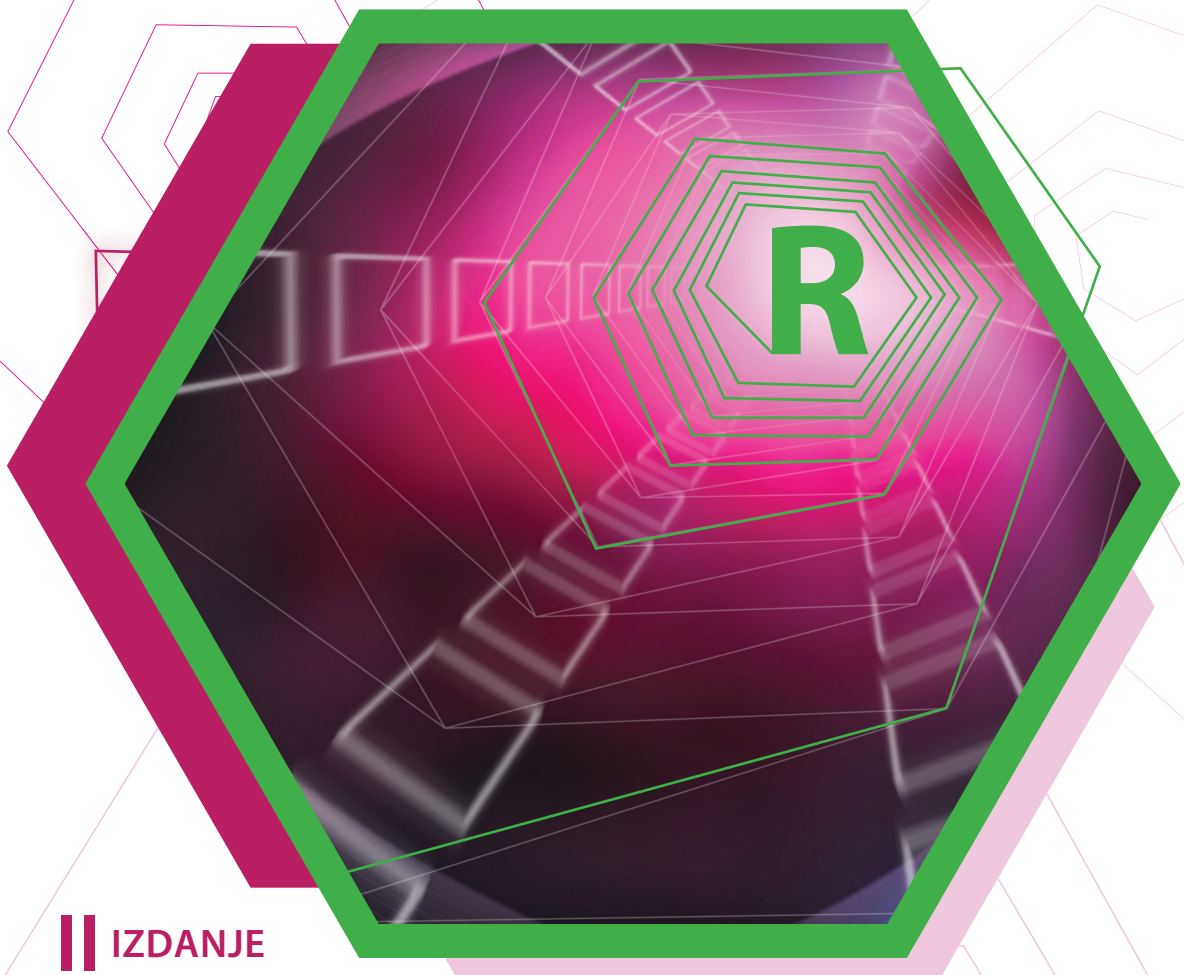


II IZDANJE

R

Analiza podataka

Tony Fischetti

kompjuter biblioteka

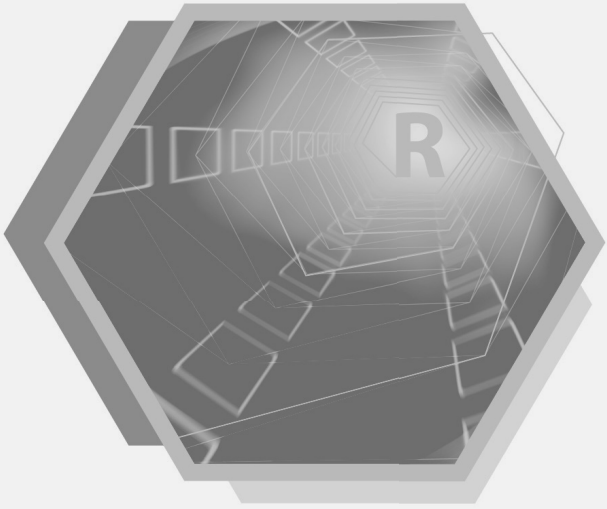
Packt›

Tony Fischetti

R

Analiza podataka

II IZDANJE



kompjuter biblioteka

Packt>

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autor: Benjamin Jakobus
Jason Marah

Prevod: Biljana Tešić

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Svetlost“, Čačak

Tiraž: 500

Godina izdanja: 2018.

Broj knjige: 504

Izdanje: Prvo

ISBN: 978-86-7310-527-7

Data Analysis with R
Second Edition

Tony Fischetti

ISBN 978-1-78839-372-0

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд,
се добија на захтев

O AUTORU

Toni Fischetti je analitičar podataka na koledžu „Factual“ i svakodnevno koristi R da bi izgradio lične sisteme za rangiranje i preporuku. Diplomirao je kognitivnu nauku na institutu „RensselaerPolytechnic“ - njegov diplomski rad je bio fokusiran na korišćenje statistike za proučavanje kratkoročne vizuelne memorije.

Uživa u pisanju softvera otvorenog koda i bloga na sajtu *On The Lambda* (`http://www.onthelambda.com/`), piše o sebi u trećem licu jednine i deli svoje znanje, koristeći jednostavan pristupačan jezik i primere.

Zahvaljujem se NYPL-u, R zajednici, milionima ljudi koji me podržavaju na Internetu, kompanijama „Toblerone“, „Ignatius“, „Lex i Pierre“ i „Bethany Wickhamu“. Posebnu zahvalnost izražavam Andrei Fischetti zbog njene ljubavi i podrške. Na kraju, zahvaljujem se mom ocu, koji je u meni razvio ljubav prema učenju i interesovanje za nauku i statistiku, a i mojoj majci na njenoj ljubavi i velikoj podršci.

O RECENZENTU

Manoj Kumar je konsultant sa više od 15 godina iskustva iz različitih oblasti i angažovan je na poboljšanju implementacije procesa i optimizacije rada u tipičnim proizvodnim okruženjima i proizvodnoj industriji korišćenjem napredne prediktivne i preskriptivne analitike, koja obuhvata mašinsko učenje, dubinsko učenje, simboličku dinamiku, dinamičku neuronsku mrežu, mehanizme električnih kola i Markovljev proces odlučivanja.

Stekao je iskustvo u sledećim oblastima:

- transport i upravljanje lancem snabdevanja
- proces i proizvodnja
- rudarstvo i energetika
- maloprodaja, prodaja robe široke potrošnje, zdravstvena zaštita, marketing i finansijsko računovodstvo

Davor Lozić je viši softverski inženjer, zainteresovan za različite teme, posebno za bezbednost računara, algoritme i strukture podataka. Upravlja timovima sa više od 15 inženjera i vanredni je profesor koji drži predavanja o sistemima baze podataka, Javi i interoperabilnosti. Možete da posetite njegov veb sajt na adresi http://varriorkitti.com. Voli mačke, pa ako imate smešne slike mačaka ili želite da razgovarate o bilo kom aspektu tehnologije, slobodno mu se obratite na navedenoj adresi.

„PACKT“ TRAŽI AUTORE KAO ŠTO STE VI

Ako ste zainteresovani da postanete autor za „Packt“, posetite stranicu authors.packtpub.com i prijavite se odmah. Mi smo radili sa hiljadama programera i profesionalaca da bismo im pomogli da podele svoj uvid sa globalnom tehničkom zajednicom. Možete da popunite osnovnu prijavu, da se prijavite za specifičnu temu za koje tražimo autore ili da nam pošaljete neku svoju ideju.

UVOD

Istaći ću odmah da postoji mnoštvo knjiga o analizi podataka i programskom jeziku R. Pretpostavljam da već znate zašto je izuzetno korisno naučiti R i analizu podataka (ako, pak, ne znate, zašto čitate ovaj predgovor?!), ali dozvolite da preporučim ovu knjigu kao vodič na vašem „putovanju".

Prvo, učenje analize podataka je za mene bilo komplikovano. Postoje osobe sa urođenim talentom za statistiku koje mogu odmah da je shvate, ali mislim da ja nisam jedan od njih. Nisam odustajao, ne zato što sam odmah razumeo analizu podataka, već zato što znao sam da je neophodna, a ja volim nauku i istraživanje. Danas volim ovu oblast samu po sebi, a ne zato što mi je potrebna, ali to sam shvatio tek posle nekoliko meseci truda. Na kraju, nakon što sam pročitao mnogo knjiga, delovi slagalice su počeli da se sklapaju. Zatim sam počeo da podučavam prijatelje koji su bili zainteresovani za ovu temu i video sam da su nailazili na iste prepreke koje sam i ja morao da prevaziđem. Mislim da mi je iskustvo u ovoj oblasti omogućilo da shvatim probleme sa kojima se susreću studenti statistike - mogu da ih razumem na način na koji drugi možda neće moći. Ne dopustite da vas uplaši to što sam ja imao poteškoća da shvatim analizu podataka, jer danas sa sigurnošću mogu reći da znam o čemu govorim.

Drugo, ova knjiga je nastala zbog frustracije što je većina tekstova o statistici napisana na način koji nije nimalo produktivan. Nasuprot tome, ja sam prihvatio jedan opušten, ali ne i suviše neozbiljan pristup.

Treće, ova knjiga sadrži mnogo podataka koje bih voleo da su zastupljeni u više izvora koje sam koristio kada sam učio R analizu podataka. Na primer, čitava poslednja lekcija posebno pokriva teme koje predstavljaju ogroman izazov za R analitičare kada prvi put primenjuju znanje na nesavršene podatke iz stvarnog sveta.

Na kraju, dugo sam razmišljao o tome kako da predstavim ovu knjigu i koji redosled tema bi bio najbolji. Napisao sam biblioteku i dizajnirao algoritme za ovaj postupak. Redosled tema u u ovoj knjizi sam pažljivo predstavio na sledeći način: (a) jedna tema se nadgrađuje na drugu, (b) obezbeđena su poglavlja jednostavnijeg sadržaja u skladu sa nivoom složenosti tema (psiholozi ovo nazivaju povremeno nagrađivanje), (c) grupisane su povezane teme i (d) smanjen je broj tema za koje je neophodno znanje iz još nenaučenih oblasti (nažalost, ovo je uobičajeno u statistici). Ovaj postupak sam detaljno opisao u tekstu na blogu koji možete pročitati na adresi http://bit.li/teach-stats.

Poenta je u tome što je ova knjiga veoma posebna – knjiga u koju sam uložio celokupno svoje znanje. Međutim, analiza podataka može biti veoma komplikovana oblast i ponekad izgleda kao da ništa nema smisla. U takvim situacijama su se mnogi drugi (čak i ja) osećali kao da „stoje u mestu“. Nemojte odustajati, jer je „nagrada“ sjajna. Ako je „smetenjak“ kao što sam ja mogao da nauči analizu podataka, možete i vi. Samo napred!

ZA KOGA JE OVA KNJIGA

Ako prvi put učite analizu podataka ili, pak, želite da poboljšate znanje koje već imate, ova knjiga će se pokazati dragocenim izvorom. Ako tražite knjigu koje će vas voditi od osnova do primene naprednih i efikasnih analitičkih metodologija i ako imate neko prethodno iskustvo u programiranju i matematici, onda je ova knjiga za vas.

ŠTA OBUHVATA OVA KNJIGA

Poglavlje 1, Osnove programskog jezika R, sadrži pregled aspekata R-a koje ćete morati da upoznate zbog sledećih poglavlja. Ovde učimo osnove R sintakse i osnovne strukture podataka R-a, pišemo funkcije, učitavamo podatke i instaliramo pakete.

U Poglavlju 2, Oblik podataka, razmatraju se univarijacioni podaci. Naučićete različite tipove podataka, opisivanje jedinstvenih podataka i vizuelizovanje oblika podataka.

Poglavlje 3, Opis veza, obuhvata multivarijacione podatke. Konkretno, naučićete tri osnovne klase bivarijacionih veza i njihov opis.

U Poglavlju 4, Verovatnoća, naučićete osnove teorije verovatnoće, Bajesovu teoremu i distribuciju verovatnoće.

U Poglavlju 5, Korišćenje podataka za uzorkovanje i procenu, razmatra se teorija uzorkovanja i procene. Pomoću primera naučićete centralnu graničnu teoremu, procenu parametara i intervale poverenja.

U Poglavlju 6, Testiranje hipoteza, predstavljeno je testiranje nulte hipoteze (NHST – Null Hypothesis Significance Testing). Naućićete mnogo popularnih testova hipoteza i njihove neparametarske alternative. Možda je najvažnije je da ćete razotkriti zablude o NHST-u.

U Poglavlju 7, Bajesove metode, predstavljena je alternativa za NHST koja se zasniva na intuitivnijem prikazu verovatnoće. Takođe ćete upoznati prednosti i nedostatke ovog pristupa.

U Poglavlju 8, Bootstrap, opisan je još jedan pristup za NHST pomoću tehnike pod nazivom ponovno uzorkovanje (resampling). Upoznaćete prednosti i nedostatke ovog pristupa. Osim toga, ovo poglavlje služi kao odlična nadgradnja sadržaja iz poglavlja 5 i 6.

U Poglavlju 9, Predviđanje kontinualnih promenljivih, započinjemo još jednu novu lekciju o prediktivnoj analitici i detaljno razmatramo linearnu regresiju. Naučićete sve što je potrebno o ovoj tehnici, kada da je koristite i na koje „zamke“ treba da obratite pažnju.

U Poglavlju 10, Predviđanje kategorijskih promenljivih, predstavljene su četiri najpopularnije tehnike klasifikacije. Koristićemo sve četiri tehnike u svim primerima, zahvaljujući čemu ćete shvatiti zašto je svaka od njih sjajna.

U Poglavlju 11, Predviđanje promena tokom vremena, završavamo lekciju o prediktivnoj analitici i predstavljamo analizu vremenskog niza i prognozu. Na kraju, razmatramo osnove jednog od odličnih metoda prognoze vremenskog niza.

Poglavlje 12, Izvori podataka, počinje lekcijom u kojoj je opisana analiza podataka u stvarnom svetu. Razmatra se upotreba različitih izvora podataka u R-u. Konkretno, naučićete kako da se povežete sa bazom podataka i da zahtevate i učitate JSON i XML pomoću primera.

U Poglavlju 13, Upravljanje nedostajućim podacima, opisani su nedostajući podaci, način za identifikovanje tipa nedostajućih podataka i dve osnovne metode za upravljanje njima.

U Poglavlju 14, Upravljanje neurednim podacima, predstavljeni su neki od problema u vezi sa korišćenjem nesavršenih podataka u praksi. Ovo obuhvata proveru neočekivanih unosa, korišćenje regexa i proveru valjanosti podataka pomoću paketa assertr.

U Poglavlju 15, Upravljanje velikim podacima, razmatrane su neke od tehnika koje mogu da se koriste za skupove podataka veće od podataka kojima se može brzo upravljati bez nekog planiranja. Ključni elementi ovog poglavlja su paralelizacija i Rcpp.

U Poglavlju 16, Korišćenje popularnih R paketa, potvrđujemo da smo već koristili mnogo popularnih paketa u ovoj lekciji, ali popunjavamo neke praznine i predstavljamo neke od najsavremenijih paketa, zahvaljujući kojima brzina i jednostavnost korišćenja postaju prioritet.

U Poglavlju 17, Reproducibilnost i najbolje tehnike, završavamo razmatranje izuzetno važne (ali često ignorisane) teme - kako da R koristite kao profesionalac. Ovo obuhvata učenje alatki, organizacije i reproducibilnosti.

Da biste dobili maksimum iz ove knjige:

Kompletan kod u ovoj knjizi je napisan za najnoviju verziju R-a 3.4.3 u vreme pisanja ove knjige. Najbolje bi bilo da ažurirate R verziju, ali skoro ceo kod bi trebalo da funkcioniše u svakoj skorijoj verziji R-a. Neki R paketi koje ćemo instalirati ipak zahtevaju najnovije verzije. Za druge softvere koji su upotrebljeni u ovoj knjizi uputstva će biti navedena prema potrebi. Međutim, ako želite da budete u startnoj prednosti, instalirajte Rstudio, JAGS i C++ kompajler (ili Rtools ako koristite Windows).

Preuzimanje fajlova primera koda

Fajlove sa primerima koda za ovu knjigu možete da preuzmete sa našeg sajta ukoliko posetite veb stranicu knjige:
http://bit.ly/2sA1t55

Da biste preuzeli kod, kliknite link “Preuzmite kod”:

http://bit.ly/2kQDjzd

Kada je fajl preuzet, raspakujte ili ekstrahujte direktorijum, koristeći najnoviju verziju:

- WinRAR/7-Zip za Windows
- Zipeg/iZip/UnRarX za Mac
- 7-Zip/PeaZip za Linux

Upotrebljene konvencije

Postoji veliki broj konvencija teksta koje su upotrebljene u ovoj knjizi.

`CodeInText`: ukazuje na reči koda u tekstu, nazive tabele baze podataka, nazive direktorijuma, nazive fajlova, ekstenzije fajlova, nazive putanje, skraćene URL-ove, korisnički unos i Twitter postove. Evo i primera: „Instalirajte preuzeti imidž fajl `WebStorm-10*.dmg` kao drugi disk na vaš sistem“.

Blok koda je prikazan na sledeći način:

```
# don‘t worry about
memorizing this temp.density
<- density(airquality$Temp)
pdf <- approxfun(temp.density$x, temp.density$y, rule=2)
integrate(pdf, 80, 90)
```

Kada želimo da privučemo pažnju na određeni deo bloka koda, relevantne linije ili stavke će biti ispisane zadebljanim slovima:

```
[default]
exten => s,1,Dial(Zap/1|30) exten => s,2,Voicemail(u100) exten =>
s,102,Voicemail(b100) exten => i,1,Voicemail(s0)
```

Svaki unos komandne linije ili ispis će biti prikazan na sledeći način:

```
table(mtcars$carb) / length(mtcars$carb)
      1       2       3       4       6       8
0.21875 0.31250 0.09375 0.31250 0.03125 0.03125
```

Zadebljana slova ukazuju na novi termin, važnu reč ili reči koje vidite na ekranu. Na primer, reči u menijima ili okvirima za dijalog prikazane su u tekstu na sledeći način: „Selektujte **System info** iz panela **Administration**“.

Napomene ili važna obaveštenja prikazani su ovako.

Saveti i trikovi su prikazani ovako.

KONTAKTIRAJTE SA NAMA

Povratne informacije od naših čitalaca su uvek dobrodošle.

Osnovne povratne informacije - Pošaljite e-mail na adresu informatori@kombib.rs i u naslovu poruke napišite naslov ove knjige. Ako imate bilo kakva pitanja o bilo kom aspektu ove knjige, pošaljite nam e-mail na adresu informatori@kombib.rs.

Štamparske greške - Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške mogu da se potkradu. Ako pronađete grešku u ovoj knjizi, bili bismo zahvalni ako biste nam to prijavili. Posetite stranicu, kliknite Ostavite komentar i unesite detalje.

Piraterija - Ako na Internetu pronađete ilegalnu kopiju naših knjiga, u bilo kojoj formi, molimo vas da nas o tome obavestite i pošaljete adresu lokacije ili naziv veb sajta. Kontaktirajte sa nama na adresiinformatori@kombib.rs i pošaljite nam link ka sumnjivom materijalu.

Ako ste zainteresovani da postanete autor - Ako postoji tema za koju ste specijalizovani i zainteresovani ste da pišete ili sarađujete na nekoj od knjiga, pogledajte vodič za autore na adresiauthors.packtpub.com.

Piraterija: Ako pronađete ilegalnu kopiju naših knjiga na Internetu, u bilo kojoj formi, molimo vas da nas o tome obavestite i pošaljete nam adresu lokacije ili naziv web sajta. Molimo vas, kontaktirajte sa nama na adresi informatori@kombib.rs i pošaljite nam link ka sumnjivom materijalu.

1

Osnove programskog jezika R

Pre nego što „zaronimo“ u (druge) zabavne tehnike (uzorkovanje višedimenzionalne distribucije verovatnoće, korišćenje konveksne optimizacije za prilagođavanje modela podataka i tako dalje), bilo bi korisno da pregledamo one aspekte R-a za koje je u sledećim poglavljima potrebno znanje. Ako sebe smatrate R guruom, trebalo bi makar da pregledate ovo poglavlje, jer ćete najverovatnije uvideti da su idiomi, paketi i stil koji su ovde predstavljeni korisni za praćenje ostatka knjige.

Ako vam R nije mnogo bitan (još uvek) i ako ga učite samo radi statistike, možete da odahnete, jer ćete moći da kod koji je naveden u ovoj knjizi pokrenete u interaktivnom R interpreteru, uz veoma male izmene. Međutim, uveren sam da ćete do kraja ove knjige da steknete poštovanje prema R-u i da dobro shvatite metode u analizi podataka.

Pokrenite vaš R interpreter i počnite da ga koristite!

UVOD U OSNOVE

U interaktivnom interpreteru R svaka linija koja počinje znakom > označava R koji zahteva unos (ako se prikaže odzivnik koji počinje znakom +, to znači da niste završili kucanje iskaza i R od vas traži da navedete ostatak izraza). Ako pritisnete taster Return, to će poslati vaš unos u R radi evaluacije. Odgovor R-a se vraća na liniju odmah nakon vašeg unosa, a zatim R od vas zahteva da unesete više podataka. Ovo se zove **REPL (Read-Evaluate-Print-Loop)**. Osim toga, R može da pročita skup komandi koje su sačuvane u datoteci pod nazivom **batch mode** (komandni režim), ali mi ćemo koristiti interaktivni režim u većem delu knjige.

Kao što možete da pretpostavite, R podržava sve poznate matematičke operacije kao i većina drugih jezika.

Aritmetika i dodela

Pogledajte sledeći primer:

```
> 2 + 2
[1] 4


> 9 / 3
[1] 3


> 5 %% 2     modulus operator (remainder of 5 divided by 2)
[1] 1
```

Sve ono što se dešava posle znaka tarabe ili znaka funte, `#`, (ili heš znaka za vas mlade) R interpreter će zanemariti. Ovo je korisno za dokumentovanje koda na prirodnom jeziku u obliku **komentara**.

U aritmetičkom izrazu sa više operacija R će slediti standardni redosled operacija iz matematike. Da biste poništili ovaj uobičajeni redosled, treba da koristite zagrade iza podizraza za koji želite da bude izvršen prvi:

```
> 3 + 2 - 10 ^ 2        # ^ is the exponent operator
[1] -95
> 3 + (2 - 10) ^ 2
[1] 67
```

U praksi su skoro svi složeni izrazi podeljeni na međuvrednosti koje su dodeljene promenljivim. Kada ove promenljive upotrebimo u budućim izrazima, to je slično kao kada bismo promenljivu zamenili vrednošću koja joj je dodeljena. Osnovni operator dodele je `<-`:

```
> # assignments follow the form VARIABLE <- VALUE
> var <- 10
> var
[1] 10
> var ^ 2
[1] 100
> VAR / 2             # variable names are case-sensitive
Error: object 'VAR' not found
```

Primetićete da u prvoj i drugoj liniji u prethodnom isečku koda nije ispisan rezultat, pa će R odmah tražiti da unesete više podataka. Zadatak operatora dodele je da promenljivoj dodeli vrednost, odnosno da izmeni postojeću vrednost promenljive. Uopšteno rečeno, operacije i funkcije koje se primenjuju na promenljive u R-u ne menjaju vrednost tih promenljivih. Umesto toga, ispisuje se rezultat operacije. Ako želite da promenljiva bude rezultat operacije, morate ponovo da je dodelite na sledeći način:

```
> var               # var is 10
[1] 10
> var ^ 2
[1] 100
> var               # var is still 10
[1] 10
> var <- var ^ 2    # no return value
> var               # var is now 100
[1] 100
```

Imajte na umu da nazivi promenljivih mogu da sadrže brojeve, donje crte i tačke – to je nešto što zbunjuje mnoge ljude koji poznaju druge programske jezike u kojima nije dozvoljeno korišćenje tačaka u nazivima promenljivih. Jedina ograničenja su da nazivi promenljivih moraju počinjati slovom (ili tačkom, a zatim slovom) i da ne smeju da budu jedna od rezervisanih reči u R-u, kao što su `TRUE`, `Inf` i tako dalje.

Iako su aritmetički operatori koje ste videli do sada sami po sebi funkcije, većina funkcija u R-u ima oblik `function_name(value(s) supplied to the function)`. Vrednosti koje se dodeljuju funkciji su njeni **argumenti**:

```
> cos(3.14159)      # cosine function
[1] -1
> cos(pi)           # pi is a constant that R provides
[1] -1
> acos(-1)          # arccosine function
[1] 3.141593
> acos(cos(pi)) + 10
[1] 13.14159
> # functions can be used as arguments to other functions
```

Verovatno se sećate sa časova matematike da je kosinus broja `pi -1` i da je arkus kosinus inverzna funkcija kosinusa.

Postoje stotine takvih korisnih funkcija definisanih u R bazi, od kojih ćete samo nekoliko videti u ovoj knjizi. Nakon sledeća dva odeljka kreiraćete svoje funkcije. Pre nego što nastavimo dalje, potrebno je da vidite neke neuobičajene vrednosti koje mogu da budu rezultat određenih operacija:

```
> 1 / 0
[1] Inf
> 0 / 0
[1] NaN
```

Uobičajeno je da se tokom praktične upotrebe R-a slučajno desi deljenje sa nulom. Kao što možete da vidite, ova nedefinisana operacija daje beskonačnu vrednost u R-u. Ako podelimo nulu sa nulom, rezultat će biti vrednost `NaN`, koja označava **nenumeričku vrednost** (Not A Number).

Logičke vrednosti i znakovi

Do sada smo koristili samo brojeve, ali u R-u postoje i drugi atomični tipovi podataka:

```
> foo <- TRUE         # foo is of the logical data type
> class(foo)          # class() tells us the type
[1] "logical"
> bar <- "hi!"        # bar is of the character data type
> class(bar)
[1] "character"
```

Logički tip vrednosti (koji se naziva i **Bulove promenljive**) može da sadrži vrednost `TRUE` ili `FALSE`, odnosno T ili F. Za ove tipove definisani su poznati operatori Bulove algebre:

```
> foo
[1] TRUE
> foo && TRUE                 # boolean and
[1] TRUE
> foo && FALSE
[1] FALSE
> foo || FALSE                # boolean or
[1] TRUE
> !foo                        # negation operator
[1] FALSE
```

U Bulovom izrazu koji sadrži logičku vrednost i broj svaki broj koji nije 0 se interpretira kao TRUE:

```
> foo && 1
[1] TRUE
> foo && 2
[1] TRUE
> foo && 0
[1] FALSE
```

Osim toga, postoje funkcije i operatori koji vraćaju sledeće logičke vrednosti:

```
> 4 < 2            # less than operator
[1] FALSE
> 4 >= 4           # greater than or equal to
[1] TRUE
> 3 == 3           # equality operator
[1] TRUE
> 3 != 2           # inequality operator
[1] TRUE
```

Baš kao što u R-u postoje funkcije koje se definišu samo za upotrebu sa numeričkim i logičkim tipovima podataka, postoje druge funkcije koje su dizajnirane za upotrebu sa znakovnim podacima (poznati su i kao **znakovni nizovi**):

```
> lang.domain <- "statistics"
> lang.domain <- toupper(lang.domain)
> print(lang.domain)
[1] "STATISTICS"
> # retrieves substring from first character to fourth
character
> substr(lang.domain, 1, 4)
[1] "STAT"
> gsub("I", "1", lang.domain)  # substitutes every "I" for
"1"
[1] "STAT1ST1CS"
> # combines character strings
> paste("R does", lang.domain, "!!!")
[1] "R does STATISTICS !!!"
```

Tok kontrole

Poslednja tema u ovom odeljku će biti konstrukcije za *tok kontrole*.

Najosnovnija konstrukcija za tok kontrole je iskaz `if`. Argumenat za iskaz `if` (ono što se nalazi između parametara) je izraz koji vraća logičku vrednost. Blok koda sa iskazom `if` se izvršava samo ako izraz ima vrednost `TRUE`:

```
> if(2 + 2 == 4)
+   print("very good")
[1] "very good"
> if(2 + 2 == 5)
+    print("all hail to the thief")
```

U kodu može da se izvrši više od jednog iskaza `if` ukoliko je za to omogućen uslov - treba samo da stavite iskaze u vitičaste zagrade ({}):

```
> if((4/2==2) && (2*2==4)){
+    print("four divided by two is two...")
+    print("and two times two is four")
+ }
[1] "four divided by two is two..."
[1] "and two times two is four"
```

Takođe možete da odredite blok koda koji će izvršiti iskaz `if` ako je uslov `FALSE`:

```
> closing.time <- TRUE
> if(closing.time){
+    print("you don't have to go home")
+    print("but you can't stay here")
+ } else{
+    print("you can stay here!")
+ }
[1] "you don't have to go home"
[1] "but you can't stay here"
> if(!closing.time){
+     print("you don't have to go home")
+     print("but you can't stay here")
+ } else{
+     print("you can stay here!")
+ }
[1] "you can stay here!"
```

Postoje druge konstrukcije za kontrolu toka (kao što su `while` i `for`), ali njih nećemo mnogo koristiti u ovom tekstu.

DOBIJANJE POMOĆI U R-U

Pre nego što nastavimo dalje, treba da u kratkom odeljku opišemo kako se dobija pomoć u R-u. U većini uputstava za R dobijanje pomoći je opisano u poslednjim odeljcima (ako je uopšte opisano). Na osnovu svog ličnog iskustva ipak smatram da je dobijanje pomoći jedna od prvih stavki koje će vam biti potrebne dok učite R. Učenje R-a ne mora da bude teško – ne žurite, postavljajte pitanja i zatražite pomoć na vreme. Samo napred!

Pomoć za R možete jednostavno dobiti u konzoli. Pokretanjem funkcije `help.start()` u odzivniku biće pokrenut pregledač sa uputstvom. U tom uputstvu možete da pročitate sve ono što vas zanima o R-u – od osnova, do detalja o tome kako funkcioniše interno.

Možete da dobijete pomoć o određenoj funkciji u R-u ako znate njen naziv. Naziv funkcije koja vas zanima možete da navedete kao argumenat u funkciji `help`. Na primer, ako želite da saznate više informacija o funkciji `gsub()`, pogledajte sledeći kod:

```
> help("gsub")
> # or simply
> ?gsub
```

Pomoću ovog koda možete da otvorite stranicu sa uputstvom na kojoj je opisano kako se koristi funkcija, uz primere korišćenja.

Pošto možete brzo da pristupite ovoj dokumentaciji, znači da se nećete osećati beznadežno kada naiđete na funkciju koju ranije niste videli. Nedostatak ovog izvanrednog i pogodnog mehanizma za pomoć je što treba da zapamtite redosled argumenata koje tražite.

Ponekad možda nećete zapamtiti tačan naziv funkcije koju tražite, ali ćete pretpostaviti koji bi mogao da bude. Ako se suočite sa ovim problemom, možete da koristite funkciju `help.search()`:

```
> help.search("chisquare")
> # or simply
> ??chisquare
```

Kada treba da pronađete značenje neke reči, najbolje bi bilo da koristite "dobri stari" veb pretraživač. Ako prvi put ne dobijete odgovarajuće rezultate, pokušajte da u pretraživač unesete termin koji se odnosi na programiranje ili statistiku.

VEKTORI

Vektori su najosnovnije strukture podataka u R-u i zaista su sveprisutni. U stvari, čak i pojedinačne vrednosti koje smo koristili do sada su, zapravo, vektori čija je dužina `1`. Zbog toga, interaktivna R konzola ispisuje `[1]`, zajedno sa svim rezultatima.

Vektori su, u osnovi, *uređena kolekcija vrednosti atomičnih podataka*. Oni mogu imati proizvoljnu dužinu (uz neka ograničenja) ili mogu biti samo jedna vrednost.

Prilikom ručne kanonske izgradnje vektora koristi se funkcija `c()`, čiji je naziv skraćenica od **combine** (kombinovati):

```
> our.vect <- c(8, 6, 7, 5, 3, 0, 9)
> our.vect
[1] 8 6 7 5 3 0 9
```

U prethodnom primeru kreirali smo numerički vektor, čija je dužina 7 (tj. Dženin telefonski broj).

Sada ćemo pokušati da znakovne podatke stavimo u ovaj vektor na sledeći način:

```
> another.vect <- c("8", 6, 7, "-", 3, "0", 9)
> another.vect
[1] "8" "6" "7" "-" "3" "0" "9"
```

R će pretvoriti sve stavke koje se nalaze u vektoru (pod nazivom **elementi**) u znakovne podatke da bi bio zadovoljen uslov prema kome svi elementi vektora moraju biti istog tipa. Slično se dešava kada koristite logičke vrednosti u vektoru sa brojevima – logičke vrednosti će biti pretvorene u 1 za `TRUE`, a u 0 za `FALSE`. Ove logičke vrednosti će biti pretvorene u `TRUE` i `FALSE` kada se koriste u vektoru koji sadrži znakove.

Izrada podskupova

Veoma često je potrebno ekstrahovati jedan ili više elemenata iz vektora. Za to se koristi tehnika pod nazivom **indeksiranje** (indexing) ili **izrada podskupova** (subsetting). Iza vektora u kvadratne zagrade (`[]`) stavljamo ceo broj koji se naziva **operator za indeksiranje**. Ovaj operator ukazuje R-u da treba da vrati elemenat na tom indeksu. Indeksi za vektore u R-u počinju na 1 i završavaju se na dužini vektora:

```
> our.vect[1]                    # to get the first value
[1] 8
> # the function length() returns the length of a vector
> length(our.vect)
[1] 7
> our.vect[length(our.vect)]     # get the last element of a vector
[1] 9
```

Napominjemo da je u prethodnom kodu funkcija upotrebljena u operatoru za indeksiranje. U ovakvom slučaju R procenjuje izraz u operatoru za indeksiranje i koristi broj koji vraća kao indeks radi ekstrahovanja.

Ako pokušamo da ekstrahujemo elemenat iz indeksa koji ne postoji, R će ispisati vrednost `NA`, što znači **nije dostupno** (not available). Tu posebnu vrednost ćete povremeno viđati u ostatku ovog teksta:

```
> our.vect[10]
[1] NA
```

Jedna od najmoćnijih funkcija u R-u je mogućnost upotrebe vektora za izradu podskupova drugih vektora:

```
> # extract the first, third, fifth, and
> # seventh element from our vector
> our.vect[c(1, 3, 5, 7)]
[1] 8 7 3 9
```

Mogućnost upotrebe vektora za indeksiranje drugih vektora možda vam sada ne izgleda korisna, ali uskoro će vam biti jasno da je takav utisak pogrešan - objasnićemo zašto je korisna.

Vektori se mogu kreirati i pomoću sekvenci:

```
> other.vector <- 1:10
> other.vector
[1]  1  2  3  4  5  6  7  8  9 10
> another.vector <- seq(50, 30, by=-2)
> another.vector
[1] 50 48 46 44 42 40 38 36 34 32 30
```

Ovde iskaz `1:10` kreira vektor od `1` do `10`. Iskaz `10:1` bi kreirao isti vektor sa 10 elemenata, ali elementi bi bili ispisani obrnutim redosledom. Funkcija `seq()` je uopštenija, jer omogućava kreiranje sekvenci pomoću koraka, osim mnogih drugih opcija.

Kombinovanjem znanja o sekvencama i vektorima za izradu podskupova vektora možemo da saznamo prvih pet cifara Dženinog broja telefona:

```
> our.vect[1:5]
[1] 8 6 7 5 3
```

Vektorizovane funkcije

Ono što čini R toliko moćnim je i činjenica da mnoge funkcije prihvataju vektore kao argumente. Ove vektorizovane funkcije su, obično, izuzetno brze i efikasne. Već smo videli jednu takvu funkciju, koja se zove `length()`, ali postoji i mnoštvo drugih:

```
> # takes the mean of a vector
> mean(our.vect)
[1] 5.428571
> sd(our.vect)    # standard deviation
[1] 3.101459
> min(our.vect)
[1] 0
> max(1:10)
[1] 10
> sum(c(1, 2, 3))
[1] 6
```

U praksi (na primer, prilikom čitanja podataka iz datoteka) uobičajeno je da se u vektorima pojave `NA` vrednosti:

```
> messy.vector <- c(8, 6, NA, 7, 5, NA, 3, 0, 9)
> messy.vector
[1]  8  6 NA  7  5 NA  3  0  9
> length(messy.vector)
[1] 9
```

Neke vektorizovane funkcije podrazumevano onemogućavaju `NA` vrednosti, pa u funkciji morate da navedete argumenat rezervisane reči, zajedno sa prvim argumentom:

```
> mean(messy.vector)
[1] NA
> mean(messy.vector, na.rm=TRUE)
[1] 5.428571
> sum(messy.vector, na.rm=FALSE)
```

```
[1] NA
> sum(messy.vector, na.rm=TRUE)
[1] 38
```

Vektori mogu da se konstruišu i iz logičkih vrednosti:

```
> log.vector <- c(TRUE, TRUE, FALSE)
> log.vector
[1]  TRUE TRUE FALSE
```

Kao što ste već videli, logičke vrednosti možemo „prisiliti“ da se ponašaju kao numeričke vrednosti ukoliko pokušamo da izračunamo vektor na sledeći način:

```
> sum(log.vector)
[1] 2
```

U suštini, rezultat će biti broj vrednosti `TRUE` u tom vektoru.

Postoje mnoge funkcije u R-u koje izračunavaju vektore i, kao rezultat, daju logičke vektore. Jedna od takvih funkcija je `is.na()`. Ona vraća logički vektor, čija je dužina ista kao dužina vektora koji je naveden kao argumenat sa vrednošću `TRUE` na mestu svake `NA` vrednosti. Da li se sećate našeg „neurednog“ vektora od malopre?

```
> messy.vector
[1]  8  6 NA  7  5 NA  3  0  9
> is.na(messy.vector)
[1] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
> #  8     6      NA    7     5      NA    3       0     9
```

Kada primenimo sve tehnike koje smo opisali, možemo da izračunamo broj `NA` vrednosti u vektoru na sledeći način:

```
> sum(is.na(messy.vector))
[1] 2
```

Kada primenite Bulove operatore na vektorima, takođe ćete, kao rezultat, dobiti logičke vektore dugačke kao vektor nad kojim se vrši operacija:

```
> our.vect > 5
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE
```

Ako želite da saznate broj cifara u Dženinom telefonskom broju koje su veće od 5, postupite na sledeći način:

```
> sum(our.vect > 5)
[1] 4
```

Napredna izrada skupova

Zar nisam pomenuo da možete da koristite vektore da biste izradili skupove drugih vektora! Kada izradimo podskupove vektora pomoću logičkih vektora iste dužine, ekstrahuju se samo elementi koji odgovaraju vrednostima `TRUE`. Srećom, sada će vam biti sve jasnije. Ako želite da iz Dženinog telefonskog broja ekstrahujete samo važeće cifre koje nemaju vrednosti `NA`, postupite na sledeći način:

```
> messy.vector[!is.na(messy.vector)]
[1] 8 6 7 5 3 0 9
```

Ovo je veoma važna funkcija R-a, pa je potrebno vreme da biste mogli da je razumete; pojavljivaće se više puta u ostatku knjige.

Logički vektor koji ima vrednost `TRUE` kada se vrednost `NA` pojavi u vektoru `merry.vector` (iz funkcije `is.na()`) negira se pomoću operatora negacije. Dobićemo vektor `TRUE` uvek kada odgovarajuća vrednost u vektoru `messy.vector` nije `NA`. Kada se ovaj logički vektor upotrebi za izradu podskupova originalnog „neurednog“ vektora, biće ekstrahovane samo vrednosti koje nisu `NA`.

Slično tome, možemo da prikažemo sve cifre iz Dženinog telefonskog broja koje su veće od 5 na sledeći način:

```
> our.vect[our.vect > 5]
[1] 8 6 7 9
```

Do sada smo prikazali samo elemente koji su ekstrahovani iz vektora. Međutim, kao što možemo da dodelimo i izmenimo promenljive, isto tako možemo da dodelimo vrednost različitim indeksima vektora i da izmenimo vektor. Na primer, ako nam Dženi kaže da smo pogrešili prvu cifru iz prvih pet cifara njenog telefonskog broja (tačan broj je 9), možemo ponovo da dodelimo samo taj elemenat, bez izmene drugih elemenata:

```
> our.vect
[1] 8 6 7 5 3 0 9
> our.vect[1] <- 9
> our.vect
[1] 9 6 7 5 3 0 9
```

Ponekad je potrebno da zamenimo sve `NA` vrednosti u vektoru vrednošću 0. Da bismo to uradili u našem „neurednom“ vektoru, možemo da izvršimo sledeću komandu:

```
> messy.vector[is.na(messy.vector)] <- 0
> messy.vector
[1] 8 6 0 7 5 0 3 0 9
```

Prethodno rešenje je elegantno, ali kreiranje kopije originalnog vektora i izmena te kopije predstavljaju bolje rešenje od izmene vektora. Kreiranje kopije vektora možete da obavite pomoću funkcije `ifelse()`.

Ova funkcija nema nikakve veze sa konstrukcijom za kontrolu if/else, jer sadrži tri argumenta: test koji vraća logičku/Bulovu vrednost, vrednost koja se koristi ako elemenat prođe test i drugu vrednost koja će se vratiti kao rezultat ako elemenat ne prođe test.

Modifikacija vektora može da se reimplementira pomoću funkcije `ifelse` na sledeći način:

```
> ifelse(is.na(messy.vector), 0, messy.vector)
[1] 8 6 0 7 5 0 3 0 9
```

Recikliranje

Poslednje važno svojstvo vektora i vektorskih operacija u R-u je da mogu da se recikliraju. Da biste razumeli šta to znači, ispitajte sledeći izraz:

```
> our.vect + 3
[1] 12  9 10  8  6  3 12
```

Ovaj izraz dodaje 3 svakoj cifri u Dženinom telefonskom broju, pa možda samo izgleda kao da R izvršava ovu operaciju između vektora i jedne vrednosti. Da li se sećate da sam rekao da su pojedinačne vrednosti, u stvari, vektori čija je dužina 1? R-u je instruisano da izvrši sabiranje po elementima vektora čija je dužina 7 i vektora čija je dužina 1. Pošto sabiranje po elementima nije definisano za vektore različitih dužina, R reciklira manje vektore, sve dok ne dostignu istu dužinu kao veći vektor. Kada su oba vektora iste dužine, R izvršava sabiranje elemenat po elemenat i vraća rezultat:

```
> our.vect + 3
[1] 12  9 10  8  6  3 12
```

Ovaj kod je jednak sledećem kodu:

```
> our.vect + c(3, 3, 3, 3, 3, 3, 3)
[1] 12  9 10  8  6  3 12
```

Ako želite da ekstrahujete svaku drugu cifru iz Dženinog telefonskog broja, postupite na sledeći način:

```
> our.vect[c(TRUE, FALSE)]
[1] 9 7 3 9
```

Ovaj način funkcioniše, zato što se vektor `c(TRUE, FALSE)` ponavlja dok ne dostigne dužinu 7, što je jednako sledećem kodu:

```
> our.vect[c(TRUE, FALSE, TRUE, FALSE, TRUE, FALSE, TRUE)]
[1] 9 7 3 9
```

Uobičajeno je u vezi sa rekcikliranjem vektora je da su se R korisnici tokom nekih aritmetičkih operacija koje uključuju vektore različite dužine susreli sa upozorenjima u R-u koja ukazuju da manji vektor ne može da bude ponovljen više puta da bi dostigao dužinu većeg vektora. Ovo nije problem kada obavljate vektorske aritmetičke operacije sa pojedinačnim vrednostima, jer 1 može da se ponavlja više puta da bi odgovarao veličini svakog vektora (koja svakako mora biti ceo broj). Međutim, problem se javlja kada dodamo broj 3 svakom drugom elementu u Dženinom telefonskom broju:

```
> our.vect + c(3, 0)
[1] 12  6 10  5  6  0 12
Warning message:
In our.vect + c(3, 0) :
  longer object length is not a multiple of shorter object length
```

Svideće vam se ova upozorenja, jer, zahvaljujući njima, nećete praviti ozbiljne greške.

Pre nego što pređete na sledeći odeljak, važno je napomenuti da u mnogim drugim programskim jezicima koristimo petlje i druge strukture za kontrolu. Iako petlje mogu da se koriste u R-u, postoji sofisticiranije rešenje – upotreba operacija sa vektorima/matricama. Rešenje da se koriste vektorizacija i recikliranje ne samo da je elegantno i kreativno, već je i mnogo efikasnije.

FUNKCIJE

Ako želimo da izvršimo neko izračunavanje za koje ne postoji funkcija u R-u, obično će biti potrebno da definišemo svoju funkciju. Prilagođena funkcija u R-u se definiše pomoću sledeće sintakse:

```
> function.name <- function(argument1, argument2, ...){
+   # some functionality
+ }
```

Na primer, ako treba da napišemo funkciju koja utvrđuje da li broj koji je naveden kao argumenat ima vrednost `even`, to možemo da uradimo na sledeći način:

```
> is.even <- function(a.number){
+   remainder <- a.number %% 2
+   if(remainder==0)
+     return(TRUE)
+   return(FALSE)
+ }
> # testing it
> is.even(10)
[1] TRUE
> is.even(9)
[1] FALSE
```

Kao primer funkcije koja koristi više argumenata, izvršićemo generalizaciju prethodne funkcije, tako što ćemo kreirati funkciju koja utvrđuje da li je prvi argumenat deljiv sa drugim argumentom:

```
> is.divisible.by <- function(large.number, smaller.number){
+   if(large.number %% smaller.number != 0)
+     return(FALSE)
+   return(TRUE)
+ }
> # testing it
> is.divisible.by(10, 2)
[1] TRUE
> is.divisible.by(10, 3)
[1] FALSE
> is.divisible.by(9, 3)
[1] TRUE
```

Naša funkcija `is.even()` sada može da bude napisana na sledeći način:

```
> is.even <- function(num){
+   is.divisible.by(num, 2)
+ }
```

Veoma često je potrebno u R-u primeniti određenu funkciju na svaki elemenat vektora. Umesto da upotrebimo petlju da bismo izvršili iteraciju elemenata vektora, kao u mnogim drugim jezicima, upotrebićemo funkciju `sapply()`, koja prihvata vektor i funkciju kao argumente, pa primenjuje funkciju na svaki elemenat i vraća vektor kao rezultat. Možemo da koristimo funkciju `sapply()` na isti način da bismo saznali koje su parne cifre u Dženinom telefonskom broju:

```
> sapply(our.vect, is.even)
[1] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
```

Funkcija `sapply` je odlična, jer prihvata svaki elemenat i koristi ga kao argumenat u funkciji `is.even()`, koja prihvata samo jedan argumenat. Ako želite da saznate cifre koje su deljive sa 3, potrebno je malo više posla.

Jedna od opcija je da definišete funkciju `is.divisible.by.three()`, koja prihvata samo jedan argumenat i koristi ga u funkciji `sapply`. Međutim, rešenje koje se češće koristi je definisanje neimenovane funkcije u telu funkcije `sapply`:

```
> sapply(our.vect, function(num){is.divisible.by(num, 3)})
[1]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
```

Ovde smo kreirali funkciju koja proverava da li je argumenat deljiv sa 3, ali je nećemo dodeliti promenljivoj, već ćemo je koristiti direktno u telu funkcije `sapply`. Ove neimenovane funkcije za jednokratnu upotrebu nazivaju se **anonimne funkcije** ili **lambda funkcije** (naziv potiče od lambda računa, koji je osmislio Alonzo Church).

Ovo je nešto naprednija upotreba R-a, ali je veoma korisna, jer se vrlo često pojavljuje u praksi.

Ako iz Dženinog telefonskog broja želimo da ekstrahujemo cifre koje su deljive sa 2 i 3, kod možemo da napišemo na sledeći način:

```
> where.even <- sapply(our.vect, is.even)
> where.div.3 <- sapply(our.vect, function(num){
+   is.divisible.by(num, 3)})
> # "&" is like the "&&" and operator but for vectors
> our.vect[where.even & where.div.3]
[1] 6 0
```

Sjajno!

Napominjemo, ako želite da budete perfekcionisti, onda ćete u telu funkcije dodati izraz za sprečavanje izračunavanja modula u kojem će prvi broj biti manji od drugog. Kada bismo dodali ovaj izraz, funkcija ne bi pogrešno označila da je broj 0 deljiv sa 2 i 3. Ja nisam perfekcionista, pa neću menjati funkciju. Korigovanje ove funkcije prepuštam kao vežbu čitaocu (perfekcionisti).

MATRICE

Osim vektorske strukture podataka, R ima matricu, okvir podataka, listu i strukturu podataka array (niz). Iako ćemo u knjizi koristiti sve ove tipove (osim nizova), potrebno je da u ovom poglavlju razmotrimo samo prva dva.

Kao i u matematici, matrica u R-u je pravougaoni niz vrednosti (jednog tipa) koje su raspoređene u redovima i kolonama i njima se može upravljati kao celinom. Operacije sa matricama su važne za analizu podataka.

Matricu možete da kreirate tako što ćete navesti vektor u funkciji `matrix()`:

```
> a.matrix <- matrix(c(1, 2, 3, 4, 5, 6))
> a.matrix
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
[5,]    5
[6,]    6
```

Ovaj kod kreira matricu sa svim navedenim vrednostima u jednoj koloni. Možemo da kreiramo sličnu matricu sa dve kolone, tako što ćemo navesti `matrix()` sa opcionim argumentom `ncol`, koji određuje broj kolona:

```
> a.matrix <- matrix(c(1, 2, 3, 4, 5, 6), ncol=2)
> a.matrix
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Možemo da kreiramo istu matricu povezivanjem dva vektora `c(1, 2, 3)` i `c(4, 5, 6)` pomoću kolona, koristeći funkciju `cbind()` na sledeći način:

```
> a2.matrix <- cbind(c(1, 2, 3), c(4, 5, 6))
```

Možemo da kreiramo transpoziciju matrice (u kojoj su redovi i kolone zamenjeni) povezivanjem ovih vektora pomoću reda:

```
> a3.matrix <- rbind(c(1, 2, 3), c(4, 5, 6))
> a3.matrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Ovo možemo izvršiti pomoću funkcije za transpoziciju matrice u R-u `t()`:

```
> t(a2.matrix)
```

Neke druge funkcije koje se mogu primeniti na celim vektorima su `rowSums()/colSums()` i `rowMeans()/colMeans()`:

```
> a2.matrix
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> colSums(a2.matrix)
[1]  6 15
> rowMeans(a2.matrix)
[1] 2.5 3.5 4.5
```

Ako vektori imaju funkciju `sapply()`, matrice imaju funkciju `apply()`. Prethodne dve funkcije se mogu napisati opširnije na sledeći način:

```
> apply(a2.matrix, 2, sum)
[1]  6 15
> apply(a2.matrix, 1, mean)
[1] 2.5 3.5 4.5
```

Ovde broj 1 označava da će R izvršiti navedenu funkciju u redovima, a broj 2 da će je izvršiti u kolonama.

Operator množenja matrice u R-u je `%*%`:

```
> a2.matrix %*% a2.matrix
Error in a2.matrix %*% a2.matrix : non-conformable arguments
```

Množenje matrice se definiše samo kada je broj kolona u prvoj matrici jednak broju redova u drugoj matrici:

```
> a2.matrix
     [,1] [,2]
[1,]  1  4
[2,]  2  5
[3,]    3    6
> a3.matrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> a2.matrix %*% a3.matrix
     [,1] [,2] [,3]
[1,]   17   22   27
[2,]   22   29   36
[3,]   27   36   45
> # dim() tells us how many rows and columns
> # (respectively) there are in the given matrix
> dim(a2.matrix)
[1] 3 2
```

Da biste izvršili indeksiranje elementa matrice u drugom redu i prvoj koloni, potrebno je da navedete oba broja u operatoru za indeksiranje:

```
> a2.matrix[2,1]
[1] 2
```

Mnogo korisnici R-a zaborave redosled kojim se indeksi moraju prikazivati. Zapamtite: prvo se prikazuje red, a zatim kolone!

Ako ostavite jedno mesto prazno, R će pretpostaviti da kao rezultat želite da prikažete jednu celu kolonu:

```
> # returns the whole second column
a2.matrix[,2]
[1] 4 5 6
> # returns the first row
> a2.matrix[1,]
[1] 1 4
```

Kao i uvek, možemo da koristimo vektore u operatoru za indeksiranje:

```
> # give me element in column 2 at the first and third row
> a2.matrix[c(1, 3), 2]
[1] 4 6
```

UČITAVANJE PODATAKA U R-U

Do sada smo unosili podatke direktno u interaktivnu R konzolu. Za svaki skup podataka koji nije jednostavan ovaj način unosa podataka nije baš najbolje rešenje. Srećom, R ima robusni skup funkcija za čitanje podataka direktno iz spoljnih datoteka.

Na vašem čvrstom disku kreirajte datoteku pod nazivom `favorites.txt`, koja izgleda ovako:

```
flavor,number
pistachio,6
mint chocolate chip,7
vanilla,5
chocolate,10
strawberry,2
neopolitan,4
```

Ovi podaci predstavljaju broj učenika u razredu koji vole određeni ukus sladoleda od sojinog mleka. Možemo da učitamo datoteku u promenljivu pod nazivom `favs` na sledeći način:

```
> favs <- read.table("favorites.txt", sep=",", header=TRUE)
```

Ako se pojavi greška koja ukazuje da ne postoje takva datoteka ili direktorijum, u R-u navedite pun naziv putanje vašeg skupa podataka ili pokrenite sledeću komandu:

```
> favs <- read.table(file.choose(), sep=",", header=TRUE)
```

Prethodna komanda prikazuje dijalog za otvaranje datoteke da biste mogli da se krećete kroz datoteku koju ste upravo kreirali.

Argumenat `sep=","` ukazuje R-u da je svaki elemenat u redu odvojen zarezom. U drugim uobičajenim formatima podataka vrednosti su razdvojene tabulatorima i vertikalnom crtom (`"|"`). Vrednost `sep` treba da bude `"\t"` i `"|"` (tim redom).

Argumenat `header=TRUE` ukazuje R-u da prvi red datoteke treba da se interpretira kao nazivi kolona. Možete da unesete komandu `?read.table` u konzolu da biste naučili više o ovim opcijama.

Čitanje iz datoteka u formatu u kojem su vrednosti razdvojene zarezom (obično sa ekstenzijom `.csv`) toliko je često da je u R-u osmišljena specifična funkcija samo za tu svrhu. Prethodni izraz za uvoz okvira podataka možete najbolje napisati na sledeći način:

```
> favs <- read.csv("favorites.txt")
```

Sada su svi podaci iz datoteke u promenljivoj klase `data.frame`. Okvir podataka (data frame) može se smatrati pravouganim nizom podataka, koji ste možda videli u programu za tabelarno izračunavanje. On se može posmatrati i kao matrica. U stvari, možemo da koristimo matrični stil indeksiranja da bismo iz okvira podataka ekstrahovali elemente. Okvir podataka se, ipak, razlikuje od matrice po tome što može da sadrži kolone sa različitim tipovima podataka. Na primer, matrica može da sadrži samo jedan od ovih tipova, a skup podataka koji smo upravo učitali sadrži znakovne podatke u prvoj i numeričke podatke u drugoj koloni.

Sada ćete videti upotrebu komande `head()`, pomoću koje ćemo prikazati prvih nekoliko linija okvira podataka:

```
> head(favs)
              flavor number
1           pistachio      6
2 mint chocolate chip      7
3             vanilla      5
4           chocolate     10
5          strawberry      2
6          neopolitan      4

> class(favs)
[1] "data.frame"
> class(favs$flavor)
[1] "factor"
> class(favs$number)
[1] "numeric"
```

U redu, lagao sam da skup podataka u primeru sadrži znakovne podatke! Pa, šta?! Tehnički, `flavor` je tip podataka `factor`, a ne znakovni tip.

Još nismo prikazali faktore, ali su zaista jednostavni. U suštini, faktori su kodiranja za kategorijalne promenljive, tj. promenljive koje prihvataju jedan od ograničenih brojeva kategorija – {`"high", "medium", and "low"`} ili {`"control","experimental"`}.

Iako su faktori izuzetno korisni u statističkom modelovanju u R-u, ono što zbunjuje nove i iskusne korisnike R-a je činjenica da R podrazumevano i automatski interpretira kolonu iz podataka koji se čitaju sa diska kao faktor ukoliko sadrže znakove. Zbog toga ćemo sprečiti ovo ponašanje ručno, tako što ćemo opcioni argumenat rezervisane reči `stringsAsfactors` dodati komandama `read.*`:

```
> favs <- read.csv("favorites.txt", stringsAsFactors=FALSE)
> class(favs$flavor)
[1] "character"
```

Sada je mnogo bolje! Ako želite da ovo ponašanje postavite kao podrazumevano, pročitajte stranicu sa uputstvom `?options`. Uvek možemo da konvertujemo faktore kasnije kada je to potrebno. Ako do sada niste primetili, koristio sam novi operator `$` - operator ekstrahovanja. Ovo je najpopularniji način za ekstrahovanje atributa (ili kolona) iz okvira podataka. Za ekstrahovanje mogu se koristiti i dvostruke kvadratne zagrade (`[[` i `]]`).

Oba načina su dodatak opciji kanonskog indeksiranja matrice. Prema tome, sledeća tri iskaza su u ovom kontekstu funkcionalno identična:

```
> favs$flavor
[1] "pistachio"           "mint chocolate chip" "vanilla"
[4] "chocolate"           "strawberry"           "neopolitan"
> favs[["flavor"]]
[1] "pistachio"           "mint chocolate chip" "vanilla"
[4] "chocolate"           "strawberry"           "neopolitan"
> favs[,1]
[1] "pistachio"           "mint chocolate chip" "vanilla"
[4] "chocolate"           "strawberry"   "neopolitan"
```

Obratite pažnju da je R, osim [1], uz rezultat sada ispisao još jedan broj u kvadratnim zagradama. To označava da je chocolate četvrti elemenat vektora koji je vraćen ekstrahovanjem.

Možete da koristite funkciju `names()` da biste prikazali listu kolona koje su dostupne u okviru podataka. Možete čak i da ponovo dodelite nazive pomoću ove iste funkcije:

```
> names(favs)
[1] "flavor" "number"
> names(favs)[1] <- "flav"
> names(favs)
[1] "flav"   "number"
```

Na kraju, možete da vidite kompaktni prikaz strukture okvira podataka pomoću funkcije `str()`:

```
> str(favs)
'data.frame': 6 obs. of   2 variables:
 $ flav  : chr  "pistachio" "mint chocolate chip" "vanilla"
 "chocolate" ...
 $ number: num  6 7 5 10 2 4
```

Zapravo, možete da primenite ovu funkciju na svaku R strukturu. Svojstvo koje menja ponašanje funkcija na osnovu tipa unosa naziva se **polimorfizam**.

UPOTREBA PAKETA

Iako su osnovne R funkcije robusne, efikasne i brojne, nismo ni na koji način ograničeni na njih! Dodatne funkcije su dostupne u vidu paketa. U stvari, ono što čini R značajnom statističkom platformom je zadivljujuće veliki broj dostupnih paketa (više od 10.000 u trenutku pisanja ove knjige). R ekosistem je fenomenalan!

Većina ovih neograničenih paketa se nalazi na mreži **Comprehensive R Archive Network (CRAN)**. Ta mreža je primarno spremište za pakete koje su napravili korisnici.

Jedan od paketa koji ćemo odmah početi da koristimo je `ggplot2`. Ovaj paket je sistem za crtanje grafikona u R-u. R baza ima sofisticirane i napredne mehanizme za grafički prikaz podataka, ali mnogi smatraju da je `ggplot2` konzistentniji i lakši za upotrebu. Osim toga, grafikoni su često estetski prijemčiviji.

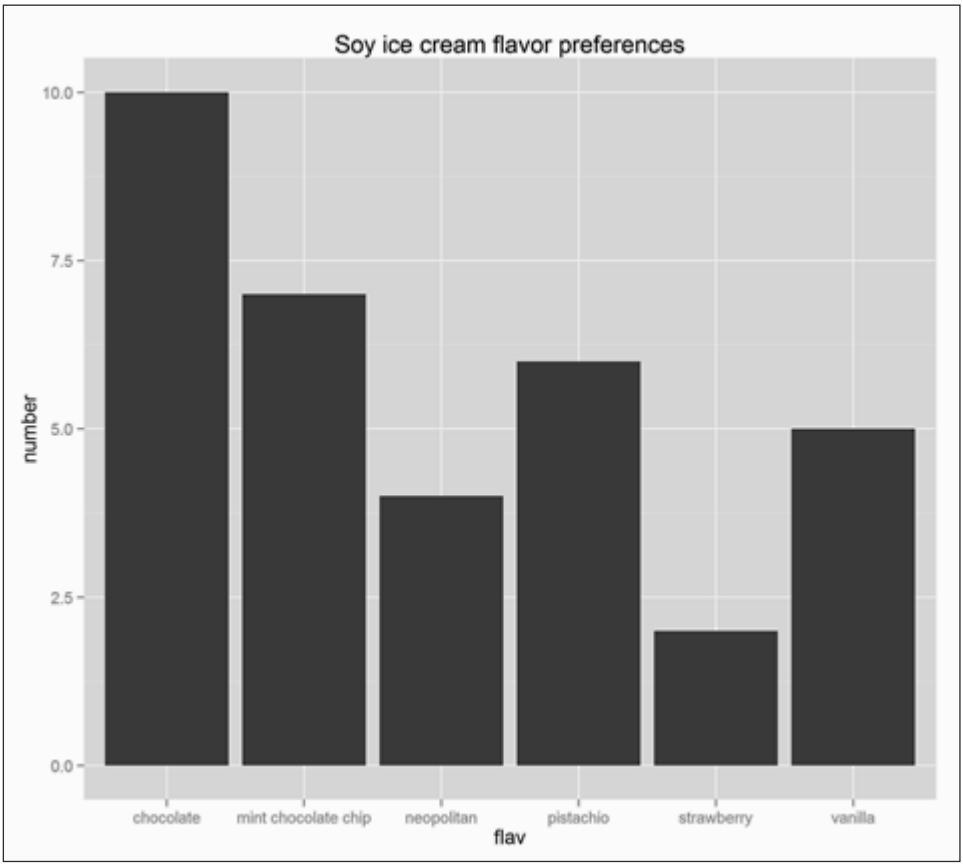
Sada ćemo instalirati ovaj paket:

```
> # downloads and installs from CRAN
> install.packages("ggplot2")
```

Pošto smo preuzeli paket, učitaćemo ga u R sesiju i testirati, tako što ćemo grafički prikazati podatke iz poslednjeg odeljka:

```
> library(ggplot2)
> ggplot(favs, aes(x=flav, y=number)) +
+    geom_bar(stat="identity") +
+    ggtitle("Soy ice cream flavor preferences")
```

Grafikon koji je generisan pomoću prethodnog koda izgleda ovako:



Slika 1.1 Omiljeni ukusi sladoleda od sojinog mleka

Niste u pravu - najbolji je sladoled od mente sa komadićima čokolade!

Još uvek ne morate da razmišljate o sintaksi funkcije `ggplot`, jer ćemo je blagovremeno razmotriti.

Instaliraćemo još neke pakete u ostatku knjige. U međuvremenu, ako želite da eksperimentišete sa još nekoliko paketa, možete da instalirate pakete `gdata` i `foreign`, koji omogućavaju da u R-u prvo uvezete Excel unakrsne tabele, a zatim SPSS datoteke podataka.

VEŽBE

Možete da uradite sledeće vežbe da biste bolje shvatili koncepte koje ste naučili iz ovog poglavlja:

- Napišite funkciju pod nazivom `simon.says`, koja prihvata znakovni niz i vraća taj niz ispisan velikim slovima nakon što je niz *Simony says:* upisan na početku datoteke.
- Napišite funkciju koja prihvata dve matrice kao argumente i vraća logičku vrednost pomoću koje se određuje da li se na matricama može izvršiti matrično množenje.
- Pronađite besplatan skup podataka na Vebu, preuzmite ga i učitajte u R. Istražite strukturu tog skupa podataka.
- Razmislite kako se na Hester Prin odrazilo kada joj je ćerka ukrasila skerletno slovo cvećem u romanu „Skerletno slovo“. U kojoj meri je ovo pokazatelj da je skerletno slovo postalo pozitivni deo Hesterinog identiteta? Potkrepite vašu tezu pomoću odlomaka iz tog romana.

REZIME

U ovoj knjizi ste upoznali R, najbolju analitičku platformu na svetu. Izgradili smo osnovu, a sada ćemo dodatno istražiti R na osnovu znanja koje ste stekli u ovom poglavlju. Do sada ste dobro upoznali osnove R-a (koje su, paradoksalno, najteži deo). Naučili ste:

- da koristite R kao "dobri stari" kalkulator za aritmetičke operacije
- da kreirate vektore, da ih koristite i da ekspresivno izrađujete njihove podskupove
- da učitate podatke sa diska
- da instalirate pakete

Niste završili učenje R-a. U stvari, predstavili smo samo osnove. Treba da naučite još mnogo štošta, pa možemo da nastavimo dalje. Uđimo u svet statistike!