R u podatkovnoj znanosti

Juzbašić, Lara

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:687041

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ODJEL ZA MATEMATIKU
Sveučilišni diplomski studij matematike
Smjer: Financijska matematika i statistika

Lara Juzbašić

R u podatkovnoj znanosti

Diplomski rad

Osijek, 2022.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ODJEL ZA MATEMATIKU
Sveučilišni diplomski studij matematike
Smjer: Financijska matematika i statistika

Lara Juzbašić

R u podatkovnoj znanosti

Diplomski rad

Mentor:
izv. prof. dr. sc. Danijel Grahovac

Osijek, 2022.

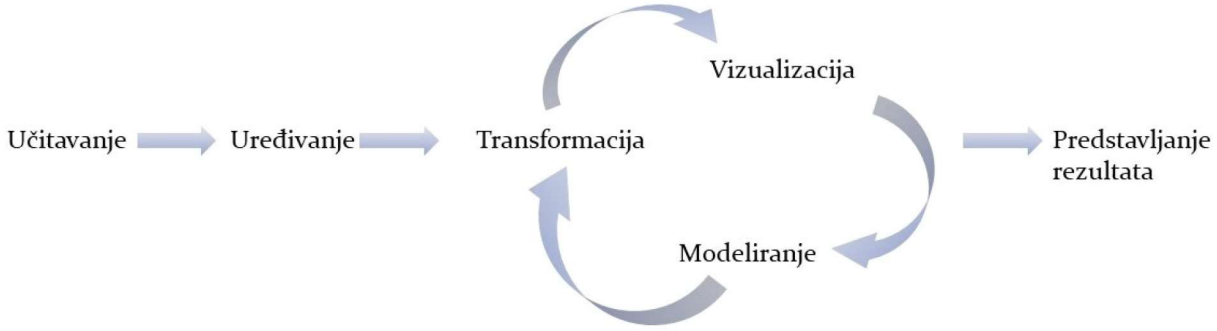
Sadržaj

Uvod

Podatkovna znanost je znanstvena disciplina koja se bavi pronalaženjem obrazaca i iščitavanjem smislenih informacija iz velikih količina podataka. Jedan od vodećih programskih jezika u podatkovnoj znanosti je R. R, osim što je programski jezik, ujedno je i okruženje namijenjeno upravljanju podacima, analizi te vizualizaciji podataka. Izveden je iz jezika S, kojeg su razvili John Chambers i njegovi suradnici u Bell laboratorijima. Ono što uporabu jezika R čini sve lakšom i jednostavnijom je nadogradnja paketima. Paketi su skupovi funkcija, dokumentacije koja opisuje kako ih koristiti i baza podataka. Kolekcija paketa koja je posebno korisna u podatkovnoj znanosti je takozvani tidyverse, čiji je tvorac Hadley Wickham. Članovi kolekcije tidyverse bit će predstavljeni u narednim poglavljima.

Analizu podataka započinjemo učitavanjem samih podataka u R. Većinu podataka nije moguće odmah analizirati, već ih je potrebno izmijeniti tako da budu u obliku pogodnom za analizu, to jest „urediti". Zatim kombiniramo vizualizaciju i transformaciju u istraživačkoj analizi podataka. Podatke transformiramo primjerice kada želimo usredotočiti istraživanje na određeni podskup skupa podataka, stvoriti novu varijablu pomoću postojećih ili izmijeniti redoslijed podataka. Vizualizaciju u istraživačkoj analizi koristimo kako bismo došli do novih uvida, ali često iz nje proizlaze i nova pitanja o podacima. Potom gradimo modele[1] koji daju odgovore na pitanja, te u konačnici predstavljamo rezultate analize drugima, pri čemu ponovno koristimo vizualizaciju podataka. Nakon izgradnje modela javljaju se nova pitanja pa je proces nerijetko cikličan. Shematski prikaz slijeda zadataka podatkovne analize možemo vidjeti na Slici 1.

U ovom radu pažnju ćemo usmjeriti na pripremu podataka te istraživačku analizu, a dotaknut ćemo se i prezentiranja rezultata analize pomoću vizualizacije podataka. U svakom dijelu analize koristit ćemo alate paketa iz kolekcije tidyverse.



Slika 1: Koraci u procesu analize podataka (inspirirano shematskim prikazom iz [6], str. 1)

[1] O modeliranju čitatelj može pročitati u [4] i [8] (poglavlje 5).

1 Tidyverse

Sve korake procesa analize podataka navedene u uvodu olakšava spomenuta skupina paketa tidyverse. Paketi ove kolekcije pružaju napredne funkcije za podatkovnu analizu. Funkcije su osmišljene na način da imaju zajedničku strukturu, što ih čini jednostavnim za korištenje. U Tablici 1 navedeni su paketi sadržani u ovoj kolekciji koje ćemo kroz rad upoznati te njihova namjena. O ostalim tidyverse paketima čitatelj može pročitati u [2].

paket	**namjena**
readr	Učitavanje podataka
tidyr	Uređivanje podataka
dplyr	Upravljanje podacima
ggplot2	Vizualizacija podataka

Tablica 1: Dio paketa kolekcije tidyverse

Temeljni objekt[2] kojim upravljamo pomoću tidyverse paketa je podatkovni okvir. Podatkovni okvir je struktura podataka konstruirana kao lista vektora jednake duljine. Svaki element liste predstavlja jedan stupac, a duljina elemenata određuje broj redaka. Primijetimo da stupci mogu sadržavati elemente različitog tipa. Tidyverse-ova verzija podatkovnog okvira naziva se tibble, o čemu će biti riječ u nastavku poglavlja.

Naziv paketa u sebi sadrži riječ „uredno" (engl. *tidy*), što upućuje na to da funkcije paketa pretpostavljaju urednost podataka s kojima rade. Urednost podrazumijeva to da su podaci organizirani prema sljedećem skupu pravila[3]:

- Svaka varijabla mora imati svoj stupac.
- Svaka opservacija mora imati svoj red.
- Svaka vrijednost mora imati svoju ćeliju.

Međutim, čim zadovoljimo dva pravila iz skupa, treće će također biti zadovoljeno pa dolazimo do kraćeg skupa uputa:

- Svaku bazu podataka stavite u jedan podatkovni okvir.
- Svaku varijablu stavite u jedan stupac.

[2]Za ostale objekte u R-u vidi [3], poglavlje 4.
[3]Pravila za uredne podatke uveo je Hadley Wickham u [6].

Pojasnimo na sljedećem primjeru prednosti korištenja tidyverse funkcija umjesto klasičnih R naredbi, pri čemu ćemo koristiti bazu podataka *airquality* ugrađenu u R. Baza sadrži podatke dnevnih mjerenja kvalitete zraka u New Yorku, od svibnja do rujna 1973. Pretpostavimo kako želimo saznati na koje dane je temperatura zraka bila veća od prosječne temperature u tom razdoblju. Koristeći osnovnu sintaksu R-a[4], možemo to saznati na dva načina (ispisat ćemo samo prvih 6 elemenata rezultata funkcijom head()):

```
# 1. način
prosjecna_temp <- mean(airquality$Temp)
veca_od_prosjecne <- airquality[airquality$Temp > prosjecna_temp, ]
veca_od_prosjecne[c("Month", "Day")]
head(dani)
# 2. način
head(airquality[airquality$Temp > mean(airquality$Temp), ][c("Month",
                                                            "Day")])
#    Month Day
# 29     5  29
# 30     5  30
# 32     6   1
# 35     6   4
# 36     6   5
# 37     6   6
```

Prvi način je postupno pridruživanje međurezultata varijablama, što rezultira mnoštvom novih varijabli koje zapravo ne koristimo u daljnjoj analizi. Drugi način je pisanje naredbe u jednom retku ugnježđivanjem funkcija. Pisanje koda na ovaj način je neintuitivno, jer korake naredbe ne pišemo redom kojim se izvršavaju (vanjska funkcija kojom započinjemo naredbu izvršava se posljednja).

Pokušajmo sada iste podatke dobiti pomoću tidyverse funkcija. Za početak instalirajmo i učitajmo tidyverse u R sljedećim naredbama:

```
install.packages("tidyverse")
library(tidyverse)
```

Instalaciju je potrebno izvršiti samo jednom, a učitavanje svaki put kada se započne nova sesija.

Sljedeći zapis ekvivalentan je prethodnim zapisima pomoću klasičnih R funkcija:

```
airquality %>%
  mutate(prosjecna_temp = mean(Temp)) %>%
```

[4]Osnovnu sintaksu R-a čitatelj može proučiti u [5].

```
  filter(Temp > prosjecna_temp) %>%
  select(Month, Day) %>%
  head()
#   Month Day
# 1     5  29
# 2     5  30
# 3     6   1
# 4     6   4
# 5     6   5
# 6     6   6
```

Naredbu započinjemo s podatkovnim okvirom „airquality", nakon čega slijedi izračun prosječne temperature, zatim odabir onih dnevnih mjerenja kod kojih je temperatura bila veća od prosječne, a potom odabir stupaca s informacijama o danu i mjesecu. Prijelaz između pojedinih koraka omogućen je operatorom %>%. Dakle, funkcije zapisujemo onim redom kojim i djeluju nad podatkovnim okvirom pa ne moramo razmišljati unaprijed kao u slučaju s ugniježđenim funkcijama. Osim toga, zapis je pregledniji pa ga je lakše razumjeti pri čitanju. Također, ovakvim zapisom minimiziramo stvaranje novih varijabli, što mu stvara prednost i pred prvim predstavljenim načinom zapisa.

U nastavku su predstavljene dvije novine u sintaksi R-a koje dolaze s ovim paketom, a koje ćemo koristiti kroz cijeli rad. Jedna od njih je operator %>% kojeg smo koristili prethodnom primjeru.

1.1 Operator cjevovoda

Prije nego li krenemo s uređivanjem podataka, predstavimo alat koji se koristi zajedno s paketima kolekcije tidyverse, takozvani operator cjevovoda, %>% (engl. *the pipe*), koji omogućuje kombiniranje višestrukih operacija uređivanja podataka tako što ulančano poziva funkcije. Smisao ovog operatora je učiniti programski kod preglednijim i samim tim lakšim za razumjeti.

Pojasnimo djelovanje ovog operatora. Neka je x podatkovni okvir nad kojim želimo izvršiti niz operacija koristeći funkcije f, g i h, tim redoslijedom. Bez korištenja cjevovoda, to bi izgledalo ovako:

```
h(g(f(x)))
```

Isti rezultat dobijemo korištenjem cjevovoda napišemo li sljedeće:

```
x %>%
  f() %>%
  g() %>%
  h()
```

Sada je jasno kako korištenje ovog operatora poboljšava čitljivost koda. Želimo li rezultat pohraniti u varijablu, možemo to jednostavno učiniti operatorom pridruživanja.

```
rezultat <- x %>%
  f() %>%
  g() %>%
  h()
```

Slično, imamo li funkciju više varijabli, sljedeća dva zapisa su ekvivalentna.

```
f(x, y)


x %>%
  f(y)
```

Upotrijebimo sada ovaj operator na stvarnom primjeru, pri čemu ćemo ponovno koristiti bazu podataka *airquality*. Želimo saznati podatke o kvaliteti zraka prvog dana u mjesecu kolovozu pomoću funkcije subset(). Bez cjevovoda to bi mogli učiniti na dva načina, ugnježđivanjem ili postupnim pridruživanjem.

```
# 1. način
subset(subset(airquality, Month == 8), Day == 1)
# 2. način
mjesec <- subset(airquality, Month == 8)
(dan <- subset(mjesec, Day == 1))
#    Ozone Solar.R Wind Temp Month Day
# 93    39      83  6.9   81     8   1
```

Pronađimo sada podatke koji nas zanimaju pomoću cjevovoda.

```
airquality %>%
  subset(Month == 8) %>%
  subset(Day == 1)
#    Ozone Solar.R Wind Temp Month Day
# 93    39      83  6.9   81     8   1
```

Operator %>% možemo čitati kao „zatim”. Tako bismo ovaj primjer pročitali na sljedeći način: dohvati bazu podataka *airquality*, zatim odaberi redove koji odgovaraju kolovozu, a zatim među njima odaberi red koji odgovara prvom danu u mjesecu.

1.2 Tibble kao nova struktura podataka

Prvi iz kolekcije paketa tidyverse kojeg ćemo proučiti je paket tibble. Istoimena struktura podataka tibble slična je tradicionalnom podatkovnom okviru, s nešto izmijenjenim svojstvima. Dizajnirana je na način da su nedostaci podatkovnog okvira ispravljeni, a svojstva koja su se pokazala učinkovita su zadržana. Tibble se može stvoriti iz pojedinačnih vektora uz pomoć funkcije tibble():

```
tibble(
  x = 1:3,
  y = 3,
  z = 2 * x + y
)
# # A tibble: 3 x 3
#       x     y     z
#   <int> <dbl> <dbl>
# 1     1     3     5
# 2     2     3     7
# 3     3     3     9
```

Primijetimo da tibble automatski pretvara varijablu y u vektor dimenzije 3, što je poznato kao svojstvo „recikliranja", te da je vektor z napisan pomoću prethodno napisanih vektora, x i y. Druga funkcija kojom možemo stvoriti tibble je tribble(), pogodna za ručni unos podataka. Tu, naime, podatke unosimo red po red, to jest transponirano u odnosu na unos u funkciju tibble(). Nazivima stupaca pridružujemo znak „∼" slijeva.

```
tribble(
  ~x, ~y, ~z,
   1, "a", 1.2,
   2, "b", 2.3
)
# # A tibble: 2 x 3
#       x y         z
#   <dbl> <chr> <dbl>
# 1     1 a       1.2
# 2     2 b       2.3
```

Tradicionalni podatkovni okvir lako se pretvara u tibble uz pomoć funkcije as_tibble().

```
(airquality_tibble <- as_tibble(airquality))
# # A tibble: 153 x 6
#   Ozone Solar.R  Wind  Temp Month   Day
```

```
#   <int>   <int> <dbl> <int> <int> <int>
# 1    41     190   7.4    67     5     1
# 2    36     118   8      72     5     2
# 3    12     149  12.6    74     5     3
# 4    18     313  11.5    62     5     4
# 5    NA      NA  14.3    56     5     5
# 6    28      NA  14.9    66     5     6
# 7    23     299   8.6    65     5     7
# 8    19      99  13.8    59     5     8
# 9     8      19  20.1    61     5     9
#10    NA     194   8.6    69     5    10
# # ... with 143 more rows
```

Iz prethodnog primjera vidimo da se pri ispisivanju strukture tibble prikazuje samo prvih deset redaka. Na taj način se sprječava pretrpavanje konzole kod ispisa velikih podatkovnih okvira. Također možemo vidjeti da je prikazan tip svake varijable ispod njenog naziva. Ukoliko ipak želimo ispisati veći (ili manji) broj redaka, koristimo funkciju print():

```
airquality_tibble %>%
  print(n = 12, width = Inf)
# # A tibble: 153 x 6
#   Ozone Solar.R  Wind  Temp Month   Day
#   <int>   <int> <dbl> <int> <int> <int>
# 1    41     190   7.4    67     5     1
# 2    36     118   8      72     5     2
# 3    12     149  12.6    74     5     3
# 4    18     313  11.5    62     5     4
# 5    NA      NA  14.3    56     5     5
# 6    28      NA  14.9    66     5     6
# 7    23     299   8.6    65     5     7
# 8    19      99  13.8    59     5     8
# 9     8      19  20.1    61     5     9
#10    NA     194   8.6    69     5    10
#11     7      NA   6.9    74     5    11
#12    16     256   9.7    69     5    12
# # ... with 141 more rows
```

Ispis svih stupaca specificirali smo navođenjem *width = Inf* unutar funkcije, a postavljanjem vrijednosti parametra n na 12 odabrali smo ispisati prvih dvanaest redaka baze.

Za razliku od tradicionalnog podatkovnog okvira, tibble ne prilagođava nazive varijabli:

```
data.frame("ime varijable" = 1) %>%
  names
# [1] "ime.varijable"
tibble("ime varijable" = 1) %>%
  names
# [1] "ime varijable"
```

Osvrnimo se još na svojstvo „recikliranja". Kada stvaramo tibble, samo varijable dimenzije 1 će se automatski „produžiti" do dimenzije ostalih elemenata ponavljanjem (recikliranjem) unesene vrijednosti. Broj redova strukture tibble određen je brojem redova prvog stupca čija je dimenzija veća od 1, te će svaki idući stupac različitog broja redova, a većeg od 1, dovesti do greške u konstrukciji.

```
tibble(
  x = 1,
  y = 1:4,
  z = 1:2
)
# Error:
# ! Tibble columns must have compatible sizes.
# * Size 4: Existing data.
# * Size 2: Column `z`.
# i Only values of size one are recycled.
```

2 Priprema podataka

U ovom dijelu rada proći ćemo kroz prvi dio procesa analize podataka, pripremu podataka. Priprema podataka započinje njihovim učitavanjem, nakon čega ih dovodimo do urednog formata kako bismo mogli nastaviti analizu pomoću tidyverse paketa.

2.1 Učitavanje podataka

Paket readr, također sadržan u kolekciji paketa tidyverse, omogućava brzo učitavanje tabličnih tekstualnih datoteka pretvarajući ih odmah u podatkovni okvir tibble. Proučit ćemo funkciju za učitavanje CSV datoteka[5], read_csv(), čija se sintaksa može primijeniti i na funkcije za učitavanje datoteka ostalih tipova. Učitajmo bazu podataka koja će nam koristiti u nastavku rada, *nobel_final.csv*[6].

```
nobel <- read_csv('nobel_final.csv')
# Rows: 923 Columns: 14
# -- Column specification --------------------------------------
# Delimiter: ","
# chr (10): firstname, surname, born_country_code, died_co...
# dbl  (4): year, share, age, age_get_prize
```

Uočavamo da se baza sastoji od 14 stupaca od kojih su 10 tipa character i 4 tipa double. Zapravo se radi o tipovima koje je funkcija pretpostavila s obzirom na podatke. Kako bismo dobili čitav popis imena stupaca i njihovih pretpostavljenih tipova te provjerili eventualne greške, koristimo funkciju spec():

```
spec(nobel)
# cols(
#   firstname = col_character(),
#   surname = col_character(),
#   born_country_code = col_character(),
#   died_country_code = col_character(),
#   gender = col_character(),
#   year = col_double(),
#   category = col_character(),
#   share = col_double(),
#   name_of_university = col_character(),
#   city_of_university = col_character(),
#   country_of_university = col_character(),
```

[5]CSV – comma-separated values
[6]Baza podataka preuzeta je s web stranice GitHub [9]

```
#   born_month = col_character(),
#   age = col_double(),
#   age_get_prize = col_double()
# )
```

Automatski određeni tipovi varijabli *gender*, *year*, *category*, *share* i *born_month* nam ne odgovaraju i želimo ih promijeniti. Navođenjem parametra col_types možemo specificirati tip pojedinog stupca:

```
nobel <- read_csv(
  "nobel_final.csv",
  col_types = cols(
    firstname = col_character(),
    surname = col_character(),
    born_country_code = col_character(),
    died_country_code = col_character(),
    gender = col_factor(),
    year = col_integer(),
    category =  col_factor(),
    share = col_integer(),
    name_of_university = col_character(),
    city_of_university = col_character(),
    country_of_university = col_character(),
    born_month = col_factor(),
    age = col_double(),
    age_get_prize = col_double()
    )
)
```

Funkcija read_csv() nam omogućava učitavanje podataka i direktnim ubacivanjem teksta:

```
read_csv("x,y,z
  x1,y1,1
  x2,y2,2
  x3,y3,3")
# # A tibble: 3 x 3
#   x     y         z
#   <chr> <chr> <dbl>
# 1 x1    y1        1
# 2 x2    y2        2
# 3 x3    y3        3
```

Iz ovog primjera vidimo da funkciji prvi red predstavlja nazive stupaca. Ukoliko baza ne sadrži nazive stupaca, možemo ih samostalno dodijeliti dodavanjem parametra col_names. Ako želimo promijeniti postojeće nazive, također je potrebno ukloniti prvi red parametrom skip:

```
read_csv("x,y,z
  x1,y1,1
  x2,y2,2
  x3,y3,3", skip = 1,
            col_names = c("stupac1", "stupac2", "stupac3"))
# # A tibble: 3 x 3
# stupac1 stupac2 stupac3
#   <chr>   <chr>     <dbl>
# 1 x1      y1            1
# 2 x2      y2            2
# 3 x3      y3            3
```

Još jedan od parametara za prilagodbu je parametar na koji služi za navođenje znakova koji predstavljaju NA vrijednosti, to jest nedostajuće vrijednosti u bazi:

```
read_csv("x,y,z
  x1,.,1
  x2,y2,.
  _,y3,3", na = c(".","_"))
# # A tibble: 3 x 3
#   x     y         z
#   <chr> <chr> <dbl>
# 1 x1    NA        1
# 2 x2    y2       NA
# 3 NA    y3        3
```

2.2 Uređivanje podataka

Nakon učitavanja podataka, potrebno je provjeriti jesu li podaci u bazi uredni. Urednost podataka važna je radi lakšeg obrađivanja i analiziranja funkcijama iz paketa kolekcije tidyverse. Većina stvarnih baza ipak ne sadrži uredne podatke te nam tidyverse i tu priskače u pomoć, ovaj put s paketom tidyr.

2.2.1 Pivotiranje

Promotrimo sljedeće podatke i provjerimo jesu li uredni.

```
prosjecne_ocjene
# # A tibble: 6 x 5
#   razred odjeljenje matematika hrvatski engleski
#   <chr>  <chr>           <dbl>    <dbl>    <dbl>
# 1 1      a                3.55     3.7      3.72
# 2 1      b                3.4      3.81     4.22
# 3 1      c                3.52     4.1      3.89
# 4 2      a                3.52     3.9      4
# 5 2      b                3.33     3.6      3.82
# 6 2      c                3.5      3.55     3.7
```

Jedna opservacija ove baze trebala bi predstavljati prosječnu ocjenu određenog razreda iz određenog predmeta. Primjećujemo da se prosječna ocjena nalazi u trima stupcima, čija imena „matematika", „hrvatski" i „engleski" predstavljaju vrijednosti varijable *predmet*. Dakle, podaci nisu uredni jer je jedna varijabla raspoređena u više stupaca. Kako bismo podatke doveli do urednog formata, koristimo funkciju pivot_longer(), koja će će tablicu „suziti" smanjenjem broja stupaca i „produljiti" povećanjem broja redaka:

```
prosjecne_ocjene <- prosjecne_ocjene %>%
  pivot_longer(c(matematika, hrvatski, engleski),
               names_to = "predmet", values_to = "ocjena")
# # A tibble: 18 x 4
#   razred odjeljenje predmet    ocjena
#   <chr>  <chr>      <chr>       <dbl>
# 1 1      a          matematika   3.55
# 2 1      a          hrvatski     3.7
# 3 1      a          engleski     3.72
# 4 1      b          matematika   3.4
# 5 1      b          hrvatski     3.81
# 6 1      b          engleski     4.22
# 7 1      c          matematika   3.52
# 8 1      c          hrvatski     4.1
# 9 1      c          engleski     3.89
#10 2      a          matematika   3.52
#11 2      a          hrvatski     3.9
#12 2      a          engleski     4
#13 2      b          matematika   3.33
```

```
#14 2      b          hrvatski     3.6
#15 2      b          engleski     3.82
#16 2      c          matematika   3.5
#17 2      c          hrvatski     3.55
#18 2      c          engleski     3.7
```

Vidimo kako smo tri problematična stupca pretvorili u novi par varijabli tako što smo funkciji pivot_longer() proslijedili tri parametra:

c(matematika, hrvatski, engleski)	vektor stupaca čija su imena vrijednosti, a ne varijable
names_to = "predmet"	naziv nove varijable u koju se premještaju nazivi stupaca
values_to = "ocjena"	naziv nove varijable u koju se premještaju vrijednosti stupaca

Prijeđimo sada na sljedeće podatke i provjerimo jesu li oni uredni.

```
vladanje
# # A tibble: 12 x 3
#   ucenik       pedagoska_mjera  broj
#   <chr>        <chr>           <dbl>
# 1 Ivan Anić    ukor                2
# 2 Ivan Anić    opomena            20
# 3 Ivan Anić    strogi ukor         1
# 4 Petar Marić  ukor                1
# 5 Petar Marić  opomena            12
# 6 Petar Marić  strogi ukor         0
# 7 Ana Ivić     ukor                0
# 8 Ana Ivić     opomena             5
# 9 Ana Ivić     strogi ukor         0
#10 Marija Perić opomena             1
#11 Marija Perić ukor                0
#12 Marija Perić strogi ukor         0
```

Opservacija ove baze trebala bi biti vladanje određenog učenika, to jest broj pojedinih pedagoških mjera koje je dobio. Svaka je opservacija raspoređena u dva retka, stoga podaci nisu uredni. U ovom slučaju podatke uređujemo funkcijom pivot_wider() koja, suprotno funkciji pivot_longer(), tablicu čini „kraćom" smanjujući broj redaka i „širom" povećavajući broj stupaca:

```
vladanje <- vladanje %>%
  pivot_wider(names_from = pedagoska_mjera, values_from = broj)
# # A tibble: 4 x 4
#   ucenik        ukor opomena `strogi ukor`
#   <chr>        <dbl>   <dbl>         <dbl>
# 1 Ivan Anić        2      20             1
# 2 Petar Marić      1      12             0
# 3 Ana Ivić         0       5             0
# 4 Marija Perić     0       1             0
```

Kategorije varijable *pedagoska_mjera* postale su nazivi novih stupaca s odgovarajućim vrijednostima iz prijašnjeg stupca *broj*. U funkciji pivot_longer() navedena su sljedeća dva parametra:

names_from = *pedagoska_mjera*	stupac iz kojeg se preuzimaju nazivi varijabli
values_from = *broj*	stupac iz kojeg se preuzimaju vrijednosti

2.2.2 Funkcije *separate* i *unite*

Podatkovni okvir „ucionice" sadrži informacije o razmještaju razrednih odjela po učionicama.

```
ucionice
# # A tibble: 6 x 2
#   razredni_odjel prostor
#   <chr>          <chr>
# 1 1a             prizemlje_3
# 2 1b             kat_2
# 3 1c             prizemlje_5
# 4 2a             prizemlje_1
# 5 2b             kat_1
# 6 2c             kat_4
```

Varijabla *prostor* sadrži dvije informacije: etažu na kojoj se nalazi učionica u koju je smješten određeni razred i broj učionice. Dakle, u jednom stupcu nalaze se dvije varijable. Još jedna funkcija paketa tidyr, funkcija separate(), omogućava razdvajanje stupca *prostor* na dvije varijable:

```
(ucionice2 <- ucionice %>%
  separate(col = prostor, into = c("etaza", "br_ucionice")))
# # A tibble: 6 x 3
```

```
#   razredni_odjel etaza     br_ucionice
#   <chr>          <chr>     <chr>
# 1 1a             prizemlje 3
# 2 1b             kat       2
# 3 1c             prizemlje 5
# 4 2a             prizemlje 1
# 5 2b             kat       1
# 6 2c             kat       4
```

Pojasnimo proslijeđene parametre:

col = prostor	naziv stupca koji rastavljamo
into = c("etaza", "br_ucionice"))	vektor naziva novih stupaca

Primijetimo kako nismo naveli znak koji je separator vrijednosti. Razlog tomu je taj što funkcija separate() rastavlja vrijednosti stupca gdje god se pojavi znak različit od slova i brojeva (ovdje je to znak „_"). Ukoliko ipak želimo sami odrediti separator, argumentu sep možemo proslijediti znak. Preostaje nam slučaj kada želimo razdvojiti vrijednosti u kojima ne postoji separator. Primjerice, htjeli bismo rastaviti stupac *razredno_odjeljenje* na dvije varijable *razred* i *odjeljenje*. To ćemo učiniti prosljeđivanjem pozicije znaka nakon kojeg želimo rastaviti vrijednosti argumentu sep:

```
(ucionice3 <- ucionice2 %>%
  separate(col = razredni_odjel, into = c("razred", "odjeljenje"),
           sep = 1))
# # A tibble: 6 x 4
#   razred odjeljenje etaza     br_ucionice
#   <chr>  <chr>      <chr>     <chr>
# 1 1      a          prizemlje 3
# 2 1      b          kat       2
# 3 1      c          prizemlje 5
# 4 2      a          prizemlje 1
# 5 2      b          kat       1
# 6 2      c          kat       4
```

Ukoliko vrijednosti stupca želimo rastaviti na više dijelova, možemo proslijediti vektor pozicija. Vrijednosti novostvorenih stupaca možemo ponovno spojiti u jedan stupac inverznom funkcijom funkciji separate(), unite():

```r
ucionice3 %>%
  unite(razredni_odjel, razred, odjeljenje, sep = "")
# # A tibble: 6 x 3
#   razredni_odjel etaza     br_ucionice
#   <chr>          <chr>     <chr>
# 1 1a             prizemlje 3
# 2 1b             kat       2
# 3 1c             prizemlje 5
# 4 2a             prizemlje 1
# 5 2b             kat       1
# 6 2c             kat       4
```

Bez navođenja argumenta sep, funkcija unite() između vrijednosti stupaca koje spaja stavlja znak „_”. Kako ovdje želimo spojiti vrijednosti bez separatora, koristimo *sep = ""*.

3 Upravljanje podatkovnim okvirima

Kada se uvjerimo da su podaci u urednom formatu, možemo upravljati njima pomoću funkcija iz paketa dplyr kako bismo ih pretvorili u željeni oblik pogodan za vizualizaciju. Sve što nam dpylr omogućava sa svojim funkcijama možemo učiniti i s osnovnom sintaksom R-a. Međutim, ona je često neintuitivna i nepraktična. Kroz ovo poglavlje vidjet ćemo kako dplyr pojednostavljuje rad s podatkovnim okvirima.

3.1 Baza podataka

Predstavimo bazu podataka *nobel_final.csv* koju smo prethodno učitali te kojom ćemo se baviti u ovom i narednim poglavljima. Baza sadrži 14 varijabli i 923 opservacija koje predstavljaju dobitnike Nobelove nagrade od 1901. do 2019. godine. Opis varijabli nalazi se u Tablici 2.

Ime varijable	**Opis varijable**
firstname	Ime dobitnika
surname	Prezime dobitnika
born_country_code	Oznaka zemlje u kojoj je dobitnik rođen
died_country_code	Oznaka zemlje u kojoj je dobitnik preminuo
gender	Spol dobitnika nagrade
year	Godina u kojoj je nagrada dodijeljena
category	Kategorija u kojoj je dodijeljena nagrada
share	Broj osoba koje su dijelile nagradu
name_of_university	Ime sveučilišta koje je dobitnik pohađao
city_of_university	Grad sveučilišta koje je dobitnik pohađao
country_of_university	Zemlja sveučilišta koje je dobitnik pohađao
born_month	Mjesec rođenja dobitnika
age	Dob dobitnika
age_get_prize	Dob dobitnika prilikom dodjele nagrade

Tablica 2: Opis varijabli

3.2 Osnovne funkcije paketa dplyr

Paket dplyr sadrži pet osnovnih funkcija za transformiranje podataka: filter(), arrange(), select(), mutate() i summarise().

3.2.1 Odabir podskupa varijabli

Za odabir stupaca podatkovnog okvira na koje se želimo fokusirati koristimo funkciju select(), koja uvelike olakšava rad s velikim bazama podataka. Podskup stupaca možemo odabrati na nekoliko načina:

- Navođenjem imena stupaca. Primjerice, pretpostavimo da nas zanimaju samo podaci o sveučilištima koje su dobitnici pohađali, stoga odabiremo stupce koji sadrže informacije o imenu sveučilišta te gradu i zemlji u kojoj se nalazi.

```
nobel %>%
  select(name_of_university, city_of_university,
         country_of_university)
```

- Navođenjem pozicija stupaca. Pozicije stupaca koje smo odabrali u prethodnom primjeru su redom 9, 10 i 11.

```
nobel %>%
  select(9,10,11)
```

- Pomoću dvotočke „:". Navođenjem dva stupca s dvotočkom među njima, osim ta dva stupca odabiremo i sve stupce između njih.

```
nobel %>%
  select(name_of_university:country_of_university)
```

- Korištenjem pomoćnih funkcija. starts_with(), ends_with(), contains() i matches() neke su od funkcija koje se mogu kombinirati unutar select() pri odabiru stupaca. Želimo li odabrati sve stupce čija imena završavaju riječju „university", koristit ćemo ends_with():

```
nobel %>%
  select(ends_with("university"))
```

U sva četiri slučaja dobije se isti rezultat:

```
# # A tibble: 923 x 3
#   name_of_university     city_of_universi~ country_of_univ~
#   <chr>                  <chr>             <chr>
# 1 Munich University      Munich            Germany
```

```
# 2 Leiden University      Leiden            the Netherlands
# 3 Amsterdam University   Amsterdam         the Netherlands
# 4 École Polytechnique    Paris             France
# 5 École municipale de p~ Paris             France
# 6 NA                     NA                NA
# 7 Sorbonne University    Paris             France
# 8 Royal Institution of ~ London            United Kingdom
# 9 Kiel University        Kiel              Germany
#10 University of Cambrid~ Cambridge         United Kingdom
# # ... with 913 more rows
```

Primijetimo da je redoslijed kojim navodimo stupce onaj redoslijed kojim će biti prikazani. Pretpostavimo sada da želimo odabrati sve osim nekoliko određenih varijabli baze. Primjerice, odabrat ćemo sve varijable osim prve četiri na sljedeći način:

```
nobel %>%
  select(-(1:4))
# # A tibble: 923 x 10
# gender  year category share name_of_univers~ city_of_univers~
#   <fct>  <int> <fct>    <int> <chr>            <chr>
# 1 male    1901 physics      1 Munich Universi~ Munich
# 2 male    1902 physics      2 Leiden Universi~ Leiden
# 3 male    1902 physics      2 Amsterdam Unive~ Amsterdam
# 4 male    1903 physics      2 École Polytechn~ Paris
# 5 male    1903 physics      4 École municipal~ Paris
# 6 female  1903 physics      4 NA               NA
# 7 female  1911 chemist~     1 Sorbonne Univer~ Paris
# 8 male    1904 physics      1 Royal Instituti~ London
# 9 male    1905 physics      1 Kiel University  Kiel
#10 male    1906 physics      1 University of C~ Cambridge
# # ... with 913 more rows, and 4 more variables:
# #   country_of_university <chr>, born_month <fct>, age <dbl>,
# #   age_get_prize <dbl>
```

Kako bismo premjestili nekoliko varijabli na početak podatkovnog okvira, nakon što ih navedemo unutar select(), dodajemo i pomoćnu funkciju everything():

```
nobel %>%
  select(gender, category, everything())
# # A tibble: 923 x 14
#   gender category  firstname      surname  born_country_co~
```

```
#   <fct>  <fct>     <chr>          <chr>    <chr>
# 1 male   physics   Wilhelm Conrad Röntgen  DE
# 2 male   physics   Hendrik A.     Lorentz  NL
# 3 male   physics   Pieter         Zeeman   NL
# 4 male   physics   Henri          Becquer~ FR
# 5 male   physics   Pierre         Curie    FR
# 6 female physics   Marie          Curie    PL
# 7 female chemistry Marie          Curie    PL
# 8 male   physics   Lord           Rayleigh GB
# 9 male   physics   Philipp        Lenard   SK
#10 male   physics   J.J.           Thomson  GB
# # ... with 913 more rows, and 9 more variables:
# #   died_country_code <chr>, year <int>, share <int>,
# #   name_of_university <chr>, city_of_university <chr>,
# #   country_of_university <chr>, born_month <fct>,
# #   age <dbl>, age_get_prize <dbl>
```

3.2.2 Filtriranje redaka

Funkcija filter() koristi se za odabir podskupa redaka iz podatkovnog okvira na temelju njihovih vrijednosti. Točnije, retke zadržava ukoliko ispunjavaju uvjete u izrazima koje proslijedimo. Na primjer, dohvatimo podatke o ženama koje su osvojile Nobelovu nagradu 2019. godine.

```
nobel %>%
  filter(gender == "female", year == 2019)
# # A tibble: 1 x 14
#   firstname surname born_country_code died_country_code gender
#   <chr>     <chr>   <chr>             <chr>              <fct>
# 1 Esther    Duflo   FR                NA                 female
# # ... with 9 more variables: year <int>, category <fct>,
# #   share <int>, name_of_university <chr>,
# #   city_of_university <chr>, country_of_university <chr>,
# #   born_month <fct>, age <dbl>, age_get_prize <dbl>
```

Vidimo kako smo izraze odvojili zarezom, koji je ovdje ekvivalentan operatoru & („i”), to jest odabran je samo onaj redak koji ispunjava sve navedene uvjete. Također možemo koristiti i operator | („ili”), kao u sljedećem primjeru gdje tražimo imena dobitnika Nobelove nagrade iz znanstvenih disciplina (kemije, fizike ili medicine).

```
nobel %>%
  filter(category == "chemistry" |
           category == "physics" |
           category == "medicine") %>%
  select(firstname, surname, category)
# # A tibble: 616 x 3
#   firstname      surname   category
#   <chr>          <chr>     <fct>
# 1 Wilhelm Conrad Röntgen   physics
# 2 Hendrik A.     Lorentz   physics
# 3 Pieter         Zeeman    physics
# 4 Henri          Becquerel physics
# 5 Pierre         Curie     physics
# 6 Marie          Curie     physics
# 7 Marie          Curie     chemistry
# 8 Lord           Rayleigh  physics
# 9 Philipp        Lenard    physics
#10 J.J.           Thomson   physics
# # ... with 606 more rows
```

Isti rezultat možemo dobiti pomoću operatora %in%:

```
nobel %>%
  filter(category %in%
           c("chemistry", "physics", "medicine")) %>%
  select(firstname, surname, category)
```

Nadalje, pronađimo podatke o dobitnicima čija je dob prilikom dodjele nagrade bila između 30 i 50 godina.

```
nobel %>%
  filter(age_get_prize > 30, age_get_prize < 50)
# # A tibble: 206 x 14
#   firstname  surname  born_country_code died_country_code
#   <chr>      <chr>    <chr>             <chr>
# 1 Hendrik A. Lorentz  NL                NL
# 2 Pieter     Zeeman   NL                NL
# 3 Pierre     Curie    FR                FR
# 4 Marie      Curie    PL                FR
# 5 Marie      Curie    PL                FR
# 6 Philipp    Lenard   SK                DE
```

```
# 7 Guglielmo  Marconi  IT                IT
# 8 Wilhelm    Wien     RU                DE
# 9 Gustaf     Dalén    SE                SE
#10 Max         von Laue DE                DE
# # ... with 196 more rows, and 10 more variables:
# #   gender <fct>, year <int>, category <fct>, share <int>,
# #   name_of_university <chr>, city_of_university <chr>,
# #   country_of_university <chr>, born_month <fct>,
# #   age <dbl>, age_get_prize <dbl>
```

Osim operatora $>$ i $<$, možemo koristiti i operatore $\geq$, $\leq$ i != pri navođenju uvjeta unutar funkcije filter().

3.2.3 Sortiranje redaka

Za promjenu redoslijeda redaka podatkovnog okvira prema vrijednostima određenih varijabli upotrijebit ćemo funkciju arrange(). Prema zadanim postavkama, reci će biti sortirani uzlazno, s tim da nedostajuće vrijednosti uvijek dolaze na kraj podatkovnog okvira. Ako nas zanima tko su najmlađi dobitnici Nobelove nagrade, možemo to saznati promjenom redoslijeda redaka prema vrijednostima varijable *age_get_prize*:

```
nobel %>%
  arrange(age_get_prize) %>%
  select(firstname, surname, age_get_prize)
# # A tibble: 923 x 3
#   firstname    surname    age_get_prize
#   <chr>        <chr>              <dbl>
# 1 Malala       Yousafzai             17
# 2 Lawrence     Bragg                 25
# 3 Nadia        Murad                 25
# 4 Werner       Heisenberg            31
# 5 Paul A.M.    Dirac                 31
# 6 Carl D.      Anderson              31
# 7 Tsung-Dao    Lee                   31
# 8 Rudolf       Mössbauer             32
# 9 Frederick G. Banting               32
#10 Mairead      Corrigan              32
# # ... with 913 more rows
```

Želimo li pak saznati tko su najstariji dobitnici Nobelove nagrade, možemo upotrijebiti funkciju desc() te sortirati podatke silaznim redoslijedom prema vrijednostima varijable *age_get_prize*:

```
nobel %>%
  arrange(desc(age_get_prize)) %>%
  select(firstname, surname, age_get_prize)
# # A tibble: 923 x 3
#   firstname surname    age_get_prize
#   <chr>     <chr>              <dbl>
# 1 John      Goodenough            97
# 2 Arthur    Ashkin                96
# 3 Leonid    Hurwicz               90
# 4 Lloyd S.  Shapley               89
# 5 Raymond   Davis Jr.             88
# 6 Doris     Lessing               88
# 7 Peyton    Rous                  87
# 8 Karl      von Frisch            87
# 9 Joseph    Rotblat               87
#10 Vitaly L. Ginzburg              87
# # ... with 913 more rows
```

Sortirati možemo i prema vrijednostima varijabli koje nisu numeričke. Na primjer, sortirajmo podatke uzlaznim abecednim redom prema vrijednostima varijable *firstname* pa silaznim prema vrijednostima varijable *surname*.

```
nobel %>%
  arrange(firstname, desc(surname))
# # A tibble: 923 x 14
#   firstname  surname     born_country_code died_country_co~
#   <chr>      <chr>       <chr>             <chr>
# 1 A. Michael Spence      US                NA
# 2 Aage N.    Bohr        DK                DK
# 3 Aaron      Klug        LT                NA
# 4 Aaron      Ciechanover IL                NA
# 5 Abdus      Salam       PK                GB
# 6 Abhijit    Banerjee    IN                NA
# 7 Abiy       Ahmed Ali   ET                NA
# 8 Ada E.     Yonath      IL                NA
# 9 Adam G.    Riess       US                NA
#10 Adolf      Windaus     DE                DE
# # ... with 913 more rows, and 10 more variables:
# #   gender <fct>, year <int>, category <fct>, share <int>,
# #   name_of_university <chr>, city_of_university <chr>,
```

```
# #   country_of_university <chr>, born_month <fct>,
# #   age <dbl>, age_get_prize <dbl>
```

3.2.4 Izmjena i stvaranje novih stupaca

Ponekad je za analizu podataka potrebno izmijeniti neku od varijabli u bazi ili stvoriti novu varijablu. Ako bismo, primjerice, htjeli znati je li pojedini dobitnik preselio u drugu zemlju tijekom života, mogli bismo dodati novi stupac koji sadrži tu informaciju. U tom slučaju možemo iskoristiti mutate(), funkciju za izmjenu ili stvaranje novih stupaca podatkovnog okvira. Novi stupac kreirat ćemo pomoću postojećih stupaca podatkovnog okvira na sljedeći način:

```
nobel %>%
  mutate(imigrant =
           (born_country_code != died_country_code)) %>%
  select(firstname, surname, born_country_code, died_country_code,
         imigrant)
# # A tibble: 923 x 5
#   firstname      surname  born_country_co~ died_country_co~
#   <chr>          <chr>    <chr>            <chr>
# 1 Wilhelm Conrad Röntgen  DE               DE
# 2 Hendrik A.     Lorentz  NL               NL
# 3 Pieter         Zeeman   NL               NL
# 4 Henri          Becquer~ FR               FR
# 5 Pierre         Curie    FR               FR
# 6 Marie          Curie    PL               FR
# 7 Marie          Curie    PL               FR
# 8 Lord           Rayleigh GB               GB
# 9 Philipp        Lenard   SK               DE
#10 J.J.           Thomson  GB               GB
# # ... with 913 more rows, and 1 more variable:
# #   imigrant <lgl>
```

Novi stupci su uvijek dodani na kraj podatkovnog okvira pa, kako bismo vidjeli one stupce koji nas zanimaju, koristimo select(). Novostvorena varijabla imigrant ima vrijednost TRUE ukoliko je uvjet *born_country_code != died_country_code* ispunjen, odnosno FALSE u suprotnom. Primijetimo da informaciju o preseljenju na ovaj način možemo dobiti jedino ukoliko je dobitnik Nobelove nagrade preminuo. U suprotnom, vrijednost varijable migrant je NA (gotovo svaka operacija koja sadrži NA vrijednost vratit će NA kao rezultat):

```
nobel %>%
  mutate(imigrant =
           (born_country_code != died_country_code)) %>%
  select(firstname, surname, born_country_code,
         died_country_code, imigrant) %>%
  filter(is.na(died_country_code))
# # A tibble: 306 x 5
#   firstname surname   born_country_code died_country_code
#   <chr>     <chr>     <chr>             <chr>
# 1 Gabriel   Lippmann  LU                NA
# 2 Chen Ning Yang      CN                NA
# 3 Tsung-Dao Lee       CN                NA
# 4 Rudolf    Mössbauer DE                NA
# 5 Leon N.   Cooper    US                NA
# 6 Leo       Esaki     JP                NA
# 7 Ivar      Giaever   NO                NA
# 8 Brian D.  Josephson GB                NA
# 9 Antony    Hewish    GB                NA
# 10 Ben R.    Mottelson US                NA
# # ... with 296 more rows, and 1 more variable:
# #   imigrant <lgl>
```

Ako nam je potreban samo novostvoreni stupac, prikladnije je umjesto mutate() koristiti funkciju transmute() koja ne zadržava stare stupce.

```
nobel %>%
  transmute(imigrant = (born_country_code != died_country_code))
# # A tibble: 923 x 1
# imigrant
#   <lgl>
# 1 FALSE
# 2 FALSE
# 3 FALSE
# 4 FALSE
# 5 FALSE
# 6 TRUE
# 7 TRUE
# 8 FALSE
# 9 TRUE
# 10 FALSE
```

```
# # ... with 913 more rows
```

Također, moguće je stvoriti više novih varijabli odjednom razdvajajući ih zarezom unutar mutate(), odnosno transmute(), te se pri stvaranju sljedeće pozvati na prethodno stvorenu varijablu. U nastavku su navedene neke od funkcija korisnih pri stvaranju novih stupaca. Pregršt takvih funkcija već se nalazi u osnovnom R-u, a ovdje ćemo navesti neke od onih koje su dio paketa tidyverse:

- Funkcijom row_number() dodajemo ID stupac podacima.

```
nobel %>%
  mutate(id = row_number()) %>%
  select(id, everything())
# # A tibble: 923 x 15
#      id firstname    surname  born_country_co~ died_country_co~
#   <int> <chr>        <chr>    <chr>            <chr>
# 1     1 Wilhelm Con~ Röntgen  DE               DE
# 2     2 Hendrik A.   Lorentz  NL               NL
# 3     3 Pieter       Zeeman   NL               NL
# 4     4 Henri        Becquer~ FR               FR
# 5     5 Pierre       Curie    FR               FR
# 6     6 Marie        Curie    PL               FR
# 7     7 Marie        Curie    PL               FR
# 8     8 Lord         Rayleigh GB               GB
# 9     9 Philipp      Lenard   SK               DE
#10    10 J.J.         Thomson  GB               GB
# # ... with 913 more rows, and 10 more variables: gender <fct>,
# #   year <int>, category <fct>, share <int>,
# #   name_of_university <chr>, city_of_university <chr>,
# #   country_of_university <chr>, born_month <fct>, age <dbl>,
# #   age_get_prize <dbl>
```

- Funkcije lag() i lead() će načiniti novi stupac koji će sadržavati vrijednosti nekog postojećeg stupca iz baze, no pomaknute za jedno mjesto „unaprijed" ili „unazad". Točnije, i-ti redak stupca stvorenog pomoću funkcije lag() sadržavat će vrijednost iz (i-1). retka originalnog stupca, odnosno (i+1). retka u slučaju da je stvoren funkcijom lead().

```
a <- tibble(x = 1:5)
a %>%
  mutate(y = lag(x), z = lead(x))
# # A tibble: 5 x 3
```

```
#       x     y     z
#   <int> <int> <int>
# 1     1    NA     2
# 2     2     1     3
# 3     3     2     4
# 4     4     3     5
# 5     5     4    NA
```

- U funkciji case_when() navodimo niz logičkih izraza, od kojih prvi istiniti određuje vrijednost varijable. Željenu vrijednost u slučaju istinitosti pojedinog iskaza stavljamo nakon znaka „∼”.

```
nobel %>%
  mutate(preminuo = case_when(
    is.na(died_country_code) ~ "ne",
    !is.na(died_country_code) ~ "da"
  )) %>%
  select(firstname, surname, died_country_code, preminuo)
# # A tibble: 923 x 4
#   firstname      surname   died_country_code preminuo
#   <chr>          <chr>     <chr>             <chr>
# 1 Wilhelm Conrad Röntgen   DE                da
# 2 Hendrik A.     Lorentz   NL                da
# 3 Pieter         Zeeman    NL                da
# 4 Henri          Becquerel FR                da
# 5 Pierre         Curie     FR                da
# 6 Marie          Curie     FR                da
# 7 Marie          Curie     FR                da
# 8 Lord           Rayleigh  GB                da
# 9 Philipp        Lenard    DE                da
#10 J.J.           Thomson   GB                da
# # ... with 913 more rows
```

Ako niti jedan od logičkih izraza nije istinit, funkcija kao vrijednost varijable vraća NA. Kako bismo to izbjegli, kao posljednji u nizu izraza stavljamo *TRUE* ∼ *(zadana vrijednost)*. Primijetimo da bismo, uzevši to u obzir, mogli prethodni kod napisati i na sljedeći način:

```
nobel %>%
  mutate(preminuo = case_when(
    is.na(died_country_code) ~ "ne",
    TRUE ~ "da"
  ))
```

- Koristeći funkciju cumall(), vrijednost u pojedinom retku novog stupca bit će TRUE ukoliko sve vrijednosti postojećeg proslijeđenog stupca do tog retka (uključujući i njega) ispunjavaju navedeni uvjet, odnosno FALSE u suprotnom. Na sličan način djeluje i funkcija cumany(), s tim da vraća vrijednost TRUE ako je uvjet ispunjen za bar jedan redak. Funkcija cummean() u pojedini redak novog stupca vraća prosjek svih vrijednosti proslijeđenog stupca do tog retka.

```
a %>% mutate(y = cumall(x < 3),
             z = cumany(x == 3),
             w = cummean(x))
# # A tibble: 5 x 4
#       x y     z          w
#   <int> <lgl> <lgl> <dbl>
# 1     1 TRUE  FALSE   1
# 2     2 TRUE  FALSE   1.5
# 3     3 FALSE TRUE    2
# 4     4 FALSE TRUE    2.5
# 5     5 FALSE TRUE    3
```

Pretpostavimo sada kako želimo sve stupce u bazi pretvoriti u stupce znakovnog tipa. U tom slučaju, za izmjenu više redaka odjednom, koristimo inačice funkcije mutate(). Prva od njih, mutate_all(), istu funkciju primjenjuje na svaki stupac podatkovnog okvira:

```
nobel %>%
  mutate_all(as.character)
# # A tibble: 923 x 14
#   firstname      surname  born_country_co~ died_country_co~
#   <chr>          <chr>    <chr>             <chr>
# 1 Wilhelm Conrad Röntgen  DE                DE
# 2 Hendrik A.     Lorentz  NL                NL
# 3 Pieter         Zeeman   NL                NL
# 4 Henri          Becquer~ FR                FR
# 5 Pierre         Curie    FR                FR
# 6 Marie          Curie    PL                FR
# 7 Marie          Curie    PL                FR
```

```
# 8 Lord           Rayleigh GB               GB
# 9 Philipp        Lenard   SK               DE
#10 J.J.           Thomson  GB               GB
# # ... with 913 more rows, and 10 more variables:
# #   gender <chr>, year <chr>, category <chr>, share <chr>,
# #   name_of_university <chr>, city_of_university <chr>,
# #   country_of_university <chr>, born_month <chr>,
# #   age <chr>, age_get_prize <chr>
```

Želimo li izmijeniti samo one stupce čije vrijednosti zadovoljavaju određene uvjete, koristimo mutate_if(). U nastavku sve stupce u bazi tipa factor pretvaramo u stupce znakovnog tipa.

```
nobel %>%
  mutate_if(is.factor, as.character) %>%
  select(gender, category, born_month, everything())
# # A tibble: 923 x 14
#  gender category  born_month firstname      surname
#  <chr>  <chr>     <chr>      <chr>          <chr>
# 1 male   physics   Mar        Wilhelm Conrad Röntgen
# 2 male   physics   Jul        Hendrik A.     Lorentz
# 3 male   physics   May        Pieter         Zeeman
# 4 male   physics   Dec        Henri          Becquerel
# 5 male   physics   May        Pierre         Curie
# 6 female physics   Nov        Marie          Curie
# 7 female chemistry Nov        Marie          Curie
# 8 male   physics   Nov        Lord           Rayleigh
# 9 male   physics   Jun        Philipp        Lenard
#10 male   physics   Dec        J.J.           Thomson
# # ... with 913 more rows, and 9 more variables:
# #   born_country_code <chr>, died_country_code <chr>,
# #   year <dbl>, share <dbl>, name_of_university <chr>,
# #   city_of_university <chr>, country_of_university <chr>,
# #   age <dbl>, age_get_prize <dbl>
```

3.2.5 Sažimanje i grupiranje podataka

Funkcija summarise() sažima retke podatkovnog okvira te vraća podatkovni okvir s jednim retkom. Vraćeni redak sadrži vrijednosti agregacijskih funkcija koje navedemo unutar summarise(). Na taj način, navodeći funkciju mean(), možemo dobiti prosječnu dob dobitnika Nobelove nagrade:

```
nobel %>% summarise(prosjecna_dob = mean(age_get_prize,
                                         na.rm = TRUE))
# # A tibble: 1 x 1
#   prosjecna_dob
#           <dbl>
# 1          59.8
```

Zbog mogućih nedostajućih vrijednosti, unutar funkcije mean() navodimo logički izraz *na.rm = TRUE*, koji osigurava da će takve vrijednosti biti uklonjene prije nastavka izračuna. Naime, ukoliko u proslijeđenim podacima postoje nedostajuće vrijednosti, mean() kao rezultat također vraća NA vrijednost. Kao i kod prethodnih funkcija paketa dplyr, u jednom pozivu funkcije summarise() također možemo izvesti više operacija. Primjerice, pronađimo minimalnu i maksimalnu dob dobitnika prilikom dodjele nagrade pomoću funkcija min() i max().

```
nobel %>% summarise(minimalna_dob = min(age_get_prize, na.rm = TRUE),
  maksimalna_dob = max(age_get_prize, na.rm = TRUE))
# # A tibble: 1 x 2
#   minimalna_dob maksimalna_dob
#           <dbl>          <dbl>
# 1            17             97
```

No, u dosadašnjim primjerima funkcija summarise()[7] nije bila izrazito korisna te bismo iste rezultate mogli dobiti i bez nje, koristeći samo agregacijske funkcije. Ova funkcija pravi smisao dobiva tek u kombinaciji s funkcijom group_by(). Funkcija group_by() grupira podatke prema jednoj ili vise varijabli, ali pritom ne mijenja izgled podatkovnog okvira, već način na koji druge funkcije rade s podacima. Druga funkcija, u našem slučaju summarise(), nakon grupiranja primijenit će se na svaki podskup podataka zasebno. Pretpostavimo sada da nas umjesto prosječne dobi svih dobitnika zanimaju prosječne dobi dobitnika po kategorijama Nobelove nagrade. Ono što ćemo učiniti kako bismo to saznali je grupiranje vrijednosti varijable *age_get_prize* prema vrijednostima varijable category, to jest prema šest različitih kategorija u kojima se nagrada dodjeljuje.

[7] R prepoznaje i britanski (summarise) i američki (summarize) pravopis.

```
nobel %>%
  group_by(category) %>%
  summarise(prosjecna_dob = mean(age_get_prize,
                                 na.rm = TRUE))
# # A tibble: 6 x 2
#   category   prosjecna_dob
#   <chr>              <dbl>
# 1 chemistry           58.9
# 2 economics           66.7
# 3 literature          64.7
# 4 medicine            58.4
# 5 peace               60.9
# 6 physics             56.2
```

Vidimo kako su prosječno najstariji dobitnici Nobelove nagrade iz ekonomije, a najmlađi iz fizike. Dodatno, promotrimo prosječne dobi po kategorijama zasebno za muške i za ženske dobitnike.

```
nobel %>%
  group_by(category, gender) %>%
  summarise(prosjecna_dob = mean(age_get_prize,
                                 na.rm = TRUE))
# `summarise()` has grouped output by 'category'. You can
# override using the `.groups` argument.
# # A tibble: 12 x 3
# # Groups:   category [6]
#   category   gender prosjecna_dob
#   <chr>      <chr>          <dbl>
# 1 chemistry  female          53.6
# 2 chemistry  male            59.0
# 3 economics  female          61.5
# 4 economics  male            66.8
# 5 literature female          62.6
# 6 literature male            65
# 7 medicine   female          62.6
# 8 medicine   male            58.1
# 9 peace      female          50.5
#10 peace      male            62.9
#11 physics    female          50.7
#12 physics    male            56.3
```

Primijetimo kako smo ovdje vrijednosti varijable *age_get_prize*, osim po kategorijama nagrada, podijelili i po spolu dobitnika. Iz primjera uočavamo kako su u svakoj kategoriji, osim medicini, ženski dobitnici prosječno mlađi od muških dobitnika.
Kada želimo poništiti grupiranje koristimo funkciju ungroup(). Osim funkcije mean(), u osnovnom R-u postoji još mnogo agregacijskih funkcija koje bismo mogli koristiti u analizi zajedno s group_by() i summarise(), a u nastavku su navedene neke od takvih funkcija koje nudi dplyr:

- Funkcija n() vraća broj vrijednosti unutar pojedine grupe određene varijable. Koristeći n(), na sljedeći način saznajemo broj dobitnika rođenih u pojedinoj zemlji:

```
nobel %>%
  group_by(born_country_code) %>%
  summarise(broj = n()) %>%
  arrange(desc(broj))
# # A tibble: 79 x 2
#   born_country_code  broj
#   <chr>             <int>
# 1 US                  274
# 2 GB                  103
# 3 DE                   82
# 4 FR                   56
# 5 SE                   29
# 6 PL                   28
# 7 JP                   27
# 8 RU                   26
# 9 CA                   20
#10 CH                   19
# # ... with 69 more rows
```

  Dodatno smo izmijenili redoslijed podataka prema vrijednostima varijable *broj* pa saznajemo kako je najviše dobitnika Nobelove nagrade rođeno u SAD-u.

- Slično, funkcija n_distinct() vraća broj jedinstvenih vrijednosti unutar pojedine grupe određene varijable. Upotrijebit ćemo ju kako bismo saznali koliko je dobitnika pohađalo pojedino sveučilište, zbog mogućih višestrukih dobitnika.

```
nobel %>%
  unite(dobitnik, firstname, surname, sep = " ") %>%
  group_by(name_of_university) %>%
  summarise(broj_dobitnika = n_distinct(dobitnik)) %>%
```

```
  arrange(desc(broj_dobitnika))
# # A tibble: 308 x 2
#   name_of_university                          broj_dobitnika
#   <chr>                                              <int>
# 1 NA                                                   226
# 2 University of California                              32
# 3 Harvard University                                    27
# 4 Massachusetts Institute of Technology (MI~            20
# 5 Stanford University                                   18
# 6 University of Chicago                                 18
# 7 University of Cambridge                               17
# 8 California Institute of Technology (Calte~            16
# 9 Columbia University                                   16
#10 Princeton University                                  15
```

Vidimo kako u 226 redaka baze nemamo informaciju o imenu sveučilišta koje je dobitnik pohađao te je time grupa koja sadrži nedostajuće vrijednosti najbrojnija. Nakon grupe nedostajućih vrijednosti, najviše redova ima grupa određena vrijednošću „University of California", to jest najviše dobitnika je pohađalo upravo to sveučilište.

- Za dohvaćanje vrijednosti prema pozicijama, upotrijebit ćemo funkcije first(), last() i nth(), koje redom dohvaćaju prvu, zadnju i n-tu vrijednost unutar pojedine grupe određenog stupca. Tako, na primjer, možemo saznati prvu godinu dodjele nagrade iz pojedine kategorije pomoću funkcije first():

```
nobel %>%
  group_by(category) %>%
  arrange(year) %>%
  summarise(prva_godina = first(year))
# # A tibble: 6 x 2
# category   prva_godina
#   <chr>            <dbl>
# 1 chemistry         1901
# 2 economics         1969
# 3 literature        1901
# 4 medicine          1901
# 5 peace             1901
# 6 physics           1901
```

Iz ovog ispisa saznajemo da se Nobelova nagrada iz ekonomije počela dodjeljivati

tek od 1969. godine. Primijetimo da je sortiranje u ovom slučaju izvršeno unutar pojedine grupe.

Ponekad je potrebno spremiti vrijednosti funkcija po grupama kako bismo ih mogli koristiti u nastavku analize. To možemo učiniti kombiniranjem funkcija group_by() i mutate():

```
nobel %>%
  group_by(category) %>%
  mutate(prosjecna_dob = mean(age_get_prize,
                              na.rm = TRUE)) %>%
  select(firstname,
         surname,
         category,
         age_get_prize,
         prosjecna_dob)
# # A tibble: 923 x 5
# # Groups:   category [6]
#   firstname   surname  category age_get_prize prosjecna_dob
#   <chr>       <chr>    <chr>            <dbl>         <dbl>
# 1 Wilhelm Co~ Röntgen  physics             56          56.2
# 2 Hendrik A.  Lorentz  physics             49          56.2
# 3 Pieter      Zeeman   physics             37          56.2
# 4 Henri       Becquer~ physics             51          56.2
# 5 Pierre      Curie    physics             44          56.2
# 6 Marie       Curie    physics             36          56.2
# 7 Marie       Curie    chemist~            44          58.9
# 8 Lord        Rayleigh physics             62          56.2
# 9 Philipp     Lenard   physics             43          56.2
#10 J.J.        Thomson  physics             50          56.2
# # ... with 913 more rows
```

U ovom primjeru funkcijom mutate() je dodan novi stupac čije su vrijednosti prosječne dobi dobitnika po kategorijama. Novododani stupac omogućuje lakše uspoređivanje dobi pojedinog dobitnika s prosječnom dobi u njegovoj kategoriji.

Još jedna funkcija koju možemo koristiti u kombinaciji s group_by() je funkcija filter(). Pretpostavimo da želimo pronaći samo podatke iz godina u kojima je broj dobitnika bio veći od prosječnog. To ćemo učiniti u dva koraka grupirajući podatke prema vrijednostima varijable *year*. Prosječan godišnji broj dobitnika dobit ćemo dvostrukom primjenom funkcije summarise(), a zatim željene retke možemo pronaći pomoću funkcije filter().

```r
nobel %>%
  group_by(year) %>%
  summarise(broj_dobitnika = n()) %>%
  summarise(mean(broj_dobitnika))
# # A tibble: 1 x 1
#   `mean(broj_dobitnika)`
#                    <dbl>
# 1                   7.96

nobel %>%
  group_by(year) %>%
  filter(n() > 7.96) %>%
  select(year,everything())
# # A tibble: 578 x 14
# # Groups:   year [54]
#    year firstname surname born_country_co~ died_country_co~
#   <dbl> <chr>     <chr>   <chr>            <chr>
# 1  1946 Percy W.  Bridgm~ US               US
# 2  1950 Cecil     Powell  GB               IT
# 3  1956 William ~ Shockl~ GB               US
# 4  1956 John      Bardeen US               US
# 5  1972 John      Bardeen US               US
# 6  1956 Walter H. Bratta~ CN               US
# 7  1958 Pavel A.  Cheren~ RU               RU
# 8  1958 Igor Y.   Tamm    RU               RU
# 9  1962 Lev       Landau  AZ               RU
#10  1963 Eugene    Wigner  HU               US
# # ... with 568 more rows, and 9 more variables:
# #   gender <chr>, category <chr>, share <dbl>,
# #   name_of_university <chr>, city_of_university <chr>,
# #   country_of_university <chr>, born_month <chr>,
# #   age <dbl>, age_get_prize <dbl>
```

Napomena. *Podatkovni okvir „nobel" tijekom ovog poglavlja je ostao nepromijenjen. Ukoliko želimo mijenjati podatkovne okvire, rezultat je potrebno pohraniti operatorom pridruživanja u originalni podatkovni okvir ili stvoriti novi.*

3.3 Spajanje podatkovnih okvira

Nerijetko su podaci baze podataka raspoređeni u više tablica. Takvu bazu podataka nazivamo relacijskom, jer na temelju relacija između pojedinih tablica njihove podatke možemo spojiti. Paket dplyr osigurava jednostavno spajanje podatkovnih okvira funkcijama spajanja. U predstavljanju ovih funkcija koristit ćemo sljedeće podatkovne okvire:

```
ocjene
# # A tibble: 10 x 3
#   ucenik       ocjena predmet_id
#   <chr>         <dbl>      <dbl>
# 1 Petar Marić       3          1
# 2 Ana Ivić          4          2
# 3 Marija Perić      5          3
# 4 Luka Horvat       4          5
# 5 Petra Kovač       5          4
# 6 Ivan Anić         2          7
# 7 Ana Ivić          5          6
# 8 Marija Perić      4          8
# 9 Luka Horvat       4         10
#10 Petra Kovać       5          1
```

```
predmeti
# # A tibble: 9 x 2
#      id predmet
#   <dbl> <chr>
# 1     1 matematika
# 2     2 hrvatski
# 3     3 engleski
# 4     4 fizika
# 5     5 kemija
# 6     6 biologija
# 7     7 povijest
# 8     8 informatika
# 9     9 geografija
```

Između podatkovnih okvira „ocjene" i „predmeti" relacija je ostvarena preko vrijednosti varijabla *predmet_id* i *id*. Te varijable, po kojima spajamo podatkovne okvire, nazivamo ključevima. Razlikujemo primarne i strane ključeve. Primarnim ključem nazivamo varijablu koja jedinstveno identificira opservacije podatkovnog okvira u kojemu se nalazi. Strani ključ je varijabla koja jedinstveno identificira opservacije nekog drugog podatkovnog ok-

vira. U ovom slučaju, varijabla *id* je jedinstveni identifikator opservacija podatkovnog okvira „predmeti" i time je primarni ključ. Varijabla *predmet_id*, koja se nalazi u podatkovnom okviru „ocjene", također jedinstveno identificira opservacije podatkovnog okvira „predmeti" te je stoga strani ključ. Funkcije spajanja će spariti dvije opservacije različitih podatkovnih okvira ukoliko imaju istu vrijednost ključa.

Želimo stvoriti podatkovni okvir koji će sadržavati ocjene učenika i nazive predmeta iz kojih su ocjene dobivene. To možemo učiniti na više različitih načina koji su pojašnjeni u nastavku.

3.3.1 Unutarnji spoj

Odlučimo li spojiti podatkovne okvire unutarnjim spojem, opservacije koje nemaju pripadnu opservaciju s istom vrijednosti ključa u drugom podatkovnom okviru neće biti sadržane u spoju. Dakle, novi podatkovni okvir sadržavat će samo one opservacije čije se vrijednosti stupca po kojemu spajamo (ključa) nalaze u oba originalna podatkovna okvira. Funkcija koju koristimo za ovakvo spajanje je inner_join(), kojoj nazive primarnog i stranog ključa prosljeđujemo pomoću parametra by.

```
ocjene %>%
  inner_join(predmeti, by = c("predmet_id" = "id"))
# # A tibble: 9 x 4
# ucenik        ocjena predmet_id predmet
#   <chr>          <dbl>      <dbl> <chr>
# 1 Petar Marić        3          1 matematika
# 2 Ana Ivić           4          2 hrvatski
# 3 Marija Perić       5          3 engleski
# 4 Luka Horvat        4          5 kemija
# 5 Petra Kovač        5          4 fizika
# 6 Ivan Anić          2          7 povijest
# 7 Ana Ivić           5          6 biologija
# 8 Marija Perić       4          8 informatika
# 9 Petra Kovać        5          1 matematika
```

Osim ključa, rezultat spajanja sadrži i sve ostale stupce originalnih podatkovnih okvira. Opservacije koje su ostale nesparene su („Luka Horvat", 4, 10) iz prvog i (9, „geografija") iz drugog podatkovnog okvira, jer se vrijednost 10 ne nalazi među vrijednostima varijable *id*, odnosno 9 među vrijednostima varijable *predmet_id*.

Ako bismo ipak htjeli zadržati neki od ta dva retka ili oba, umjesto unutarnjeg trebamo koristiti vanjski spoj.

3.3.2 Vanjski spojevi

Želimo spojiti podatkovne okvire „ocjene” i „predmeti” na način da zadržimo sve opservacije okvira „ocjene”, bez obzira na to jesu li sparene s nekom od opservacija okvira „predmeti”. To možemo učiniti lijevim (vanjskim) spojem koristeći funkciju left_join():

```
ocjene %>%
  left_join(predmeti, by = c("predmet_id" = "id"))
# # A tibble: 10 x 4
#   ucenik       ocjena predmet_id predmet
#   <chr>         <dbl>      <dbl> <chr>
# 1 Petar Marić       3          1 matematika
# 2 Ana Ivić          4          2 hrvatski
# 3 Marija Perić      5          3 engleski
# 4 Luka Horvat       4          5 kemija
# 5 Petra Kovač       5          4 fizika
# 6 Ivan Anić         2          7 povijest
# 7 Ana Ivić          5          6 biologija
# 8 Marija Perić      4          8 informatika
# 9 Luka Horvat       4         10 NA
#10 Petra Kovać       5          1 matematika
```

Primijetimo da je sada redak („Luka Horvat”, 4, 10) podatkovnog okvira „ocjene” uključen u rezultat, s nedostajućom vrijednosti u stupcu *predmet*. Pretpostavimo da želimo zadržati sve opservacije obaju podatkovnih okvira u njihovom spoju. Takav spoj naziva se puni (vanjski) spoj, a dobiva se funkcijom full_join():

```
ocjene %>%
  full_join(predmeti, by = c("predmet_id" = "id"))
# # A tibble: 11 x 4
#   ucenik       ocjena predmet_id predmet
#   <chr>         <dbl>      <dbl> <chr>
# 1 Petar Marić       3          1 matematika
# 2 Ana Ivić          4          2 hrvatski
# 3 Marija Perić      5          3 engleski
# 4 Luka Horvat       4          5 kemija
# 5 Petra Kovač       5          4 fizika
# 6 Ivan Anić         2          7 povijest
# 7 Ana Ivić          5          6 biologija
# 8 Marija Perić      4          8 informatika
# 9 Luka Horvat       4         10 NA
```

```
#10 Petra Kovač       5          1 matematika
#11 NA                NA         9 geografija
```

Sve opservacije čije se vrijednosti ključeva podudaraju sparene su, a ostale su dodane s nedostajućim vrijednostima na mjestu stupaca koji pripadaju drugom podatkovnom okviru. Tako je sada i redak (9, „geografija”) podatkovnog okvira „predmeti” uključen u rezultat.

Ostale vrste spojeva čitatelj može proučiti u [10] (poglavlje 15).

4 Vizualizacija podataka

Posljednji iz skupine paketa tidyverse kojeg ćemo predstaviti je ggplot2. Ovaj paket služi za vizualizaciju podataka, čime nam pomaže u razumijevanju podataka te prenošenju rezultata analize drugima. Zasniva se na principu takozvane grafičke gramatike, koja ga čini elegantnijim načinom za grafičko prikazivanje podataka od grafičkog paketa ugrađenog u R. Pojam grafičke gramatike uveo je Leland Wilkinson u [7].

Prije same vizualizacije potrebno je provjeriti jesu li podaci uredni, jer funkcije paketa ggplot2 upravo to pretpostavljaju. Zatim odabiremo prikladnu metodu prikazivanja podataka koja ovisi o tipu podataka i onome što iz njih želimo saznati. Sve ostale karakteristike grafa su unaprijed zadane, ali se mogu i mijenjati. U nastavku ćemo objasniti pojam grafičke gramatike te predstaviti osnovne načine grafičkog prikazivanja podataka pomoću ovog paketa.

4.1 Grafička gramatika

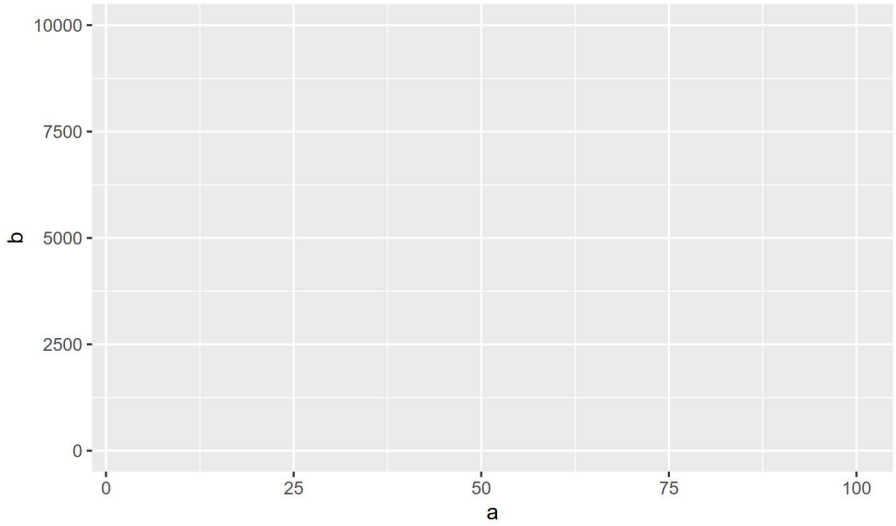
Pristup kreiranju grafova kombiniranjem različitih komponenti nazvan je grafičkom gramatikom. Za stvaranje grafa potrebne su najmanje tri komponente: podaci, komponenta estetike i geometrije. Spojene, komponente zajedno čine jedan sloj. Slojevi grafa obično imaju nešto zajedničko, najčešće su to različiti prikazi istih podataka. Pojedine slojeve međusobno spajamo u jedan graf operatorom „+”. Ovakav pristup stvaranju grafa čini taj proces prilagodljivim na promjene. Naime, kako postepeno kreiramo komponente grafa, tako ih pojedinačno možemo i mijenjati.

Krenimo od komponente podataka, to jest podataka koje želimo grafički prikazati. Za potrebe ovog potpoglavlja stvorit ćemo jednostavan podatkovni okvir „kvadrirani”:

```
kvadrirani <- tibble(
  a = 1:100,
  b = a ^ 2
)
```

Vrijednosti stvorenih varijabli prenijet ćemo na graf pomoću komponente estetike funkcijom ggplot() na sljedeći način:

```
ggplot(data = kvadrirani, mapping = aes(x = a, y = b))
```

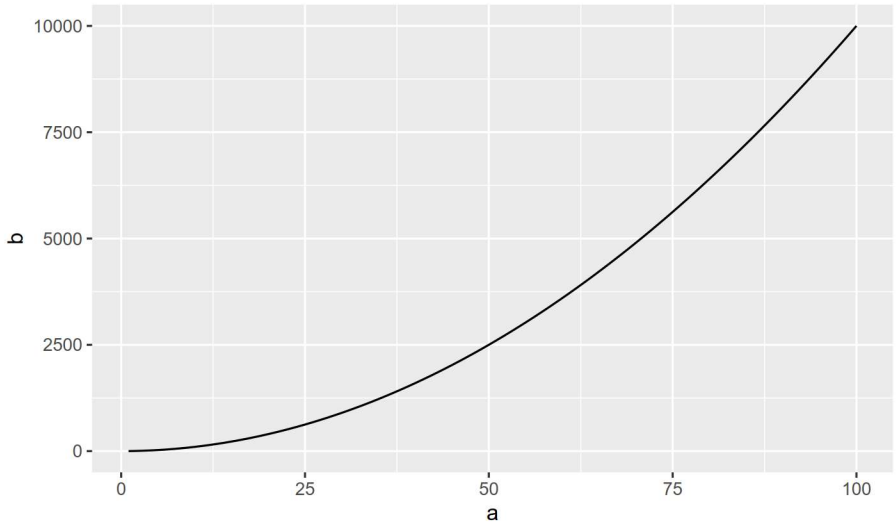


Slika 2: Primjer komponenti podataka i estetike

Pomoćnom funkcijom aes() prenijeli smo vrijednosti varijabli a i b na apscisu, odnosno ordinatu. Osim koordinatnih osi, postoje i druge estetike s kojima ćemo se susresti u nastavku.

U sljedećem koraku dodajemo i treću komponentu, geometriju, te time stvaramo jedan sloj grafa.

```
ggplot(data = kvadrirani, mapping = aes(x = a, y = b)) +
  geom_line()
```



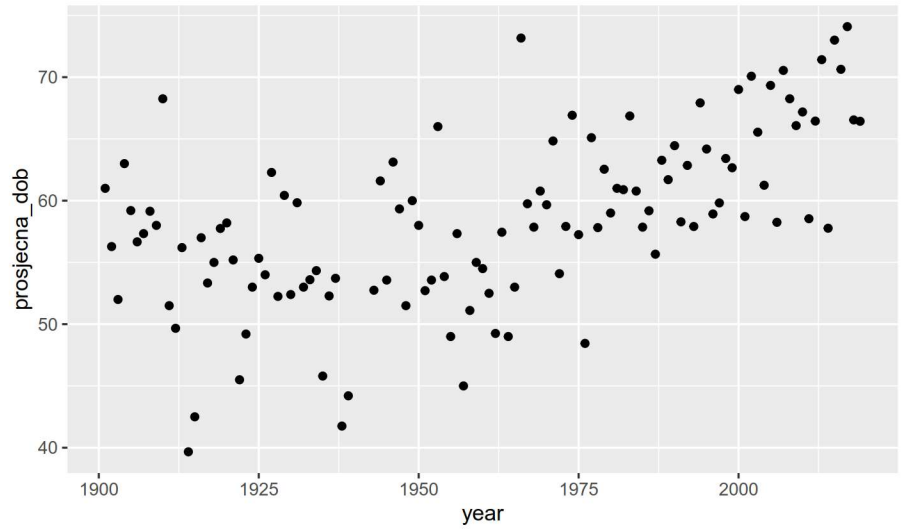
Slika 3: Primjer grafa koji se sastoji od jednog sloja

Funkcija geom_line(), koja stvara linijski dijagram, u ovom slučaju predstavlja komponentu geometrije. Korištenjem opreatora „+”, nakon ovog možemo dodati i druge slojeve na graf, čiji će geometrijski oblici biti određeni nekom drugom funkcijom iz familije geom. Više o grafičkoj gramatici može se pročitati u [11]. Ostali osnovni tipovi grafova, odnosno vrste geom funkcija bit će predstavljeni u sljedećem potpoglavlju.

4.2 Osnovni tipovi grafova

U ovom potpoglavlju upoznat ćemo osnovne tipove grafova, koje pri stvaranju grafa odabiremo pripadnom geom funkcijom. Vizualizirat ćemo podatke iz baze podataka „nobel”, čije smo varijable predstavili u prethodnom poglavlju. Prvo želimo provjeriti kako se mijenjala prosječna dob dobitnika Nobelove nagrade kroz godine. To možemo učiniti crtanjem dijagrama raspršenosti, kojeg koristimo kada želimo prikazati odnos između dvije numeričke varijable.

```
nobel %>%
  group_by(year) %>%
  summarise(prosjecna_dob = mean(age_get_prize)) %>%
  ggplot(mapping = aes(year, prosjecna_dob)) +
  geom_point()
```

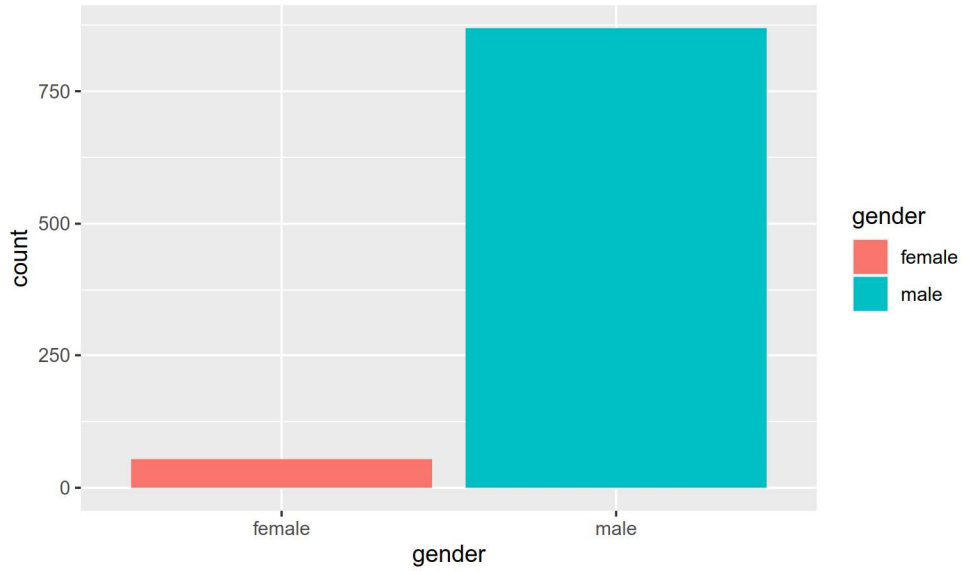


Slika 4: Odnos godine dodjele nagrade i prosječne dobi dobitnika prikazan dijagramom raspršenosti

Iz grafičkog prikaza može se naslutiti kako se prosječna dob dobitnika s godinama povećava. Razlog tomu može biti sve veći broj kandidata koji uzrokuje sve duže vrijeme čekanja od postignuća do dobitka nagrade.

Nadalje, iz dosadašnje analize mogli smo zaključiti kako je broj žena među dobitnicima znatno manji od broja muškaraca. Pogledajmo grafički prikaz zastupljenosti muškaraca i žena među dobitnicima pomoću funkcije geom_bar(). Ovom funkcijom crtamo stupčasti dijagram koji se koristi za uvid u zastupljenost pojedinih kategorija kvalitativne varijable.

```
nobel %>% ggplot() +
  geom_bar(mapping = aes(x = gender, fill = gender))
```



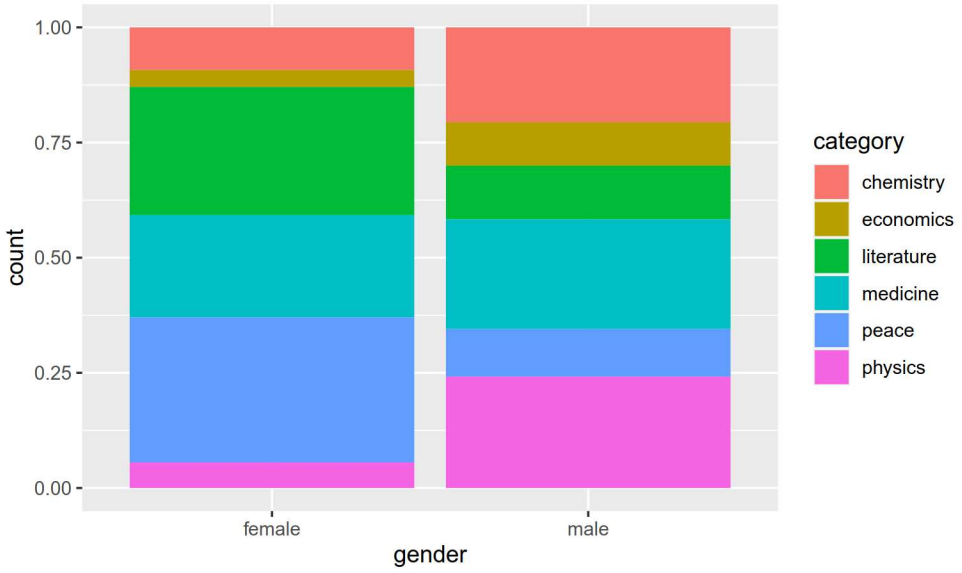
Slika 5: Stupčasti dijagram frekvencija dobitnika po spolu

Vidimo kako je od ukupnog broja dodijeljenih nagrada izrazito mali broj pripao ženama. Primijetimo da smo stupčasti dijagram obojili s obzirom na spol pomoću parametra fill.

Ono što bi nas također moglo zanimati je zastupljenost kategorija među nagradama koje su osvojile žene i nagradama koje su osvojili muškarci. Zbog različite zastupljenosti žena i muškaraca u pojedinim područjima, možemo pretpostaviti kako je zastupljenost kategorija u ove dvije skupine različita. Za provjeru ove pretpostavke također koristimo funkciju geom_bar().

```
nobel %>%
  ggplot(mapping = aes(x = gender, fill = category)) +
  geom_bar(position = "fill")
```

Iz stupčastog dijagrama na Slici 6 možemo vidjeti da su među nagradama osvojenim od strane žena najzastupljenije nagrade za književnost i mir, dok su među nagradama koje su osvojili muškarci najzastupljenije nagrade za fiziku i medicinu.

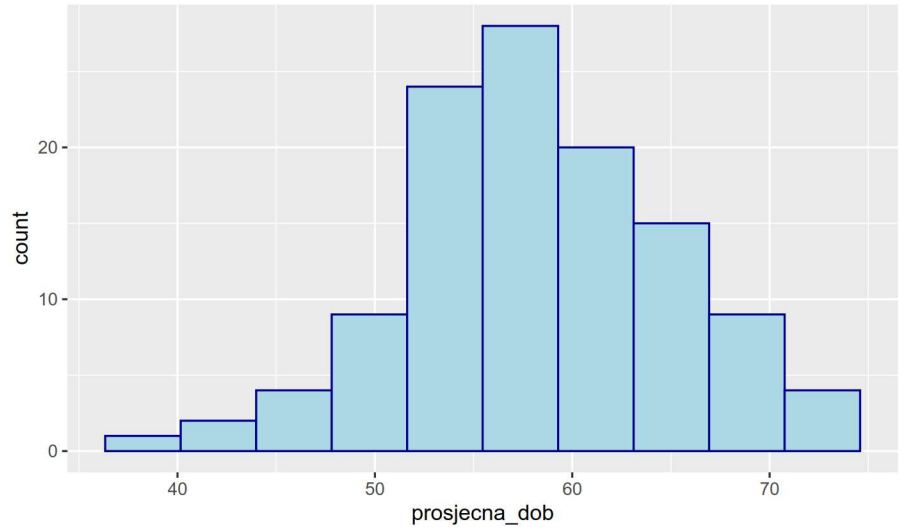


Slika 6: Stupčasti dijagram relativnih frekvencija kategorija prema spolu

Uočimo kako u ovom grafu promatramo relativne frekvencije, što smo postigli navođenjem *position = "fill"* unutar funkcije geom_bar().

Sljedeći tip grafa koji ćemo predstaviti je histogram, pogodan za grafički prikaz numeričkih podataka. Provjerimo pomoću histograma koja je prosječna dob dobitnika najzastupljenija među godinama dodjele. Funkcija koju pritom koristimo je geom_histogram().

```
nobel %>%
  group_by(year) %>%
  summarise(broj_dobitnika = n()) %>%
  ggplot(mapping = aes(broj_dobitnika)) +
  geom_histogram(color="darkblue", fill="lightblue", bins = 8)
```

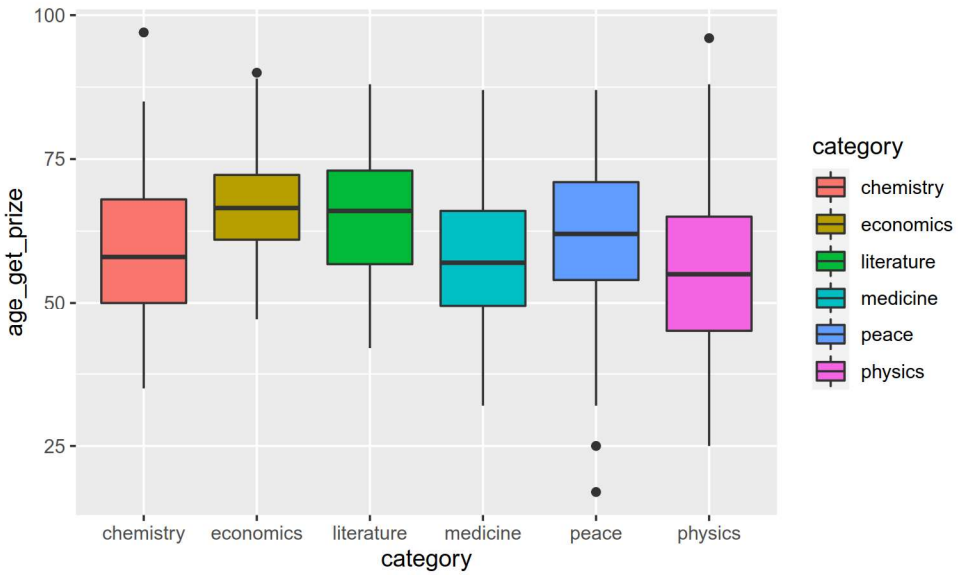


Slika 7: Histogram prosječne dobi dobitnika

Parametru bins prosljeđujemo željeni broj stupaca histograma. Visina pojedinog stupca odgovara frekvenciji pripadnog intervala pa vidimo kako su najbrojnije godine u kojima je prosječna dob dobitnika bila u intervalu od 55 do 60 godina.

Za kraj ćemo grafički prikazati odnos između numeričke i kvalitativne varijable, odnosno varijabli *age_get_prize* i *category*. U tome će nam poslužiti funkcija geom_boxplot() kojom dobivamo kutijasti dijagram.

```
nobel %>% ggplot(mapping = aes(x = category,
                               y = age_get_prize)) +
  geom_boxplot(aes(fill = category))
```



Slika 8: Kutijasti dijagram dobi dobitnika prema kategorijama

Gornje i donje stranice pravokutnika označavaju gornji, odnosno donji kvartil, a crta po pravokutniku označava medijan. Možemo naslutiti da su najmlađi među dobitnicima dobitnici nagrade iz fizike. Uočavamo i stršeće vrijednosti, prikazane izdvojenim točkama, među vrijednostima dobi dobitnika nagrade iz kemije, ekonomije, mira i fizike.

Detaljnije o grafičkom prikazivanju podataka može se pročitati u [1] (poglavlje 3).

Literatura

[1] Y. W. Huynh, *R for Graduate Students*, 2019.

[2] S. Locke, *Data Manipulation in R*, Locke Data Ltd, 2017.

[3] R. D. Peng, *R Programming for Data Science*, 2022.

[4] T. Timbers, T. Campbell, M. Lee, *Data Science: A First Introduction*, 2022.

[5] W. N. Venables, D. M. Smith, An Introduction to R, 2022.

[6] H. Wickham, G. Grolemund, *R for data science: import, tidy, transform, visualize, and model data*, O'Reilly Media, Inc., 2016.

[7] L. Wilkinson, *The Grammar of Graphics*, 1999.

[8] C. Wright, S. E. Ellis, S. C. Hicks, R. D. Peng, *Tidyverse Skills for Data Science*, 2021.

[9] https://github.com/BahramJannesar/NobelPrize/tree/master/dataset

[10] https://stat545.com/

[11] http://vita.had.co.nz/papers/layered-grammar.pdf

Sažetak

U ovom radu prolazimo kroz pojedine dijelove procesa analize podataka korištenjem programskog jezika R s alatima kolekcije paketa tidyverse. Na početku rada navodimo karakteristike spomenute kolekcije paketa i predstavljamo operator cjevovoda te novu strukturu podataka tibble. Nakon toga, bavimo se učitavanjem i uređivanjem podataka pomoću paketa readr i tidyr. Potom prikazujemo temeljne načine manipulacije podacima upravljanjem podatkovnim okvirom funkcijama iz paketa dplyr. U zadnjem dijelu rada upoznajemo se s pojmom grafičke gramatike te vizualiziramo podatke koristeći pritom napredni grafički paket ggplot2.

Ključne riječi: R, tidyverse, podatkovni okvir, operator cjevovoda, tibble, readr, tidyr, dplyr, vizualizacija podataka, ggplot2, grafička gramatika

R in data science

Summary

In this paper we deal with data analysis process using the programming language R with the tools of the package collection called tidyverse. At the beginning of the paper we list the characteristics of this package collection and introduce the so-called pipe operator, as well as the new data structure called tibble. After that, we deal with importing and tidying data using the readr and tidyr packages. Then we cover the fundamentals of data manipulation using functions from the dplyr package. Finally, we explain the concept of grammar of graphics and visualize data using the advanced visualization package ggplot2.

Key words: R, tidyverse, data frame, the pipe, tibble, readr, tidyr, dplyr, data visualization, ggplot2, the grammar of graphics

Životopis

Lara Juzbašić rođena je 23.08.1997. godine u Slavonskom Brodu. Pohađala je Osnovnu školu fra Bernardina Tome Leakovića u Bošnjacima. Nakon završenog osnovnoškolskog obrazovanja, 2012. godine upisuje Opću gimnaziju u Županji. Godine 2016. upisuje preddiplomski studij Matematike na Odjelu za matematiku u Osijeku te ga završava 2019. godine sa završnim radom „Spektralna teorija normalnih operatora", pod mentorstvom doc. dr. sc. Suzane Miodragović. Iste godine upisuje diplomski studij Financijska matematika i statistika na istom Odjelu.