Tjedan 3: Rad s podacima u tidyverse

Od sirovih podataka do analizi spremnog dataseta

2025-03-08

Table of contents

```
library(tidyverse)
library(janitor)
```

Ishodi učenja

Nakon ovog predavanja moći ćete

1. Objasniti zašto je čišćenje i transformacija podataka najvažniji (i najdugotrajniji) korak u svakoj analizi.
2. Koristiti `clean_names()` za standardizaciju imena stupaca i prepoznati zašto je

to važno za ponovljivost.
3. Koristiti `filter()` za odabir redova po jednom ili više uvjeta, uključujući kombinacije logičkih operatora i rad s nedostajućim vrijednostima.
4. Koristiti `select()` za odabir, preimenovanje i preuređivanje stupaca, uključujući pomoćne funkcije poput `starts_with()`, `ends_with()` i `contains()`.
5. Koristiti `mutate()` za kreiranje novih varijabli, transformaciju postojećih i rekodiranje vrijednosti pomoću `case_when()` i `if_else()`.
6. Koristiti `arrange()` za sortiranje podataka po jednom ili više stupaca u uzlaznom i silaznom redoslijedu.
7. Kombinirati dplyr glagole u pipeline koristeći pipe operator za složene transformacije podataka.
8. Prepoznati tipične probleme u sirovim podacima (nekonzistentno kodiranje, mješoviti tipovi, nedostajuće vrijednosti) i primijeniti odgovarajuće strategije čišćenja.

1 Prljava tajna analize podataka

Postoji jedna stvar o kojoj vam udžbenici statistike rijetko govore. Otvorite bilo koji udžbenik i vidjet ćete poglavlje o t-testu, poglavlje o regresiji, poglavlje o ANOVA-i. Sve lijepo i uredno. Ali nitko vam ne kaže da ćete 80% vremena u bilo kojoj analizi provesti na nečemu što se ne pojavljuje ni u jednom od tih poglavlja. Radi se o čišćenju i pripremi podataka.

Ovo nije pretjerivanje. Stvarni podaci su gotovo uvijek neuredni. Anketa prikupljena putem Google Formsa dolazi s imenima stupaca poput "Koliko često pratite vijesti na društvenim mrežama? (odaberite jedan odgovor)". Ispitanici u polje za spol upisuju "Ženski", "ženski", "Ž", "female" i "Zensko", a sve to treba biti ista kategorija. Stupac koji bi trebao sadržavati brojeve sadrži i tekst poput "ne gledam" ili prazne ćelije. Neki ispitanici imaju 19 godina, a jedan ima 199 jer mu je prst skliznuo na tipkovnici.

Sve ovo morate riješiti prije nego što možete izračunati ijedan prosjek ili napraviti ijedno testiranje hipoteza. I upravo zato je ovaj tjedan posvećen manipulaciji podacima. Naučit ćemo pet temeljnih funkcija iz paketa dplyr (`filter()`, `select()`, `mutate()`, `summarise()`, `group_by()`) plus alate za čišćenje i preoblikovanje iz paketa tidyr i janitor. Ove funkcije, spojene pipe operatorom u elegantne pipeline, čine okosnicu svake analize podataka u R-u.

Navarro u knjizi (poglavlja 4 i 7) pokriva sličan teren, ali u base R sintaksi. Mi ćemo koristiti tidyverse pristup koji je čitljiviji i konzistentniji. Kad ste jednom naučili logiku dplyr glagola, ista logika se primjenjuje na svaki dataset, svaki problem, svaku analizu.

---

2 Naši podaci: anketa o medijskim navikama studenata

Na ovom predavanju koristit ćemo simulirani dataset koji oponaša ono što biste zaista dobili kad biste proveli online anketu među studentima. Dataset je namjerno neuredan jer želimo vježbati čišćenje podataka na realističnom primjeru.

Učitajmo podatke i pogledajmo s čime se suočavamo.

```
raw <- read_csv("../resources/datasets/media_habits_raw.csv")
glimpse(raw)
```

```
Rows: 250
Columns: 17
$ `ID respondenta`                    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,~
$ Timestamp                           <dttm> 2025-03-28 17:05:00, 2025-04-
20 2~
$ Dob                                 <dbl> 20, 27, 27, 18, 25, 26, 28, 26, 22~
$ Spol                                <chr> "ženski", "Muški", "muški", "femal~
$ Grad                                <chr> "Zagreb", "Zadar", "Zagreb", "Spli~
$ `Godina studija`                    <chr> "2", "3", "1", "1", "2", "2.", "2"~
$ `TV (min/dan)`                      <chr> "0", "0", "65", NA, NA, "91", "91"~
$ `Portali (min/dan)`                 <dbl> 40, 20, 0, 11, 32, 25, 81, 28, 37,~
$ `Društvene mreže (min/dan)`         <dbl> 59, 101, 177, 71, 161, 155, 114, 1~
$ `Radio (min/dan)`                   <dbl> 49, NA, 0, NA, 26, NA, 0, 0, 17, 0~
$ `Podcast (min/dan)`                 <dbl> 89, 0, 49, 0, 0, NA, NA, 31, 0, 19~
$ `Povjerenje TV (1-10)`              <dbl> 2, 3, 4, 5, 5, 4, 3, 6, 6, 7, 2, 7~
$ `Povjerenje portali (1-10)`         <dbl> 6, 5, 6, 6, 3, 7, 7, 1, 7, 5, 6, 5~
$ `Povjerenje društvene mreže (1-10)` <dbl> 4, 3, 1, 4, 4, 7, 2, 3, 4, 6, 1, 2~
$ `Broj platformi`                    <dbl> 9, 5, 7, 6, 5, 2, 1, 8, 5, 7, 6, 6~
$ `Koje platforme koristi`            <chr> "Snapchat, WhatsApp, Facebook", "F~
$ `Koliko često prati vijesti`        <chr> "više puta dnevno", "nekoliko puta~
```

Već na prvi pogled vidimo nekoliko problema. Imena stupaca sadrže razmake, zagrade i dijakritičke znakove, što otežava rad u R-u. Stupci poput `Spol` imaju nekonzistentne vrijednosti. Stupac `TV (min/dan)` sadrži i brojeve i tekst ("ne gledam") i prazne ćelije, pa ga je R učitao kao tekst umjesto broja.

Pogledajmo prvih nekoliko redova detaljnije.

```
raw |>
  head(10)
```

```
# A tibble: 10 x 17
   `ID respondenta` Timestamp             Dob Spol   Grad     `Godina studija`
              <dbl> <dttm>              <dbl> <chr>  <chr>    <chr>
```

```
 1                1 2025-03-28 17:05:00    20 ženski Zagreb   2
 2                2 2025-04-20 21:11:00    27 Muški  Zadar    3
 3                3 2025-04-18 14:48:00    27 muški  Zagreb   1
 4                4 2025-03-28 12:54:00    18 female Split    1
 5                5 2025-03-21 18:06:00    25 Ženski Zagreb   2
 6                6 2025-04-14 20:26:00    26 M      Zagreb   2.
 7                7 2025-04-22 15:48:00    28 m      Zagreb   2
 8                8 2025-03-04 19:04:00    26 ženski Karlovac 2
 9                9 2025-03-12 12:17:00    22 female Split    2
10               10 2025-03-19 18:17:00    21 ž      Osijek   1
# i 11 more variables: `TV (min/dan)` <chr>, `Portali (min/dan)` <dbl>,
#   `Društvene mreže (min/dan)` <dbl>, `Radio (min/dan)` <dbl>,
#   `Podcast (min/dan)` <dbl>, `Povjerenje TV (1-10)` <dbl>,
#   `Povjerenje portali (1-10)` <dbl>,
#   `Povjerenje društvene mreže (1-10)` <dbl>, `Broj platformi` <dbl>,
#   `Koje platforme koristi` <chr>, `Koliko često prati vijesti` <chr>
```

Ovo je tipičan izgled sirovih podataka iz ankete. Prije bilo kakve analize, moramo napraviti čišćenje. Krenimo redom.

---

3 Korak nula: čišćenje imena stupaca

Prva stvar koju radimo sa svakim novim datasetom je standardizacija imena stupaca. Imena poput `TV (min/dan)` i `Povjerenje društvene mreže (1-10)` su problematična jer sadrže razmake, zagrade i specijalne znakove. Kad ih želite koristiti u kodu, morate ih stavljati u obrnute navodnike (poput `` `TV (min/dan)` ``). To je neugodno, nečitljivo i podložno greškama.

Paket janitor ima funkciju `clean_names()` koja automatski pretvara sva imena u snake_case format, uključujući mala slova, zamjenu razmaka podvlakama i uklanjanje specijalnih znakova.

```
raw <- raw |>
  clean_names()

names(raw)
```

```
 [1] "id_respondenta"                "timestamp"
 [3] "dob"                           "spol"
 [5] "grad"                          "godina_studija"
 [7] "tv_min_dan"                    "portali_min_dan"
 [9] "drustvene_mreze_min_dan"       "radio_min_dan"
```

```
[11] "podcast_min_dan"                  "povjerenje_tv_1_10"
[13] "povjerenje_portali_1_10"          "povjerenje_drustvene_mreze_1_10"
[15] "broj_platformi"                   "koje_platforme_koristi"
[17] "koliko_cesto_prati_vijesti"
```

Usporedite ova imena s originalnima. Umjesto `Povjerenje društvene mreže (1-10)` sada imamo `povjerenje_drustvene_mreze_1_10`. Duže jest, ali potpuno funkcionalno u R kodu bez ikakvih navodnika ili zagrada. Ovo je mala investicija koja štedi mnogo frustracije.

> **Praktični savjet**
>
> Navikajte se da `clean_names()` bude prva stvar koju pozovete nakon `read_csv()`. Možete to čak staviti u isti pipeline: `raw <- read_csv("datoteka.csv") |> clean_names()`. Ovo je toliko standardna praksa da mnogi R korisnici to rade automatski za svaki dataset, čak i kad su imena stupaca već uredna. Bolje spriječiti nego liječiti.

Sad kad imamo čista imena, možemo krenuti s pravim poslom. Pogledajmo strukturu nakon čišćenja.

```
glimpse(raw)
```

```
Rows: 250
Columns: 17
$ id_respondenta                  <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,~
$ timestamp                       <dttm> 2025-03-28 17:05:00, 2025-04-
20 21:11~
$ dob                             <dbl> 20, 27, 27, 18, 25, 26, 28, 26, 22, 21~
$ spol                            <chr> "ženski", "Muški", "muški", "female", ~
$ grad                            <chr> "Zagreb", "Zadar", "Zagreb", "Split", ~
$ godina_studija                  <chr> "2", "3", "1", "1", "2", "2.", "2", "2~
$ tv_min_dan                      <chr> "0", "0", "65", NA, NA, "91", "91", "0~
$ portali_min_dan                 <dbl> 40, 20, 0, 11, 32, 25, 81, 28, 37, 5, ~
$ drustvene_mreze_min_dan         <dbl> 59, 101, 177, 71, 161, 155, 114, 119, ~
$ radio_min_dan                   <dbl> 49, NA, 0, NA, 26, NA, 0, 0, 17, 0, 0,~
$ podcast_min_dan                 <dbl> 89, 0, 49, 0, 0, NA, NA, 31, 0, 19, 0,~
$ povjerenje_tv_1_10              <dbl> 2, 3, 4, 5, 5, 4, 3, 6, 6, 7, 2, 7, 7,~
$ povjerenje_portali_1_10         <dbl> 6, 5, 6, 6, 3, 7, 7, 1, 7, 5, 6, 5, 5,~
$ povjerenje_drustvene_mreze_1_10 <dbl> 4, 3, 1, 4, 4, 7, 2, 3, 4, 6, 1, 2, 3,~
$ broj_platformi                  <dbl> 9, 5, 7, 6, 5, 2, 1, 8, 5, 7, 6, 6, 2,~
$ koje_platforme_koristi          <chr> "Snapchat, WhatsApp, Facebook", "Faceb~
$ koliko_cesto_prati_vijesti      <chr> "više puta dnevno", "nekoliko puta tje~
```

4 filter() za odabir redova po uvjetu

Funkcija `filter()` je dplyr glagol za odabir redova koji zadovoljavaju jedan ili više uvjeta. Rezultat je tibble koji sadrži samo retke za koje su svi uvjeti TRUE. Redovi za koje je uvjet FALSE ili NA se odbacuju.

4.1 Osnovni uvjeti

```
# Samo ispitanici iz Zagreba
raw |>
  filter(grad == "Zagreb") |>
  nrow()
```

```
[1] 100
```

```
# Ispitanici mlađi od 21
raw |>
  filter(dob < 21) |>
  head(5)
```

```
# A tibble: 5 x 17
  id_respondenta timestamp             dob spol  grad  godina_studija tv_min_dan
           <dbl> <dttm>              <dbl> <chr> <chr> <chr>          <chr>
1              1 2025-03-28 17:05:00    20 žens~ Zagr~ 2              0
2              4 2025-03-28 12:54:00    18 fema~ Split 1              <NA>
3             13 2025-03-25 15:16:00    20 muški Zagr~ 1              0
4             14 2025-04-20 11:09:00    20 Žens~ Zagr~ 2              0
5             16 2025-04-07 18:06:00    19 muški Split 1              0
# i 10 more variables: portali_min_dan <dbl>, drustvene_mreze_min_dan <dbl>,
#   radio_min_dan <dbl>, podcast_min_dan <dbl>, povjerenje_tv_1_10 <dbl>,
#   povjerenje_portali_1_10 <dbl>, povjerenje_drustvene_mreze_1_10 <dbl>,
#   broj_platformi <dbl>, koje_platforme_koristi <chr>,
#   koliko_cesto_prati_vijesti <chr>
```

Svaki poziv `filter()` zapravo evaluira logički izraz za svaki redak. Za prvi primjer, R prolazi kroz svaki od 250 redova i provjerava je li vrijednost u stupcu `grad` jednaka "Zagreb". Retci za koje je odgovor TRUE ostaju, ostali nestaju.

4.2 Kombiniranje uvjeta

Snaga `filter()` dolazi do izražaja kad kombinirate više uvjeta. Unutar jednog `filter()` poziva, uvjeti odvojeni zarezom automatski se kombiniraju s I operatorom (`&`).

```
# Ispitanici iz Zagreba mlađi od 22
# Zarez između uvjeta je ekvivalentan &
raw |>
  filter(grad == "Zagreb", dob < 22) |>
  nrow()
```

```
[1] 41
```

```
# Isto kao:
raw |>
  filter(grad == "Zagreb" & dob < 22) |>
  nrow()
```

```
[1] 41
```

Oba pristupa daju identičan rezultat. Zarez je kraći za pisanje, `&` je eksplicitniji. Koristite što vam je čitljivije.

Za ILI uvjete, morate eksplicitno koristiti `|` operator.

```
# Ispitanici iz Zagreba ILI Splita
raw |>
  filter(grad == "Zagreb" | grad == "Split") |>
  count(grad)
```

```
# A tibble: 2 x 2
  grad       n
  <chr>  <int>
1 Split     44
2 Zagreb   100
```

```
# Elegantnije s %in%
raw |>
  filter(grad %in% c("Zagreb", "Split", "Rijeka")) |>
  count(grad, sort = TRUE)
```

```
# A tibble: 3 x 2
  grad       n
  <chr>  <int>
1 Zagreb   100
2 Split     44
3 Rijeka    18
```

Operator `%in%` smo upoznali prošli tjedan. U kontekstu `filter()` je izuzetno koristan jer zamjenjuje dugačke nizove ILI uvjeta jednim kompaktnim izrazom. Kad imate više od dvije kategorije, uvijek koristite `%in%`.

4.3 Filtriranje numeričkih raspona

```
# Ispitanici koji koriste društvene mreže između 60 i 180 minuta dnevno
raw |>
  filter(drustvene_mreze_min_dan >= 60, drustvene_mreze_min_dan <= 180) |>
  nrow()
```

```
[1] 202
```

```
# Alternativa s between()
raw |>
  filter(between(drustvene_mreze_min_dan, 60, 180)) |>
  nrow()
```

```
[1] 202
```

Funkcija `between(x, left, right)` je kratica za `x >= left & x <= right`. Oba pristupa daju isti rezultat, ali `between()` je čitljiviji kad filtrirate po rasponu.

4.4 Filtriranje teksta

Za tekstualne stupce, osim točnog podudaranja (`==`) i pripadnosti skupu (`%in%`), koristimo funkciju `str_detect()` iz paketa stringr (dio tidyverse) za pretraživanje po uzorku.

```
# Ispitanici čije platforme uključuju "Instagram" (bilo gdje u tekstu)
raw |>
  filter(str_detect(koje_platforme_koristi, "Instagram")) |>
  nrow()
```

```
[1] 39
```

```r
# Ispitanici koji prate vijesti barem jednom dnevno
raw |>
  filter(str_detect(koliko_cesto_prati_vijesti, "dnevno")) |>
  count(koliko_cesto_prati_vijesti)
```

```
# A tibble: 2 x 2
  koliko_cesto_prati_vijesti     n
  <chr>                      <int>
1 jednom dnevno                 60
2 više puta dnevno              81
```

Funkcija `str_detect()` vraća TRUE ako tekstualni uzorak postoji bilo gdje u vrijednosti. Ovo je mnogo fleksibilnije od `==` jer ne zahtijeva točno podudaranje. Na primjer, `str_detect(x, "dnevno")` hvata i "više puta dnevno" i "jednom dnevno".

4.5 filter() i nedostajuće vrijednosti

Važno svojstvo `filter()` je da **automatski odbacuje retke s NA u uvjetu**. Ovo je uglavnom poželjno ponašanje, ali morate biti svjesni da se događa.

```r
# Koliko redova imamo ukupno?
nrow(raw)
```

```
[1] 250
```

```r
# Koliko ima NA u stupcu radio_min_dan?
sum(is.na(raw$radio_min_dan))
```

```
[1] 32
```

```r
# filter s numeričkim uvjetom na stupcu s NA
raw |>
  filter(radio_min_dan > 0) |>
  nrow()
```

```
[1] 92
```

Rezultat ne uključuje retke s NA u stupcu `radio_min_dan`. Ako želite eksplicitno zadržati retke s NA, morate to navesti.

```r
# Zadrži retke gdje je radio > 0 ILI je NA
raw |>
  filter(radio_min_dan > 0 | is.na(radio_min_dan)) |>
  nrow()
```

```
[1] 124
```

```r
# Zadrži SAMO retke s NA
raw |>
  filter(is.na(radio_min_dan)) |>
  nrow()
```

```
[1] 32
```

```r
# Izbaci retke s NA (zadrži samo kompletne)
raw |>
  filter(!is.na(radio_min_dan)) |>
  nrow()
```

```
[1] 218
```

Kombinacija `filter(!is.na(stupac))` je način da zadržite samo retke s poznatim vrijednostima u tom stupcu. Alternativno, funkcija `drop_na()` iz paketa tidyr uklanja retke koji imaju NA u bilo kojem stupcu (ili u specificiranim stupcima).

```r
# Ukloni retke s NA u specifičnom stupcu
raw |>
  drop_na(radio_min_dan) |>
  nrow()
```

```
[1] 218
```

```r
# Ukloni retke s NA u BILO KOJEM stupcu (agresivno!)
raw |>
  drop_na() |>
  nrow()
```

```
[1] 149
```

Primijetite drastičnu razliku. Kad koristimo `drop_na()` bez argumenata, gubimo mnogo redova jer se uklanjaju svi retci koji imaju NA u ijednom stupcu. U praksi, `drop_na()` bez argumenata se rijetko koristi jer je previše agresivan. Bolje je ciljano raditi s NA u stupcima koji vas zapravo zanimaju.

> **!** Važna napomena
>
> Svaki put kad koristite `filter()` ili `drop_na()`, dokumentirajte koliko redova ste izgubili i zašto. Ako ste od 250 ispitanika zadržali samo 150, to je informacija koju morate navesti u metodološkom dijelu rada. Čitatelj mora znati na koliko se opažanja vaši rezultati temelje i zašto su neka isključena.

---

5 select() za odabir i preimenovanje stupaca

Dok `filter()` radi s redovima, `select()` radi sa stupcima. Koristi se za tri svrhe, a to su odabir stupaca koji vam trebaju, uklanjanje stupaca koji vam ne trebaju i preimenovanje stupaca.

5.1 Odabir po imenu

```
# Odabir specifičnih stupaca
raw |>
  select(id_respondenta, dob, spol, grad) |>
  head(5)
```

```
# A tibble: 5 x 4
  id_respondenta   dob spol   grad
           <dbl> <dbl> <chr>  <chr>
1              1    20 ženski Zagreb
2              2    27 Muški  Zadar
3              3    27 muški  Zagreb
4              4    18 female Split
5              5    25 Ženski Zagreb
```

```
# Odabir raspona stupaca (od do)
raw |>
  select(id_respondenta:godina_studija) |>
  head(5)
```

```
# A tibble: 5 x 6
  id_respondenta timestamp             dob spol   grad   godina_studija
           <dbl> <dttm>              <dbl> <chr>  <chr>  <chr>
1              1 2025-03-28 17:05:00    20 ženski Zagreb 2
```

```
2              2 2025-04-20 21:11:00    27 Muški  Zadar  3
3              3 2025-04-18 14:48:00    27 muški  Zagreb 1
4              4 2025-03-28 12:54:00    18 female Split  1
5              5 2025-03-21 18:06:00    25 Ženski Zagreb 2
```

Stupce navodite po imenu, bez navodnika. Operator `:` bira sve stupce između dva navedena, uključujući oba krajnja. Ovo je praktično kad su relevantni stupci jedan do drugoga u datasetu.

5.2 Uklanjanje stupaca

Minus ispred imena stupca znači "sve osim ovoga".

```
# Sve osim timestampa i ID-a
raw |>
  select(-timestamp, -id_respondenta) |>
  names()
```

```
 [1] "dob"                              "spol"
 [3] "grad"                             "godina_studija"
 [5] "tv_min_dan"                       "portali_min_dan"
 [7] "drustvene_mreze_min_dan"          "radio_min_dan"
 [9] "podcast_min_dan"                  "povjerenje_tv_1_10"
[11] "povjerenje_portali_1_10"          "povjerenje_drustvene_mreze_1_10"
[13] "broj_platformi"                   "koje_platforme_koristi"
[15] "koliko_cesto_prati_vijesti"
```

```
# Uklanjanje raspona
raw |>
  select(-(povjerenje_tv_1_10:povjerenje_drustvene_mreze_1_10)) |>
  names()
```

```
 [1] "id_respondenta"              "timestamp"
 [3] "dob"                         "spol"
 [5] "grad"                        "godina_studija"
 [7] "tv_min_dan"                  "portali_min_dan"
 [9] "drustvene_mreze_min_dan"     "radio_min_dan"
[11] "podcast_min_dan"             "broj_platformi"
[13] "koje_platforme_koristi"      "koliko_cesto_prati_vijesti"
```

5.3 Pomoćne funkcije za odabir

Kad imate mnogo stupaca, ručno nabrajanje postaje nepraktično. dplyr nudi pomoćne funkcije za pametni odabir.

```
# Stupci čije ime počinje s "povjerenje"
raw |>
  select(starts_with("povjerenje")) |>
  head(3)
```

```
# A tibble: 3 x 3
  povjerenje_tv_1_10 povjerenje_portali_1_10 povjerenje_drustvene_mreze_1_10
               <dbl>                   <dbl>                            <dbl>
1                  2                       6                                4
2                  3                       5                                3
3                  4                       6                                1
```

```
# Stupci čije ime završava s "dan"
raw |>
  select(ends_with("dan")) |>
  head(3)
```

```
# A tibble: 3 x 5
  tv_min_dan portali_min_dan drustvene_mreze_min_dan radio_min_dan
  <chr>                <dbl>                   <dbl>         <dbl>
1 0                       40                      59            49
2 0                       20                     101            NA
3 65                       0                     177             0
# i 1 more variable: podcast_min_dan <dbl>
```

```
# Stupci čije ime sadrži "min"
raw |>
  select(contains("min")) |>
  head(3)
```

```
# A tibble: 3 x 5
  tv_min_dan portali_min_dan drustvene_mreze_min_dan radio_min_dan
  <chr>                <dbl>                   <dbl>         <dbl>
1 0                       40                      59            49
2 0                       20                     101            NA
3 65                       0                     177             0
# i 1 more variable: podcast_min_dan <dbl>
```

```
# Samo numerički stupci
raw |>
  select(where(is.numeric)) |>
  head(3)
```

```
# A tibble: 3 x 10
  id_respondenta   dob portali_min_dan drustvene_mreze_min_dan radio_min_dan
           <dbl> <dbl>           <dbl>                   <dbl>         <dbl>
1              1    20              40                      59            49
2              2    27              20                     101            NA
3              3    27               0                     177             0
# i 5 more variables: podcast_min_dan <dbl>, povjerenje_tv_1_10 <dbl>,
#   povjerenje_portali_1_10 <dbl>, povjerenje_drustvene_mreze_1_10 <dbl>,
#   broj_platformi <dbl>
```

Funkcija `starts_with()` bira stupce čije ime počinje zadanim tekstom. `ends_with()` bira po završetku. `contains()` traži tekst bilo gdje u imenu. `where()` prima funkciju za provjeru tipa i bira stupce koji zadovoljavaju taj uvjet. Ove funkcije postaju neprocjenjive kad radite s datasetima koji imaju 50 ili 100 stupaca (što nije neuobičajeno u anketnim istraživanjima).

5.4 Preimenovanje stupaca

Unutar `select()` možete preimenovati stupac sintaksom `novo_ime = staro_ime`. Ili koristite zasebnu funkciju `rename()` koja preimenu stupce ali zadrži sve ostale.

```
# Preimenovanje unutar select (odabire SAMO navedene stupce)
raw |>
  select(
    id = id_respondenta,
    dob,
    spol,
    sm_minuta = drustvene_mreze_min_dan
  ) |>
  head(3)
```

```
# A tibble: 3 x 4
     id   dob spol    sm_minuta
  <dbl> <dbl> <chr>       <dbl>
1     1    20 ženski         59
2     2    27 Muški         101
3     3    27 muški         177
```

```
# rename() mijenja imena ali zadržava sve stupce
raw |>
  rename(
    id = id_respondenta,
    sm_minuta = drustvene_mreze_min_dan
  ) |>
  names()
```

```
 [1] "id"                                "timestamp"
 [3] "dob"                               "spol"
 [5] "grad"                              "godina_studija"
 [7] "tv_min_dan"                        "portali_min_dan"
 [9] "sm_minuta"                         "radio_min_dan"
[11] "podcast_min_dan"                   "povjerenje_tv_1_10"
[13] "povjerenje_portali_1_10"           "povjerenje_drustvene_mreze_1_10"
[15] "broj_platformi"                    "koje_platforme_koristi"
[17] "koliko_cesto_prati_vijesti"
```

Razlika je važna. `select()` s preimenovanjem zadržava samo stupce koje ste naveli. `rename()` zadržava sve stupce i samo mijenja imena onih koje ste specificirali. U praksi, `rename()` je sigurniji izbor kad želite samo promijeniti ime jednog ili dva stupca bez gubitka ostalih.

5.5 Preuređivanje stupaca

Funkcija `relocate()` premješta stupce na drugu poziciju u datasetu.

```
# Premjesti grad na početak (odmah nakon ID-a)
raw |>
  relocate(grad, .after = id_respondenta) |>
  head(3)
```

```
# A tibble: 3 x 17
  id_respondenta grad  timestamp             dob spol  godina_studija tv_min_dan
           <dbl> <chr> <dttm>              <dbl> <chr> <chr>          <chr>
1              1 Zagr~ 2025-03-28 17:05:00    20 žens~ 2              0
2              2 Zadar 2025-04-20 21:11:00    27 Muški 3              0
3              3 Zagr~ 2025-04-18 14:48:00    27 muški 1              65
# i 10 more variables: portali_min_dan <dbl>, drustvene_mreze_min_dan <dbl>,
#   radio_min_dan <dbl>, podcast_min_dan <dbl>, povjerenje_tv_1_10 <dbl>,
#   povjerenje_portali_1_10 <dbl>, povjerenje_drustvene_mreze_1_10 <dbl>,
#   broj_platformi <dbl>, koje_platforme_koristi <chr>,
#   koliko_cesto_prati_vijesti <chr>
```

```r
# Premjesti sve numeričke stupce na kraj
raw |>
  relocate(where(is.numeric), .after = last_col()) |>
  head(3)
```

```
# A tibble: 3 x 17
  timestamp           spol   grad   godina_studija tv_min_dan
  <dttm>              <chr>  <chr>  <chr>          <chr>
1 2025-03-28 17:05:00 ženski Zagreb 2              0
2 2025-04-20 21:11:00 Muški  Zadar  3              0
3 2025-04-18 14:48:00 muški  Zagreb 1              65
# i 12 more variables: koje_platforme_koristi <chr>,
#   koliko_cesto_prati_vijesti <chr>, id_respondenta <dbl>, dob <dbl>,
#   portali_min_dan <dbl>, drustvene_mreze_min_dan <dbl>, radio_min_dan <dbl>,
#   podcast_min_dan <dbl>, povjerenje_tv_1_10 <dbl>,
#   povjerenje_portali_1_10 <dbl>, povjerenje_drustvene_mreze_1_10 <dbl>,
#   broj_platformi <dbl>
```

`relocate()` ne dodaje niti uklanja stupce, samo ih premješta. Ovo je korisno za organizaciju dataseta kad želite da relevantni stupci budu jedni do drugih.

---

6 mutate() za kreiranje i transformaciju varijabli

Funkcija `mutate()` je najsvestraniji dplyr glagol. Služi za kreiranje novih stupaca na temelju postojećih, transformaciju postojećih stupaca i rekodiranje vrijednosti. Rezultat je tibble s istim brojem redova ali potencijalno novim ili izmijenjenim stupcima.

6.1 Kreiranje novih varijabli

```r
# Ukupno dnevno korištenje medija (portal + društvene mreže)
raw |>
  mutate(
    ukupno_digital = portali_min_dan + drustvene_mreze_min_dan
  ) |>
  select(id_respondenta, portali_min_dan, drustvene_mreze_min_dan, ukupno_digital) |>
  head(8)
```

```
# A tibble: 8 x 4
  id_respondenta portali_min_dan drustvene_mreze_min_dan ukupno_digital
           <dbl>           <dbl>                   <dbl>          <dbl>
1              1              40                      59             99
2              2              20                     101            121
3              3               0                     177            177
4              4              11                      71             82
5              5              32                     161            193
6              6              25                     155            180
7              7              81                     114            195
8              8              28                     119            147
```

`mutate()` evaluira izraz na desnoj strani znaka jednakosti za svaki redak i rezultat pohranjuje u novi stupac nazvan imenom na lijevoj strani. Kao i kod vektoriziranih operacija, R automatski primjenjuje operaciju redak po redak.

Možete kreirati više stupaca u jednom `mutate()` pozivu, i kasniji stupci mogu koristiti ranije definirane.

```
raw |>
  mutate(
    ukupno_digital = portali_min_dan + drustvene_mreze_min_dan,
    ukupno_sati = ukupno_digital / 60,
    iznad_2_sata = ukupno_sati > 2
  ) |>
  select(id_respondenta, ukupno_digital, ukupno_sati, iznad_2_sata) |>
  head(8)
```

```
# A tibble: 8 x 4
  id_respondenta ukupno_digital ukupno_sati iznad_2_sata
           <dbl>          <dbl>       <dbl> <lgl>
1              1             99        1.65 FALSE
2              2            121        2.02 TRUE
3              3            177        2.95 TRUE
4              4             82        1.37 FALSE
5              5            193        3.22 TRUE
6              6            180        3    TRUE
7              7            195        3.25 TRUE
8              8            147        2.45 TRUE
```

Primijetite da smo u istom `mutate()` pozivu najprije izračunali `ukupno_digital`, zatim ga koristili za izračun `ukupno_sati`, a onda `ukupno_sati` za logički stupac `iznad_2_sata`. Ova mogućnost referiranja na upravo kreirane stupce čini `mutate()` izuzetno moćnim.

6.2 Transformacija postojećih stupaca

`mutate()` može i prepisati postojeći stupac.

```
# Zaokruži portal minute na desetice (prepisuje stupac)
raw |>
  mutate(
    portali_min_dan = round(portali_min_dan, -1)
  ) |>
  select(id_respondenta, portali_min_dan) |>
  head(8)
```

```
# A tibble: 8 x 2
  id_respondenta portali_min_dan
           <dbl>           <dbl>
1              1              40
2              2              20
3              3               0
4              4              10
5              5              30
6              6              20
7              7              80
8              8              30
```

Kad date mutate stupcu isto ime kao postojeći stupac, novi vrijednosti zamjenjuju stare. Ovo je korisno za čišćenje podataka (na primjer, pretvorbu teksta u mala slova), ali budite oprezni jer originalne vrijednosti nestaju. Dobra praksa je raditi transformacije na kopiji dataseta, ne na originalu.

6.3 Čišćenje stupca spol pomoću str_to_lower() i case_when()

Pogledajmo koliko je neuredan stupac `spol` u našim podacima.

```
raw |>
  count(spol, sort = TRUE)
```

```
# A tibble: 12 x 2
   spol       n
   <chr>  <int>
 1 Muški     48
 2 Ženski    45
 3 muški     42
 4 ženski    31
```

```
 5 Ž          16
 6 male       14
 7 M          11
 8 Musko      11
 9 ž          10
10 m           9
11 Zensko      7
12 female      6
```

Imamo dvanaestak varijanti istih dviju kategorija. “Ženski”, “ženski”, “Ž”, “ž”, “Zensko”, “female” bi sve trebalo biti jedna kategorija. Ovo je klasičan problem u anketnim podacima i jedan od najčešćih razloga za čišćenje.

Funkcija `case_when()` je najfleksibilniji alat za rekodiranje. Radi kao niz IF-THEN pravila. Za svaki redak, R provjerava uvjete redom i dodjeljuje vrijednost prvog uvjeta koji je ispunjen.

```
raw <- raw |>
  mutate(
    spol_clean = case_when(
      str_to_lower(spol) %in% c("ženski", "ž", "zensko", "female") ~ "ženski",
      str_to_lower(spol) %in% c("muški", "m", "musko", "male") ~ "muški",
      .default = "ostalo"
    )
  )

raw |>
  count(spol_clean)
```

```
# A tibble: 2 x 2
  spol_clean     n
  <chr>      <int>
1 muški        135
2 ženski       115
```

Raščlanimo ovaj kod. Funkcija `str_to_lower()` pretvara tekst u mala slova, čime eliminiramo razliku između “Ženski” i “ženski”. Zatim `%in%` provjerava pripada li vrijednost jednom od navedenih oblika. Ako da, dodjeljuje standardizirani oblik. Argument `.default` hvata sve što ne odgovara nijednom uvjetu.

Rezultat je čist stupac `spol_clean` s tri konzistentne kategorije umjesto dvanaest neujednačenih varijanti.

Praktični savjet

Kad čistite tekstualne podatke, uvijek najprije pretvorite u mala slova pomoću `str_to_lower()`. Ovo odmah eliminira najčešći izvor nekonzistentnosti (razliku u kapitalizaciji) i smanjuje broj slučajeva koje morate pokriti u `case_when()`. Redoslijed je važan. Najprije trebate koristiti `str_to_lower()`, pa tek onda provjerite uvjete.

6.4 Rekodiranje numeričkih varijabli u kategorije

Čest zadatak u komunikologiji je pretvaranje kontinuirane varijable u kategorije. Na primjer, umjesto točne dobi, želimo dobne skupine.

```
raw <- raw |>
  mutate(
    dobna_skupina = case_when(
      dob < 20 ~ "18-19",
      dob < 22 ~ "20-21",
      dob < 24 ~ "22-23",
      dob >= 24 ~ "24+"
    )
  )

raw |>
  count(dobna_skupina)
```

```
# A tibble: 4 x 2
  dobna_skupina     n
  <chr>         <int>
1 18-19            73
2 20-21            61
3 22-23            47
4 24+              69
```

Redoslijed uvjeta u `case_when()` je bitan. R provjerava uvjete odozgo prema dolje i dodjeljuje vrijednost prvog ispunjenog uvjeta. Ako osoba ima 19 godina, prvi uvjet (`dob < 20`) je TRUE i dodjeljuje se "18-19". R ne provjerava preostale uvjete. Zato uvjete postavljamo od najspecifičnijeg prema najopćenitijem.

6.5 if_else(): binarno rekodiranje

Za jednostavne da/ne situacije, `if_else()` je kraći od `case_when()`.

```r
raw <- raw |>
  mutate(
    visoko_koristenje_sm = if_else(drustvene_mreze_min_dan > 120, "visoko", "nisko/umjeren
    prati_vijesti_cesto = if_else(
      koliko_cesto_prati_vijesti %in% c("više puta dnevno", "jednom dnevno"),
      TRUE,
      FALSE
    )
  )

raw |>
  count(visoko_koristenje_sm)
```

```
# A tibble: 2 x 2
  visoko_koristenje_sm     n
  <chr>                <int>
1 nisko/umjereno         117
2 visoko                 133
```

Funkcija `if_else()` prima tri argumenta, uključujući uvjet, vrijednost za TRUE i vrijednost za FALSE. Prednost nad base R `ifelse()` je što `if_else()` strogo provjerava tipove i daje razumljivije greške kad nešto ne štima.

6.6 Čišćenje stupca godina studija

Pogledajmo još jedan neuredan stupac.

```r
raw |>
  count(godina_studija, sort = TRUE)
```

```
# A tibble: 11 x 2
   godina_studija     n
   <chr>          <int>
 1 2                 54
 2 1                 52
 3 3                 38
 4 3.                22
 5 1.                19
 6 4                 19
 7 druga             14
 8 5                 12
 9 2.                 7
10 treća              7
11 prva               6
```

Imamo “1”, “1.”, “prva”, “2”, “2.”, “druga” i tako dalje. Sve to treba svesti na konzistentne brojeve.

```
raw <- raw |>
  mutate(
    godina_clean = case_when(
      str_to_lower(godina_studija) %in% c("1", "1.", "prva") ~ 1,
      str_to_lower(godina_studija) %in% c("2", "2.", "druga") ~ 2,
      str_to_lower(godina_studija) %in% c("3", "3.", "treća", "treca") ~ 3,
      str_to_lower(godina_studija) %in% c("4", "4.", "četvrta", "cetvrta") ~ 4,
      str_to_lower(godina_studija) %in% c("5", "5.", "peta") ~ 5,
      .default = NA_real_
    )
  )

raw |>
  count(godina_clean)
```

```
# A tibble: 5 x 2
  godina_clean     n
         <dbl> <int>
1            1    77
2            2    75
3            3    67
4            4    19
5            5    12
```

Ovaj put smo neprepoznate vrijednosti kodirali kao `NA_real_` (NA numeričkog tipa) umjesto tekstualne kategorije. To je ispravniji pristup kad očekujemo numerički rezultat. Ako neka vrijednost ne odgovara nijednom poznatom obrascu, bolje je eksplicitno reći “ne znam” (NA) nego nagađati.

6.7 Rad s problematičnim numeričkim stupcima

Prisjetimo se da je stupac `tv_min_dan` učitan kao tekst jer sadrži i brojeve i tekst (“ne gledam”) i prazne ćelije. Moramo ga pretvoriti u broj.

```
# Pogledajmo problematične vrijednosti
raw |>
  count(tv_min_dan, sort = TRUE) |>
  head(15)
```

```
# A tibble: 15 x 2
```

```
   tv_min_dan     n
   <chr>      <int>
 1 0             85
 2 <NA>          41
 3 ne gledam      6
 4 71             4
 5 10             3
 6 112            3
 7 119            3
 8 26             3
 9 48             3
10 49             3
11 51             3
12 7              3
13 76             3
14 82             3
15 104            2
```

```
# "ne gledam" tretiramo kao 0, prazne kao NA
raw <- raw |>
  mutate(
    tv_minuta = case_when(
      tv_min_dan == "ne gledam" ~ 0,
      tv_min_dan == "" ~ NA_real_,
      .default = as.numeric(tv_min_dan)
    )
  )

# Provjera
raw |>
  select(tv_min_dan, tv_minuta) |>
  filter(is.na(tv_minuta) | tv_minuta == 0) |>
  head(10)
```

```
# A tibble: 10 x 2
   tv_min_dan tv_minuta
   <chr>          <dbl>
 1 0                  0
 2 0                  0
 3 <NA>              NA
 4 <NA>              NA
 5 0                  0
 6 <NA>              NA
 7 0                  0
 8 0                  0
```

```
 9 0                    0
10 <NA>                NA
```

Sada imamo čist numerički stupac `tv_minuta` u kojem je "ne gledam" pretvoreno u 0 (jer osoba zaista ne gleda TV, dakle 0 minuta), a prazne ćelije su NA (jer ne znamo koliko ta osoba gleda TV).

Ova razlika između 0 i NA je konceptualno važna i vraća nas na tipove nedostajućih vrijednosti koje smo spominjali u tjednu 2. Nula znači "znamo odgovor, i odgovor je ništa". NA znači "ne znamo odgovor".

---

7 arrange(): sortiranje podataka

Funkcija `arrange()` sortira retke po vrijednostima jednog ili više stupaca. Po defaultu sortira uzlazno (od najmanjeg prema najvećem ili abecedno). Za silazno sortiranje koristimo `desc()`.

```
# Sortirano po dobi (najmlađi prvi)
raw |>
  select(id_respondenta, dob, grad, drustvene_mreze_min_dan) |>
  arrange(dob) |>
  head(8)
```

```
# A tibble: 8 x 4
  id_respondenta   dob grad    drustvene_mreze_min_dan
           <dbl> <dbl> <chr>                     <dbl>
1              4    18 Split                        71
2             23    18 Split                       145
3             25    18 Osijek                      184
4             31    18 Zagreb                       97
5             39    18 Šibenik                     154
6             44    18 Zagreb                      152
7             45    18 Pula                         86
8             57    18 Zagreb                      192
```

```
# Sortirano po korištenju društvenih mreža (najviše korištenja prvo)
raw |>
  select(id_respondenta, dob, grad, drustvene_mreze_min_dan) |>
  arrange(desc(drustvene_mreze_min_dan)) |>
  head(8)
```

```
# A tibble: 8 x 4
  id_respondenta   dob grad           drustvene_mreze_min_dan
           <dbl> <dbl> <chr>                            <dbl>
1            149    22 Zadar                              296
2             21    21 Slavonski Brod                     271
3             42    25 Zagreb                             246
4             87    18 Zagreb                             246
5            141    23 Split                              222
6            181    22 Split                              217
7             16    19 Split                              214
8             79    22 Rijeka                             206
```

7.1 Sortiranje po više stupaca

Kad sortirate po više stupaca, prvi stupac ima prioritet. Unutar istih vrijednosti prvog stupca, koristi se drugi za razrješenje.

```
# Sortiraj po gradu (abecedno), unutar grada po dobi (silazno)
raw |>
  select(id_respondenta, grad, dob, drustvene_mreze_min_dan) |>
  arrange(grad, desc(dob)) |>
  head(12)
```

```
# A tibble: 12 x 4
   id_respondenta grad        dob drustvene_mreze_min_dan
            <dbl> <chr>     <dbl>                   <dbl>
 1            241 Dubrovnik    27                     165
 2            198 Dubrovnik    25                      99
 3             34 Dubrovnik    23                     151
 4            222 Dubrovnik    21                     109
 5            243 Dubrovnik    20                     117
 6            129 Dubrovnik    19                     203
 7             75 Karlovac     28                     115
 8              8 Karlovac     26                     119
 9            229 Karlovac     24                     134
10             15 Karlovac     23                      35
11             73 Karlovac     21                      20
12             95 Karlovac     20                     147
```

Vidimo da su najprije svi ispitanici iz Dubrovnika (abecedno prvi), unutar kojih je najstariji na vrhu. Zatim dolazi Karlovac, pa dalje.

7.2 Gdje se NA pojavljuje pri sortiranju?

```
# NA vrijednosti uvijek idu na kraj, bez obzira na smjer
raw |>
  select(id_respondenta, radio_min_dan) |>
  arrange(radio_min_dan) |>
  tail(8)
```

```
# A tibble: 8 x 2
  id_respondenta radio_min_dan
           <dbl>         <dbl>
1            207            NA
2            211            NA
3            217            NA
4            218            NA
5            219            NA
6            220            NA
7            233            NA
8            244            NA
```

R stavlja NA na kraj sortiranog dataseta, neovisno o tome sortirate li uzlazno ili silazno. Ovo je korisno znati jer ćete ponekad htjeti vidjeti retke s nedostajućim vrijednostima, a oni su uvijek na dnu.

---

8 Kombiniranje glagola u pipeline

Prava snaga dplyr-a nije u pojedinačnim glagolima nego u njihovoj kombinaciji. Pipe operator (`|>`) omogućuje ulančavanje operacija u jednu koherentnu sekvencu koja čita dataseta od početka do kraja, korak po korak.

Pogledajmo realistični primjer. Želimo odgovoriti na pitanje te saznati koji gradovi imaju studente koji najviše koriste društvene mreže.

```
raw |>
  filter(dob >= 18, dob <= 25) |>
  select(grad, drustvene_mreze_min_dan) |>
  group_by(grad) |>
  summarise(
    n = n(),
    prosjek = round(mean(drustvene_mreze_min_dan), 1),
```

```
    .groups = "drop"
  ) |>
  filter(n >= 5) |>
  arrange(desc(prosjek))
```

```
# A tibble: 11 x 3
   grad               n prosjek
   <chr>          <int>   <dbl>
 1 Slavonski Brod     8   141.
 2 Zadar             11   137.
 3 Dubrovnik          5   136.
 4 Pula               9   134.
 5 Split             40   133.
 6 Šibenik            9   127.
 7 Osijek            18   124.
 8 Zagreb            79   116.
 9 Rijeka            18   112.
10 Varaždin           5    79.6
11 Karlovac           6    79.3
```

Čitamo odozgo prema dolje. Uzmi sirove podatke. Zadrži samo ispitanike između 18 i 25 godina. Odaberi samo stupce za grad i minute korištenja. Grupiraj po gradu. Izračunaj broj ispitanika i prosječno korištenje za svaki grad. Zadrži samo gradove s barem 5 ispitanika (da prosjeci budu smisleni). Sortiraj po prosječnom korištenju od najvišeg prema najnižem.

Svaki korak je sam za sebe jasan, a zajedno tvore kompletnu analizu. Ovo je radni obrazac koji ćete koristiti stotine puta.

Pogledajmo drugi primjer. Trebamo profil tipičnog korisnika koji često prati vijesti.

```
raw |>
  filter(prati_vijesti_cesto == TRUE) |>
  summarise(
    n = n(),
    prosjek_dob = round(mean(dob), 1),
    prosjek_sm_min = round(mean(drustvene_mreze_min_dan), 1),
    prosjek_portal_min = round(mean(portali_min_dan), 1),
    prosjek_trust_portal = round(mean(povjerenje_portali_1_10), 1),
    prosjek_trust_sm = round(mean(povjerenje_drustvene_mreze_1_10), 1)
  )
```

```
# A tibble: 1 x 6
      n prosjek_dob prosjek_sm_min prosjek_portal_min prosjek_trust_portal
  <int>       <dbl>          <dbl>              <dbl>                <dbl>
1   141        21.9           122.               44.2                  4.9
# i 1 more variable: prosjek_trust_sm <dbl>
```

Ljudi koji prate vijesti barem jednom dnevno provode određen broj minuta na portalima i društvenim mrežama te imaju specifičan profil povjerenja u različite medije. Ova tablica daje bogat uvid u jednom pipeline.

8.1 Pipeline za čišćenje podataka

Uobičajena praksa je napisati jedan veliki pipeline za čišćenje koji pretvara sirove podatke u analizi spreman dataset. Evo kako bi to izgledalo za naše podatke.

```
clean <- raw |>
  # Preimenovanje stupaca za čitljivost
  rename(
    id = id_respondenta,
    sm_min = drustvene_mreze_min_dan,
    portal_min = portali_min_dan,
    trust_tv = povjerenje_tv_1_10,
    trust_portal = povjerenje_portali_1_10,
    trust_sm = povjerenje_drustvene_mreze_1_10,
    n_platformi = broj_platformi,
    vijesti_frekvencija = koliko_cesto_prati_vijesti
  ) |>
  # Korištenje već očišćenih stupaca
  select(
    id, dob, spol_clean, grad, godina_clean,
    tv_minuta, portal_min, sm_min, radio_min_dan, podcast_min_dan,
    trust_tv, trust_portal, trust_sm,
    n_platformi, vijesti_frekvencija,
    dobna_skupina, visoko_koristenje_sm, prati_vijesti_cesto
  ) |>
  # Završno preimenovanje čistih stupaca
  rename(
    spol = spol_clean,
    godina = godina_clean,
    radio_min = radio_min_dan,
    podcast_min = podcast_min_dan
  )

glimpse(clean)
```

```
Rows: 250
Columns: 18
$ id                  <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15~
$ dob                 <dbl> 20, 27, 27, 18, 25, 26, 28, 26, 22, 21, 22, 27, 2~
$ spol                <chr> "ženski", "muški", "muški", "ženski", "ženski", "~
```

```
$ grad                <chr> "Zagreb", "Zadar", "Zagreb", "Split", "Zagreb", "~
$ godina              <dbl> 2, 3, 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1~
$ tv_minuta           <dbl> 0, 0, 65, NA, NA, 91, 91, 0, 76, 66, NA, 109, 0, ~
$ portal_min          <dbl> 40, 20, 0, 11, 32, 25, 81, 28, 37, 5, 38, 44, 26,~
$ sm_min              <dbl> 59, 101, 177, 71, 161, 155, 114, 119, 56, 40, 129~
$ radio_min           <dbl> 49, NA, 0, NA, 26, NA, 0, 0, 17, 0, 0, 13, 0, 0, ~
$ podcast_min         <dbl> 89, 0, 49, 0, 0, NA, NA, 31, 0, 19, 0, 29, 22, 25~
$ trust_tv            <dbl> 2, 3, 4, 5, 5, 4, 3, 6, 6, 7, 2, 7, 7, 5, 4, 2, 5~
$ trust_portal        <dbl> 6, 5, 6, 6, 3, 7, 7, 1, 7, 5, 6, 5, 5, 5, 4, 6, 4~
$ trust_sm            <dbl> 4, 3, 1, 4, 4, 7, 2, 3, 4, 6, 1, 2, 3, 5, 2, 2, 4~
$ n_platformi         <dbl> 9, 5, 7, 6, 5, 2, 1, 8, 5, 7, 6, 6, 2, 7, 4, 3, 6~
$ vijesti_frekvencija <chr> "više puta dnevno", "nekoliko puta tjedno", "više~
$ dobna_skupina       <chr> "20-21", "24+", "24+", "18-19", "24+", "24+", "24~
$ visoko_koristenje_sm <chr> "nisko/umjereno", "nisko/umjereno", "visoko", "ni~
$ prati_vijesti_cesto <lgl> TRUE, FALSE, TRUE, FALSE, TRUE, FALSE, FALSE, TRU~
```

Sada imamo čist dataset `clean` s razumljivim imenima stupaca, konzistentnim kodiranjem spola i godine studija, numeričkim stupcem za TV minute i binarnim varijablama za visoko korištenje i praćenje vijesti. Ovaj dataset je spreman za deskriptivnu statistiku i vizualizaciju.

> U svakom projektu analize podataka, trebali biste imati jasnu granicu između sirovih podataka (koje nikad ne mijenjate) i čistih podataka (koje kreirate skriptom iz sirovih). Skripta za čišćenje je vaš zapis svakog koraka, i svaki korišten uvjet mora biti dokumentiran komentarima.

---

9 Brzi pregled očišćenog dataseta

Provjerimo da je čišćenje uspjelo i iskoristimo priliku da povežemo sve naučene glagole.

```
# Distribucija po spolu
clean |>
  count(spol)
```

```
# A tibble: 2 x 2
  spol       n
  <chr>  <int>
1 muški    135
2 ženski   115
```

```r
# Distribucija po gradu (top 5)
clean |>
  count(grad, sort = TRUE) |>
  head(5)
```

```
# A tibble: 5 x 2
  grad       n
  <chr>  <int>
1 Zagreb   100
2 Split     44
3 Osijek    23
4 Rijeka    18
5 Zadar     15
```

```r
# Prosječno korištenje medija po dobnim skupinama
clean |>
  group_by(dobna_skupina) |>
  summarise(
    n = n(),
    sm_prosjek = round(mean(sm_min), 1),
    portal_prosjek = round(mean(portal_min), 1),
    tv_prosjek = round(mean(tv_minuta, na.rm = TRUE), 1),
    .groups = "drop"
  )
```

```
# A tibble: 4 x 5
  dobna_skupina     n sm_prosjek portal_prosjek tv_prosjek
  <chr>         <int>      <dbl>          <dbl>      <dbl>
1 18-19            73       122            42.3       33
2 20-21            61       120            43.6       34.4
3 22-23            47       120.           42.9       35.6
4 24+              69       122.           42.9       36.1
```

Tablica pokazuje jasne razlike. Studenti različitih dobnih skupina imaju različite obrasce korištenja medija. Najmlađi (18 do 19) provode najviše vremena na društvenim mrežama, dok je korištenje portala ravnomjernije raspoređeno. TV je konzistentno najniži oblik medijske konzumacije u svim skupinama, što je očekivano za studentsku populaciju.

```r
# Povjerenje u medije - tko kome vjeruje
clean |>
  summarise(
    trust_tv_prosjek = round(mean(trust_tv), 1),
    trust_portal_prosjek = round(mean(trust_portal), 1),
    trust_sm_prosjek = round(mean(trust_sm), 1)
  )
```

```
# A tibble: 1 x 3
  trust_tv_prosjek trust_portal_prosjek trust_sm_prosjek
             <dbl>                <dbl>            <dbl>
1              4.5                    5              3.4
```

```
# Povjerenje po spolu
clean |>
  group_by(spol) |>
  summarise(
    n = n(),
    trust_sm = round(mean(trust_sm), 1),
    trust_portal = round(mean(trust_portal), 1),
    .groups = "drop"
  )
```

```
# A tibble: 2 x 4
  spol       n trust_sm trust_portal
  <chr>  <int>    <dbl>        <dbl>
1 muški    135      3.3          4.9
2 ženski   115      3.5          5.1
```

Studenti u prosjeku najviše vjeruju portalima, zatim televiziji, a najmanje društvenim mrežama. Ovo je zanimljiv nalaz jer istovremeno na društvenim mrežama provode daleko najviše vremena. Provode li ljudi najviše vremena na medijima kojima najmanje vjeruju? Ili se povjerenje gradi korištenjem? Ovo su pitanja na koja ćemo se vraćati kad budemo radili korelacije i regresiju u kasnijim tjednima.

---

10 group_by() i summarise() za statistike po grupama

Kombinaciju `group_by()` i `summarise()` smo već koristili u dosadašnjim primjerima, ali zaslužuje detaljniju obradu jer je ovo daleko najvažniji obrazac u cijelom tidyverse radnom toku. Gotovo svaka analiza u komunikologiji uključuje usporedbu između grupa: razlikuju li se muškarci i žene po korištenju medija? Razlikuju li se gradovi po povjerenju? Razlikuju li se generacije po izvorima vijesti?

10.1 Osnovna logika

`group_by()` dijeli tibble na nevidljive podskupove prema vrijednostima jednog ili više stupaca. Sam po sebi ne proizvodi nikakav vidljiv rezultat. Ali kad nakon njega pozovete `summarise()`, izračun se ponavlja zasebno za svaki podskup.

```r
# Prosječno korištenje društvenih mreža po spolu
clean |>
  group_by(spol) |>
  summarise(
    n = n(),
    prosjek_sm = round(mean(sm_min), 1),
    sd_sm = round(sd(sm_min), 1),
    medijan_sm = median(sm_min),
    .groups = "drop"
  )
```

```
# A tibble: 2 x 5
  spol       n prosjek_sm sd_sm medijan_sm
  <chr>  <int>      <dbl> <dbl>      <dbl>
1 muški    135       117.  49.2        115
2 ženski   115       126.  47.3        129
```

Argument `.groups = "drop"` na kraju govori R-u da ukloni grupiranje nakon izračuna. Bez njega, rezultirajući tibble bi ostao grupiran, što može uzrokovati neočekivano ponašanje u kasnijim operacijama. Dobra praksa je uvijek eksplicitno navesti `.groups = "drop"`.

10.2 Grupiranje po više varijabli

```r
# Korištenje po spolu i dobnoj skupini
clean |>
  group_by(spol, dobna_skupina) |>
  summarise(
    n = n(),
    prosjek_sm = round(mean(sm_min), 1),
    prosjek_portal = round(mean(portal_min), 1),
    .groups = "drop"
  ) |>
  filter(spol != "ostalo") |>
  arrange(dobna_skupina, spol)
```

```
# A tibble: 8 x 5
  spol   dobna_skupina     n prosjek_sm prosjek_portal
  <chr>  <chr>         <int>      <dbl>          <dbl>
1 muški  18-19            39       122.           40.2
2 ženski 18-19            34       122.           44.7
3 muški  20-21            36       114.           41.4
4 ženski 20-21            25       128.           46.8
```

```
5 muški  22-23            24       110.          44.1
6 ženski 22-23            23       130.          41.7
7 muški  24+              36       119.          42.9
8 ženski 24+              33       126.          43
```

Kad grupirate po više varijabli, `summarise()` izračunava statistike za svaku kombinaciju tih varijabli. S dva spola i četiri dobne skupine dobivate osam grupa (ili manje, ako neke kombinacije nemaju opažanja). Filtrirali smo kategoriju "ostalo" jer s malim brojem opažanja statistike nisu pouzdane.

10.3 count() kao kratica

Funkcija `count()` je zapravo kratica za `group_by() |> summarise(n = n()) |> ungroup()`. Koristite je kad vam treba samo prebrojavanje.

```r
# Ovo:
clean |>
  count(grad, sort = TRUE) |>
  head(5)
```

```
# A tibble: 5 x 2
  grad       n
  <chr>  <int>
1 Zagreb   100
2 Split     44
3 Osijek    23
4 Rijeka    18
5 Zadar     15
```

```r
# Je ekvivalentno ovome:
clean |>
  group_by(grad) |>
  summarise(n = n(), .groups = "drop") |>
  arrange(desc(n)) |>
  head(5)
```

```
# A tibble: 5 x 2
  grad       n
  <chr>  <int>
1 Zagreb   100
2 Split     44
3 Osijek    23
4 Rijeka    18
5 Zadar     15
```

Obje verzije daju identičan rezultat, ali `count()` štedi tri reda koda. Za jednostavno prebrojavanje uvijek koristite `count()`.

10.4 group_by() s mutate()

Manje poznata ali izuzetno korisna kombinacija je `group_by()` s `mutate()`. Umjesto da sažima podatke u jednu vrijednost po grupi (kao `summarise()`), `mutate()` dodaje novu kolonu svakom retku, ali izračun se radi unutar grupe.

```
# Z-score korištenja društvenih mreža UNUTAR svake dobne skupine
clean |>
  group_by(dobna_skupina) |>
  mutate(
    sm_prosjek_grupe = mean(sm_min),
    sm_z = round((sm_min - mean(sm_min)) / sd(sm_min), 2)
  ) |>
  ungroup() |>
  select(id, dob, dobna_skupina, sm_min, sm_prosjek_grupe, sm_z) |>
  head(10)
```

```
# A tibble: 10 x 6
      id   dob dobna_skupina sm_min sm_prosjek_grupe  sm_z
   <dbl> <dbl> <chr>          <dbl>            <dbl> <dbl>
 1     1    20 20-21             59             120. -1.24
 2     2    27 24+              101             122. -0.53
 3     3    27 24+              177             122.  1.33
 4     4    18 18-19             71             122. -0.98
 5     5    25 24+              161             122.  0.94
 6     6    26 24+              155             122.  0.79
 7     7    28 24+              114             122. -0.21
 8     8    26 24+              119             122. -0.09
 9     9    22 22-23             56             120. -1.2
10    10    21 20-21             40             120. -1.62
```

Primijetite `ungroup()` na kraju. Kad koristite `group_by()` s `mutate()`, grupiranje ostaje aktivno nakon mutatea (za razliku od `summarise()` koji ga automatski smanjuje). Uvijek dodajte `ungroup()` kad završite s grupiranim operacijama da izbjegnete iznenađenja.

---

11 across() za istu operaciju na više stupaca

Do sada smo u `summarise()` ručno pisali svaku statistiku za svaki stupac. Kad imate pet ili deset numeričkih stupaca, to postaje zamorno. Funkcija `across()` rješava ovaj problem jer primjenjuje istu funkciju (ili više funkcija) na više stupaca odjednom.

```
# Prosjek za sve stupce koji sadrže "trust" u imenu
clean |>
  summarise(
    across(starts_with("trust"), ~round(mean(.x), 1))
  )
```

```
# A tibble: 1 x 3
  trust_tv trust_portal trust_sm
     <dbl>        <dbl>    <dbl>
1      4.5            5      3.4
```

Sintaksa `~round(mean(.x), 1)` koristi lambda notaciju (tilda formula). `.x` je placeholder za svaki stupac na koji se `across()` primjenjuje. Ovo se čita kao "za svaki stupac koji počinje s trust, izračunaj zaokruženi prosjek".

11.1 Više funkcija odjednom

```
# Prosjek i SD za stupce s minutama
clean |>
  summarise(
    across(
      c(sm_min, portal_min, tv_minuta),
      list(
        prosjek = ~round(mean(.x, na.rm = TRUE), 1),
        sd = ~round(sd(.x, na.rm = TRUE), 1)
      ),
      .names = "{.col}_{.fn}"
    )
  )
```

```
# A tibble: 1 x 6
  sm_min_prosjek sm_min_sd portal_min_prosjek portal_min_sd tv_minuta_prosjek
           <dbl>     <dbl>              <dbl>         <dbl>              <dbl>
1           121.      48.5               42.9          22.3               34.7
# i 1 more variable: tv_minuta_sd <dbl>
```

Kad proslijedite imenovu listu funkcija, `across()` kreira zasebne stupce za svaku kombinaciju stupca i funkcije. Argument `.names = "{.col}_{.fn}"` kontrolira kako se novi stupci imenuju — `{.col}` je ime izvornog stupca, `{.fn}` je ime funkcije iz liste.

11.2 across() s group_by()

Kombinacija `across()` i `group_by()` omogućuje izračun više statistika za više stupaca po grupama, u jednom kompaktnom pozivu.

```
clean |>
  group_by(dobna_skupina) |>
  summarise(
    n = n(),
    across(
      c(sm_min, portal_min, trust_sm, trust_portal),
      list(M = ~round(mean(.x, na.rm = TRUE), 1)),
      .names = "{.col}_{.fn}"
    ),
    .groups = "drop"
  )
```

```
# A tibble: 4 x 6
  dobna_skupina     n sm_min_M portal_min_M trust_sm_M trust_portal_M
  <chr>         <int>    <dbl>        <dbl>      <dbl>          <dbl>
1 18-19            73      122         42.3        3.5            4.9
2 20-21            61      120         43.6        3.4            4.9
3 22-23            47      120.        42.9        3.5            5
4 24+              69      122.        42.9        3.3            5.2
```

U jednom pozivu dobivamo prosjeke četiri varijable za svaku dobnu skupinu, plus broj opažanja. Ovo je obrazac koji ćete koristiti za izradu tablica deskriptivnih statistika u akademskim radovima.

11.3 across() s mutate()

`across()` radi i unutar `mutate()` za transformaciju više stupaca odjednom.

```
# Centriranje svih trust varijabli (oduzimanje prosjeka)
clean |>
  mutate(
    across(
      starts_with("trust"),
      ~.x - mean(.x),
```

```
      .names = "{.col}_cent"
    )
  ) |>
  select(id, starts_with("trust")) |>
  head(5)
```

```
# A tibble: 5 x 7
     id trust_tv trust_portal trust_sm trust_tv_cent trust_portal_cent
  <dbl>    <dbl>        <dbl>    <dbl>         <dbl>              <dbl>
1     1        2            6        4        -2.48             0.992
2     2        3            5        3        -1.48            -0.00800
3     3        4            6        1        -0.476            0.992
4     4        5            6        4         0.524            0.992
5     5        5            3        4         0.524           -2.01
# i 1 more variable: trust_sm_cent <dbl>
```

Ovo je osobito korisno za standardizaciju ili transformaciju velikog broja varijabli u jednom koraku.

> **Praktični savjet**
>
> Funkcija `across()` čini kod kompaktnijim ali i teže čitljivim za početnike. Ako vam lambda notacija (`~mean(.x)`) izgleda zbunjujuće, nema ništa loše u tome da najprije pišete svaku statistiku ručno, a `across()` počnete koristiti kad se osjećate ugodno s osnovnim glagolima. Cilj je čitljivost, ne kratkoća.

---

12 pivot_longer() i pivot_wider() za preoblikovanje podataka

Ponekad podaci dolaze u obliku koji nije pogodan za analizu ili vizualizaciju i moramo ih preoblikovati. Dva najčešća slučaja su pretvaranje širokog formata u dugački i obrnuto.

12.1 Tidy data — princip urednih podataka

Wickham (2014) definira **uredne podatke** (tidy data) kao tablicu u kojoj svaki redak predstavlja jedno opažanje, svaki stupac jednu varijablu i svaka ćelija jednu vrijednost. Zvuči jednostavno, ali mnogi dataseti ne zadovoljavaju ovaj princip.

Pogledajmo konkretan primjer. Naši podaci o povjerenju imaju tri zasebna stupca: `trust_tv`, `trust_portal`, `trust_sm`. Za neke analize (posebno vizualizaciju), bilo bi korisnije imati jedan stupac `medij` s vrijednostima "TV", "portal" i "društvene mreže" i jedan stupac `povjerenje` s numeričkom ocjenom.

12.2 pivot_longer() za pretvaranje od širokog prema dugačkom formatu

```
trust_long <- clean |>
  select(id, dob, spol, dobna_skupina, trust_tv, trust_portal, trust_sm) |>
  pivot_longer(
    cols = starts_with("trust"),
    names_to = "medij",
    values_to = "povjerenje",
    names_prefix = "trust_"
  )

trust_long |>
  head(12)
```

```
# A tibble: 12 x 6
      id   dob spol   dobna_skupina medij  povjerenje
   <dbl> <dbl> <chr>  <chr>         <chr>       <dbl>
 1     1    20 ženski 20-21         tv              2
 2     1    20 ženski 20-21         portal          6
 3     1    20 ženski 20-21         sm              4
 4     2    27 muški  24+           tv              3
 5     2    27 muški  24+           portal          5
 6     2    27 muški  24+           sm              3
 7     3    27 muški  24+           tv              4
 8     3    27 muški  24+           portal          6
 9     3    27 muški  24+           sm              1
10     4    18 ženski 18-19         tv              5
11     4    18 ženski 18-19         portal          6
12     4    18 ženski 18-19         sm              4
```

Funkcija `pivot_longer()` pretvara stupce u redove. Ključni argumenti funkcije su sljedeći.

`cols` specificira koje stupce pretvaramo (ovdje sve koji počinju s "trust").

`names_to` je ime novog stupca koji će sadržavati imena izvornih stupaca.

`values_to` je ime novog stupca koji će sadržavati vrijednosti iz izvornih stupaca.

`names_prefix` uklanja zajednički prefiks iz imena (bez njega bismo imali "trust_tv" umjesto "tv").

Iz originalnih 250 redova (jedan po ispitaniku) dobili smo 750 redova (tri po ispitaniku, jedan za svaki tip medija). Ovo je dugački format.

Sad možemo lako izračunati prosječno povjerenje po tipu medija.

```
trust_long |>
  group_by(medij) |>
  summarise(
    prosjek = round(mean(povjerenje), 2),
    sd = round(sd(povjerenje), 2),
    .groups = "drop"
  )
```

```
# A tibble: 3 x 3
  medij  prosjek    sd
  <chr>    <dbl> <dbl>
1 portal    5.01  1.73
2 sm        3.41  1.72
3 tv        4.48  1.99
```

Ili po tipu medija i dobnoj skupini.

```
trust_long |>
  group_by(dobna_skupina, medij) |>
  summarise(
    prosjek = round(mean(povjerenje), 1),
    .groups = "drop"
  ) |>
  arrange(dobna_skupina, medij)
```

```
# A tibble: 12 x 3
   dobna_skupina medij  prosjek
   <chr>         <chr>    <dbl>
 1 18-19         portal     4.9
 2 18-19         sm         3.5
 3 18-19         tv         4.6
 4 20-21         portal     4.9
 5 20-21         sm         3.4
 6 20-21         tv         4.8
 7 22-23         portal     5
 8 22-23         sm         3.5
 9 22-23         tv         4.1
10 24+           portal     5.2
11 24+           sm         3.3
12 24+           tv         4.4
```

Ova tablica jasno pokazuje obrasce koje bi bilo teško vidjeti u širokom formatu. Dugački format je posebno koristan za vizualizaciju jer ggplot2 (koji ćemo učiti sljedeći tjedan) radi prirodno s dugačkim podacima.

12.3 pivot_wider() za pretvaranje od dugačkog prema širokom formatu

Obrnuta operacija, `pivot_wider()`, pretvara redove u stupce. Korisna je kad želite tablicu u obliku koji je čitljiv za ljude (široki format), a ne za računalo (dugački format).

```
# Prosječno povjerenje po dobnoj skupini i mediju, u širokom formatu
trust_long |>
  group_by(dobna_skupina, medij) |>
  summarise(prosjek = round(mean(povjerenje), 1), .groups = "drop") |>
  pivot_wider(
    names_from = medij,
    values_from = prosjek
  )
```

```
# A tibble: 4 x 4
  dobna_skupina portal    sm    tv
  <chr>          <dbl> <dbl> <dbl>
1 18-19            4.9   3.5   4.6
2 20-21            4.9   3.4   4.8
3 22-23            5     3.5   4.1
4 24+              5.2   3.3   4.4
```

Rezultat je tablica s jednim retkom po dobnoj skupini i jednim stupcem po tipu medija. Ovo je format koji biste stavili u izvještaj ili akademski rad jer je lako čitljiv.

Argumenti su zrcalni u odnosu na `pivot_longer()`:

`names_from` je stupac čije će vrijednosti postati imena novih stupaca.

`values_from` je stupac čije će vrijednosti popuniti nove stupce.

12.4 Primjer s minutama korištenja

Isti obrazac primjenjujemo i na podatke o korištenju medija.

```
# Pretvorba minuta korištenja u dugački format
koristenje_long <- clean |>
  select(id, dob, spol, dobna_skupina, tv_minuta, portal_min, sm_min) |>
  pivot_longer(
    cols = c(tv_minuta, portal_min, sm_min),
    names_to = "medij",
    values_to = "minuta"
  ) |>
  mutate(
    medij = case_when(
```

```
      medij == "tv_minuta" ~ "TV",
      medij == "portal_min" ~ "Portali",
      medij == "sm_min" ~ "Društvene mreže"
    )
  )

# Prosječno korištenje po tipu medija
koristenje_long |>
  group_by(medij) |>
  summarise(
    prosjek = round(mean(minuta, na.rm = TRUE), 1),
    medijan = median(minuta, na.rm = TRUE),
    .groups = "drop"
  ) |>
  arrange(desc(prosjek))
```

```
# A tibble: 3 x 3
  medij           prosjek medijan
  <chr>             <dbl>   <dbl>
1 Društvene mreže   121.      124
2 Portali            42.9      43
3 TV                 34.7      17
```

Društvene mreže dominiraju s velikim razmakom. TV je daleko na dnu. Ovi podaci su za studentsku populaciju, pa ne iznenađuju, ali upravo ovakve tablice čine temelj svakog izvještaja o medijskim navikama.

> **!** Važna napomena
>
> Zapamtite sljedeće pravilo. `pivot_longer()` koristite kad želite pretvoriti podatke iz oblika čitljivog za ljude u oblik pogodan za analizu i vizualizaciju. `pivot_wider()` koristite kad želite rezultate pretvoriti natrag u oblik čitljiv za ljude (za tablice u izvještajima). Tipičan radni tok uključuje sljedeće korake: učitajte podatke, pretvorite u dugački format, analizirajte i pretvorite rezultate u široki format za prezentaciju.

---

13 Spajanje tablica pomoću left_join()

U stvarnim istraživanjima, podaci rijetko dolaze u jednoj tablici. Možda imate jednu tablicu s demografskim podacima ispitanika i drugu s rezultatima eksperimenta. Ili jednu tablicu s podacima o člancima i drugu s podacima o komentarima. Da biste ih analizirali zajedno, morate ih spojiti.

Kreirajmo pomoćnu tablicu za demonstraciju.

```
# Tablica s informacijama o gradovima
gradovi_info <- tibble(
  grad = c("Zagreb", "Split", "Rijeka", "Osijek", "Zadar", "Dubrovnik",
           "Slavonski Brod", "Pula", "Karlovac", "Varaždin", "Šibenik", "Sisak"),
  regija = c("Središnja", "Dalmacija", "Primorje", "Slavonija", "Dalmacija", "Dalmacija",
             "Slavonija", "Istra", "Središnja", "Sjever", "Dalmacija", "Središnja"),
  populacija_tis = c(770, 160, 108, 96, 70, 41, 50, 52, 46, 41, 34, 33)
)

gradovi_info
```

```
# A tibble: 12 x 3
   grad           regija    populacija_tis
   <chr>          <chr>              <dbl>
 1 Zagreb         Središnja            770
 2 Split          Dalmacija            160
 3 Rijeka         Primorje             108
 4 Osijek         Slavonija             96
 5 Zadar          Dalmacija             70
 6 Dubrovnik      Dalmacija             41
 7 Slavonski Brod Slavonija             50
 8 Pula           Istra                 52
 9 Karlovac       Središnja             46
10 Varaždin       Sjever                41
11 Šibenik        Dalmacija             34
12 Sisak          Središnja             33
```

Sada možemo spojiti ovu tablicu s našim čistim podacima da svakom ispitaniku dodamo informaciju o regiji i populaciji grada.

```
clean_s_regijom <- clean |>
  left_join(gradovi_info, by = "grad")

clean_s_regijom |>
  select(id, grad, regija, populacija_tis, sm_min) |>
  head(10)
```

```
# A tibble: 10 x 5
      id grad     regija    populacija_tis sm_min
   <dbl> <chr>    <chr>              <dbl>  <dbl>
 1     1 Zagreb   Središnja            770     59
 2     2 Zadar    Dalmacija             70    101
```

```
 3     3 Zagreb   Središnja            770    177
 4     4 Split    Dalmacija            160     71
 5     5 Zagreb   Središnja            770    161
 6     6 Zagreb   Središnja            770    155
 7     7 Zagreb   Središnja            770    114
 8     8 Karlovac Središnja             46    119
 9     9 Split    Dalmacija            160     56
10    10 Osijek   Slavonija             96     40
```

Funkcija `left_join()` spaja dvije tablice po zajedničkom stupcu (ovdje `grad`). Za svaki redak u lijevoj tablici (`clean`), traži podudarajući redak u desnoj tablici (`gradovi_info`) i dodaje stupce iz desne tablice. Ako nema podudaranja (na primjer, grad koji nije u tablici `gradovi_info`), dobivamo NA.

Argument `by = "grad"` specificira koji stupac koristimo za podudaranje. Ako se stupac za spajanje različito zove u dvjema tablicama, koristimo sintaksu `by = c("ime_lijevo" = "ime_desno")`.

Sad možemo analizirati podatke po regijama.

```r
clean_s_regijom |>
  group_by(regija) |>
  summarise(
    n = n(),
    prosjek_sm = round(mean(sm_min), 1),
    prosjek_trust_sm = round(mean(trust_sm), 1),
    .groups = "drop"
  ) |>
  filter(!is.na(regija)) |>
  arrange(desc(prosjek_sm))
```

```
# A tibble: 6 x 4
  regija        n prosjek_sm prosjek_trust_sm
  <chr>     <int>      <dbl>            <dbl>
1 Istra        10       140                3
2 Dalmacija    74       132.               3.2
3 Slavonija    31       127.               3.5
4 Središnja   110       115.               3.6
5 Primorje     18       112.               3.4
6 Sjever        7        79                3.3
```

Ovo je moć spajanja tablica — informacija koja je bila u zasebnoj tablici sada je dio naše analize i omogućuje grupiranje po varijabli koja nije postojala u izvornim podacima.

13.1 Vrste joinova

`left_join()` je daleko najčešći join i jedini koji ćete trebati u većini situacija. Ali vrijedi znati da postoje i drugi.

`left_join(a, b)` zadržava sve retke iz a, dodaje podudarajuće iz b. Ako nema podudaranja, NA.

`inner_join(a, b)` zadržava samo retke koji postoje u obje tablice.

`full_join(a, b)` zadržava sve retke iz obje tablice, s NA gdje nema podudaranja.

`anti_join(a, b)` zadržava retke iz a koji nemaju podudaranje u b. Korisno za pronalaženje nepodudarajućih zapisa.

```
# Ima li ispitanika iz gradova koji nisu u našoj tablici?
clean |>
  anti_join(gradovi_info, by = "grad") |>
  count(grad)
```

```
# A tibble: 0 x 2
# i 2 variables: grad <chr>, n <int>
```

`anti_join()` je odličan dijagnostički alat jer otkriva retke koji se ne mogu spojiti. U ovom slučaju vidimo gradove koji postoje u anketi ali ne u našoj tablici gradova.

---

14 Stringovi — osnove rada s tekstom

U komunikologiji se često radi s tekstualnim podacima, kao što su imena platformi, naslovi članaka i otvoreni odgovori u anketama. Paket stringr (dio tidyverse) pruža konzistentan skup funkcija za rad s tekstom.

Već smo koristili `str_to_lower()` i `str_detect()`. Pogledajmo još nekoliko korisnih funkcija.

```
# Stupac s platformama je slobodan tekst s više unosa
raw |>
  select(id_respondenta, koje_platforme_koristi) |>
  head(5)
```

```
# A tibble: 5 x 2
  id_respondenta koje_platforme_koristi
           <dbl> <chr>
1              1 Snapchat, WhatsApp, Facebook
2              2 Facebook, YouTube
3              3 WhatsApp
4              4 Pinterest, LinkedIn
5              5 Viber, Snapchat, Reddit
```

```
# Koliko ispitanika koristi Instagram (bilo gdje u tekstu)?
raw |>
  mutate(koristi_instagram = str_detect(koje_platforme_koristi, "Instagram")) |>
  count(koristi_instagram)
```

```
# A tibble: 2 x 2
  koristi_instagram     n
  <lgl>             <int>
1 FALSE               211
2 TRUE                 39
```

14.1 Brojanje i izdvajanje uzoraka

```
# Koliko platformi svaki ispitanik navodi (brojeći zareze + 1)?
raw |>
  mutate(
    navedeno_platformi = str_count(koje_platforme_koristi, ",") + 1
  ) |>
  select(id_respondenta, koje_platforme_koristi, navedeno_platformi) |>
  head(8)
```

```
# A tibble: 8 x 3
  id_respondenta koje_platforme_koristi                navedeno_platformi
           <dbl> <chr>                                            <dbl>
1              1 Snapchat, WhatsApp, Facebook                         3
2              2 Facebook, YouTube                                    2
3              3 WhatsApp                                             1
4              4 Pinterest, LinkedIn                                  2
5              5 Viber, Snapchat, Reddit                              3
6              6 Pinterest, YouTube                                   2
7              7 WhatsApp                                             1
8              8 Reddit, Pinterest, WhatsApp, Twitter/X               4
```

Funkcija `str_count()` broji koliko se puta uzorak pojavljuje u tekstu. Budući da su platforme odvojene zarezima, broj zareza plus jedan daje broj navedenih platformi. Ovo je primjer kako tekstualne operacije pomažu u izvlačenju numeričkih informacija iz nestrukturiranih podataka.

14.2 Zamjena i čišćenje teksta

```
# Zamjena "Twitter/X" s "X" za konzistentnost
raw |>
  mutate(
    platforme_clean = str_replace(koje_platforme_koristi, "Twitter/X", "X")
  ) |>
  filter(str_detect(koje_platforme_koristi, "Twitter")) |>
  select(koje_platforme_koristi, platforme_clean) |>
  head(5)
```

```
# A tibble: 5 x 2
  koje_platforme_koristi                    platforme_clean
  <chr>                                     <chr>
1 Reddit, Pinterest, WhatsApp, Twitter/X    Reddit, Pinterest, WhatsApp, X
2 YouTube, Twitter/X, Pinterest, WhatsApp   YouTube, X, Pinterest, WhatsApp
3 Pinterest, Instagram, Telegram, Twitter/X Pinterest, Instagram, Telegram, X
4 Reddit, Twitter/X, Pinterest              Reddit, X, Pinterest
5 LinkedIn, Telegram, Snapchat, Twitter/X   LinkedIn, Telegram, Snapchat, X
```

Funkcija `str_replace()` zamjenjuje prvo pojavljivanje uzorka, a `str_replace_all()` zamjenjuje sva pojavljivanja. Funkcija `str_trim()` uklanja razmake s početka i kraja teksta, što je korisno kad ispitanici slučajno unesu razmak.

---

15 Sve zajedno — kompletna analiza od sirovih do gotovih podataka

Zaokružimo ovo predavanje tako da napišemo kompletnu analizu koja prolazi kroz sve faze, a to su učitavanje, čišćenje, transformacija, analiza i prezentacija rezultata. Ovo je obrazac koji ćete ponavljati u svakom projektu.

```r
# FAZA 1: Učitavanje i čišćenje
anketa_clean <- read_csv("../resources/datasets/media_habits_raw.csv") |>
  clean_names() |>
  mutate(
    # Čišćenje spola
    spol = case_when(
      str_to_lower(spol) %in% c("ženski", "ž", "zensko", "female") ~ "ženski",
      str_to_lower(spol) %in% c("muški", "m", "musko", "male") ~ "muški",
      .default = NA_character_
    ),
    # Čišćenje godine studija
    godina = case_when(
      str_to_lower(godina_studija) %in% c("1", "1.", "prva") ~ 1L,
      str_to_lower(godina_studija) %in% c("2", "2.", "druga") ~ 2L,
      str_to_lower(godina_studija) %in% c("3", "3.", "treća", "treca") ~ 3L,
      str_to_lower(godina_studija) %in% c("4", "4.") ~ 4L,
      str_to_lower(godina_studija) %in% c("5", "5.") ~ 5L,
      .default = NA_integer_
    ),
    # TV minute: text -> broj
    tv_min = case_when(
      tv_min_dan == "ne gledam" ~ 0,
      tv_min_dan == "" ~ NA_real_,
      .default = as.numeric(tv_min_dan)
    ),
    # Dobna skupina
    dobna_sk = case_when(
      dob < 20 ~ "18-19",
      dob < 22 ~ "20-21",
      dob < 24 ~ "22-23",
      dob >= 24 ~ "24+"
    )
  ) |>
  # Odabir i preimenovanje konačnih stupaca
  select(
    id = id_respondenta,
    dob, spol, grad, godina,
    tv_min,
    portal_min = portali_min_dan,
    sm_min = drustvene_mreze_min_dan,
    trust_tv = povjerenje_tv_1_10,
    trust_portal = povjerenje_portali_1_10,
    trust_sm = povjerenje_drustvene_mreze_1_10,
    vijesti = koliko_cesto_prati_vijesti,
    dobna_sk
```

```
  )

glimpse(anketa_clean)
```

```
Rows: 250
Columns: 13
$ id           <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17~
$ dob          <dbl> 20, 27, 27, 18, 25, 26, 28, 26, 22, 21, 22, 27, 20, 20, 2~
$ spol         <chr> "ženski", "muški", "muški", "ženski", "ženski", "muški", ~
$ grad         <chr> "Zagreb", "Zadar", "Zagreb", "Split", "Zagreb", "Zagreb",~
$ godina       <int> 2, 3, 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 1, 2, ~
$ tv_min       <dbl> 0, 0, 65, NA, NA, 91, 91, 0, 76, 66, NA, 109, 0, 0, 56, 0~
$ portal_min   <dbl> 40, 20, 0, 11, 32, 25, 81, 28, 37, 5, 38, 44, 26, 65, 35,~
$ sm_min       <dbl> 59, 101, 177, 71, 161, 155, 114, 119, 56, 40, 129, 95, 72~
$ trust_tv     <dbl> 2, 3, 4, 5, 5, 4, 3, 6, 6, 7, 2, 7, 7, 5, 4, 2, 5, 3, 4, ~
$ trust_portal <dbl> 6, 5, 6, 6, 3, 7, 7, 1, 7, 5, 6, 5, 5, 5, 4, 6, 4, 5, 6, ~
$ trust_sm     <dbl> 4, 3, 1, 4, 4, 7, 2, 3, 4, 6, 1, 2, 3, 5, 2, 2, 4, 3, 6, ~
$ vijesti      <chr> "više puta dnevno", "nekoliko puta tjedno", "više puta dn~
$ dobna_sk     <chr> "20-21", "24+", "24+", "18-19", "24+", "24+", "24+", "24+~
```

```
# FAZA 2: Provjera
# Koliko NA po stupcu?
anketa_clean |>
  summarise(across(everything(), ~sum(is.na(.)))) |>
  pivot_longer(everything(), names_to = "stupac", values_to = "n_NA") |>
  filter(n_NA > 0)
```

```
# A tibble: 1 x 2
  stupac  n_NA
  <chr>  <int>
1 tv_min    41
```

```
# FAZA 3: Deskriptivna analiza
# Korištenje medija po dobnoj skupini
anketa_clean |>
  group_by(dobna_sk) |>
  summarise(
    n = n(),
    across(
      c(sm_min, portal_min, tv_min),
      list(M = ~round(mean(.x, na.rm = TRUE), 1)),
      .names = "{.col}_{.fn}"
    ),
```

```
    .groups = "drop"
  )
```

```
# A tibble: 4 x 5
  dobna_sk     n sm_min_M portal_min_M tv_min_M
  <chr>    <int>    <dbl>        <dbl>    <dbl>
1 18-19       73     122          42.3     33
2 20-21       61     120          43.6     34.4
3 22-23       47     120.         42.9     35.6
4 24+         69     122.         42.9     36.1
```

```
# Povjerenje po tipu medija (dugački format za lakšu usporedbu)
anketa_clean |>
  pivot_longer(
    cols = starts_with("trust"),
    names_to = "medij",
    values_to = "povjerenje",
    names_prefix = "trust_"
  ) |>
  mutate(
    medij = case_when(
      medij == "tv" ~ "Televizija",
      medij == "portal" ~ "Web portali",
      medij == "sm" ~ "Društvene mreže"
    )
  ) |>
  group_by(medij) |>
  summarise(
    M = round(mean(povjerenje), 2),
    SD = round(sd(povjerenje), 2),
    Med = median(povjerenje),
    .groups = "drop"
  ) |>
  arrange(desc(M))
```

```
# A tibble: 3 x 4
  medij              M    SD   Med
  <chr>          <dbl> <dbl> <dbl>
1 Web portali     5.01  1.73     5
2 Televizija      4.48  1.99     4
3 Društvene mreže 3.41  1.72     3
```

```r
# Tko prati vijesti, a tko ne?
anketa_clean |>
  mutate(
    cesto_prati = vijesti %in% c("više puta dnevno", "jednom dnevno")
  ) |>
  group_by(cesto_prati) |>
  summarise(
    n = n(),
    prosjek_dob = round(mean(dob), 1),
    prosjek_sm = round(mean(sm_min), 1),
    prosjek_trust_portal = round(mean(trust_portal), 1),
    prosjek_trust_sm = round(mean(trust_sm), 1),
    .groups = "drop"
  )
```

```
# A tibble: 2 x 6
  cesto_prati     n prosjek_dob prosjek_sm prosjek_trust_portal prosjek_trust_sm
  <lgl>       <int>       <dbl>      <dbl>                <dbl>            <dbl>
1 FALSE         109        21.6       120.                  5.2              3.3
2 TRUE          141        21.9       122.                  4.9              3.5
```

Ova kompletna analiza, od učitavanja sirovih podataka do gotovih tablica, stane u manje od 80 redova koda. Svaki korak je dokumentiran, ponovljiv i čitljiv. Ako sutra dobijete ažurirane podatke s još 100 ispitanika, pokrenete istu skriptu i dobijete ažurirane rezultate. To je suština ponovljive analize.

> Cilj čišćenja podataka nije savršenstvo. Cilj je da od neurednog, nekonzistentnog i djelomično nepoznatog skupa podataka stvorite skup koji je dovoljno uredan i dokumentiran da možete s povjerenjem raditi statističku analizu i transparentno komunicirati svaki izbor koji ste napravili.

! Ključni zaključci

1. Čišćenje i priprema podataka oduzima 80% vremena u bilo kojoj analizi. Stvarni podaci su gotovo uvijek neuredni i zahtijevaju sistematično čišćenje prije ikakve statističke analize.
2. Funkcija `clean_names()` iz paketa janitor standardizira imena stupaca u snake_case format. Koristite je odmah nakon učitavanja svakog dataseta.
3. `filter()` odabire retke po uvjetu. Automatski odbacuje retke s NA. Koristite `%in%` za provjeru pripadnosti skupu, `between()` za raspone i `str_detect()` za pretraživanje teksta.

4. `select()` odabire, uklanja i preuređuje stupce. Pomoćne funkcije `starts_with()`, `ends_with()`, `contains()` i `where()` omogućuju pametan odabir. `rename()` mijenja imena bez gubitka stupaca.

5. `mutate()` kreira nove stupce i transformira postojeće. `case_when()` je alat za složeno rekodiranje. `if_else()` za binarno. Razlika između 0 i NA je konceptualno važna.

6. `arrange()` sortira retke. `desc()` za silazni smjer. NA uvijek na kraj.

7. `group_by() |> summarise()` je temeljni obrazac za izračun statistika po grupama. Uvijek dodajte `.groups = "drop"`. `count()` je kratica za prebrojavanje.

8. `across()` primjenjuje istu operaciju na više stupaca odjednom. Kombinira se i sa `summarise()` i s `mutate()`.

9. `pivot_longer()` pretvara stupce u redove (široki u dugački format). `pivot_wider()` pretvara redove u stupce. Dugački format je pogodan za analizu i vizualizaciju, široki za prezentaciju.

10. `left_join()` spaja dvije tablice po zajedničkom stupcu. Koristite ga kad trebate kombinirati podatke iz više izvora.

11. stringr funkcije (`str_detect()`, `str_to_lower()`, `str_replace()`, `str_count()`) omogućuju rad s tekstualnim podacima. Bitne za čišćenje anketnih podataka.

12. Svaka analiza ima jasne faze, a to su učitavanje, čišćenje, provjera, analiza i prezentacija. Dokumentirajte svaki korak i svaki izbor (osobito koliko redova gubite filtiranjem).

⚠ Priprema za sljedeći tjedan

Sljedeći tjedan bavimo se **deskriptivnom statistikom**: mjerama centralne tendencije (prosjek, medijan, mod), mjerama varijabilnosti (varijanca, standardna devijacija, IQR), korelacijama i standardnim rezultatima (z-scores). Sve ćemo raditi kroz `summarise()` i `group_by()` koje ste upravo naučili.
Za pripremu napravite sljedeće:

1. Ponovite kompletni pipeline čišćenja iz ovog predavanja. Pokrenite ga red po red i provjerite da razumijete svaki korak.
2. Pokušajte odgovoriti na pitanje: razlikuje li se prosječno povjerenje u društvene mreže između ispitanika koji prate vijesti često i onih koji ne prate? (Hint: `mutate()` za kreiranje binarne varijable, `group_by() |> summarise()` za usporedbu.)

3. Pretvorite podatke o korištenju (TV, portali, društvene mreže) u dugački format pomoću `pivot_longer()` i izračunajte prosječno korištenje po tipu medija i spolu.
4. Pročitajte poglavlje 5 iz knjige Navarro (Learning Statistics with R) o deskriptivnoj statistici. Fokusirajte se na koncepte, ne na R kod (jer knjiga koristi base R).

---

16 Dodatno čitanje

Obavezno

Wickham, H. & Grolemund, G. (2023). *R for Data Science* (2nd edition), Chapters 4, 5 i 6. Besplatno dostupno na r4ds.hadley.nz. Poglavlje 4 pokriva transformaciju podataka, poglavlje 5 organizaciju radnog toka, poglavlje 6 preoblikovanje podataka s pivot funkcijama.

Navarro, D. (2018). *Learning Statistics with R*, Chapters 4 i 7. Besplatno dostupno na learningstatisticswithr.com. Pokrivaju sličan teren u base R sintaksi.

Preporučeno

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, 59(10). Besplatno dostupno na vita.had.co.nz/papers/tidy-data.pdf. Klasičan rad koji definira princip urednih podataka.

Firke, S. (2023). *janitor: Simple Tools for Examining and Cleaning Dirty Data*. Dokumentacija paketa na sfirke.github.io/janitor. Osim `clean_names()`, paket sadrži i `tabyl()` za brze tablice frekvencija i `remove_empty()` za uklanjanje praznih redova i stupaca.

---

17 Pojmovnik

Pojam	Objašnjenje
dplyr	R paket iz tidyverse ekosustava za manipulaciju podacima. Sadrži glagole `filter()`, `select()`, `mutate()`, `summarise()`, `arrange()`, `group_by()` i druge.
`filter()`	dplyr glagol za odabir redova koji zadovoljavaju logički uvjet. Automatski odbacuje retke s NA u uvjetu.

Pojam	Objašnjenje
`select()`	dplyr glagol za odabir, uklanjanje i preuređivanje stupaca. Podržava pomoćne funkcije poput `starts_with()`, `contains()` i `where()`.
`mutate()`	dplyr glagol za kreiranje novih stupaca ili transformaciju postojećih. Novi stupci mogu referirati na upravo kreirane.
`arrange()`	dplyr glagol za sortiranje redova po vrijednostima stupaca. `desc()` za silazno sortiranje.
`summarise()`	dplyr glagol za sažimanje podataka u jednu vrijednost po grupi (ili za cijeli dataset). Koristi se s agregatnim funkcijama poput `mean()`, `sd()`, `n()`.
`group_by()`	dplyr glagol koji dijeli podatke u grupe po jednoj ili više varijabli. Sve naknadne operacije se izvršavaju zasebno za svaku grupu.
`ungroup()`	dplyr glagol koji uklanja grupiranje. Koristite nakon `group_by() |> mutate()` da izbjegnete neočekivano ponašanje.
`count()`	Kratica za `group_by() |> summarise(n = n()) |> ungroup()`. Prebrojava opažanja po kategorijama.
`across()`	Funkcija za primjenu iste operacije na više stupaca odjednom. Radi unutar `summarise()` i `mutate()`.
`rename()`	dplyr glagol za preimenovanje stupaca bez gubitka ostalih. Sintaksa: `rename(novo = staro)`.
`relocate()`	dplyr glagol za premještanje stupaca na drugu poziciju u datasetu.
`case_when()`	Funkcija za složeno rekodiranje s više uvjeta. Svaki uvjet ima oblik `uvjet ~ vrijednost`. Provjerava uvjete redom.
`if_else()`	Funkcija za binarno rekodiranje. Prima uvjet, vrijednost za TRUE i vrijednost za FALSE.
`between()`	Pomoćna funkcija: provjera je li vrijednost unutar raspona. Kratica za `x >= left & x <= right`.
`str_detect()`	stringr funkcija: provjerava sadrži li tekst zadani uzorak. Vraća TRUE/FALSE.

Pojam	Objašnjenje
`str_to_lower()`	stringr funkcija: pretvara tekst u mala slova. Korisna za standardizaciju.
`str_replace()`	stringr funkcija: zamjenjuje prvo pojavljivanje uzorka u tekstu. `str_replace_all()` zamjenjuje sva.
`str_count()`	stringr funkcija: broji pojavljivanja uzorka u tekstu.
`pivot_longer()`	tidyr funkcija za pretvaranje stupaca u redove (široki u dugački format). Ključni argumenti: `cols`, `names_to`, `values_to`.
`pivot_wider()`	tidyr funkcija za pretvaranje redova u stupce (dugački u široki format). Ključni argumenti: `names_from`, `values_from`.
`left_join()`	dplyr funkcija za spajanje dviju tablica po zajedničkom stupcu. Zadržava sve retke iz lijeve tablice.
`inner_join()`	Spajanje koje zadržava samo retke koji postoje u obje tablice.
`anti_join()`	Spajanje koje zadržava retke iz lijeve tablice koji nemaju podudaranje u desnoj. Dijagnostički alat.
`clean_names()`	janitor funkcija: pretvara imena stupaca u snake_case. Uklanja razmake, zagrade i specijalne znakove.
`drop_na()`	tidyr funkcija za uklanjanje redova s NA. Može se primijeniti na cijeli dataset ili specifične stupce.
Tidy data (uredni podaci)	Princip organizacije podataka: svaki redak je opažanje, svaki stupac varijabla, svaka ćelija vrijednost.
Široki format	Organizacija podataka u kojoj su različita mjerenja iste varijable raspoređena u zasebne stupce. Čitljiv za ljude.
Dugački format	Organizacija podataka u kojoj su različita mjerenja u zasebnim redovima s identifikacijskim stupcem. Pogodan za analizu i vizualizaciju.
Pipeline	Niz operacija spojenih pipe operatorom (`|>`) koji transformira podatke korak po korak.
Sirovi podaci (raw data)	Podaci u izvornom obliku, prije čišćenja. Ne bi ih trebalo mijenjati izravno.
Čisti podaci (clean data)	Podaci nakon standardizacije, rekodiranja i provjere. Spremni za analizu.

Pojam	Objašnjenje
Rekodiranje	Pretvaranje vrijednosti varijable u standardizirani oblik (npr. svih varijanti spola u “ženski”/“muški”).