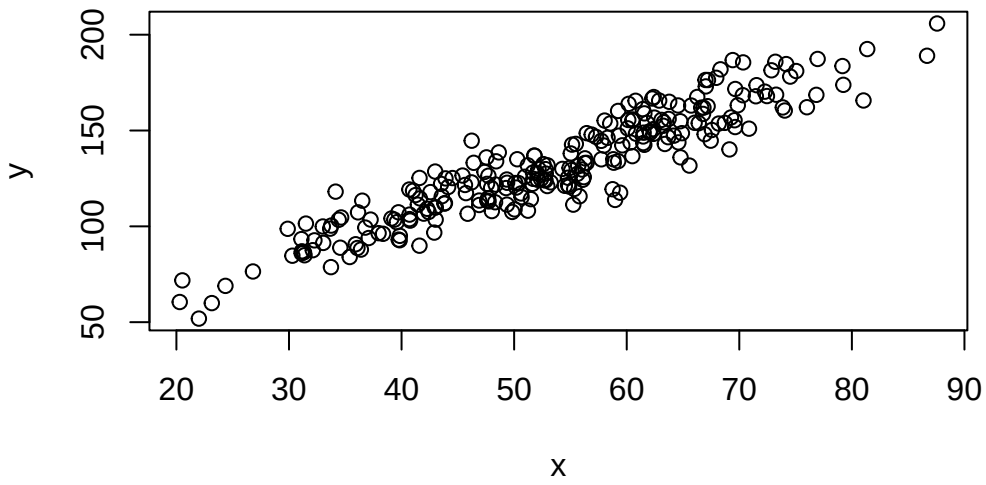
No, ako bismo grafički prikazali broj dolazaka i odlazaka turista iz promatranih EU zemalja (podatkovni okvir Tur_eu), onda dobivamo **dijagram rasipanja**. Dodatno, specificiran je naslov koristeći parametar *main* te nazivi apscise (koristeći *xlab*) i ordinate (koristeći *ylab*).

```
> plot(Tur_eu[ , 2:3],
+      main = "Dolasci i odlasci turista iz promatranih EU zemalja",
+      xlab = "Broj dolazaka turista",
+      ylab = "Broj odlazaka turista")
```



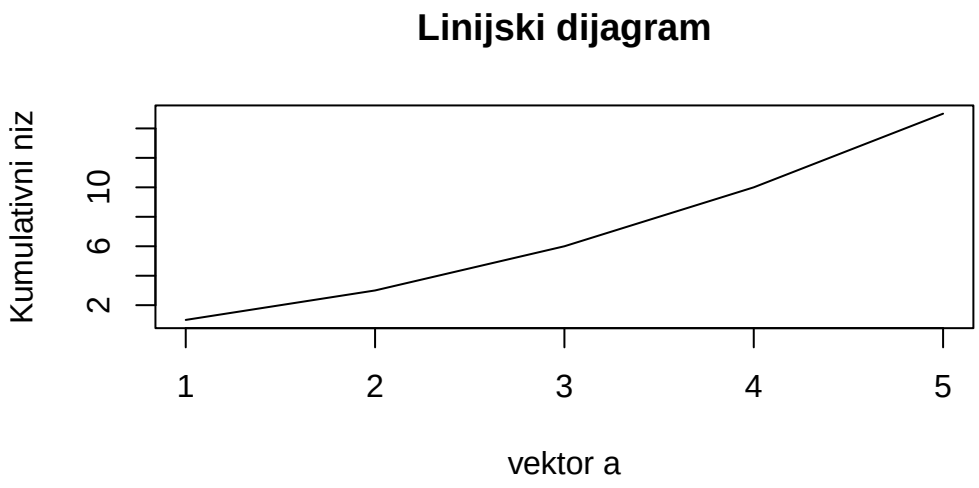
Ovaj tip grafa prikladini je za veći broj opažanja, kao u sljedećem primjeru, u kojem se simulira povezanost dviju varijabli:

```
> x <- rnorm(250, 55, 15)
> z <- rnorm(250, 10, 10)
> y <- 15 + z + 2 * x
>
> plot(x, y)
```



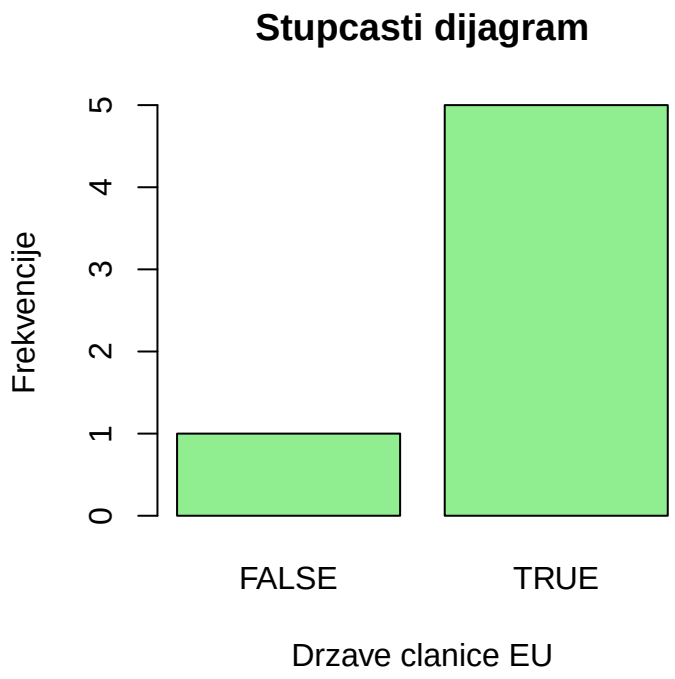
Osim toga, `plot()` se može koristiti za kreiranje linijskog dijagrama:

```
> y <- cumsum(a)
>
> plot(a, y, type = "l", main = "Linijski dijagram",
+      xlab = "vektor a", ylab = "Kumulativni niz")
```



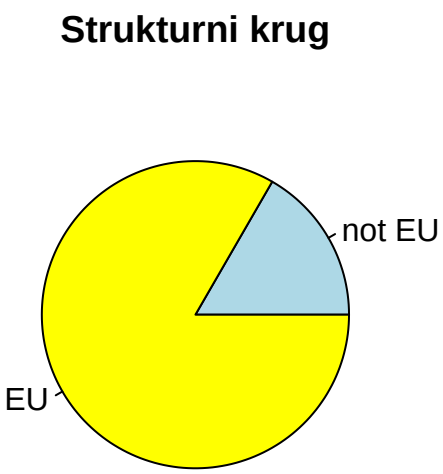
Nadalje, moguće je kreiranje stupčastog dijagrama koristeći `barplot()` naredbu. U ranijim primjerima nisu korištene izmjene u bojama, ali se mogu provesti na isti način i to podešavanjem parametra *col*.

```
> counts <- table(Tur_ord$EU)
>
> # Kreiranje stupčastog dijagrama
> barplot(counts,
+         main = "Stupcasti dijagram",
+         xlab = "Drzave clanice EU", ylab = "Frekvencije",
+         col = "lightgreen")
```



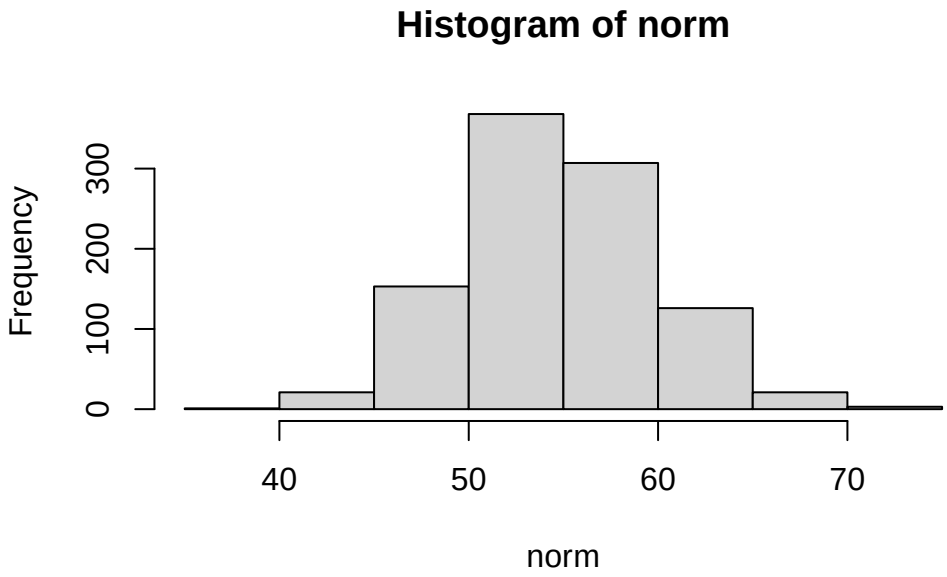
Na sličan način moguće je kreirati i strukturni krug, poznatiji kao torta dijagram.

```
> pie(counts, labels = c("not EU", "EU"),
+     main = "Strukturni krug",
+     col = c("lightblue", "yellow"))
```



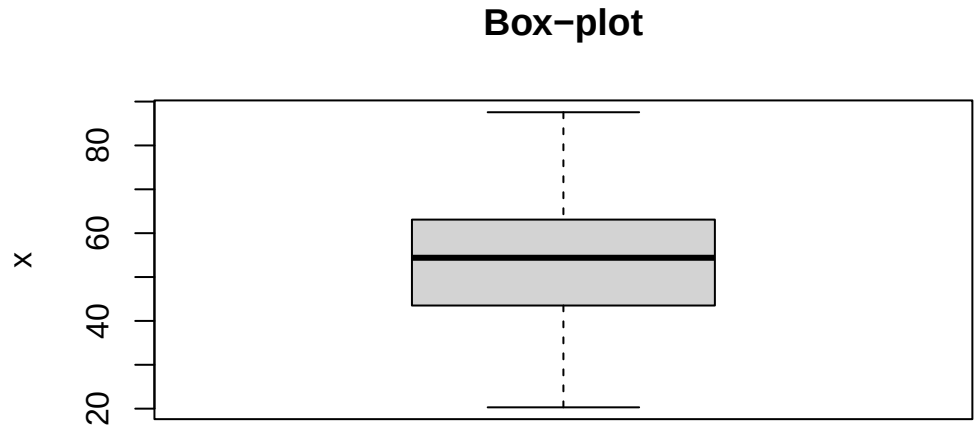
Jedan od najčešće korištenih grafičkih prikaza svakako je histogram. Prikazat ćemo histogram ranije kreiranog vektora *norm*. Hist ima više različitih parametara koji se mogu definirati, a između ostalih i pravilo prema kojem se definiraju razredi.

```
> hist(norm)
```



Uz histogram, izuzetno se često koristi i box-plot dijagram (ili kutijasti dijagram). Ovaj je dijagram vrlo koristan za dobivanje uvida u raspršenost podataka.

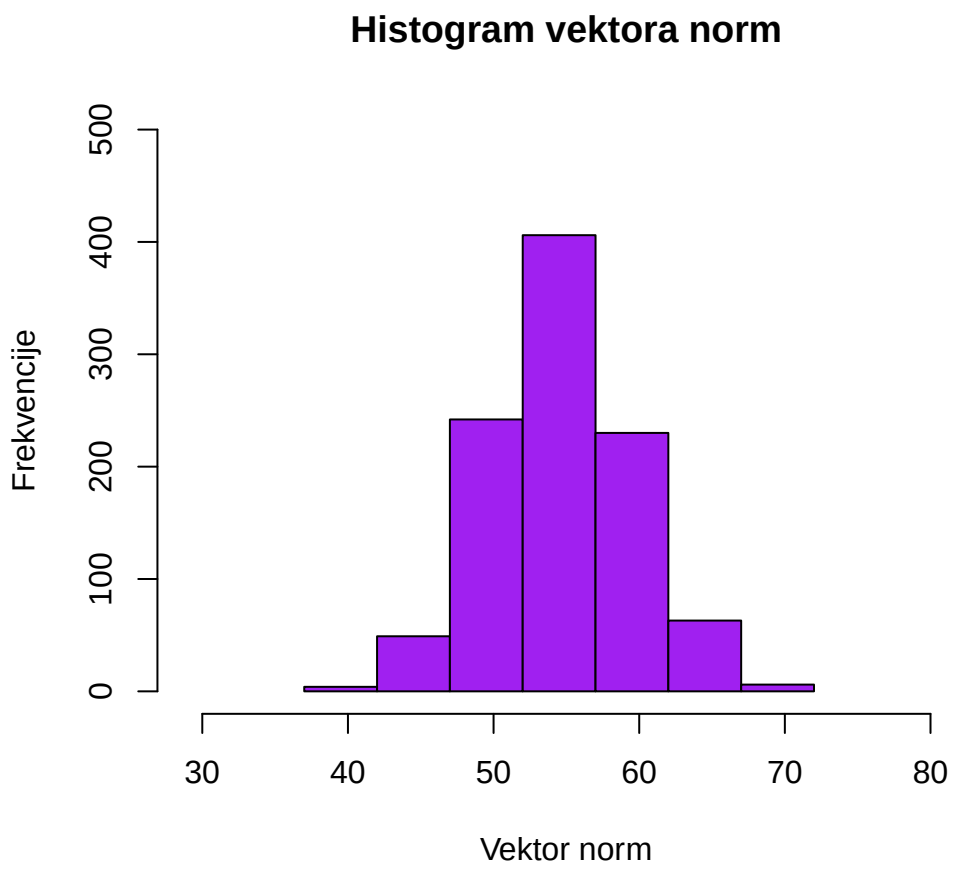
```
> boxplot(x, main = "Box-plot", ylab = "x")
```



11.2 Prilagodba grafikona

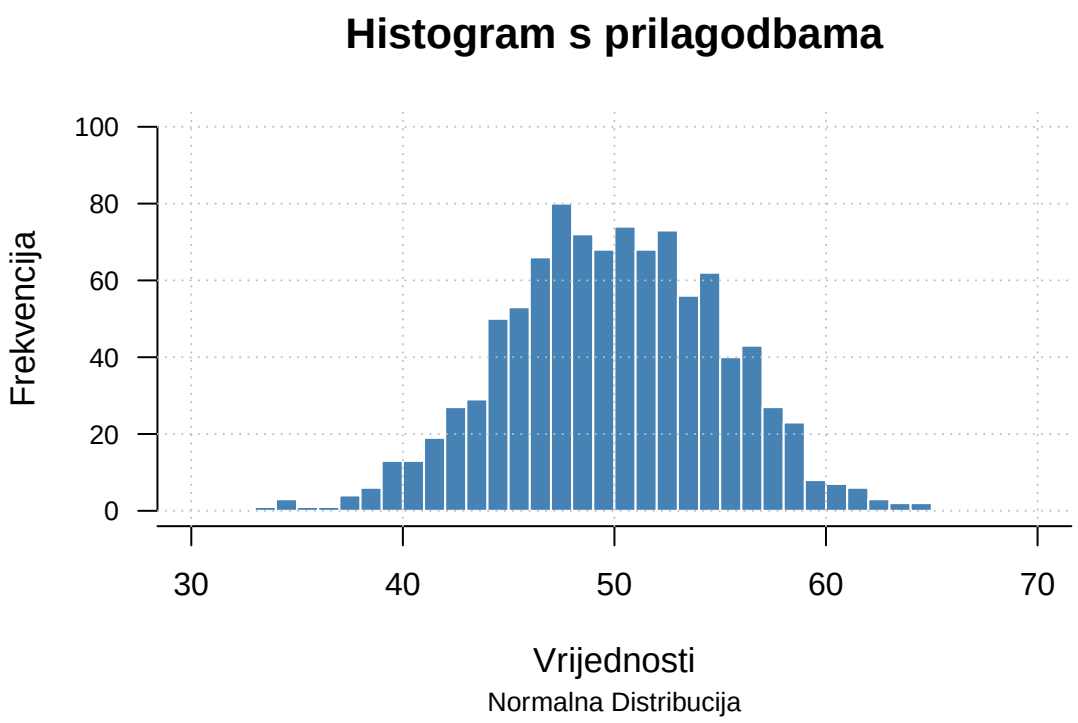
Osnovna postavka je korištenje Sturgesovog pravila pri kreiranju razreda, no granice razreda se mogu zadati u obliku vektora.

```
> razredi <- seq(round(min(norm)-2), round(max(norm)+2), 5)
>
> hist(norm,
+      breaks = razredi,                 #granice razreda
+      main = "Histogram vektora norm", #promjena naziva grafa
+      xlim = c(round(min(norm)-10), round(max(norm)+10)),  # promjena raspona apscise
+      xlab = "Vektor norm",             #promjena naziva apscise
+      ylim = c(0, 500),                   #promjena raspona ordinate
+      ylab = "Frekvencije",             #promjena naziva ordinate
+      col = "purple"                    #promjena boje površine histograma
+      )
```



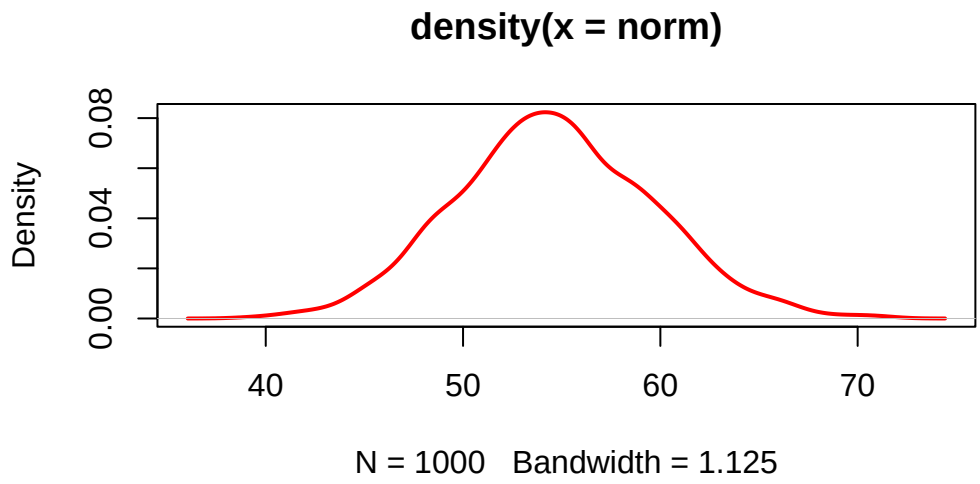
Evo jednog primjera s jako puno podešenih parametara pri crtanju histograma.

```
> data <- rnorm(1000, mean = 50, sd = 5)
>
> # Stvaranje prilagođenog histograma
> hist(data,
+      breaks = 30,                    # Broj stupaca
+      col = "steelblue",              # Boja stupaca
+      border = "white",               # Boja ruba stupaca
+      main = "Histogram s prilagodbama", # Naslov histograma
+      xlab = "Vrijednosti",           # Oznaka za x-os
+      ylab = "Frekvencija",           # Oznaka za y-os
+      xlim = c(30, 70),               # Granica za x-os
+      ylim = c(0, 100),               # Granica za y-os
+      las = 1,                        # Orjentacija oznaka osi
+      cex.axis = 0.8,                 # Veličina oznaka osi
+      cex.lab = 1.1,                  # Veličina naslova osi
+      cex.main = 1.3,                 # Veličina glavnog naslova
+      cex.sub = 0.8,                  # Veličina podnaslova
+      sub = "Normalna Distribucija",  # Podnaslov
+      xaxt = "n")                     # Sakriti oznake x-osi
>
> # Dodavanje prilagođenih oznaka x-osi
> axis(1, at = seq(20, 80, by = 10), labels = seq(20, 80, by = 10))
>
> # Dodavanje prilagođene mreže
> grid(nx = NULL, ny = NULL, col = "gray", lty = "dotted")
```



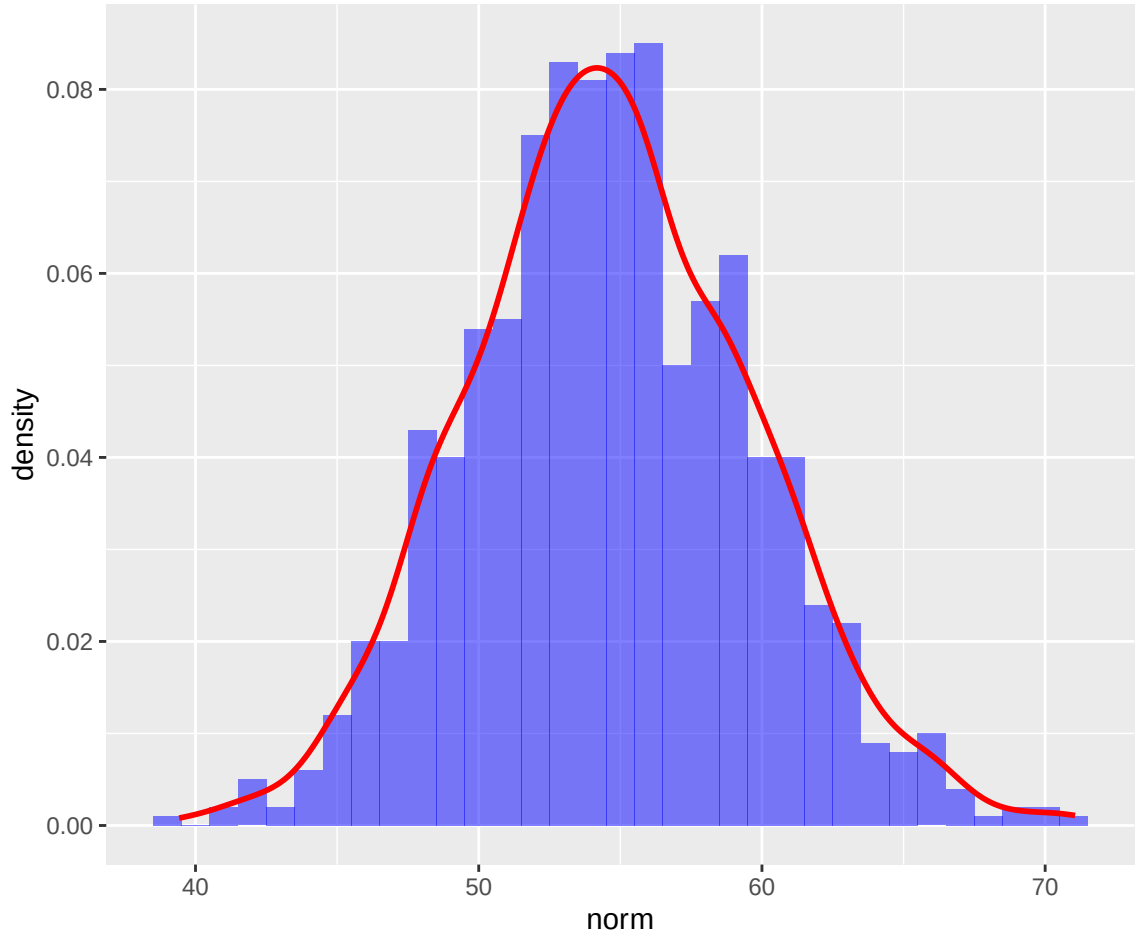
Nadalje, za vizualno preispitivanje i stvaranje procjene funkcije gustoće vjerojatnosti za kontinuirani skup podataka koristi se `density()` u kombinaciji s naredbom `plot()`.

```
> dens <-density(norm)
> plot(dens, col = "red", lwd = 2)
```



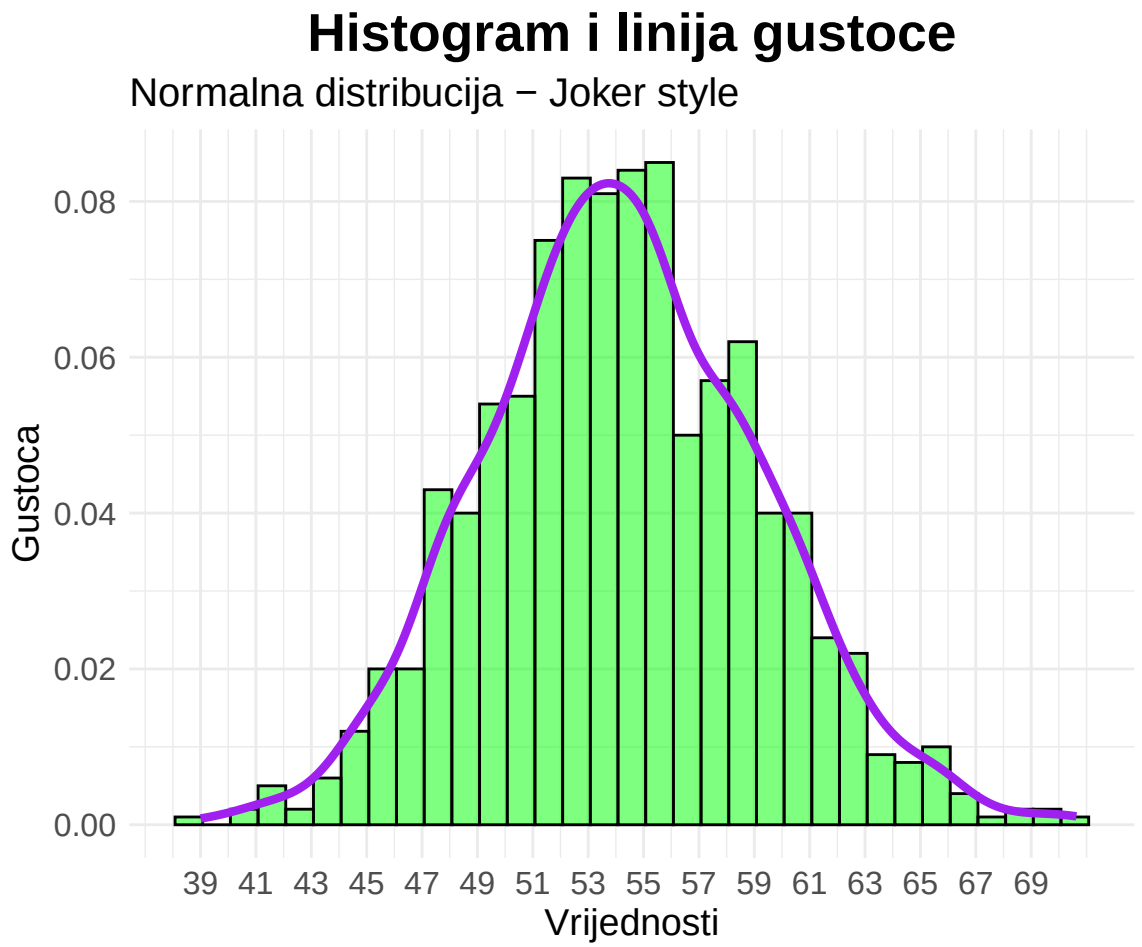
Nešto opsežnije mogućnosti grafičkih prikaza s više prilagođenih postavki mogu se ostvariti koristeći paket ggplot2 (Wickham and Wickham 2016). Ovdje je prikazan primjer u kojem se pomoću ggplot2 kreiraju slojevi grafičkog prikaza, uključujući histogram s prilagođenom bojom, širinom stupa i prozirnosti te krivulju gustoće s prilagođenom bojom i debljinom linije.

```
> library(ggplot2)
> norm<-as.data.frame(norm)
> ggplot(norm, aes(x = norm)) +
+   geom_histogram(aes(y = after_stat(density)),
+                  binwidth = 1, fill = "blue", alpha = 0.5) +
+   geom_density(color = "red", linewidth = 1)
```



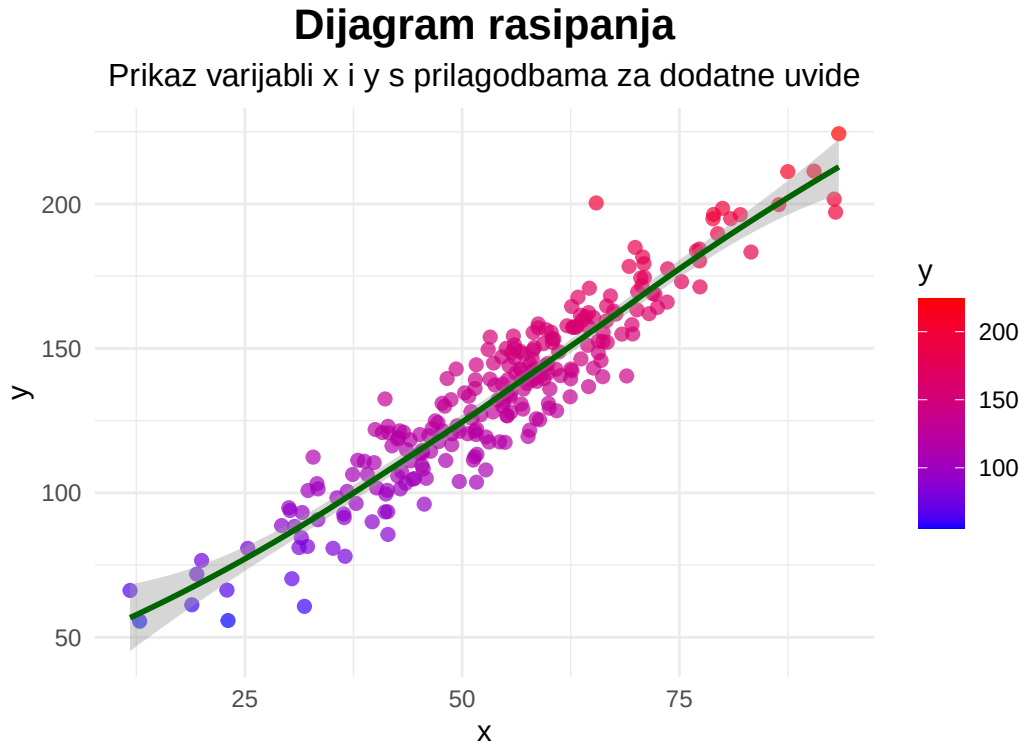
Nakon toga, prelazimo na grafički prikaz histograma i krivulje gustoće s većom količinom podešenih parametara. Objašnjenja za podešene parametre unesena su u kodu.

```
> # postavljanje broja znamenki u ispisu
> # (u ovom slučaju, u ispisu brojeva na osima)
> options(digits = 2)
>
> # Kreiranje histograma s linijom gustoće koristeći ggplot2 s prilagodbama
> ggplot(norm, aes(x = norm)) +
+   # dodavanje histograma
+   geom_histogram(aes(y = after_stat(density)),
+                  # crtanje histograma sa širinom stupca podešenom na 1
+                  binwidth = 1,
+                  # ispuna histograma
+                  fill = "green",
+                  # parametar koji kontrolira prozirnost elemenata na grafikonu
+                  alpha = 0.5,
+                  # boja obruba
+                  color = "black") +
+   # Dodavanje linije gustoće
+   # dodavanje linije gustoće, boja i debljina linije
+   geom_density(color = "purple", size = 1.5) +
+   # Prilagodba naslova i oznaka osi
+   # naslov
+   # U pravilu, paketi u R-u nisu prilagođeni za ispis naziva koji sadrže
+   # dijakritičke znakove
+   labs(title = "Histogram i linija gustoce",
+        # podnaslov
+        subtitle = "Normalna distribucija - Joker style",
+        # naziv apscise
+        x = "Vrijednosti",
+        # naziv ordinate
+        y = "Gustoca") +
+   # prilagodba oznaka na osima
+   scale_x_continuous(breaks = seq(min(norm), max(norm), by = 2)) +
+   # Prilagodba tema
+   theme_minimal() +
+   # Prilagodba elemenata tema
+   theme(
+     plot.title = element_text(size = 20, face = "bold", hjust = 0.5),
+     plot.subtitle = element_text(size = 15),
+     axis.title.x = element_text(size = 14),
+     axis.title.y = element_text(size = 14),
+     axis.text = element_text(size = 12)
+   )
```


Histogram i linija gustoce
Normalna distribucija − Joker style
Gustoca
0.08
0.06
0.04
0.02
0.00
39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69
Vrijednosti

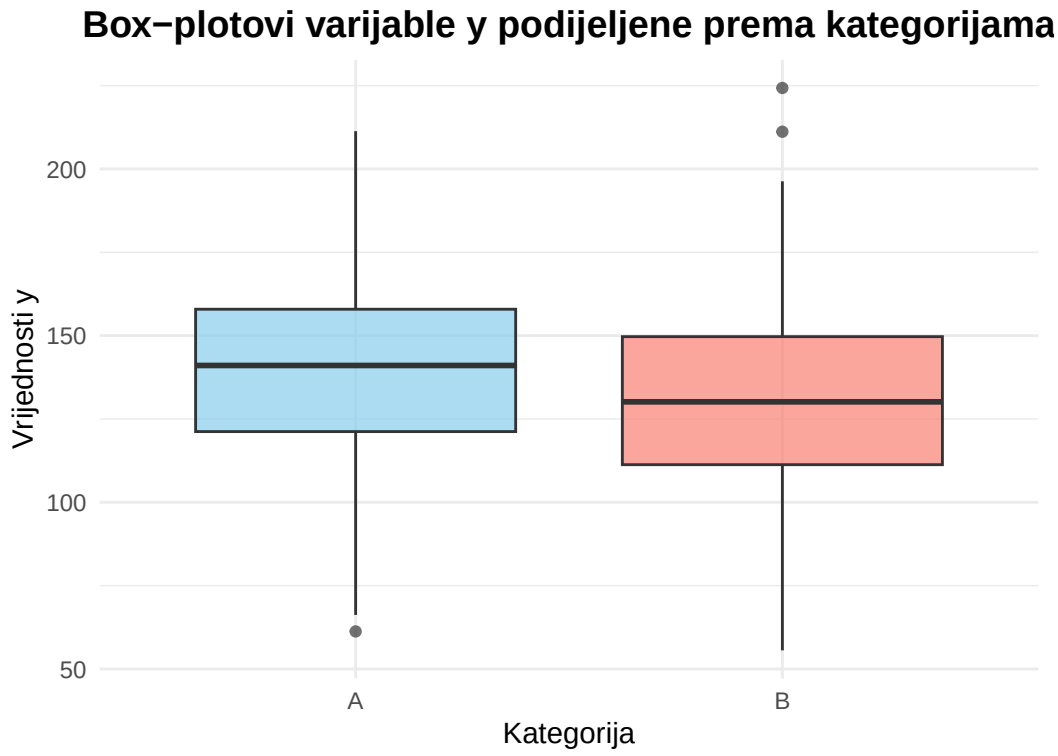
Na sličan način, možemo unaprijediti prikaz dijagrama raspipanja, koji je ranije kreiran koristeći naredbu `plot()`.

```
> x <- rnorm(250, 55, 15)
> z <- rnorm(250, 10, 10)
> y <- 15 + z + 2 * x
> data <- data.frame(x = x, y = y)
> ggplot(data, aes(x = x, y = y)) +
+   geom_point(aes(color = y), size = 2, alpha = 0.7) +
+   scale_color_gradient(low = "blue", high = "red") +
+   geom_smooth(method = "loess", se = TRUE, color = "darkgreen") +
+   labs(
+     title = "Dijagram rasipanja",
+     subtitle = "Prikaz varijabli x i y s prilagodbama za dodatne uvide",
+     x = "x",
+     y = "y"
+   ) +
+   theme_minimal() +
+   theme(
+     plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
+     plot.subtitle = element_text(hjust = 0.5, size = 12),
+     legend.position = "right"
+   )
```



Recimo da je ovom skupu podataka pridružena još jedna, kategorijska varijabla, koja se odnosi na treću promatranu karakteristiku ovog skupa. Promotrimo kako možemo prikazati varijablu y s obzirom na podjelu koju stvara dodana varijabla.

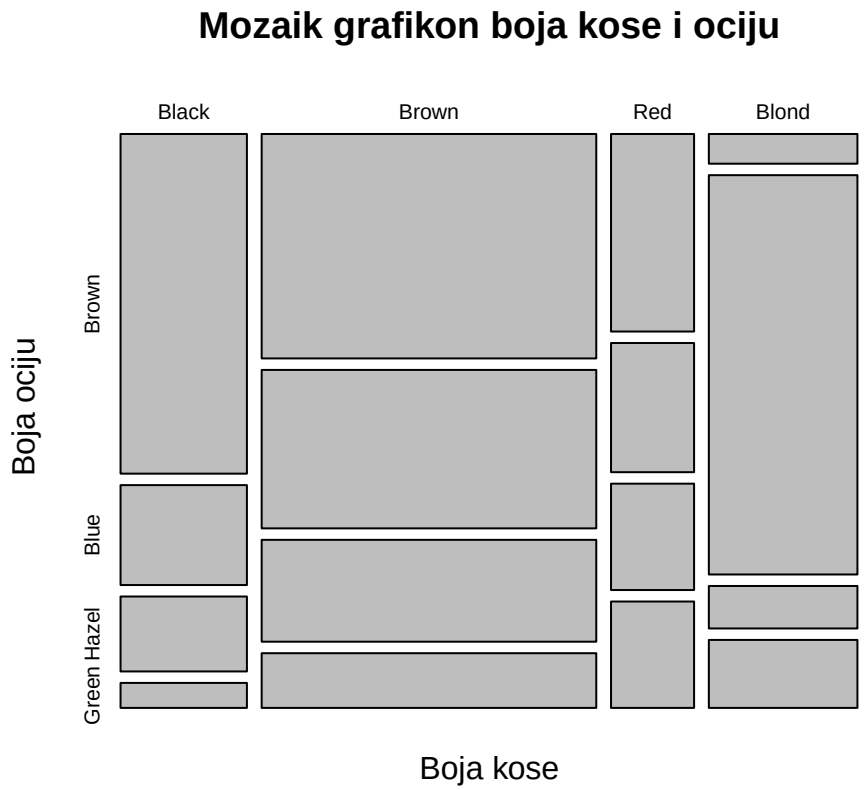
```
> category <- factor(sample(c("A", "B"), 250, replace = TRUE))
> data <- data.frame(x = x, y = y, category = category)
>
> ggplot(data, aes(x = category, y = y, fill = category)) +
+   geom_boxplot(alpha = 0.7) +
+   scale_fill_manual(values = c("skyblue", "salmon")) +
+   labs(
+     title = "Box-plotovi varijable y podijeljene prema kategorijama",
+     x = "Kategorija",
+     y = "Vrijednosti y"
+   ) +
+   theme_minimal() +
+   theme(
+     plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
+     legend.position = "none"
+   )
```



Ovakav prikaz omogućuje usporedbu raspršenosti podskupina podataka.

Osim navedenih, mozaik grafikon (*engl. mosaic plot*) je vizualni prikaz koji se koristi za prikazivanje podataka iz kombiniranih tablica (tablica kontingence). Za prikaz opisnih podataka ciljano je kreiran paket vcd (Meyer et al. 2023).

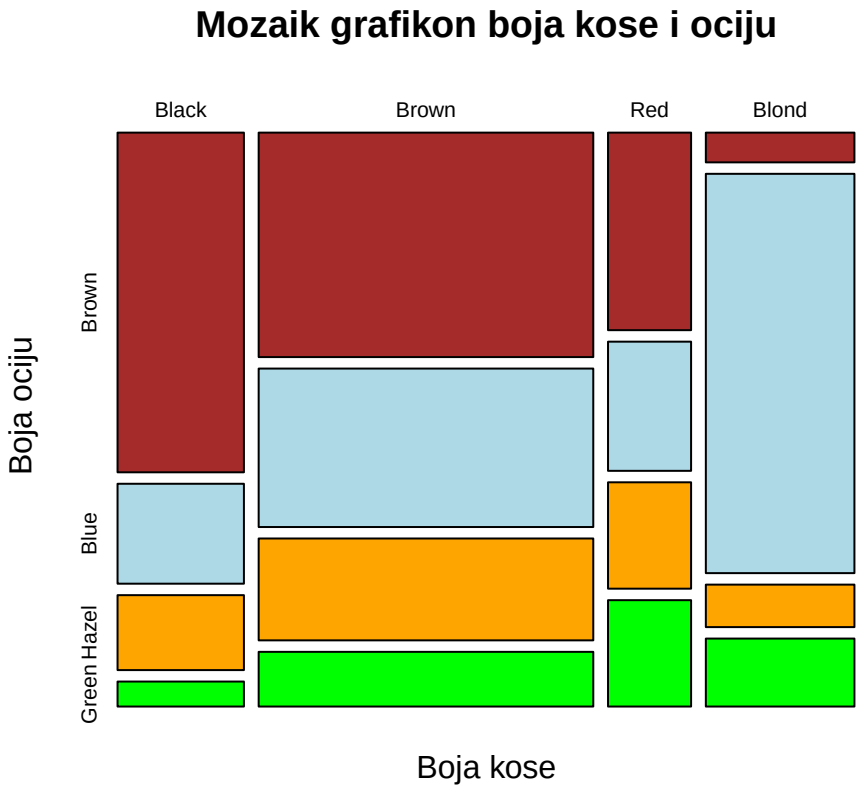
```
> library(vcd)
>
> # Primjer tablice kontingence
>
> # učitavanje postojećih podataka iz paketa vcd
> data(HairEyeColor)
> tablica <- margin.table(HairEyeColor, margin = c(1, 2))
>
> # Kreiranje mozaik grafikona
> mosaicplot(tablica, main = "Mozaik grafikon boja kose i ociju",
+            xlab = "Boja kose", ylab = "Boja ociju")
```



I u ovom se grafikonu mogu dodatno prilagođavati boje.

```
> boje <- c("brown", "lightblue", "orange", "green")
>
```

```
> mosaicplot(tablica, main = "Mozaik grafikon boja kose i ociju",
+            xlab = "Boja kose", ylab = "Boja ociju",
+            color = boje)
```



11.3 Prednosti i nedostaci vizualizacija u R-u te najčešće pogreške

Prednosti vizualizacije podataka u R-u

- **Raznolikost tipova grafova**: R podržava širok spektar tipova grafova i vizualizacija, omogućavajući prilagodbu prikaza podacima i analitičkim ciljevima,
- **fleksibilnost i prilagodba**: napredne opcije za prilagodbu omogućuju precizno podešavanje izgleda grafova, od boja i tipova linija do složenih rasporeda i integracija s drugim elementima kao što su tekst i elementi oblikovanja,
- **integracija s analitičkim radnim tokovima**: vizualizacije se lako integriraju s ostatkom analitičkog procesa u R-u, omogućavajući brzu iteraciju između analize i vizualizacije.

Uobičajene greške pri kreiranju vizualizacija

- **Previše detalja ili složenih elemenata** može dovesti do konfuznih grafova koji otežavaju razumijevanje podataka,
- **Zanemarivanje ciljne publike**: vizualizacije koje nisu prilagođene razumijevanju i potrebama ciljne publike mogu rezultirati neučinkovitom komunikacijom,
- **Neadekvatna priprema podataka**, odnosno greške u pripremi podataka, kao što su neobrađene iznimke ili nepravilno skaliranje, mogu dovesti do netočnih vizualizacija ili vizualizacija koje mogu biti pogrešno protumačene.

Najbolje prakse za kreiranje vizualizacija

- **Jednostavnost** (usmjerenost na ključne elemente koji komuniciraju podatke),
- **Kontrast i boja** (poboljšanje čitljivosti),
- **Preglednost i pristupačnost** (grafovi razumljivi širokom rasponu korisnika).

Isprobajte različite tipove grafova

Različiti skupovi podataka i analitički ciljevi zahtijevaju različite pristupe vizualizaciji. Eksperimentiranje s različitim tipovima grafova može otkriti učinkovitije načine prikaza podataka.

Upotreba paketa za vizualizaciju poput ggplot2

Paket ggplot2 nudi moćan i fleksibilan sustav za kreiranje kompleksnih vizualizacija podataka s gramatikom grafičkog dizajna. Iskoristite njegove napredne mogućnosti za stvaranje visokokvalitetnih grafova.

Pitanja za ponavljanje

1. Kako funkcija `plot()` određuje što će se prikazati na grafu ako se ne dodaju dodatni parametri? Prikazati primjer korištenja `plot()` za dijagram rasipanja.

2. Koje parametre možete prilagoditi pomoću `plot()` kako biste dodali naslov, oznake osi, i prilagodili boje u grafu?

3. Koja je razlika između dijagrama rasipanja i linijskog dijagrama? Kako promijeniti `plot()` funkciju tako da prikazuje linijski dijagram?

4. Objasnite primjenu `barplot()` funkcije i navedite primjer kako kreirati stupčasti dijagram s prilagođenom bojom.

5. Kako funkcija `hist()` automatski postavlja granice razreda, i koje su dodatne prilagodbe koje možete izvršiti na histogramu (npr. promjena boje, dodavanje naziva osi)?

6. Kada biste koristili `pie()` funkciju za vizualizaciju podataka, i koji su osnovni parametri za prilagodbu torta dijagrama?

7. Što omogućava `density()` funkcija, i kako koristiti `plot()` za prikaz krivulje gustoće?

8. Objasnite razliku između klasičnog histograma i histogram prikaza pomoću ggplot2 paketa, te kako dodati liniju gustoće na histogram koristeći `geom_density()`.

9. Kada se koristi mozaik grafikon, i kako se prilagođava boja prikaza pomoću `mosaicplot()`?

10. Koje su najbolje prakse za odabir boja i nivoa složenosti u vizualizacijama, posebno kada je ciljna publika šira i manje tehnički orijentirana?

12 Primjer upravljanja podacima

Ovdje će se prikazati osnove upravljanja podacima na datasetu 'starwars' iz paketa dplyr (Yarberry and Yarberry 2021). Ovo poglavlje pruža praktičan uvid u primjenu tehnika upravljanja podacima u R-u, koristeći dataset starwars iz paketa dplyr. Kroz primjere se ilustrira kako pristupiti podacima, odabrati i filtrirati relevantne varijable te kako raditi s podacima koji nedostaju. Posebna pažnja posvećena je osnovnim tehnikama čišćenja podataka: zamjena vrijednosti koje nedostaju, formatiranje podataka te pretvaranje višestrukih unosa u jedinstvene zapise. Prikazani su primjeri kako se koristiti složenim tabličnim prikazima, poput tablica kontingencije, dok se dodatnim statističkim izračunima i vizualizacijama analizira raspodjela i povezanost unutar podataka.

```
> library(tidyverse)
> library(dplyr)
```

```
> data("starwars")
```

```
> data("starwars")
> glimpse(starwars)
```

```
## Rows: 87
## Columns: 14
## $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Leia Or~
## $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 180, 2~
## $ mass       <dbl> 77, 75, 32, 136, 49, 120, 75, 32, 84, 77, 84, NA, 112, 80, ~
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "brown", N~
## $ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "light", "~
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue", "blue",~
## $ birth_year <dbl> 19, 112, 33, 42, 19, 52, 47, NA, 24, 57, 42, 64, 200, 29, 4~
## $ sex        <chr> "male", "none", "none", "male", "female", "male", "female",~
## $ gender     <chr> "masculine", "masculine", "masculine", "masculine", "femini~
## $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan", "T~
## $ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human", "Huma~
## $ films      <list> <"A New Hope", "The Empire Strikes Back", "Return of the J~
## $ vehicles   <list> <"Snowspeeder", "Imperial Speeder Bike">, <>, <>, <>, "Imp~
## $ starships  <list> <"X-wing", "Imperial shuttle">, <>, <>, "TIE Advanced x1",~
```

Prvo, vidimo da podatkovni okvir sadrži 14 varijabli sa 87 opažanja:

- varijabla ime je tipa „character", kao što i očekujemo
- varijabla visine je numerička cjelobrojna varijabla, „int"
- varijabla masa/ težina je numerička varijable, „double"
- varijable boja kose, boja kože, boja očiju, spol, rod, planet s kojeg potječu i vrsta su tipa „character
- filmovi, vozila i svemirski brodovi su liste

Dodatni uvid u strukturu podataka:

```
> head(starwars)
```

```
## # A tibble: 6 x 14
##   name      height  mass hair_color skin_color eye_color birth_year sex   gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
## 1 Luke Sky~    172    77 blond      fair       blue            19   male  mascu~
## 2 C-3PO        167    75 <NA>       gold       yellow         112   none  mascu~
## 3 R2-D2         96    32 <NA>       white, bl~ red             33   none  mascu~
## 4 Darth Va~    202   136 none       white      yellow          41.9 male  mascu~
## 5 Leia Org~    150    49 brown      light      brown           19   fema~ femin~
## 6 Owen Lars    178   120 brown, gr~ light      blue            52   male  mascu~
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

Ovdje ćemo se detaljnije pozabaviti podatkovnim okvirom, pa ćemo odabrati nekoliko varijabli za daljnje uvide. Odabirom stupaca s određenim varijablama (bilo putem rednih brojeva stupaca ili naziva) kreirat ćemo novi podatkovni skup.

```
> Starwars <- as.data.frame(starwars[,c(1:4,6,10,11)])
```

```
> glimpse(Starwars)
```

```
## Rows: 87
## Columns: 7
## $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Leia Or~
## $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 180, 2~
## $ mass       <dbl> 77, 75, 32, 136, 49, 120, 75, 32, 84, 77, 84, NA, 112, 80, ~
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "brown", N~
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue", "blue",~
## $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan", "T~
## $ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human", "Huma~
```

Provjeru podataka koji nedostaju izvršit ćemo kombinacijom dvije funkcije, `summary()` i `is.na()`. Druga funkcija služi provjeri nalaze li se u varijabli NA (vrijednosti koje nedostaju), a `summary()` će poslužiti za jasan prikaz.

```
> summary(is.na(Starwars))
```

```
##     name           height           mass         hair_color
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:87        FALSE:81        FALSE:59        FALSE:82
##                  TRUE :6         TRUE :28        TRUE :5
##  eye_color       homeworld        species
##  Mode :logical   Mode :logical   Mode :logical
##  FALSE:87        FALSE:77        FALSE:83
##                  TRUE :10        TRUE :4
```

Za 6 likova nedostaje podatak o visini, za 28 nedostaje podatak o težini, za 5 nedostaje podatak o boji kose…. Ukratko, is.na(starwars) provjerava je li vrijednost prisutna (FALSE) ili nedostaje (TRUE), dok summary te podatke sažima u zbroj podataka koji su navedeni i onih koji nedostaju.

Recimo da pretpostavljamo da bismo mogli znati podatke koji nedostaju o vrsti, pa želimo identificirati gdje se nalaze NA, kako bismo ih eventualno izmijenili.

```
> na_pozicije <- which(is.na(Starwars$species))
> na_pozicije
```

```
## [1] 18 59 60 81
```

NA se nalaze na pozicijama 18, 59, 60 i 81. Moramo provjeriti o kojim se likovima radi.

```
> Starwars$name[na_pozicije]
```

```
## [1] "Jek Tono Porkins" "Gregar Typho"     "Cord\xe9"         "Sly Moore"
```

Kratkim pretraživanjem, možemo saznati kojim vrstama pripadaju ovi likovi:

- Jek Tono Porkins: Human
- Gregar Typho: Human
- Cordé: Human
- Sly Moore: Umbaran

Sada možemo zamijeniti postojeće NA poznatim podacima. To možemo učiniti na dva načina. S obzirom da ih je malo, možemo koristiti „ručni” unos:

```
> Starwars$species[18] <- "Human"
> Starwars$species[59] <- "Human"
> Starwars$species[60] <- "Human"
> Starwars$species[81] <- "Umbaran"
```

Provjera:

```
> summary(is.na(Starwars$species))
```

```
##    Mode   FALSE
## logical      87
```

Svi su podaci uspješno zamijenjeni i ta varijabla više ne sadrži NA. Kad bi u pitanju bilo više vrijednosti, onda bi bilo praktičnije prvo kreirati vektor koji mapira imena likova na njihove vrste. Potom bi se koristila petlja for ili vektorizirana funkcija za pregled redova u podatkovnom okviru i zamijenile NA vrijednosti odgovarajućim vrstama:

```
> species_replacements <- c("Jek Tono Porkins" = "Human",
+                           "Gregar Typho" = "Human",
+                           "Cordé" = "Human",
+                           "Sly Moore" = "Umbaran")
> for (name in names(species_replacements)) {
+     na_indices <- is.na(Starwars$species) & Starwars$name == name
+     Starwars$species[na_indices] <- species_replacements[name]
+ }
```

Ovaj kod prvo identificira redove u kojima je species NA i gdje se ime lika podudara s ključem u species_replacements. Zatim zamjenjuje NA odgovarajućom vrstom.

Sljedeće, provjeravamo za boju kose.

```
> na_pozicije <- which(is.na(Starwars$hair_color))
> na_pozicije
```

```
## [1]  2  3  8 15 16
```

```
> Starwars$name[na_pozicije]
```

```
## [1] "C-3PO"                 "R2-D2"                 "R5-D4"
## [4] "Greedo"                "Jabba Desilijic Tiure"
```

Za navedene likove boja kose nije primjenjiva. Kako se ne bismo bavili s NA, možemo ih zamijeniti s, na primjer „Unknown" i odmah provjeriti provedbu zamjene.

```
> Starwars$hair_color[na_pozicije]<- "Unknown"
>
> summary(is.na(Starwars$hair_color))
```

```
##    Mode   FALSE
## logical      87
```

```
> unique(Starwars$hair_color)
```

```
##  [1] "blond"         "Unknown"       "none"          "brown"
##  [5] "brown, grey"   "black"         "auburn, white" "auburn, grey"
##  [9] "white"         "grey"          "auburn"        "blonde"
```

Druga stvar koja se može primijetiti kod boje kose jest da se pojavljuje Unknown i unknown kao zasebna „boja", iako se radi o pojmovno istoj stvari. To je zbog toga što je R osjetljiv na velika i mala slova. No to možemo jednostavno riješiti tako što ćemo sve znakove u varijabli hair_color podesiti na mala slova.

```
> Starwars$hair_color<-tolower(Starwars$hair_color)
> unique(Starwars$hair_color)
```

```
##  [1] "blond"         "unknown"       "none"          "brown"
##  [5] "brown, grey"   "black"         "auburn, white" "auburn, grey"
##  [9] "white"         "grey"          "auburn"        "blonde"
```

Druga stvar koju se može primijetiti kod boje kose jest da da su za neke likove zapisane dvije boje. Recimo da želimo zadržati samo prvu zapisanu boju. To znači da prvo moramo razdvojiti zapisane boje, a potom pohraniti prvu boju u varijablu hair_color.

```
> hair_colors_first <- sapply(strsplit(Starwars$hair_color, ", "), `[`, 1)
> hair_colors_first
```

```
##  [1] "blond"   "unknown" "unknown" "none"    "brown"   "brown"   "brown"
##  [8] "unknown" "black"   "auburn"  "blond"   "auburn"  "brown"   "brown"
## [15] "unknown" "unknown" "brown"   "brown"   "white"   "grey"    "black"
## [22] "none"    "none"    "black"   "none"    "none"    "auburn"  "brown"
## [29] "brown"   "none"    "brown"   "none"    "blond"   "brown"   "none"
## [36] "none"    "none"    "brown"   "black"   "none"    "black"   "black"
## [43] "none"    "none"    "none"    "none"    "none"    "none"    "none"
## [50] "none"    "white"   "none"    "black"   "none"    "none"    "none"
## [57] "none"    "none"    "black"   "brown"   "brown"   "none"    "black"
## [64] "black"   "brown"   "white"   "black"   "black"   "blonde"  "none"
## [71] "none"    "none"    "white"   "none"    "none"    "none"    "none"
## [78] "none"    "brown"   "brown"   "none"    "none"    "black"   "brown"
## [85] "brown"   "none"    "none"
```

```
> Starwars$hair_color <- hair_colors_first
> unique(Starwars$hair_color)
```

```
## [1] "blond"   "unknown" "none"    "brown"   "black"   "auburn"  "white"
## [8] "grey"    "blonde"
```

Uočavamo još dvije stvari: blond i blonde su sinonimi, stoga i to treba korigirati. Također, unknown i none bi također mogli biti vezani za istu situaciju te none možemo zamijeniti s unknown.

```
> # Zamjena "blond" s "blonde"
> Starwars$hair_color <- gsub("\\bblond\\b", "blonde", Starwars$hair_color)
> # \\b označava granicu riječi, što osigurava da se zamjena dogodi samo onda
> #kada se „blond" pojavljuje kao zasebna riječ, a ne kao dio druge riječi
>
> # Zamjena „none" s „unknown"
> Starwars$hair_color <- gsub("none", "unknown", Starwars$hair_color)
>
> unique(Starwars$hair_color)
```

```
## [1] "blonde"  "unknown" "brown"   "black"   "auburn"  "white"   "grey"
```

Ovime smo završili čišćenje podataka u varijabli hair_color. Daljnja analiza ovisi o kvaliteti podataka. Ako podaci nisu pravilno zapisani, potrebno je provesti „čišćenje" podataka, na način da se podaci ne ujednače na pravilan način, riješi se pitanje NA, definira prikladan tip varijable i slično.

Recimo da nas zbunjuje korištenje engleskih naziva i zbog toga želimo promijeniti nazive varijabli u podatkovnom okviru.

```
> colnames(Starwars) <-c("ime", "visina", "tezina",
+                        "boja_kose", "boja_ociju",
+                        "planet", "vrsta")
> str(Starwars)
```

```
## 'data.frame':    87 obs. of  7 variables:
##  $ ime       : chr  "Luke Skywalker" "C-3PO" "R2-D2" "Darth Vader" ...
##  $ visina    : int  172 167 96 202 150 178 165 97 183 182 ...
##  $ tezina    : num  77 75 32 136 49 120 75 32 84 77 ...
##  $ boja_kose : chr  "blonde" "unknown" "unknown" "unknown" ...
##  $ boja_ociju: chr  "blue" "yellow" "red" "yellow" ...
##  $ planet    : chr  "Tatooine" "Tatooine" "Naboo" "Tatooine" ...
##  $ vrsta     : chr  "Human" "Droid" "Droid" "Human" ...
```

Recimo da nas zanima koliko iznosi prosječna visina i težina za svaku vrstu. Prvo bismo morali provjeriti koliko ima različitih vrsta, a potom za svaku izračunati visinu i težinu. Nakon toga, mogli bismo to prikazati u tablici.

```
> unique(Starwars$vrsta)
```

```
##  [1] "Human"          "Droid"          "Wookiee"        "Rodian"
##  [5] "Hutt"           "Yoda's species" "Trandoshan"     "Mon Calamari"
##  [9] "Ewok"           "Sullustan"      "Neimodian"      "Gungan"
## [13] "Toydarian"      "Dug"            "Zabrak"         "Twi'lek"
## [17] "Aleena"         "Vulptereen"     "Xexto"          "Toong"
## [21] "Cerean"         "Nautolan"       "Tholothian"     "Iktotchi"
## [25] "Quermian"       "Kel Dor"        "Chagrian"       "Geonosian"
## [29] "Mirialan"       "Clawdite"       "Besalisk"       "Kaminoan"
## [33] "Skakoan"        "Muun"           "Togruta"        "Kaleesh"
## [37] "Umbaran"        "Pau'an"
```

Koristeći naredbu `unique()`, utvrđujemo da postoji 38 jedinstvenih naziva za vrste. No, izgledno je da će se neke vrste pojavljivati samo jedanput. Za njih nećemo računati prosječne vrijednosti. To znači da prvo moramo utvrditi frekvencije pojavljivanja vrsta. To možemo učiniti na sljedeći način.

```
> vrsta_frekv <- table(Starwars$vrsta)
> vrsta_frekv
```

```
##
##         Aleena       Besalisk         Cerean       Chagrian       Clawdite
##             1              1              1              1              1
##          Droid            Dug           Ewok      Geonosian         Gungan
##             6              1              1              1              3
##          Human           Hutt       Iktotchi        Kaleesh       Kaminoan
##             38             1              1              1              2
##        Kel Dor       Mirialan   Mon Calamari           Muun       Nautolan
##             1              2              1              1              1
##      Neimodian         Pau'an       Quermian         Rodian        Skakoan
##             1              1              1              1              1
##      Sullustan     Tholothian        Togruta          Toong      Toydarian
##             1              1              1              1              1
##     Trandoshan        Twi'lek        Umbaran     Vulptereen        Wookiee
##             1              2              1              1              2
##          Xexto Yoda's species         Zabrak
##             1              1              2
```

```
> vrsta_multiple <- which(vrsta_frekv > 1)
> vrsta_multiple_nazivi <- names(vrsta_multiple)
> vrsta_multiple_nazivi
```

```
## [1] "Droid"    "Gungan"   "Human"    "Kaminoan" "Mirialan" "Twi'lek"  "Wookiee"
## [8] "Zabrak"
```

Za svaku od ovih vrsta izračunat ćemo prosjek visine i težine temeljem podskupa podataka samo za tu vrstu. To se može učiniti na više načina, a ovdje će se prikazati dva načina.

U prvom slučaju koristimo dplyr paket i *filtriranje* skupa podataka, a potom *grupiranje* i *sažimanje*.

```
> filtered_data <- Starwars %>%
+                  filter(vrsta %in% vrsta_multiple_nazivi)
>
> prosjek_visina_tezina <- filtered_data %>%
+   group_by(vrsta) %>%
+   summarise(
+     prosjek_visina = mean(visina, na.rm = TRUE),
+     prosjek_tezina = mean(tezina, na.rm = TRUE)
+                     )
> prosjek_visina_tezina
```

```
## # A tibble: 8 x 3
##   vrsta    prosjek_visina prosjek_tezina
```

```
##   <chr>             <dbl>          <dbl>
## 1 Droid              131.           69.8
## 2 Gungan             209.           74
## 3 Human              178.           82.8
## 4 Kaminoan           221            88
## 5 Mirialan           168            53.1
## 6 Twi'lek            179            55
## 7 Wookiee            231           124
## 8 Zabrak             173            80
```

Alternativno, možemo koristiti pristup „pješke" upotrebom podskupa podataka. Na taj način izračunavamo prosjek visine za pojedinu vrstu temeljem filtracije za jednu vrstu. Isto to učinimo za težinu, a potom ponovimo postupak za preostale vrste. Na kraju objedinimo podatke u tablicu.

```
> prosjek_visine_Droid <- mean(Starwars$visina[Starwars$vrsta=="Droid"],
+                              na.rm = TRUE)
>
> # Ovdje moramo koristiti 'na.rm = TRUE' zato jer varijable
> # visine i težine sadrže nepoznanice koje nismo korigirali
> # zbog na.rm će funkcija mean() ignorirati vrijednosti koje nedostaju;
> # u suprotnom bismo dobili NA ili error
>
> prosjek_tezine_Droid <- mean(Starwars$tezina[Starwars$vrsta=="Droid"],
+                              na.rm = TRUE)
>
> prosjek_visine_Gungan <- mean(Starwars$visina[Starwars$vrsta=="Gungan"],
+                               na.rm = TRUE)
> prosjek_tezine_Gungan <- mean(Starwars$tezina[Starwars$vrsta=="Gungan"],
+                               na.rm = TRUE)
>
> prosjek_visine_Human <- mean(Starwars$visina[Starwars$vrsta=="Human"],
+                              na.rm = TRUE)
> prosjek_tezine_Human <- mean(Starwars$tezina[Starwars$vrsta=="Human"],
+                              na.rm = TRUE)
>
> prosjek_visine_Kaminoan <- mean(Starwars$visina[Starwars$vrsta=="Kaminoan"],
+                                 na.rm = TRUE)
> prosjek_tezine_Kaminoan <- mean(Starwars$tezina[Starwars$vrsta=="Kaminoan"],
+                                 na.rm = TRUE)
>
> prosjek_visine_Mirialan <- mean(Starwars$visina[Starwars$vrsta=="Mirialan"],
+                                 na.rm = TRUE)
> prosjek_tezine_Mirialan <- mean(Starwars$tezina[Starwars$vrsta=="Mirialan"],
+                                 na.rm = TRUE)
>
> prosjek_visine_Twilek <- mean(Starwars$visina[Starwars$vrsta=="Twi'lek"],
+                               na.rm = TRUE)
```

```
> prosjek_tezine_Twilek <- mean(Starwars$tezina[Starwars$vrsta=="Twi'lek"],
+                               na.rm = TRUE)
>
> prosjek_visine_Wookiee <- mean(Starwars$visina[Starwars$vrsta=="Wookiee"],
+                                na.rm = TRUE)
> prosjek_tezine_Wookiee <- mean(Starwars$tezina[Starwars$vrsta=="Wookiee"],
+                                na.rm = TRUE)
>
> prosjek_visine_Zabrak <- mean(Starwars$visina[Starwars$vrsta=="Zabrak"],
+                               na.rm = TRUE)
> prosjek_tezine_Zabrak<- mean(Starwars$tezina[Starwars$vrsta=="Zabrak"],
+                              na.rm = TRUE)
>
> # Kreiramo tablicu prikaza
>
> visina_tezina_vrsta <-matrix(c("Droid",
+                                prosjek_visine_Droid, prosjek_tezine_Droid,
+                                "Gungan",
+                                prosjek_visine_Gungan, prosjek_tezine_Gungan,
+                                "Human",
+                                prosjek_visine_Human, prosjek_tezine_Human,
+                                "Kaminoan",
+                                prosjek_visine_Kaminoan, prosjek_tezine_Kaminoan,
+                                "Mirialan",
+                                prosjek_visine_Mirialan, prosjek_tezine_Mirialan,
+                                "Twi'lek",
+                                prosjek_visine_Twilek, prosjek_tezine_Twilek,
+                                "Wookiee",
+                                prosjek_visine_Wookiee, prosjek_tezine_Wookiee,
+                                "Zabrak",
+                                prosjek_visine_Zabrak, prosjek_tezine_Zabrak),
+                              nrow = 8, byrow = TRUE)
>
> colnames(visina_tezina_vrsta) <- c("Vrsta", "Prosječna visina",
+                                    "Prosječna težina")
>
> visina_tezina_vrsta<-as.table(visina_tezina_vrsta)
> visina_tezina_vrsta
```

```
##   Vrsta    Prosječna visina Prosječna težina
## A Droid    131.2            69.75
## B Gungan   208.666666666667 74
## C Human    177.636363636364 82.7818181818182
## D Kaminoan 221              88
## E Mirialan 168              53.1
## F Twi'lek  179              55
## G Wookiee  231              124
```

```
## H Zabrak   173              80
```

Rezultat je isti, iako je prvi pristup puno brži. No, ako preferirate pratiti izračune „korak po korak", drugi pristup će vam biti prihvatljiviji unatoč većem broju linija koda.

Recimo da bismo htjeli vizualno prikazati distribucije visina po vrstama i pretpostavljamo da su visine normalno distribuirane varijable za svaku od vrsta. U tom slučaju, potrebna nam je standardna devijacija za svaku vrstu.

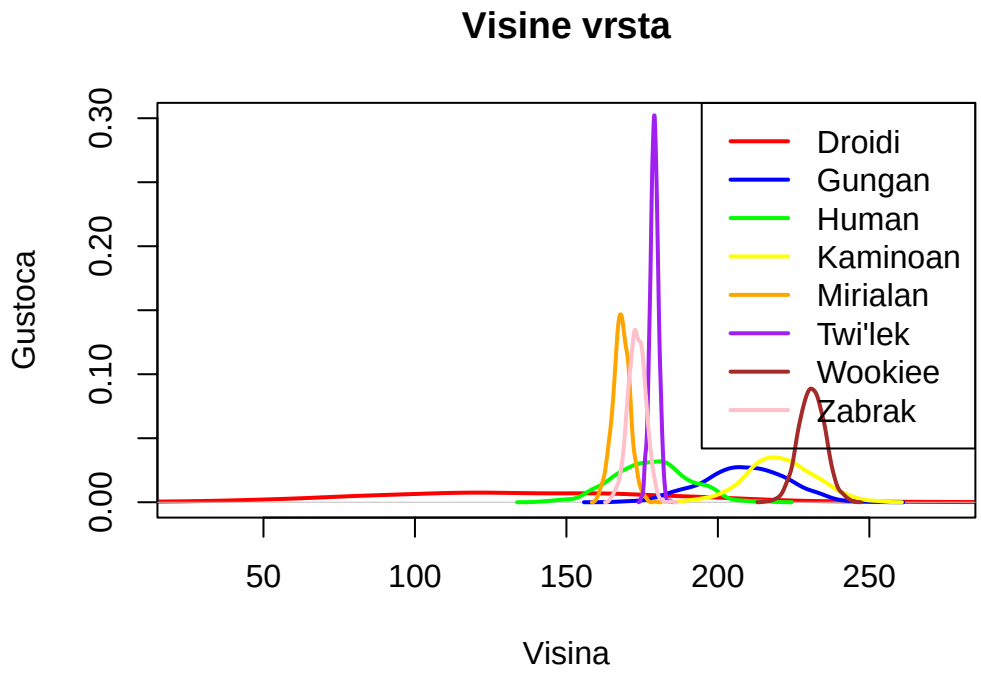
Treba napomenuti da ovo činimo zbog primjera za vježbu i grafičkog prikaza te u inferencijalnoj statistici nećete običavati koristiti podatke ovako malih uzoraka da biste pretpostavljali distribuciju bilo koje karakteristike čitave vrste ili bilo koje veće populacije.

```
> prosjek_sd_visina <- filtered_data %>%
+   group_by(vrsta) %>%
+   summarise(
+     prosjek_visina = mean(visina, na.rm = TRUE),
+     sd_visina = sd(visina, na.rm = TRUE)
+                       )
> prosjek_sd_visina
```

```
## # A tibble: 8 x 3
##   vrsta    prosjek_visina sd_visina
##   <chr>             <dbl>     <dbl>
## 1 Droid              131.     49.1
## 2 Gungan             209.     14.2
## 3 Human              178.     12.1
## 4 Kaminoan           221      11.3
## 5 Mirialan           168       2.83
## 6 Twi'lek            179       1.41
## 7 Wookiee            231       4.24
## 8 Zabrak             173       2.83
```

```
> Droidi <- density(rnorm(1000,prosjek_sd_visina$prosjek_visina[1],
+                         prosjek_sd_visina$sd_visina[1]))
> Gungan <- density(rnorm(1000,prosjek_sd_visina$prosjek_visina[2],
+                         prosjek_sd_visina$sd_visina[2]))
> Human <- density(rnorm(1000,prosjek_sd_visina$prosjek_visina[3],
+                        prosjek_sd_visina$sd_visina[3]))
> Kaminoan <- density(rnorm(1000,prosjek_sd_visina$prosjek_visina[4],
+                           prosjek_sd_visina$sd_visina[4]))
> Mirialan  <- density(rnorm(1000,prosjek_sd_visina$prosjek_visina[5],
+                            prosjek_sd_visina$sd_visina[5]))
> Twilek  <- density(rnorm(1000,prosjek_sd_visina$prosjek_visina[6],
+                          prosjek_sd_visina$sd_visina[6]))
> Wookiee  <- density(rnorm(1000,prosjek_sd_visina$prosjek_visina[7],
+                           prosjek_sd_visina$sd_visina[7]))
> Zabrak  <- density(rnorm(1000,prosjek_sd_visina$prosjek_visina[8],
```

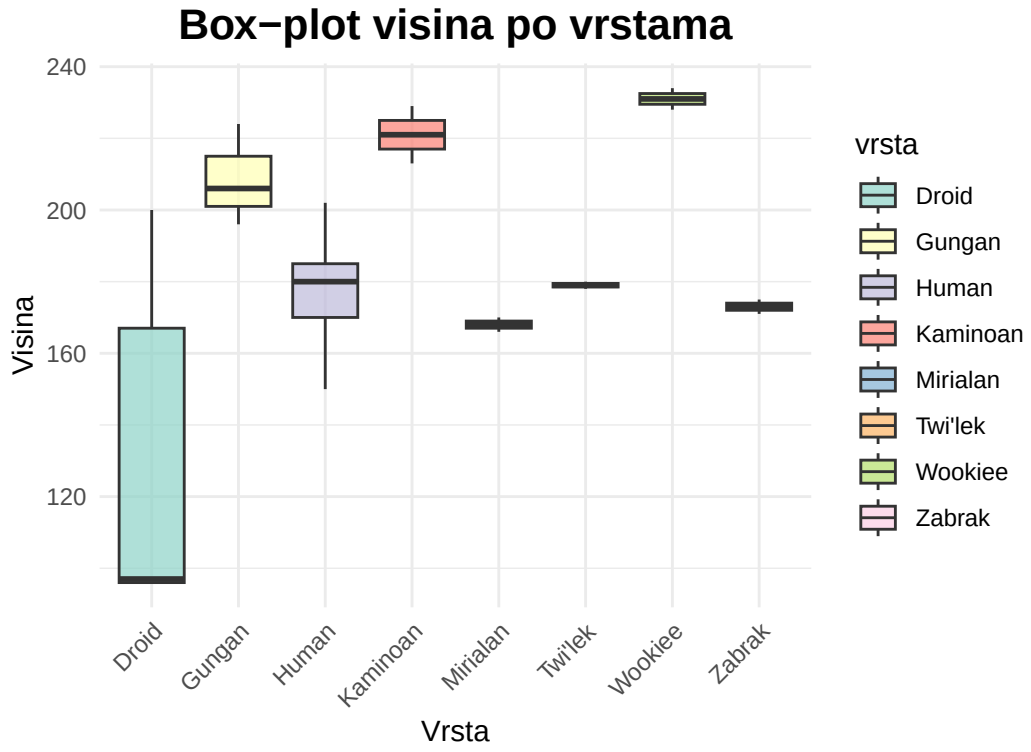
```
+                          prosjek_sd_visina$sd_visina[8]))
>
> plot(Droidi, main="Visine vrsta", xlab="Visina", ylab="Gustoća",
+      col="red", lwd=2, xlim=c(25,275), ylim=c(0,0.3))
> lines(Gungan, col="blue", lwd=2)
> lines(Human, col="green", lwd=2)
> lines(Kaminoan, col="yellow", lwd=2)
> lines(Mirialan, col="orange", lwd=2)
> lines(Twilek, col="purple", lwd=2)
> lines(Wookiee, col="brown", lwd=2)
> lines(Zabrak, col="pink", lwd=2)
> legend("topright", legend=c("Droidi", "Gungan", "Human", "Kaminoan",
+                             "Mirialan", "Twi'lek", "Wookiee", "Zabrak"),
+        col=c("red", "blue", "green", "yellow",
+              "orange", "purple", "brown", "pink"), lwd=2)
```



Alternativno, možda želimo dobiti uvid u distribuciju podataka o opaženim visinama (a ne generiranim vrijednostima) koristeći box-plot. Ako kreiramo usporedne box-plotove za sve vrste iz podskupa *filtered_data*, moći ćemo ih međusobno usporediti.

```
> ggplot(filtered_data, aes(x = vrsta, y = visina, fill = vrsta)) +
+   geom_boxplot(alpha = 0.7) +
+   labs(
+     title = "Box-plot visina po vrstama",
+     x = "Vrsta",
```

```
+     y = "Visina"
+   ) +
+   theme_minimal() +
+   theme(
+     plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
+     axis.text.x = element_text(angle = 45, hjust = 1)
+   ) +
+   scale_fill_brewer(palette = "Set3")
```



Box-plot prikazuje značajne razlike u visinama među vrstama. Kaminoan i Wookiee ističu se kao najviše vrste, s visokim vrijednostima medijana, dok Droidi i Mirialan imaju najniže medijane. Raspon visina je najširi kod Droida, što sugerira veliku varijabilnost unutar te vrste, dok vrste poput Wookiee i Zabrak imaju uže raspone, što znači da su visine unutar tih vrsta konzistentnije. Human i Gungan imaju slične raspone varijacija.

Neke vrste, poput Kaminoan, nemaju izdvojenice, dok druge vrste pokazuju visine koje su izvan tipičnih vrijednosti. Ova usporedba omogućuje bolji uvid u razlike među vrstama, naglašavajući kako se vrste razlikuju ne samo po prosječnoj visini već i po varijabilnosti i ekstremnim vrijednostima unutar svojih distribucija.

No, za pokazatelje deskriptivne statistike, možemo, na primjer, koristiti i paket *summarytools*. S obzirom da želimo izračunati pokazatelje za svaku vrstu zasebno, kombinirat ćemo s naredbom `stby()` iz paketa *dplyr*. Prva naredba, kojom bismo utvrdili sve vrste, neće se pokrenuti, jer je ispis podugačak. No, možete isprobati sami. Drugom skupinom naredbi kreira se podskup podataka koji obuhvaća dvije vrsta (Droide i ljude), a potom se izračunavaju pokazatelji deskriptivne statistike za svaku vrstu likova.

```
> # stby(data = Starwars, INDICES = Starwars$vrsta,
> #      FUN = function(x) descr(Starwars[,2:3], stats = "all"))
```

```
> starwars_subset <- Starwars %>%
+   filter(Starwars$vrsta %in% c("Human", "Droid"))
>
> stby(data = starwars_subset[, c("tezina", "visina")],
+                INDICES = starwars_subset$vrsta,
+                FUN = descr,
+                stats = "all")
```

```
## Descriptive Statistics
## starwars_subset
## Group: vrsta = Droid
## N: 6
##
##                     tezina   visina
## ----------------- -------- --------
##              Mean    69.75   131.20
##           Std.Dev    51.03    49.15
##               Min    32.00    96.00
##                Q1    32.00    96.00
##            Median    53.50    97.00
##                Q3   107.50   167.00
##               Max   140.00   200.00
##               MAD    31.88     1.48
##               IQR    59.25    71.00
##                CV     0.73     0.37
##          Skewness     0.45     0.41
##       SE.Skewness     1.01     0.91
##          Kurtosis    -1.95    -2.02
##           N.Valid     4.00     5.00
##         Pct.Valid    66.67    83.33
##
## Group: vrsta = Human
## N: 38
##
##                     tezina   visina
## ----------------- -------- --------
##              Mean    82.78   177.64
##           Std.Dev    19.38    12.12
##               Min    45.00   150.00
##                Q1    77.00   170.00
##            Median    79.00   180.00
##                Q3    84.00   185.00
##               Max   136.00   202.00
##               MAD     5.93    11.86
```

```
##                IQR     7.00    15.00
##                 CV     0.23     0.07
##           Skewness     0.80    -0.54
##        SE.Skewness     0.49     0.41
##           Kurtosis     1.46    -0.17
##            N.Valid    22.00    33.00
##          Pct.Valid    57.89    86.84
```

Droidi su prosječno teški 69.75 kg, uz standardno odstupanje težina od prosjeka za 51.03 kg ukazuje na relativno veliku varijabilnost. U malim uzorcima uobičajeno se opaža veća varijabilnost u podacima. Raspon težina, od 32 kg do 140 kg, dodatno potvrđuje postojanje ekstremnih vrijednosti. Polovica Droida u uzorku teška je 53 kg ili manje od toga, dok je preostala polovica teška toliko ili više. Središnjih 50% podataka o visini nalazi se između 55 i 85 kg. Distribucija težine Droida blago je pozitivno asimetrična, dok je distribucija vrhom zaobljenija od normalne.

Droidi su prosječno visoki 131.20 cm, uz standardno odstupanje visine od prosjeka za 49.15 cm. Raspon varijacija visina kreće se od 96 do 200 cm. Središnjih 50% Droida visoko je između 96 i 167 cm, ukazujući na izdvojenice s desne strane distribucije. To potvrđuje i pokazatelj asimetrije, ukazujući na to da je distribucija visina Droida blago pozitivno asimetrična. Slično kao i distribucija težina, distribucija visina Droida je vrhom zaobljenija od normalne.

Ljudi u uzorku su prosječno teški 82.78 kg, uz standardno odstupanje težine od prosjeka za 19.38 kg, što ukazuje na umjerenu varijabilnost težina među ljudima. Polovica ljudi u uzorku ima težinu od 79 kg ili manje, dok druga polovica ima težinu veću od toga. Prosjek je veći od medijana, što upućuje na postojanje nekolicine velikih vrijednosti u nizu, koje odvlače prosjek k većim vrijednostima. Raspon težina kreće se od minimalnih 45 kg do maksimalnih 136 kg, dok se središnjih 50% podataka o težini nalazi u intervalu između 77 kg i 84 kg. Distribucija težine je blago pozitivno asimetrična, što potvrđuje da postoji nekoliko ljudi s većim težinama koje odstupaju od većine opažanja. Kurtosis, ili zaobljenost distribucije, ukazuje na izduženiji vrh distribucije u odnosu na normalnu i veću koncentraciju opažanja oko prosjeka.

Prosječna visina ljudi u uzorku iznosi 177.64 cm, uz standardno odstupanje od 12.12 cm, što ukazuje na nisku varijabilnost visina. Raspon visina kreće se od minimalnih 150 cm do maksimalnih 202 cm. Središnjih 50% ljudi visoko je između 170 cm i 185 cm. Distribucija visina je blago negativno asimetrična, što znači da postoji nekoliko ljudi s nižim visinama koje povlače distribuciju ulijevo. Kurtosis ukazuje na distribuciju koja je oblikom blizu normalnoj distribuciji, s blago zaobljenijim vrhom.

No, može nas zanimati i postoji li veza između visina i težina likova Starwarsa.

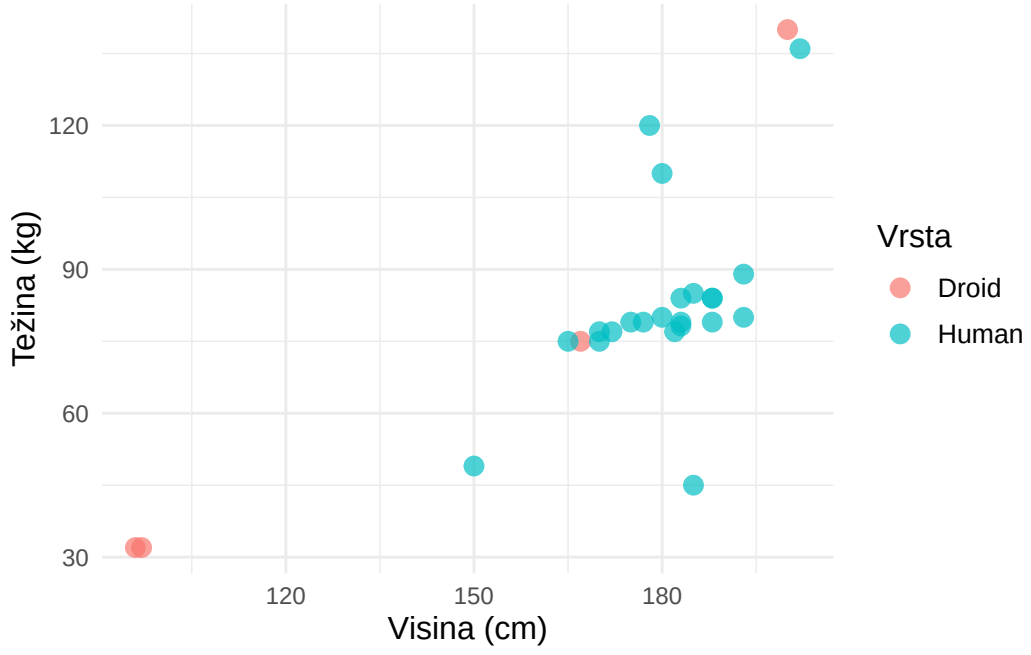
```
> starwars_clean <- Starwars %>%
+   select(vrsta, tezina, visina) %>%
+   filter(!is.na(visina) & !is.na(tezina))
>
> cor(starwars_clean[2:3], use = "pairwise.complete.obs")
```

```
##        tezina visina
## tezina   1.00   0.13
## visina   0.13   1.00
```

Vrijednost korelacije 0.13 sugerira da postoji vrlo slab pozitivan odnos između težine i visine. To znači da, u prosjeku, s povećanjem visine, težina blago raste. Ipak, radi se o vrlo slaboj povezanosti, koja ne ukazuje na izraženu pravilnost.

Budući da skup podataka sadrži različite vrste likova, moguće je da razlike između skupina značajno smanjuju ukupnu korelaciju. Na primjer, Droidi mogu imati različite obrasce povezanosti težine i visine u usporedbi s ljudima. Bilo bi korisno izračunati korelacije unutar specifičnih skupina (npr. za ljude i Droide) kako bismo vidjeli postoji li jača povezanost u homogenijim podskupinama.

```
> library(ggplot2)
>
> ggplot(data = starwars_subset, aes(x = visina, y = tezina, color = vrsta)) +
+   geom_point(size = 3, alpha = 0.7) +
+   labs(
+     x = "Visina (cm)",
+     y = "Težina (kg)",
+     color = "Vrsta"
+   ) +
+   theme_minimal() +
+   theme(
+     plot.title = element_text(hjust = 0.5, size = 14),
+     axis.title = element_text(size = 12),
+     legend.title = element_text(size = 12),
+     legend.text = element_text(size = 10)
+   )
```




```
> cor_droid <- starwars_subset %>%
+   filter(vrsta == "Droid") %>%
```

```
+   select(tezina, visina) %>%
+   cor(use = "pairwise.complete.obs")
> cor_droid
```

```
##        tezina visina
## tezina   1.00   0.96
## visina   0.96   1.00
```

Koeficijent korelacije između težine i visine Droida iznosi 0.96, što ukazuje na vrlo snažnu pozitivnu linearnu vezu. To znači da, kako se povećava visina Droida, težina također gotovo proporcionalno raste. Ovaj visok koeficijent može biti posljedica malog uzorka Droida i homogennosti njihovih karakteristika u skupu podataka, ali je vjerojatnije da je težina rezultat njihove visine i materijala koji su korišteni u njihovoj konstrukciji (veći Droidi zahtijevaju više materijala, što povećava njihovu težinu). S obzirom da Droidi nemaju biološke varijacije poput ljudi (npr., razlike u masi mišića, gustoći kostiju ili postotku masti), veza između težine i visine je jača.

```
> cor_human <- starwars_subset %>%
+   filter(vrsta == "Human") %>%
+   select(tezina, visina) %>%
+   cor(use = "pairwise.complete.obs")
> cor_human
```

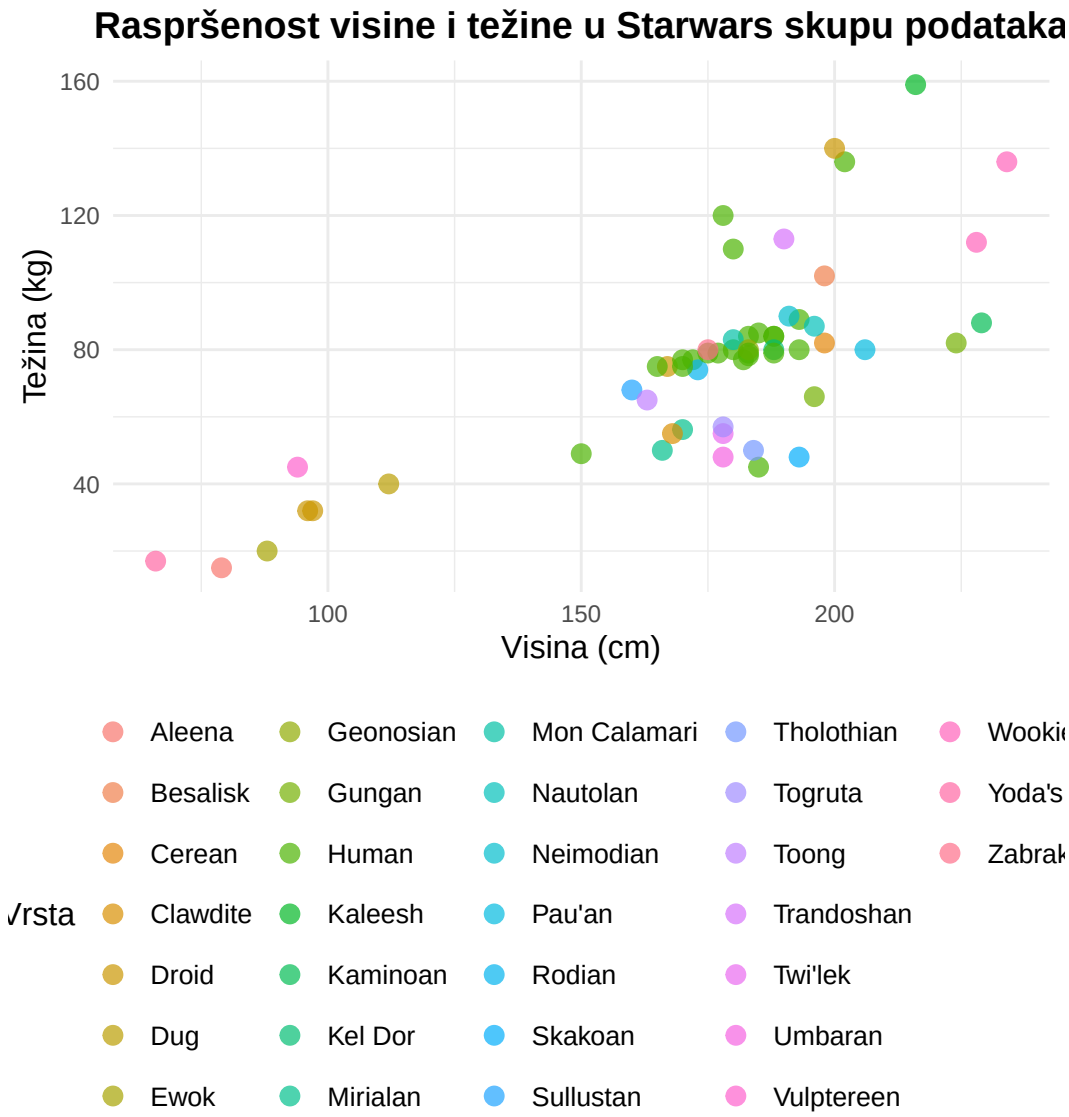
```
##        tezina visina
## tezina   1.00   0.51
## visina   0.51   1.00
```

Koeficijent krelacije između težine i visine iznosi 0.51, što ukazuje na umjerenu pozitivnu linearnu vezu. Ovo je očekivano jer su kod ljudi visina i težina povezane, ali nisu savršeno proporcionalne (npr., razlike u građi tijela, masi mišića, itd.). Dakle umjerena pozitivna korelacija je u skladu s raznolikošću podataka o ljudima u uzorku.

Također, sličnu analizu možemo provesti i za sve vrste, no ovdje ćemo samo grafički prikazati uvide koje možemo dobiti dijagramom rasipanja.

```
> library(ggplot2)
> library(dplyr)
>
> starwars_clean <- Starwars %>%
+   # odabiremo opažanja koja ne sadrže NA
+   # te eliminiramo jednu izdvojenicu
+   filter(!is.na(visina) & !is.na(tezina) & tezina <= 1000) %>%
+   select(vrsta, visina, tezina)
>
> ggplot(data = starwars_clean, aes(x = visina, y = tezina, color = vrsta)) +
+   geom_point(size = 3, alpha = 0.7) +
+   labs(
```

```
+     title = "Raspršenost visine i težine u Starwars skupu podataka",
+     x = "Visina (cm)",
+     y = "Težina (kg)",
+     color = "Vrsta"
+   ) +
+   theme_minimal() +
+   theme(
+     plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
+     axis.title = element_text(size = 12),
+     legend.title = element_text(size = 12),
+     legend.text = element_text(size = 10),
+     legend.position = "bottom"
+   )
```



Ako želite, možete samostalno provesti daljnju analizu visina i težina za svaku vrstu, po uzoru na

postupak proveden za Droide i ljude.

Sljedeće nas zanima koji likovi iz određenih vrsta imaju visinu i težinu iznad dvije standardne devijacije od prosjeka za svoju vrstu? Odgovor na to pitanje iziskuje nekoliko koraka u analizi. Prvo je potrebno izračunati prosječnu visinu i težinu za svaku vrstu u skupu podataka. Nakon toga trebamo filtrirati likove koji su iznad prosječne visine i težine za svoju vrstu. I naravno, na kraju treba ispisati ili prikazati rezultate. Ovo je izvrstan primjer na kojem se može prikazati kreiranje vlastitih funkcija u kontekstu analize podataka.

```
> library(dplyr)
>
> # Funkcija za pronalazak likova iznad prosječne visine i težine
> iznadprosjecni_likovi <- function(data, vrsta, visina, tezina) {
+
+   # 1. Grupiranje podataka po vrsti i računanje prosjeka
+   avg_stats <- data %>%
+     group_by(!!sym(vrsta)) %>%
+     summarize(
+       avg_visina = mean(!!sym(visina), na.rm = TRUE),
+       sd_visina = sd(!!sym(visina), na.rm = TRUE),
+       avg_tezina = mean(!!sym(tezina), na.rm = TRUE),
+       sd_tezina = sd(!!sym(tezina), na.rm = TRUE)
+     )
+
+   # 2. Spajanje originalnih podataka s prosjecima
+   data_with_avg <- data %>%
+     left_join(avg_stats, by = vrsta)
+
+   # 3. Filtriranje likova čija visina i težina odstupaju
+   #    više od 2 standardne devijacije iznad prosjeka
+   above_2sd <- data_with_avg %>%
+     filter(
+       !!sym(visina) > avg_visina + 2 * sd_visina &
+       !!sym(tezina) > avg_tezina + 2 * sd_tezina
+     ) %>%
+     select(ime, !!sym(vrsta), !!sym(visina), !!sym(tezina))
+
+   return(above_2sd)
+ }
>
> # Primjena funkcije
> likovi <- iznadprosjecni_likovi(
+   data = Starwars,
+   vrsta = "vrsta",
+   visina = "visina",
+   tezina = "tezina"
+ )
>
```

```
> # Prikaz rezultata
> likovi
```

```
##           ime vrsta visina tezina
## 1 Darth Vader Human    202    136
```

Recimo da se sad želimo usredotočiti na raznovrsnost imena i želimo ih prvo poredati po abecedi. Jedan od načina kako možemo dobiti takav prikaz i čitav skup podataka poredati prema imenima likova je koristeći funkciju `arrange()`.

```
> sorted_Starwars <- arrange(Starwars, ime)
> # kad bismo htjeli poredati silazno po imenima Z->A,
> # onda bi umjesto imena upisali desc(ime)
> str(sorted_Starwars)
```

```
## 'data.frame':    87 obs. of  7 variables:
##  $ ime       : chr  "Ackbar" "Adi Gallia" "Anakin Skywalker" "Arvel Crynyd" ...
##  $ visina    : int  180 184 188 NA 178 NA 191 166 163 165 ...
##  $ tezina    : num  83 50 84 NA 55 NA NA 50 65 75 ...
##  $ boja_kose : chr  "unknown" "unknown" "blonde" "brown" ...
##  $ boja_ociju: chr  "orange" "blue" "blue" "brown" ...
##  $ planet    : chr  "Mon Cala" "Coruscant" "Tatooine" NA ...
##  $ vrsta     : chr  "Mon Calamari" "Tholothian" "Human" "Human" ...
```

Ako nas zanima koliko je jedinstvenih imena, odnosno likova, obuhvaćeno ovim skupom podataka, možemo koristiti funkciju `unique()`.

```
> unique(sorted_Starwars$ime)
```

```
##  [1] "Ackbar"                "Adi Gallia"            "Anakin Skywalker"
##  [4] "Arvel Crynyd"          "Ayla Secura"           "BB8"
##  [7] "Bail Prestor Organa"   "Barriss Offee"         "Ben Quadinaros"
## [10] "Beru Whitesun Lars"    "Bib Fortuna"           "Biggs Darklighter"
## [13] "Boba Fett"             "Bossk"                 "C-3PO"
## [16] "Captain Phasma"        "Chewbacca"             "Cliegg Lars"
## [19] "Cord\xe9"              "Darth Maul"            "Darth Vader"
## [22] "Dexter Jettster"       "Dooku"                 "Dorm\xe9"
## [25] "Dud Bolt"              "Eeth Koth"             "Finis Valorum"
## [28] "Finn"                  "Gasgano"               "Greedo"
## [31] "Gregar Typho"          "Grievous"              "Han Solo"
## [34] "IG-88"                 "Jabba Desilijic Tiure" "Jango Fett"
## [37] "Jar Jar Binks"         "Jek Tono Porkins"      "Jocasta Nu"
## [40] "Ki-Adi-Mundi"          "Kit Fisto"             "Lama Su"
## [43] "Lando Calrissian"      "Leia Organa"           "Lobot"
## [46] "Luke Skywalker"        "Luminara Unduli"       "Mace Windu"
```

```
## [49] "Mas Amedda"            "Mon Mothma"            "Nien Nunb"
## [52] "Nute Gunray"           "Obi-Wan Kenobi"        "Owen Lars"
## [55] "Padm\xe9 Amidala"      "Palpatine"             "Plo Koon"
## [58] "Poe Dameron"           "Poggle the Lesser"     "Quarsh Panaka"
## [61] "Qui-Gon Jinn"          "R2-D2"                 "R4-P17"
## [64] "R5-D4"                 "Ratts Tyerel"          "Raymus Antilles"
## [67] "Rey"                   "Ric Oli\xe9"           "Roos Tarpals"
## [70] "Rugor Nass"            "Saesee Tiin"           "San Hill"
## [73] "Sebulba"               "Shaak Ti"              "Shmi Skywalker"
## [76] "Sly Moore"             "Tarfful"               "Taun We"
## [79] "Tion Medon"            "Wat Tambor"            "Watto"
## [82] "Wedge Antilles"        "Wicket Systri Warrick" "Wilhuff Tarkin"
## [85] "Yarael Poof"           "Yoda"                  "Zam Wesell"
```

Recimo da nas u sljedećem koraku zanimaju boje kose prema vrsti likova te apsolutne frekvencije u kojima se pojedine kombinacije pojavljuju. Za to nam je potrebna tablica kontingence.

```
> contingency_table <- table(Starwars$boja_kose, Starwars$vrsta)
>
> # Prikazivanje tablice
> contingency_table
```

```
##
##           Aleena Besalisk Cerean Chagrian Clawdite Droid Dug Ewok Geonosian
##   auburn       0        0      0        0        0     0   0    0         0
##   black        0        0      0        0        0     0   0    0         0
##   blonde       0        0      0        0        1     0   0    0         0
##   brown        0        0      0        0        0     0   0    1         0
##   grey         0        0      0        0        0     0   0    0         0
##   unknown      1        1      0        1        0     6   1    0         1
##   white        0        0      1        0        0     0   0    0         0
##
##           Gungan Human Hutt Iktotchi Kaleesh Kaminoan Kel Dor Mirialan
##   auburn       0     3    0        0       0        0       0        0
##   black        0     9    0        0       0        0       0        2
##   blonde       0     3    0        0       0        0       0        0
##   brown        0    16    0        0       0        0       0        0
##   grey         0     1    0        0       0        0       0        0
##   unknown      3     4    1        1       1        2       1        0
##   white        0     2    0        0       0        0       0        0
##
##           Mon Calamari Muun Nautolan Neimodian Pau'an Quermian Rodian Skakoan
##   auburn             0    0        0         0      0        0      0       0
##   black              0    0        0         0      0        0      0       0
##   blonde             0    0        0         0      0        0      0       0
##   brown              0    0        0         0      0        0      0       0
##   grey               0    0        0         0      0        0      0       0
```

```
##   unknown            1    1        1         1      1        1      1       1
##   white              0    0        0         0      0        0      0       0
##
##           Sullustan Tholothian Togruta Toong Toydarian Trandoshan Twi'lek
##   auburn          0          0       0     0         0          0       0
##   black           0          0       0     0         1          0       0
##   blonde          0          0       0     0         0          0       0
##   brown           0          0       0     0         0          0       0
##   grey            0          0       0     0         0          0       0
##   unknown         1          1       1     1         0          1       2
##   white           0          0       0     0         0          0       0
##
##           Umbaran Vulptereen Wookiee Xexto Yoda's species Zabrak
##   auburn        0          0       0     0              0      0
##   black         0          0       0     0              0      1
##   blonde        0          0       0     0              0      0
##   brown         0          0       2     0              0      0
##   grey          0          0       0     0              0      0
##   unknown       1          1       0     1              0      1
##   white         0          0       0     0              1      0
```

S obzirom da postoji puno jedinstvenih kombinacija i jako puno kombinacija koje se ne pojavljuju, možemo isto ispitati za podskup vrsta za koje smo ranije utvrdili da su učestalije i spremili u df filtered_data.

```
> contingency_table <- table(filtered_data$boja_kose, filtered_data$vrsta)
>
> # Prikazivanje tablice
> contingency_table
```

```
##
##           Droid Gungan Human Kaminoan Mirialan Twi'lek Wookiee Zabrak
##   auburn      0      0     3        0        0       0       0      0
##   black       0      0     9        0        2       0       0      1
##   blonde      0      0     3        0        0       0       0      0
##   brown       0      0    16        0        0       0       2      0
##   grey        0      0     1        0        0       0       0      0
##   unknown     6      3     4        2        0       2       0      1
##   white       0      0     2        0        0       0       0      0
```

Svi Droidi imaju nepoznatu boju kose, odnosno nemaju kose. Slična je situacija i sa svim Gunganima, Kaminoanima, Twi'lekima i jednim/om Zabrakom. Kosa ljudi se pojavljuje u najviše boja. Temeljem ovog uzorka, moglo bi se zaključiti da Mirilalani imaju crnu kosu, Wookiee imaju smeđu kosu, a Zabraci imaju crnu ili nemaju kose.

Što bi nas moglo zanimati dalje? Moglo bi nas zanimati postoji li statistički značajna razlika u težinama s obzirom na vrstu. Mogli bismo htjeti kreirati dodatne grafičke prikaze… Uz dovoljno podataka i predznanja te malo znatiželje, dostupni skupovi podataka u R-u mogu postati nepresušna oaza za vježbanje i kalibriranje vještina upravljanja podacima.

Pitanja za ponavljanje

1. Koje funkcije možete koristiti za pregled strukture skupa podataka i uvida u varijable s nedostajućim vrijednostima?

2. Kako biste identificirali sve pozicije s vrijednostima koje nedostaju za određenu varijablu, npr. *species*?

3. Na koji način možemo ručno zamijeniti specifične NA vrijednosti u varijabli species u skupu podataka *Starwars*?

4. Objasnite kako koristiti funkciju `tolower()` za ujednačavanje teksta u varijabli *hair_color*. Koji je problem riješen ovom funkcijom?

5. Kako bi izgledao kod za zamjenu riječi „blond“ s „blonde“ u varijabli *hair_color*? Koji je dodatni simbol korišten kako bi se zamjena izvršila samo za tu riječ?

6. Na koji način biste prikazali različite vrste unutar skupa podataka *Starwars* i koliko puta se svaka od njih pojavljuje?

7. Koristeći *dplyr*, kako biste dobili prosječne vrijednosti visine i težine po vrsti u skupu podataka?

8. Na koji način biste kreirali vizualni prikaz distribucija visina za svaku vrstu? Koje pretpostavke su ovdje korištene?

9. Kako biste poredali podatke u *Starwars* prema imenima likova u uzlaznom redoslijedu koristeći *dplyr*?

10. Na koji način biste ispitali odnos između boje kose i vrste likova te prikazali apsolutne frekvencije za kombinacije pomoću tablice kontingencije?

Zadaci

Otvorite novi R Script dokument.

1. Kreirajte vektor *z* koji će sadržavati sve parne brojeve između 50 i 100.
2. Provjerite tip *z*-a.
3. Izračunajte prosjek vrijednosti varijable *z*.
4. Kreirajte podatkovni okvir prema podacima iz tablice na 32. stranici u *Statističkim informacijama*
5. Pretvorite podatkovni okvir iz prošlog zadatka u tablicu.
6. Kreirajte vektor temeljem podataka iz tablice na 33. stranici *Statističkih informacija*, na način da koristite samo opažanja iz 2020. godine za minimalno šest varijabli.
7. Kreirajte vektor temeljem podataka iz tablice na 33. stranici *Statističkih informacija*, na način da koristite samo opažanja iz 2019. godine za iste varijable koje ste odabrali u prethodnom zadatku.
8. Vektorima iz 6. i 7. zadatka pridružite nazive.
9. Spojite vektore iz 6. i 7. zadatka.
10. Vektor iz prethodnog zadatka pretvorite u matricu.
11. Matrici iz prethodnog zadatka pridodajte opažanja iz 2018., za iste varijable, iz istog izvora podataka.
12. Matricu pretvorite u podatkovni okvir.
13. Iz podatkovnog okvira obrišite jedan podatak iz 2018. godine.
14. Iz podatkovnog okvira obrišite jednu, proizvoljno odabranu, varijablu.
15. Izračunajte nekoliko statističkih pokazatelja za preostale varijable.
16. Je li moguće spojiti dva podatkovna okvira koja ste kreirali? Obrazložite odgovor.
17. Ako vas zanima udio nezaposlenih žena prema razinama obrazovanja, na koji način možete sortirati/filtrirati podatkovni okvir?
18. Na stranici 36. *Statističkih informacija* pogledajte dio podataka pod nazivom *Upisani studenti*. Na koje je sve načine moguće podatke zapisati u R-u? O kakvim se podacima radi? Što su opažanja, a što su varijable?
19. Na stranici 53 istog dokumenta, nalazi se tablica „BRUTO DOMAĆI PROIZVOD REPUBLIKE HRVATSKE". Razmislite o kakvim se podacima radi, što su varijable, a što opažanja i na koji bi način bilo potrebno upisati podatke u R da se kreira podatkovni okvir.
20. Ako niste ranije, bilo bi dobro sad proći sve primjere naredbi u ovom dokumentu. Osobitu pozornost posvetite onim funkcijama koje su vam nejasne i ne zaboravite koristiti prozor Help. Ako i nakon toga imate poteškoće, javite se za konzultacije ili pitajte na početku sljedećeg sata.

Otvorite novi R Script dokument.

1. Iz paketa *nycflights13* uvezite podatke *flights*.
2. Izdvojite sve letove koji su poletjeli iz zračne luke JFK i imali kašnjenje pri polasku veće od 60 minuta.
3. Sortirajte letove prema duljini leta (*air_time*) u silaznom redoslijedu.
4. Dodajte novu varijablu koja prikazuje razliku između planiranog i stvarnog vremena dolaska (*arr_delay - dep_delay*).
5. Kreirajte novi podatkovni okvir koji uključuje samo varijable *origin*, *dest*, *air_time*, i *distance*.

6. Koliko je ukupno letova bilo iz svake zračne luke (*origin*)?
7. Koji je bio najdulji let (*air_time*) iz zračne luke EWR?
8. Kolika je prosječna udaljenost (*distance*) letova s obzirom na polazište (*origin*)?
9. Kreirajte histogram koji prikazuje distribuciju kašnjenja pri polasku (*dep_delay*).
10. Izračunajte pokazatelje deskriptivne statistike za varijable *air_time* i *distance*.

Otvorite novi R Script dokument.

1. Učitajte podatke dostupne na: "http://sites.williams.edu/rdeveaux/files/2014/09/Saratoga.txt"
2. Jesu li cijene kuća uz obalu (`Waterfront == 1`) više u usporedbi s onima koje nisu uz obalu? Odgovor potkrijepite grafički.
3. Kako se cijene nekretnina mijenjaju s obzirom na broj soba (*Rooms*)?
4. Jesu li cijene novoizgrađene nekretnine (`New.Construct == 1`) više u odnosu na one koje nisu novogradnja?
5. Imaju li kuće s više kamina (*Fireplaces*) veću prosječnu cijenu? Kako se cijena razlikuje za kuće s jednim, dva ili više kamina?
6. Kako je omjer broja kupaonica (*Bathrooms*) i broja soba (*Rooms*) vezan uz cijenu kuće? Jesu li kuće s većim brojem kupaonica po sobi skuplje?
7. Izračunajte pokazatelje deskriptivne statistike za sve kvantitativne varijable.
8. Kreirajte prikladan grafički prikaz za svaku varijablu u ovom podatkovnom okviru.
9. Korištenjem grafičkog prikaza provjerite razlike u medijalnim cijenama kuća s obzirom na centralno grijanje (*Central.Air*)?
10. Kreirajte tablice frekvencija za sve kvalitativne varijable o ovom skupu podataka.

Otvorite novi R Script dokument.

1. U svibnju 2024., više od 65000 programera odgovorilo je na godišnju anketu o kodiranju, tehnologijama i alatima koje koriste i koje žele naučiti, umjetnoj inteligenciji i iskustvu programera na poslu. Podaci su dostupni za preuzimanje putem linka u .zip datoteci. Ta datoteka uključuje provedeni upitnik (u pdf formatu) te dva .csv dokumenta. Učitajte *survey_results_public.csv*.
2. Izračunajte udio vrijednosti koje nedostaju za svaku varijablu. Je li možda samo pitanje razlog izostanka odgovora?
3. Pronađite 10 najčešćih zemalja (*Country*) među ispitanicima i izračunajte njihovu relativnu učestalost.
4. Kreirajte tablicu frekvencija za starosne skupine (*Age*) i prikažite podatke grafički.
5. Kreirajte tablicu kontingence za varijable *Employment* i *LearnCode*.
6. Izdvojite jedinstvene vrijednosti za sve *OfficeStackAsync* te sve *OfficeStackSync* varijable, a nakon toga utvrdite učestalosti. Koje su najpoželjnije baze?
7. Izdvojite najčešće jezike koje ispitanici koriste (*LanguageHaveWorkedWith*) i prikažite njihove frekvencije.Koristeći tablicu kontingencije, povežite s načinom na koji su naučili kodirati (*LearnCode*)
8. Izdvojite sve varijable koje sadrže *Admired* u nazivu i utvrdite učestalosti svakog modaliteta.
9. Izdvojite i grafički prikažite najčešće načine upotrebe AI-a.
10. Zadovoljstvo poslom mjereno je varijablama koje u nazivu sadrže *JobSatPoints*. Kreirajte novu varijablu, na način da ona predstavlja niz prosječnih opažanja po recima, a nakon toga, tu novu varijablu prikažite histogramom i box-plotom.

11. Pronađite jezike koje ispitanici žele koristiti (*LanguageWantToWorkWith*) i usporedite s jezicima koje trenutno koriste.
12. Izračunajte osnovne statističke pokazatelje za godišnje plaće (*ConvertedCompYearly*), uklanjajući vrijednosti koje nedostaju.
13. Prikažite distribuciju veličina organizacija (*OrgSize*) u kojima ispitanici rade.
14. Izračunajte prosječno radno iskustvo u godinama (*WorkExp*) za ispitanike koji rade puno radno vrijeme.
15. Postoji li veza između radnog iskustva u godinama (*WorkExp*) i godišnjih plaća (*ConvertedCompYearly*) zaposlenika? Je li ta veza jača ili slabija, ako se odnos tih varijabli ispituje zasebno za ispitanike po državama?

Važno je ne zaboraviti - na kraju rada **pohraniti** projekt/ skriptu…

13 Prilagodba R-a

U R-u, globalne opcije (`options()`), koriste se za postavljanje ili ispitivanje globalnih opcija koje utječu na ponašanje cijelog R okruženja. Ove opcije mogu biti vrlo korisne za prilagodbu radnih procesa, ponašanja grafičkih prikaza, izlaza i drugih aspekata R sesije. Evo nekoliko često korištenih globalnih opcija u R-u:

- `options(scipen=999)`: Ova opcija kontrolira kako se prikazuje znanstvena notacija. Veća vrijednost za scipen favorizira prikazivanje brojeva u obliku fiksnog broja umjesto u znanstvenoj notaciji.
- `options(digits=3)`: Postavlja broj značajnih znamenki koji se prikazuju u izlazima. Na primjer, options(digits=7) prikazuje do 7 značajnih znamenki (treba voditi računa da se radi o broju znamenki, a ne decimalnih mjesta).
- `options(stringsAsFactors=FALSE)`: Ova opcija određuje treba li automatski pretvoriti znakovne vektore u faktore prilikom kreiranja podatkovnih okvira. Postavljanje na FALSE sprečava automatsku pretvorbu.
- `options(warn=)`: Kontrolira kako R upravlja upozorenjima. Na primjer, options(warn=-1) će potisnuti sva upozorenja, dok options(warn=2) pretvara upozorenja u greške.
- `options(error=)`: Omogućuje korisnicima da postave vlastitu funkciju za upravljanje greškama.
- `options(width=)`: Postavlja širinu ispisa u konzoli. Korisno za upravljanje širinom ispisa kada radite s velikim skupovima podataka.
- `options(prompt="R>," continue=" ")`: Ove opcije omogućuju prilagodbu prompta u R konzoli.
- `options(expressions=)`: Postavlja maksimalan broj izraza koji se mogu ugnijezditi. Povećanje ove vrijednosti može biti korisno u situacijama kada radite s vrlo složenim ili duboko ugniježđenim izrazima.
- `options(viewer=)`: Postavlja funkciju koja se koristi za prikaz HTML sadržaja, na primjer u R Markdownu.
- `options(par(mfrow=c(,)))`: Ove opcije se koriste za postavljanje izgleda grafičkih prikaza (broj redaka i stupaca) kada se crta više grafikona u jednom prozoru. Na primjer, `par(mfrow=c(2,2))` će postaviti raspored tako da se u prozoru mogu nacrtati četiri grafikona, raspoređena u 2 reda i 2 stupca.

Ove opcije su korisne za fino podešavanje različitih aspekata R sesije kako bi bolje odgovarali vašim potrebama i stilu rada.

Literatura

Allen, David E. 2017. "Practical Aspects of R in Finance, Management Information, and Decision Sciences." *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3085054.

Berkelaar, Michel, and et al. 2023. "lpSolve: Interface to 'Lp_solve' v. 5.5 to Solve Linear/Integer Programs." https://CRAN.R-project.org/package=lpSolve.

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2023. "Shiny: Web Application Framework for R." https://cran.r-project.org/web/packages/shiny/index.html.

Dayal, Vikram. 2015. "An Introduction to R for Quantitative Economics." *SpringerBriefs in Economics*.

Douglas, Alex, Deon Roos, Francesca Mancini, Ana Couto, and David Lusseau. 2022. "An Introduction to R." *URL Https://Intro2r. Com*.

González-Pérez, Beatriz, Victoria López, and Juan Sampedro. 2014. "Programming Global and Local Sequence Alignment by Using R." In *Knowledge Engineering and Management*, edited by Fuchun Sun, Tianrui Li, and Hongbo Li, 214:341–52. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-37832-4_31.

Hair, Joseph F., G. Tomas M. Hult, Christian M. Ringle, Marko Sarstedt, Nicholas P. Danks, and Soumya Ray. 2021. "Overview of R and RStudio." In *Partial Least Squares Structural Equation Modeling (PLS-SEM) Using R*, 31–47. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-80519-7_2.

Henningsen, Arne. 2022. "Linprog: Linear Programming / Optimization." https://CRAN.R-project.org/package=linprog.

Horton, Nicholas J., and Ken Kleinman. 2015. *Using R and RStudio for Data Management, Statistical Analysis, and Graphics*. CRC Press.

Irizarry, Rafael A. 2019. *Introduction to Data Science: Data Analysis and Prediction Algorithms with R*. CRC Press.

Konis, Kjell, Florian Schwendinger, and Kurt Hornik. 2023. "lpSolveAPI: R Interface to 'Lp_solve' Version 5.5.2.0." https://CRAN.R-project.org/package=lpSolveAPI.

Kuhn, Max, Jed Wing, Steve Weston, and Andre Williams. 2007. "The Caret Package." *Gene Expr*.

Kumbatović, Maja, Damir Kasum, and Tarzan Legović. 2004. "Uvod u Korištenje r-a." CRAN.

Meyer, David, Achim Zeileis, Kurt Hornik, Florian Gerbert, and Michael Friendly. 2023. "Vcd: Visualizing Categorical Data." https://cran.r-project.org/web/packages/vcd/index.html.

Miller, Allan M. 2017. "Review of Practical Data Science with R by Nina Zumel and John Mount." *ACM SIGACT News* 48 (1): 18–22. https://doi.org/10.1145/3061640.3061644.

Müller, Kirill, Hadley Wickham, Romain Francois, Jennifer Bryan, and RStudio. 2023. "Tibble: Simple Data Frames." https://cran.r-project.org/web/packages/tibble/index.html.

R Core Team, R. Developement Core. 2010. "R: A Language and Environment for Statistical Computing." *(No Title)*.

Ripley, Brian D. 2001. "The R Project in Statistical Computing." *MSOR Connections. The Newsletter of the LTSN Maths, Stats & OR Network* 1 (1): 23–25.

Spinu, V., G. Grolemund, and H. Wickham. 2021. "Make Dealing with Dates a Little Easier." *An R Package 'Lubridate*.

Van der Loo, Mark PJ. 2012. *Learning RStudio for R Statistical Computing*. Packt Publishing Ltd.

Venables, William N., David M. Smith, and R. Development Core Team. 2009. *An Introduction to R*. Network Theory Limited Bristol.

Venables, William N, David M Smith, R Development Core Team, et al. 2009. "An Introduction to r." Network Theory Limited London, UK.

Wickham, Hadley. 2019. *Stringr: Simple, Consistent Wrappers for Common String Operations. R Package Version 1.4. 0*.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, and Jim Hester. 2019. "Welcome to the Tidyverse." *Journal of Open Source Software* 4 (43): 1686.

Wickham, Hadley, and Jennifer Bryan. 2023. *R Packages*. " O'Reilly Media, Inc.".

Wickham, Hadley, Mine Çetinkaya-Rundel, and Garrett Grolemund. 2023. *R for Data Science*. " O'Reilly Media, Inc.".

Wickham, Hadley, Lionel Henry, and RStudio. 2023. "Purrr: Functional Programming Tools." https://cran.r-project.org/web/packages/purrr/index.html.

Wickham, Hadley, Jim Hester, Winston Chang, Jennifer Bryan, and RStudio. 2022. "Devtools: Tools to Make Developing R Packages Easier." https://cran.r-project.org/web/packages/devtools/index.html.

Wickham, Hadley, Jim Hester, Romain Francois, Jennifer Bryan, Shelby Bearrows, and P. B. C. Posit. 2023. "Package 'Readr'." *Read Rectangular Text Data. Available Online: Https://Cran. R-Project. Org/Web/Packages/Readr/Readr. Pdf (Accessed on 28 June 2022)*.

Wickham, Hadley, and Hadley Wickham. 2016. "Getting Started with Ggplot2." *Ggplot2: Elegant Graphics for Data Analysis*, 11–31.

Wickham, Hadley, and Maintainer Hadley Wickham. 2017. "Package 'Tidyr'." *Easily Tidy Data with'spread'and'gather ()'Functions*.

Xie, Yihui, and et al. 2023. "Knitr: A General-Purpose Package for Dynamic Report Generation in R." https://CRAN.R-project.org/package=knitr.

Yarberry, William, and William Yarberry. 2021. "Dplyr." *CRAN Recipes: DPLYR, Stringr, Lubridate, and RegEx in R*, 1–58.

Ypma, Jelmer, Steven G. Johnson, Aymeric Stamm, Hans W. Borchers, Dirk Eddelbuette, Brian Ripley, Kurt Hornik, et al. 2024. "Package 'Nloptr'." https://cran.r-project.org/web/packages/nloptr/nloptr.pdf.

Zhao, Yanchang, and Yonghua Cen. 2013. *Data Mining Applications with R*. Academic Press.