Tjedan 4: Programiranje u R-u

Funkcije, uvjeti i ponovljive analize

2025-03-29

Table of contents

1 Koliko programiranja treba komunikolog? 3

2 Naši podaci: newsletter kampanje 3

3 Zašto funkcije? Problem kopiranja koda 4

4 Pisanje vlastite funkcije 6
- 4.1 Anatomija funkcije . . . 6
- 4.2 Funkcija s više argumenata . . . 7
- 4.3 Default vrijednosti argumenata . . . 7
- 4.4 Funkcija koja radi s tibbleom . . . 8
- 4.5 Funkcija koja vraća graf . . . 9

5 Uvjetne naredbe: if i else 11
- 5.1 Osnovna sintaksa . . . 11
- 5.2 if, else if, else . . . 12
- 5.3 Razlika između if/else i if_else()/case_when() . . . 13
- 5.4 Uvjeti u funkcijama: validacija ulaza . . . 13

6 For petlje: ponavljanje operacija 14
- 6.1 Osnovna for petlja . . . 14
- 6.2 For petlja za generiranje rezultata . . . 15
- 6.3 For petlja za generiranje grafova . . . 16

7 map(): moderna alternativa petljama 17
- 7.1 Osnovni map() . . . 17
- 7.2 Skraćena lambda sintaksa . . . 18
- 7.3 Varijante map-a . . . 19
- 7.4 map() unutar tibble radnog toka . . . 19

8 DRY princip i organizacija skripte **20**
8.1 Primjer: parametri na jednom mjestu . . . 20
8.2 Struktura analitičke skripte . . . 21
8.3 Pomoćne funkcije na vrhu skripte . . . 23

9 Praktični primjer: automatizirana analiza po kampanjama **24**

10 Rad s više datoteka **26**
10.1 Pronalaženje datoteka . . . 27
10.2 Učitavanje više datoteka odjednom . . . 27
10.3 Dodavanje informacije o izvoru . . . 28

11 Debugging: pronalaženje i ispravljanje grešaka **28**
11.1 Čitanje poruka o greškama . . . 28
11.2 Strategija: izoliraj problem . . . 29
11.3 print() i glimpse() kao dijagnostika . . . 30
11.4 Česte greške i rješenja . . . 31

12 Quarto: integracija koda, teksta i rezultata **32**
12.1 Struktura Quarto dokumenta . . . 32
12.2 Chunk opcije za kontrolu ispisa . . . 33
12.3 Inline R kod . . . 33
12.4 Quarto vs R skripta: kad koristiti što . . . 33

13 Funkcionali za složenije radne tokove **34**
13.1 walk(): map() bez povratne vrijednosti . . . 34
13.2 map2(): paralelna iteracija preko dva vektora . . . 34
13.3 imap(): iteracija s indeksom . . . 35
13.4 possibly(): zaštita od grešaka . . . 36

14 Kompletna analiza: automatizirani izvještaj o kampanjama **37**

15 Dodatno čitanje **43**

16 Pojmovnik **43**

```
library(tidyverse)
```

Ishodi učenja

Nakon ovog predavanja moći ćete

1. Objasniti zašto su vlastite funkcije korisne za izbjegavanje ponavljanja koda i smanjenje grešaka.
2. Napisati vlastitu R funkciju s argumentima i podrazumijevanim (default) vrijednostima.
3. Koristiti uvjetne naredbe (`if`, `else`, `if_else()`, `case_when()`) za kontrolu toka

programa.
4. Koristiti `for` petlje za ponavljanje operacija nad skupom elemenata.
5. Koristiti `map()` funkcije iz paketa purrr kao modernu alternativu petljama.
6. Primijeniti principe DRY (Don't Repeat Yourself) na pisanje analitičkih skripti.
7. Organizirati analitičku skriptu s jasnom strukturom, uključujući učitavanje, čišćenje, analizu, vizualizaciju i izvoz.
8. Prepoznati kada je pisanje vlastite funkcije isplativije od kopiranja koda.

1 Koliko programiranja treba komunikolog?

Ovo pitanje zaslužuje iskren odgovor. Ne trebate postati softverski inženjer. Ne trebate znati pisati web aplikacije, baze podataka ili algoritme strojnog učenja. Ali trebate znati dovoljno programiranja da vaše analize budu **ponovljive**, **prilagodljive** i **manje podložne greškama**.

Zamislite sljedeću situaciju. Radite analizu medijskih navika za klijenta. Napravili ste čišćenje podataka, deskriptivnu statistiku, osam grafova i izvještaj. Klijent je zadovoljan, ali tjedan dana kasnije kaže: "Dobili smo još 200 odgovora na anketu, možete li ponoviti analizu s novim podacima?" Ako ste sve radili ručno u Excelu, to znači ponoviti svaki korak od nule. Ako ste napisali R skriptu, to znači promijeniti jednu liniju koda (putanju do nove datoteke) i pokrenuti skriptu. Pet sekundi umjesto pet sati.

Programiranje u kontekstu analize podataka nije apstraktno akademsko znanje. To je praktična vještina koja vas čini bržima, preciznijima i profesionalnijima. U ovom tjednu naučit ćemo tri temeljne programerske koncepte (funkcije, uvjetne naredbe i iteraciju) i pokazati kako ih koristiti u kontekstu koji je relevantan za komunikologe.

---

2 Naši podaci: newsletter kampanje

Ovaj tjedan koristimo dataset o 50 newsletter kampanja jednog informativnog portala. Za svaku kampanju imamo podatke o tipu, stilu naslova, vremenu slanja, broju pretplatnika, open rateu (postotak otvaranja), click rateu (postotak klikova) i drugim metrikama.

```
nl <- read_csv("../resources/datasets/newsletter_campaigns.csv")
glimpse(nl)
```

```
Rows: 50
Columns: 13
$ campaign_id      <chr> "NL-001", "NL-002", "NL-003", "NL-004", "NL-
005", "NL~
```

```
$ campaign_type    <chr> "special_report", "weekly_digest", "special_report", ~
$ subject_style    <chr> "personalizirani", "hitno", "upitni", "informativni",~
$ day_sent         <chr> "petak", "petak", "utorak", "ponedjeljak", "utorak", ~
$ send_hour        <dbl> 8, 11, 8, 9, 20, 16, 13, 6, 11, 19, 11, 19, 9, 15, 18~
$ subscribers      <dbl> 11770, 14266, 10652, 23113, 9847, 9150, 23450, 12798,~
$ open_rate        <dbl> 0.2410, 0.2696, 0.3023, 0.2134, 0.2887, 0.1921, 0.288~
$ click_rate       <dbl> 0.0858, 0.0065, 0.0309, 0.0656, 0.0273, 0.0651, 0.015~
$ unsubscribe_rate <dbl> 0.00447, 0.00283, 0.00519, 0.00000, 0.00082, 0.00391,~
$ word_count       <dbl> 499, 378, 437, 545, 559, 146, 428, 210, 309, 519, 376~
$ n_links          <dbl> 2, 9, 4, 1, 3, 1, 10, 3, 8, 4, 9, 3, 8, 7, 5, 7, 1, 5~
$ has_image        <lgl> TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, FALS~
$ revenue          <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 4659.53, 0.~
```

```r
nl |>
  count(campaign_type, sort = TRUE)
```

```
# A tibble: 5 x 2
  campaign_type      n
  <chr>          <int>
1 special_report    17
2 weekly_digest     12
3 sponsored          8
4 breaking_news      7
5 event_promo        6
```

Ovo je manji dataset od prethodnih tjedana, ali upravo to ga čini pogodnim za učenje programiranja. S 50 redova možete vidjeti svaki korak i razumjeti što vaš kod radi.

---

3 Zašto funkcije? Problem kopiranja koda

Krenimo od konkretnog problema. Recimo da za svaki tip kampanje želite izračunati sažetak s prosjekom, medijanom i standardnom devijacijom open ratea. Jedan pristup je kopiranje koda.

```r
# Sažetak za weekly digest
nl |>
  filter(campaign_type == "weekly_digest") |>
  summarise(
    n = n(),
    or_prosjek = round(mean(open_rate), 3),
    or_medijan = round(median(open_rate), 3),
```

```
    or_sd = round(sd(open_rate), 3)
  )
```

```
# A tibble: 1 x 4
      n or_prosjek or_medijan or_sd
  <int>      <dbl>      <dbl> <dbl>
1    12      0.289       0.29 0.066
```

```
# Sažetak za breaking news (kopiran kod s jednom promjenom)
nl |>
  filter(campaign_type == "breaking_news") |>
  summarise(
    n = n(),
    or_prosjek = round(mean(open_rate), 3),
    or_medijan = round(median(open_rate), 3),
    or_sd = round(sd(open_rate), 3)
  )
```

```
# A tibble: 1 x 4
      n or_prosjek or_medijan or_sd
  <int>      <dbl>      <dbl> <dbl>
1     7      0.198      0.195 0.055
```

Ovo radi, ali ima tri problema. Prvo, ako želite promijeniti izračun (recimo dodati trimmed mean), morate to napraviti na svakom mjestu gdje ste kopirali kod. Drugo, svako kopiranje je prilika za grešku. Možda zaboravite promijeniti ime kampanje na jednom mjestu. Treće, kad imate pet ili deset tipova kampanja, kod postaje nepregledano dugačak.

Naravno, za ovaj specifični problem znamo elegantno rješenje s `group_by()`.

```
nl |>
  group_by(campaign_type) |>
  summarise(
    n = n(),
    or_prosjek = round(mean(open_rate), 3),
    or_medijan = round(median(open_rate), 3),
    or_sd = round(sd(open_rate), 3),
    .groups = "drop"
  )
```

```
# A tibble: 5 x 5
  campaign_type      n or_prosjek or_medijan or_sd
  <chr>          <int>      <dbl>      <dbl> <dbl>
```

```
1 breaking_news      7      0.198      0.195 0.055
2 event_promo        6      0.25       0.23  0.086
3 special_report    17      0.259      0.255 0.067
4 sponsored          8      0.248      0.246 0.043
5 weekly_digest     12      0.289      0.29  0.066
```

Ali `group_by()` ne rješava svaki problem. Kad trebate ponoviti složeniju analizu (koja uključuje čišćenje, više izračuna, graf i tablicu) za različite podskupove podataka, vlastite funkcije postaju nezamjenjive.

---

4 Pisanje vlastite funkcije

Funkcija u R-u je objekt koji prima ulazne podatke (argumente), izvršava niz operacija i vraća rezultat. Već koristite funkcije svaki dan; na primjer, `mean()`, `filter()` i `ggplot()` su sve funkcije koje je netko napisao. Sad ćete naučiti pisati vlastite.

4.1 Anatomija funkcije

```
# Funkcija koja pretvara postotke u razlomke
postotak_u_razlomak <- function(postotak) {
  postotak / 100
}

postotak_u_razlomak(25)
```

```
[1] 0.25
```

```
postotak_u_razlomak(73.5)
```

```
[1] 0.735
```

Raščlanimo sintaksu. `postotak_u_razlomak` je ime funkcije (kao ime bilo kojeg objekta, dodjeljujemo ga s `<-`). Ključna riječ `function()` govori R-u da kreiramo funkciju. Unutar zagrada su argumenti (ulazni podatci). Unutar vitičastih zagrada `{}` je tijelo funkcije (operacije koje se izvršavaju). Zadnji izraz u tijelu je povratna vrijednost (ono što funkcija vraća).

4.2 Funkcija s više argumenata

```
# Funkcija za izračun engagement ratea
engagement_rate <- function(clicks, opens) {
  rate <- clicks / opens
  round(rate, 4)
}

engagement_rate(clicks = 150, opens = 1200)
```

```
[1] 0.125
```

```
engagement_rate(clicks = 80, opens = 500)
```

```
[1] 0.16
```

Funkcija prima dva argumenta i vraća zaokruženi omjer. Kad pozivate funkciju, argumente možete navesti po imenu (`clicks = 150`) ili po poziciji. Po imenu je sigurnije jer nije bitno kojim redoslijedom ih navedete.

4.3 Default vrijednosti argumenata

Ponekad želite da argument ima podrazumijevanu (default) vrijednost koju korisnik može promijeniti ako želi.

```
# Funkcija za sažetak numeričke varijable
sazetak_varijable <- function(x, decimale = 2) {
  tibble(
    n = length(x),
    n_NA = sum(is.na(x)),
    prosjek = round(mean(x, na.rm = TRUE), decimale),
    medijan = round(median(x, na.rm = TRUE), decimale),
    sd = round(sd(x, na.rm = TRUE), decimale),
    min = round(min(x, na.rm = TRUE), decimale),
    max = round(max(x, na.rm = TRUE), decimale)
  )
}

# Korištenje s default decimala (2)
sazetak_varijable(nl$open_rate)
```

```
# A tibble: 1 x 7
      n  n_NA prosjek medijan    sd   min   max
  <int> <int>   <dbl>   <dbl> <dbl> <dbl> <dbl>
1    50     0    0.25    0.25  0.07  0.11  0.41
```

```
# Korištenje s 4 decimale
sazetak_varijable(nl$open_rate, decimale = 4)
```

```
# A tibble: 1 x 7
      n  n_NA prosjek medijan     sd   min   max
  <int> <int>   <dbl>   <dbl>  <dbl> <dbl> <dbl>
1    50     0   0.255   0.252 0.0676 0.115 0.408
```

Argument `decimale = 2` ima default vrijednost 2. Ako ga ne navedete pri pozivu, koristi se 2. Ako ga eksplicitno navedete, koristi se vaša vrijednost. Ovo čini funkciju fleksibilnom bez opterećivanja korisnika nepotrebnim odlukama.

4.4 Funkcija koja radi s tibbleom

Funkcije koje primaju cijeli tibble i koriste dplyr glagole unutar sebe su izuzetno korisne u praksi.

```
# Funkcija za sažetak kampanje po tipu
sazetak_kampanje <- function(data, tip) {
  data |>
    filter(campaign_type == tip) |>
    summarise(
      tip = tip,
      n = n(),
      prosjek_or = round(mean(open_rate), 3),
      prosjek_ctr = round(mean(click_rate), 4),
      ukupni_doseg = sum(subscribers),
      .groups = "drop"
    )
}

sazetak_kampanje(nl, "weekly_digest")
```

```
# A tibble: 1 x 5
  tip               n prosjek_or prosjek_ctr ukupni_doseg
  <chr>         <int>      <dbl>       <dbl>        <dbl>
1 weekly_digest    12      0.289       0.029       217303
```

```
sazetak_kampanje(nl, "breaking_news")
```

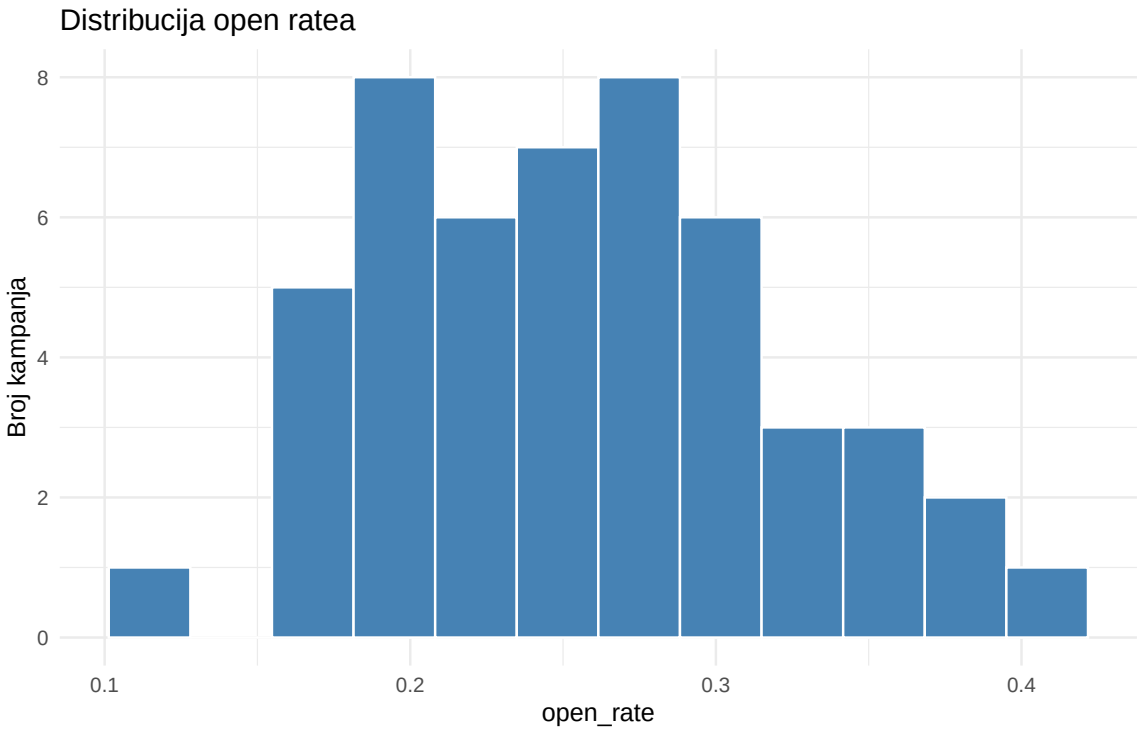
```
# A tibble: 1 x 5
  tip               n prosjek_or prosjek_ctr ukupni_doseg
  <chr>         <int>      <dbl>       <dbl>        <dbl>
1 breaking_news     7      0.198      0.0382       120123
```

Sad umjesto kopiranja pet blokova koda, pozivamo jednu funkciju s različitim argumentom. Ako želite promijeniti izračun (dodati novu metriku), mijenjate na jednom mjestu i promjena se automatski primjenjuje svugdje.

4.5 Funkcija koja vraća graf

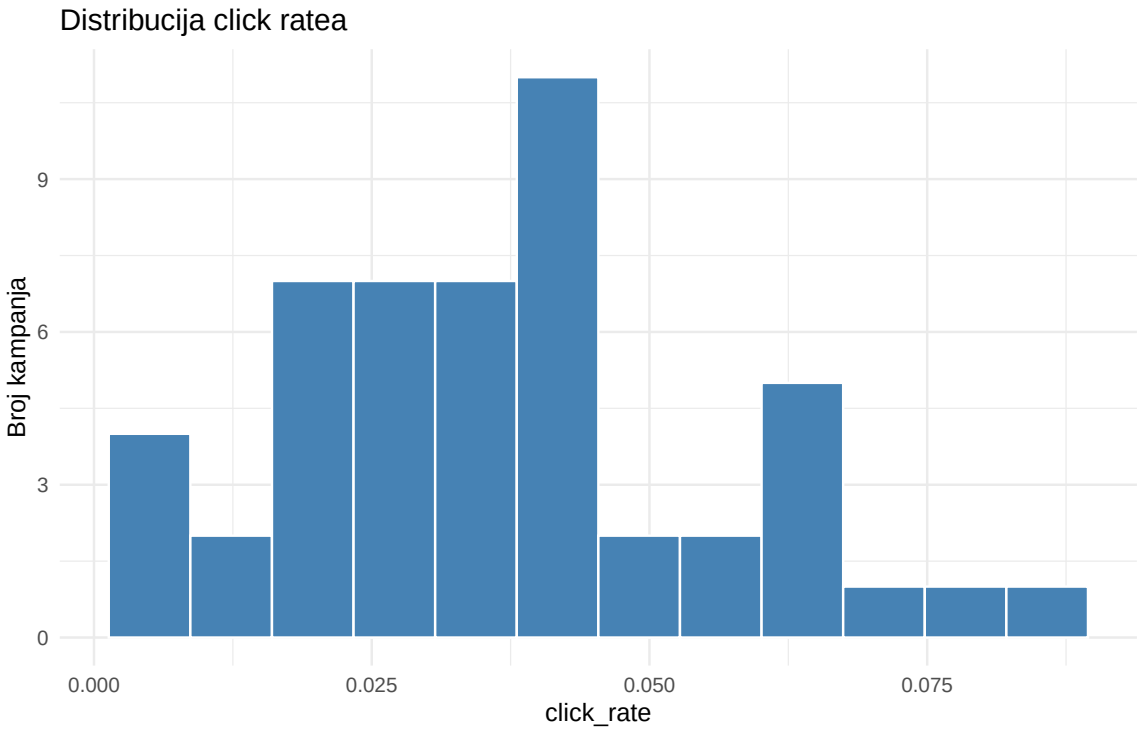
Funkcije mogu vraćati bilo koji R objekt, uključujući ggplot grafove.

```
graf_distribucije <- function(data, varijabla, naslov) {
  data |>
    ggplot(aes(x = .data[[varijabla]])) +
    geom_histogram(fill = "steelblue", color = "white", bins = 12) +
    labs(title = naslov, x = varijabla, y = "Broj kampanja") +
    theme_minimal()
}

graf_distribucije(nl, "open_rate", "Distribucija open ratea")
```



Konstrukcija `.data[[varijabla]]` omogućuje prosljeđivanje imena stupca kao teksta. Ovo je tehnički detalj tidyverse programiranja koji je koristan kad pišete funkcije koje rade s različitim stupcima.

```
graf_distribucije(nl, "click_rate", "Distribucija click ratea")
```



Ista funkcija, druga varijabla, novi graf. Ovo je suština DRY principa—napišete logiku jednom i koristite je koliko god puta trebate.

> **Praktični savjet**
>
> Prema Pravilu tri, ako ste kopirali isti blok koda tri puta ili više, vrijeme je da ga pretvorite u funkciju. Dva kopiranja su još prihvatljiva (ponekad je brže kopirati nego pisati funkciju), ali tri signaliziraju obrazac koji će se ponavljati i dalje. Funkcija vam štedi vrijeme dugoročno i smanjuje rizik od grešaka pri kopiranju.

---

5 Uvjetne naredbe: if i else

Uvjetne naredbe omogućuju R-u da donese odluku—ako je uvjet ispunjen, napravi jedno, inače napravi drugo. Već smo koristili `if_else()` i `case_when()` unutar `mutate()` za rekodiranje varijabli. Sad učimo klasične `if`/`else` naredbe koje rade izvan tibble konteksta.

5.1 Osnovna sintaksa

```r
prosjek_or <- mean(nl$open_rate)

if (prosjek_or > 0.25) {
  cat("Prosječni open rate je iznad 25%, što je odličan rezultat.\n")
} else {
  cat("Prosječni open rate je ispod 25%, ima prostora za poboljšanje.\n")
}
```

```
Prosječni open rate je iznad 25%, što je odličan rezultat.
```

R evaluira uvjet u zagradi. Ako je TRUE, izvršava kod u prvom bloku. Ako je FALSE, izvršava kod u else bloku. Funkcija `cat()` ispisuje tekst u konzolu (slično `print()`, ali bez dodatnih oznaka).

5.2 if, else if, else

Za više od dva ishoda, koristite `else if`.

```r
ocijeni_kampanju <- function(open_rate) {
  if (open_rate > 0.30) {
    "izvrsna"
  } else if (open_rate > 0.20) {
    "dobra"
  } else if (open_rate > 0.10) {
    "prosječna"
  } else {
    "loša"
  }
}

ocijeni_kampanju(0.35)
```

```
[1] "izvrsna"
```

```r
ocijeni_kampanju(0.22)
```

```
[1] "dobra"
```

```r
ocijeni_kampanju(0.08)
```

```
[1] "loša"
```

Uvjeti se provjeravaju redom, od vrha prema dnu. Čim je jedan uvjet TRUE, pripadajuća vrijednost se vraća i R ne provjerava preostale uvjete. Zato uvjete postavljamo od najstrožeg prema najblažem.

5.3 Razlika između if/else i if_else()/case_when()

Ovo je česta točka zbunjenosti. Postoje dva različita sustava uvjetnog izvršavanja u R-u i svaki ima svoje mjesto.

Klasični `if/else` radi s jednom vrijednošću. Koristi se u funkcijama, skriptama i kontroli toka programa. Nije vektoriziran, što znači da ne može obrađivati cijeli stupac odjednom.

`if_else()` i `case_when()` su vektorizirane funkcije. Rade s cijelim vektorom (stupcem) odjednom i koriste se unutar `mutate()` za rekodiranje varijabli u tibbleu.

```
# if_else() unutar mutate: radi na cijelom stupcu
nl |>
  mutate(
    ocjena = if_else(open_rate > 0.25, "iznad prosjeka", "ispod prosjeka")
  ) |>
  count(ocjena)
```

```
# A tibble: 2 x 2
  ocjena             n
  <chr>          <int>
1 ispod prosjeka    25
2 iznad prosjeka    25
```

```
# Klasični if/else: radi s jednom vrijednošću
# (koristili smo ga u funkciji ocijeni_kampanju)
```

Pravilo je jednostavno. Unutar `mutate()` koristite `if_else()` ili `case_when()`. Izvan `mutate()`, u funkcijama i skriptama, koristite klasični `if/else`.

5.4 Uvjeti u funkcijama: validacija ulaza

Praktična primjena `if/else` u funkcijama je provjera jesu li ulazni podaci ispravni.

```
izracunaj_ctr <- function(clicks, impressions) {
  if (impressions <= 0) {
    warning("Broj impresija mora biti pozitivan. Vraćam NA.")
    return(NA_real_)
  }
```

```r
  if (clicks < 0) {
    warning("Broj klikova ne može biti negativan. Vraćam NA.")
    return(NA_real_)
  }

  round(clicks / impressions, 4)
}

izracunaj_ctr(150, 5000)
```

```
[1] 0.03
```

```r
izracunaj_ctr(150, 0)
```

```
[1] NA
```

```r
izracunaj_ctr(-10, 5000)
```

```
[1] NA
```

Funkcija `warning()` ispisuje upozorenje ali ne zaustavlja izvršavanje. Funkcija `return()` eksplicitno vraća vrijednost i izlazi iz funkcije. Bez `return()`, funkcija bi nastavila izvršavanje i pokušala podijeliti s nulom.

Validacija ulaza je ono što razdvaja robusne funkcije od krhkih. Kad pišete funkciju za sebe, možda znate da nikad nećete unijeti negativan broj. Ali kad tu funkciju koristi netko drugi (ili vi za šest mjeseci, kad ste zaboravili detalje), validacija sprečava tihe greške.

---

6 For petlje: ponavljanje operacija

Petlja je naredba koja ponavlja blok koda za svaki element u skupu. `for` petlja u R-u ima jednostavnu sintaksu.

6.1 Osnovna for petlja

```r
tipovi <- unique(nl$campaign_type)

for (tip in tipovi) {
  n <- nl |> filter(campaign_type == tip) |> nrow()
  cat(tip, ":", n, "kampanja\n")
}
```

```
special_report : 17 kampanja
weekly_digest : 12 kampanja
breaking_news : 7 kampanja
sponsored : 8 kampanja
event_promo : 6 kampanja
```

R prolazi kroz svaki element vektora `tipovi`, dodjeljuje ga varijabli `tip`, i izvršava kod u tijelu petlje. Kad se tijelo izvrši za zadnji element, petlja završava.

6.2 For petlja za generiranje rezultata

Čest obrazac je korištenje petlje za prikupljanje rezultata u listu ili tibble.

```r
# Inicijalizirajte praznu listu za rezultate
rezultati <- list()

for (tip in tipovi) {
  saz <- nl |>
    filter(campaign_type == tip) |>
    summarise(
      tip = tip,
      n = n(),
      prosjek_or = round(mean(open_rate), 3),
      prosjek_ctr = round(mean(click_rate), 4)
    )

  rezultati[[tip]] <- saz
}

# Spojite sve rezultate u jedan tibble
bind_rows(rezultati)
```

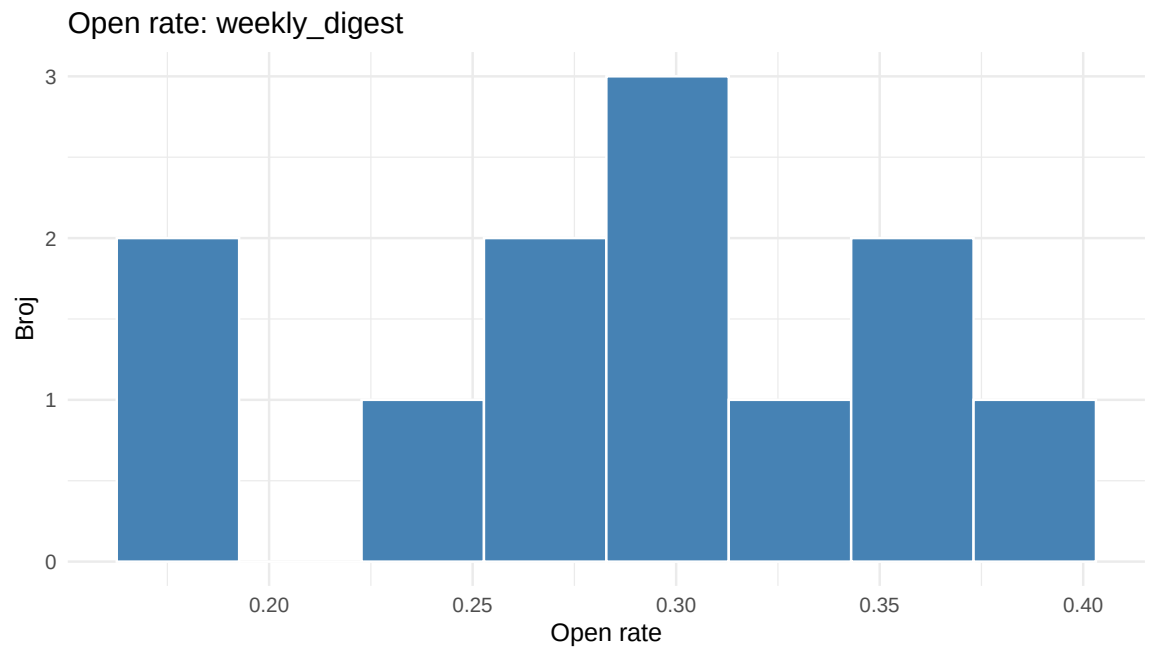
```
# A tibble: 5 x 4
  tip                n prosjek_or prosjek_ctr
  <chr>          <int>      <dbl>       <dbl>
1 special_report    17      0.259      0.0519
```

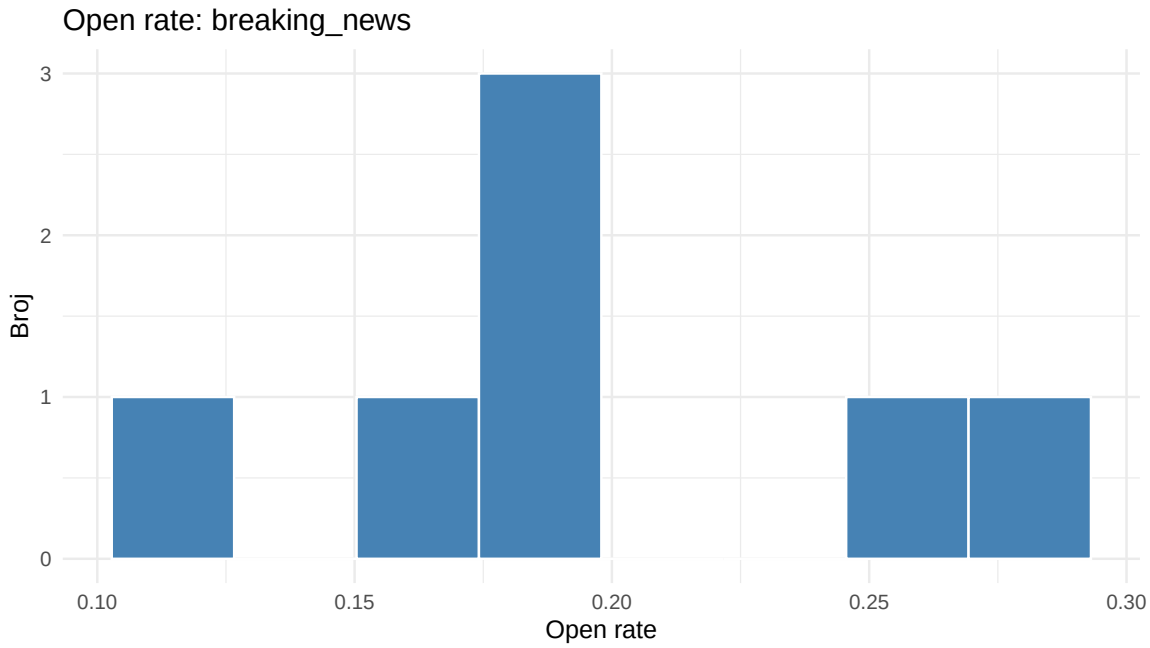
```
2 weekly_digest      12       0.289       0.029
3 breaking_news       7       0.198       0.0382
4 sponsored           8       0.248       0.0223
5 event_promo         6       0.25        0.0294
```

Kreiramo praznu listu `rezultati`, u svakoj iteraciji računamo sažetak i pohranjujemo ga u listu pod imenom tipa kampanje, a na kraju sve spajamo u jedan tibble s `bind_rows()`.

6.3 For petlja za generiranje grafova

```r
# Generiranje grafa za svaki tip kampanje
for (tip in c("weekly_digest", "breaking_news")) {
  p <- nl |>
    filter(campaign_type == tip) |>
    ggplot(aes(x = open_rate)) +
    geom_histogram(fill = "steelblue", color = "white", bins = 8) +
    labs(
      title = paste("Open rate:", tip),
      x = "Open rate",
      y = "Broj"
    ) +
    theme_minimal()

  print(p)
}
```





Unutar for petlje, ggplot grafove morate eksplicitno ispisati s `print()`. Izvan petlje, R automatski ispisuje zadnji objekt, ali unutar petlje to ne radi. Ovo je čest izvor frustracije za početnike.

> **!** Važna napomena
>
> For petlje u R-u nisu pogrešne ni zastarjele, ali za većinu zadataka u tidyverse ekosustavu postoje elegantnije alternative. `group_by() |> summarise()` zamjenjuje petlje za grupirane sažetke. `across()` zamjenjuje petlje za primjenu iste operacije na više stupaca. `map()` iz paketa purrr zamjenjuje petlje za primjenu funkcije na svaki element liste ili vektora. Petlje koristite kad alternative ne postoje ili kad je petlja jasnija (što se ponekad događa).

---

7 map(): moderna alternativa petljama

Paket purrr (dio tidyverse) pruža obitelj `map()` funkcija koje primjenjuju funkciju na svaki element vektora ili liste. Rezultat ovisi o varijanti map-a koju koristite.

7.1 Osnovni map()

```r
tipovi <- unique(nl$campaign_type)

# map() vraća listu
rezultati <- map(tipovi, function(tip) {
  nl |>
    filter(campaign_type == tip) |>
    summarise(
      tip = tip,
      n = n(),
      prosjek_or = round(mean(open_rate), 3)
    )
})

bind_rows(rezultati)
```

```
# A tibble: 5 x 3
  tip                n prosjek_or
  <chr>          <int>      <dbl>
1 special_report    17      0.259
2 weekly_digest     12      0.289
3 breaking_news      7      0.198
4 sponsored          8      0.248
5 event_promo        6      0.25
```

`map()` prima vektor (ili listu) i funkciju, primjenjuje funkciju na svaki element i vraća listu rezultata. Ovo je funkcionalni ekvivalent for petlje ali u kompaktnijem obliku.

7.2 Skraćena lambda sintaksa

Umjesto `function(tip) { ... }` možete koristiti skraćenu lambda sintaksu s tildom.

```r
# Skraćena lambda sintaksa: \(x) umjesto function(x)
rezultati <- map(tipovi, \(tip) {
  nl |>
    filter(campaign_type == tip) |>
    summarise(tip = tip, n = n(), prosjek_or = round(mean(open_rate), 3))
})

bind_rows(rezultati)
```

```
# A tibble: 5 x 3
  tip                n prosjek_or
```

```
  <chr>          <int>      <dbl>
1 special_report    17      0.259
2 weekly_digest     12      0.289
3 breaking_news      7      0.198
4 sponsored          8      0.248
5 event_promo        6      0.25
```

Notacija `\(tip)` je R-ova nova (od verzije 4.1) skraćenica za `function(tip)`. Obje verzije rade identično, ali `\(x)` je kraća za pisanje.

7.3 Varijante map-a

`map()` uvijek vraća listu. Kad znate kakav tip rezultata očekujete, koristite specifičniju varijantu.

```
# map_dbl() vraća numerički vektor
prosjeci <- map_dbl(tipovi, \(tip) {
  nl |>
    filter(campaign_type == tip) |>
    pull(open_rate) |>
    mean()
})

tibble(tip = tipovi, prosjek_or = round(prosjeci, 3))
```

```
# A tibble: 5 x 2
  tip            prosjek_or
  <chr>               <dbl>
1 special_report      0.259
2 weekly_digest       0.289
3 breaking_news       0.198
4 sponsored           0.248
5 event_promo         0.25
```

`map_dbl()` vraća numerički (double) vektor umjesto liste. `map_chr()` vraća tekstualni vektor. `map_lgl()` vraća logički. `map_df()` vraća tibble (spaja sve rezultate). Koristite specifičnu varijantu kad god možete jer je rezultat predvidljiviji i jednostavniji za daljnji rad.

7.4 map() unutar tibble radnog toka

Najelegantnija primjena `map()` je unutar tibble radnog toka, kombinirano s `nest()` i `unnest()`.

```r
nl |>
  group_by(campaign_type) |>
  nest() |>
  mutate(
    n = map_dbl(data, nrow),
    prosjek_or = map_dbl(data, \(df) mean(df$open_rate)),
    prosjek_ctr = map_dbl(data, \(df) mean(df$click_rate))
  ) |>
  select(-data) |>
  mutate(across(c(prosjek_or, prosjek_ctr), \(x) round(x, 4)))
```

```
# A tibble: 5 x 4
# Groups:   campaign_type [5]
  campaign_type      n prosjek_or prosjek_ctr
  <chr>          <dbl>      <dbl>       <dbl>
1 special_report    17      0.259      0.0519
2 weekly_digest     12      0.289      0.029
3 breaking_news      7      0.198      0.0382
4 sponsored          8      0.248      0.0223
5 event_promo        6      0.250      0.0294
```

Funkcija `nest()` pakira podatke svake grupe u zasebni tibble unutar liste-stupca `data`. Zatim `map_dbl()` primjenjuje funkciju na svaki od tih ugniježđenih tibbleova. Rezultat je jedan redak po grupi s izračunatim metrikama.

Ovo je napredni obrazac koji ćete cijeniti kad budete radili složenije analize (na primjer, fitanje zasebnog regresijskog modela za svaku grupu u tjednu 14).

---

8 DRY princip i organizacija skripte

DRY (Don't Repeat Yourself) je programerski princip koji kaže da svaka informacija u kodu treba postojati na jednom mjestu. Kad se ponavljate, stvarate više točaka koje trebate ažurirati kad nešto promijenite, a to je recept za greške.

8.1 Primjer: parametri na jednom mjestu

```r
# PARAMETRI (mijenjajte ovdje, promjena se propagira svugdje)
min_kampanja_za_analizu <- 5
decimale <- 3
boja_grafova <- "steelblue"
kategorije_interesa <- c("weekly_digest", "breaking_news", "special_report")

# ANALIZA (koristi parametre odozgo)
nl_filtered <- nl |>
  filter(campaign_type %in% kategorije_interesa)

nl_filtered |>
  group_by(campaign_type) |>
  summarise(
    n = n(),
    prosjek_or = round(mean(open_rate), decimale),
    prosjek_ctr = round(mean(click_rate), decimale),
    .groups = "drop"
  ) |>
  filter(n >= min_kampanja_za_analizu)
```

```
# A tibble: 3 x 4
  campaign_type      n prosjek_or prosjek_ctr
  <chr>          <int>      <dbl>       <dbl>
1 breaking_news      7      0.198       0.038
2 special_report    17      0.259       0.052
3 weekly_digest     12      0.289       0.029
```

Svi ključni parametri su definirani na jednom mjestu na vrhu. Kad klijent kaže "pokaži mi analizu samo za weekly digest i special report", mijenjate jednu liniju i cijela analiza se ažurira. Ovo je fundamentalno drugačije od traženja i zamjenjivanja vrijednosti razbacanih po cijelom kodu.

8.2 Struktura analitičke skripte

Dobro organizirana skripta ima jasne sekcije. Svaka sekcija radi jednu stvar i jasno je označena.

```r
# ============================================
# Analiza newsletter kampanja
# Autor: Ime Prezime
# Datum: 2025-03-29
# Opis: Sažetak performansi email kampanja
# ============================================
```

```r
# 1. PAKETI ----
library(tidyverse)

# 2. PARAMETRI ----
input_file <- "../resources/datasets/newsletter_campaigns.csv"
output_dir <- "../outputs/"
min_n <- 5

# 3. UČITAVANJE ----
raw <- read_csv(input_file)

# 4. ČIŠĆENJE ----
clean <- raw |>
  filter(!is.na(open_rate)) |>
  mutate(
    campaign_type = factor(campaign_type),
    ocjena = case_when(
      open_rate > 0.30 ~ "izvrsna",
      open_rate > 0.20 ~ "dobra",
      open_rate > 0.10 ~ "prosjecna",
      .default = "losa"
    )
  )

# 5. ANALIZA ----
sazetak <- clean |>
  group_by(campaign_type) |>
  summarise(
    n = n(),
    M_or = mean(open_rate),
    SD_or = sd(open_rate),
    M_ctr = mean(click_rate),
    .groups = "drop"
  )

# 6. VIZUALIZACIJA ----
graf <- ggplot(clean, aes(x = campaign_type, y = open_rate)) +
  geom_boxplot(fill = "steelblue", alpha = 0.6) +
  theme_minimal() +
  labs(title = "Open rate po tipu kampanje")

# 7. IZVOZ ----
write_csv(sazetak, paste0(output_dir, "sazetak_kampanja.csv"))
ggsave(paste0(output_dir, "boxplot_open_rate.png"), graf, width = 8, height = 5)
```

Komentari s četiri crtice (`# 1. PAKETI ----`) stvaraju navigacijske oznake u Positronu (ili RStudiju) koje omogućuju brzo skakanje između sekcija. Ovo je konvencija, ne sintaktičko pravilo, ali je široko prihvaćena u R zajednici.

> Vaša skripta je vaš laboratorijski dnevnik. Svaki korak je dokumentiran, svaka odluka komentirana, svaki rezultat ponovljiv. Netko (uključujući vas za šest mjeseci) mora moći pokrenuti skriptu od početka do kraja i dobiti identične rezultate.

8.3 Pomoćne funkcije na vrhu skripte

Kad imate funkcije koje koristite na više mjesta u analizi, definirajte ih odmah nakon učitavanja paketa. Ovo ih čini vidljivima kroz cijelu skriptu.

```
# Pomoćne funkcije za newsletter analizu
sazetak_metrike <- function(data, metrika, decimale = 3) {
  data |>
    summarise(
      M = round(mean(.data[[metrika]], na.rm = TRUE), decimale),
      Med = round(median(.data[[metrika]], na.rm = TRUE), decimale),
      SD = round(sd(.data[[metrika]], na.rm = TRUE), decimale),
      Min = round(min(.data[[metrika]], na.rm = TRUE), decimale),
      Max = round(max(.data[[metrika]], na.rm = TRUE), decimale)
    )
}

ocjena_kampanje <- function(open_rate) {
  case_when(
    open_rate > 0.30 ~ "izvrsna",
    open_rate > 0.20 ~ "dobra",
    open_rate > 0.10 ~ "prosjecna",
    .default = "losa"
  )
}

# Korištenje pomoćnih funkcija
nl |>
  group_by(campaign_type) |>
  sazetak_metrike("open_rate")
```

```
# A tibble: 5 x 6
  campaign_type      M   Med    SD   Min   Max
  <chr>          <dbl> <dbl> <dbl> <dbl> <dbl>
1 breaking_news  0.198 0.195 0.055 0.115 0.281
2 event_promo    0.25  0.23  0.086 0.161 0.408
```

```
3 special_report 0.259 0.255 0.067 0.155 0.393
4 sponsored      0.248 0.246 0.043 0.191 0.309
5 weekly_digest  0.289 0.29  0.066 0.178 0.388
```

```r
nl |>
  mutate(ocjena = ocjena_kampanje(open_rate)) |>
  count(ocjena, sort = TRUE)
```

```
# A tibble: 3 x 2
  ocjena        n
  <chr>     <int>
1 dobra        25
2 prosjecna    14
3 izvrsna      11
```

Definirajući `ocjena_kampanje()` kao funkciju, logiku rekodiranja pišete jednom. Ako se kriteriji promijene (recimo, prag za "izvrsno" padne na 0.28), mijenjate na jednom mjestu.

---

9 Praktični primjer: automatizirana analiza po kampanjama

Spojimo sve naučene koncepte u jednom praktičnom primjeru. Cilj je napisati kod koji za svaki tip kampanje generira sažetak tablica i graf, koristeći funkcije, map i DRY principe.

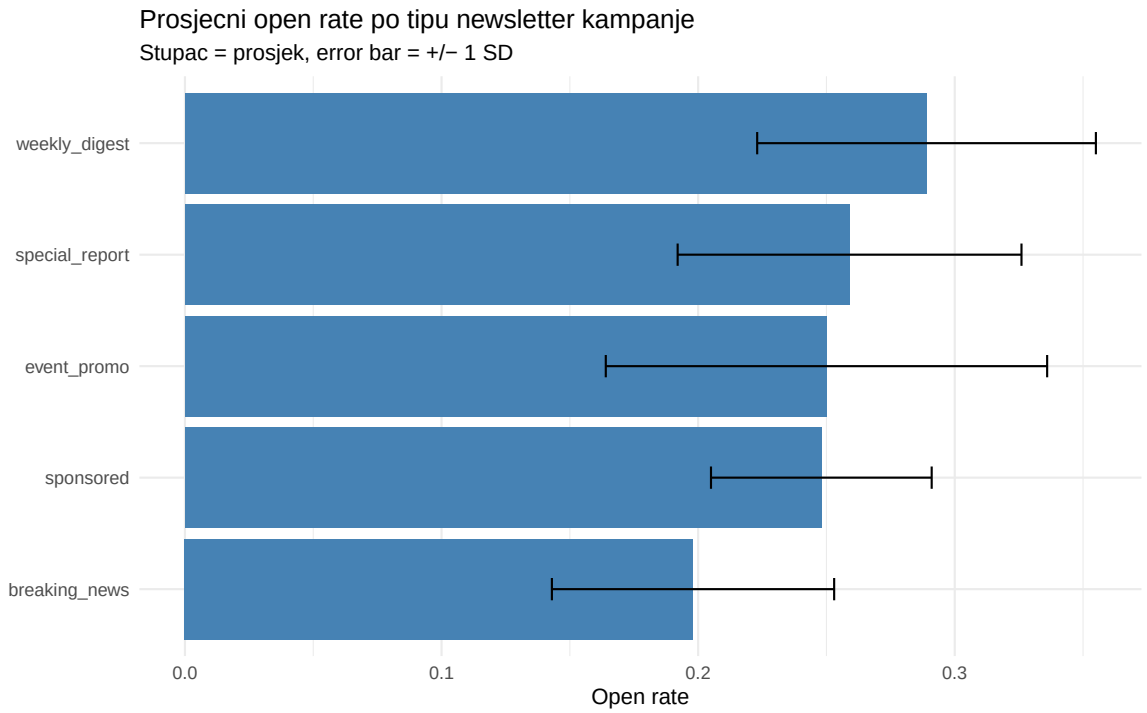
```r
# Funkcija za kompletnu analizu jednog tipa kampanje
analiziraj_tip <- function(data, tip) {
  podaci <- data |> filter(campaign_type == tip)

  if (nrow(podaci) < 3) {
    return(NULL)  # Preskoči tipove s premalo podataka
  }

  saz <- podaci |>
    summarise(
      tip = tip,
      n = n(),
      or_M = round(mean(open_rate), 3),
      or_SD = round(sd(open_rate), 3),
      ctr_M = round(mean(click_rate), 4),
      prosj_pretplatnika = round(mean(subscribers), 0),
      prosj_rijeci = round(mean(word_count), 0)
    )
```

```
  saz
}

# Primjena na sve tipove
svi_tipovi <- unique(nl$campaign_type)

rezultati <- map(svi_tipovi, \(tip) analiziraj_tip(nl, tip)) |>
  bind_rows()

rezultati |>
  arrange(desc(or_M))
```

```
# A tibble: 5 x 7
  tip                n  or_M or_SD  ctr_M prosj_pretplatnika prosj_rijeci
  <chr>          <int> <dbl> <dbl>  <dbl>              <dbl>        <dbl>
1 weekly_digest     12 0.289 0.066 0.029                18109          380
2 special_report    17 0.259 0.067 0.0519               17581          486
3 event_promo        6 0.25  0.086 0.0294               12938          210
4 sponsored          8 0.248 0.043 0.0223               13609          204
5 breaking_news      7 0.198 0.055 0.0382               17160          140
```

Ovaj pristup ima nekoliko prednosti. Logika analize je enkapsulirana u jednu funkciju. Validacija (`if (nrow(podaci) < 3)`) osigurava da ne radimo besmislene izračune na premalo podataka. `map()` elegantno primjenjuje funkciju na sve tipove. Rezultat je čist tibble sortiran po open rateu.

```
# Vizualizacija rezultata
rezultati |>
  mutate(tip = fct_reorder(tip, or_M)) |>
  ggplot(aes(x = tip, y = or_M)) +
  geom_col(fill = "steelblue") +
  geom_errorbar(
    aes(ymin = or_M - or_SD, ymax = or_M + or_SD),
    width = 0.2
  ) +
  coord_flip() +
  labs(
    title = "Prosječni open rate po tipu newsletter kampanje",
    subtitle = "Stupac = prosjek, error bar = +/- 1 SD",
    x = NULL,
    y = "Open rate"
  ) +
  theme_minimal()
```



Error barovi (crte pogreške) prikazuju jednu standardnu devijaciju iznad i ispod prosjeka, dajući vizualni uvid u varijabilnost unutar svake kategorije. Breaking news kampanje imaju viši prosječni open rate ali i veću varijabilnost, dok su sponsored kampanje konzistentno niže.

Podsjetnik

U prvom dijelu naučili smo pisati vlastite funkcije, koristiti uvjetne naredbe, for petlje i map() iz paketa purrr. U ovom dijelu primjenjujemo te vještine na realne radne tokove, kao što su rad s više datoteka, debugging, Quarto izvještaji i kompletna automatizirana analiza.

10 Rad s više datoteka

U praksi, podaci rijetko dolaze u jednoj datoteci. Možda imate zasebne CSV datoteke za svaki mjesec, za svaku kampanju ili za svaki izvor podataka. Umjesto ručnog učitavanja svake datoteke, možete automatizirati taj proces koristeći koncepte koje smo upravo naučili.

10.1 Pronalaženje datoteka

Funkcija `list.files()` pronalazi datoteke u direktoriju po zadanom uzorku.

```
# Popis svih CSV datoteka u datasets direktoriju
csv_datoteke <- list.files(
  path = "../resources/datasets/",
  pattern = "\\.csv$",
  full.names = TRUE
)

csv_datoteke
```

```
 [1] "../resources/datasets/ab_test_headlines.csv"
 [2] "../resources/datasets/article_engagement.csv"
 [3] "../resources/datasets/article_visuals.csv"
 [4] "../resources/datasets/instagram_ab_test.csv"
 [5] "../resources/datasets/media_habits_raw.csv"
 [6] "../resources/datasets/media_population.csv"
 [7] "../resources/datasets/media_survey_chi2.csv"
 [8] "../resources/datasets/media_trust.csv"
 [9] "../resources/datasets/news_credibility.csv"
[10] "../resources/datasets/newsletter_campaign.csv"
[11] "../resources/datasets/newsletter_campaigns.csv"
[12] "../resources/datasets/social_engagement.csv"
[13] "../resources/datasets/social_media_survey.csv"
[14] "../resources/datasets/social_posts.csv"
[15] "../resources/datasets/tiktok_usage.csv"
```

Argument `pattern = "\\.csv$"` koristi regularni izraz za pronalaženje datoteka koje završavaju s .csv. `full.names = TRUE` vraća kompletne putanje (ne samo imena datoteka), što je bitno jer ih trebamo za učitavanje.

10.2 Učitavanje više datoteka odjednom

Kombinirajmo `list.files()`, `map()` i `bind_rows()` za učitavanje i spajanje svih CSV datoteka u jednom koraku.

```
# Učitaj sve CSV datoteke i spoji ih
svi_podaci <- csv_datoteke |>
  map(\(f) read_csv(f, show_col_types = FALSE)) |>
  bind_rows()
```

Ovo je moćan obrazac. `map()` primjenjuje `read_csv()` na svaku putanju, vraćajući listu tibbleova. `bind_rows()` ih vertikalno spaja u jedan veliki tibble. Ako datoteke imaju iste stupce, rezultat je jednostavna konkatenacija. Ako se stupci razlikuju, `bind_rows()` popunjava nedostajuće s NA.

10.3 Dodavanje informacije o izvoru

Često želite znati iz koje datoteke dolazi koji redak. Funkcija `set_names()` pomaže.

```
# Učitaj sve datoteke i dodaj stupac s imenom datoteke
svi_podaci <- csv_datoteke |>
  set_names() |>
  map(\(f) read_csv(f, show_col_types = FALSE)) |>
  bind_rows(.id = "izvor")
```

Argument `.id = "izvor"` u `bind_rows()` kreira novi stupac `izvor` koji sadrži ime elementa liste (u ovom slučaju putanju datoteke). Ovo je korisno za praćenje porijekla podataka.

> **Praktični savjet**
>
> Obrazac `list.files() |> map(read_csv) |> bind_rows()` je jedan od najkorisnijih obrazaca u cijelom R radnom toku. Naučite ga napamet. Koristit ćete ga svaki put kad dobijete podatke razdijeljene u više datoteka (mjesečni izvještaji, odvojene ankete, logovi po danima).

---

11 Debugging: pronalaženje i ispravljanje grešaka

Greške su neizbježan dio programiranja. Pitanje nije hoćete li naletjeti na grešku, nego koliko ćete brzo identificirati i ispraviti problem. R daje poruke o greškama koje su ponekad jasne, a ponekad kriptične. Evo strategija za sustavno traženje problema.

11.1 Čitanje poruka o greškama

```
# Tipična greška: objekt ne postoji
nl |>
  filter(kampanja_tip == "weekly_digest")
# Error: object 'kampanja_tip' not found

# Čitamo: R ne može naći objekt 'kampanja_tip'
```

```
# Rješenje: provjerimo imena stupaca
names(nl)
# Ah, stupac se zove 'campaign_type', ne 'kampanja_tip'
```

Poruka "object not found" gotovo uvijek znači jednu od tri stvari. Ili ste napravili tipfeler u imenu, ili objekt još nije kreiran (izvršili ste kod izvan redoslijeda), ili je objekt u drugom okruženju (na primjer, kreiran unutar funkcije ali ne i izvan nje).

11.2 Strategija: izoliraj problem

Kad imate dugački pipeline koji ne radi, razbijte ga na dijelove i pokrenite svaki zasebno.

```
# Umjesto pokretanja cijelog pipelinea odjednom:
# nl |> filter(...) |> mutate(...) |> group_by(...) |> summarise(...)

# Pokrenite korak po korak:
korak1 <- nl |> filter(campaign_type == "weekly_digest")
korak1  # Provjerite: izgleda li ovo kako očekujete?
```

```
# A tibble: 12 x 13
   campaign_id campaign_type subject_style   day_sent    send_hour subscribers
   <chr>       <chr>         <chr>           <chr>           <dbl>       <dbl>
 1 NL-002      weekly_digest hitno           petak              11       14266
 2 NL-007      weekly_digest upitni          srijeda            13       23450
 3 NL-009      weekly_digest personalizirani subota             11       12444
 4 NL-011      weekly_digest personalizirani utorak             11       23941
 5 NL-016      weekly_digest informativni    cetvrtak           15       12188
 6 NL-022      weekly_digest hitno           utorak             18       12768
 7 NL-026      weekly_digest brojke          utorak              7       12035
 8 NL-032      weekly_digest personalizirani utorak             18       18310
 9 NL-034      weekly_digest brojke          cetvrtak           20       24014
10 NL-035      weekly_digest hitno           ponedjeljak         7       19975
11 NL-037      weekly_digest personalizirani nedjelja           18       20070
12 NL-038      weekly_digest informativni    nedjelja           14       23842
# i 7 more variables: open_rate <dbl>, click_rate <dbl>,
#   unsubscribe_rate <dbl>, word_count <dbl>, n_links <dbl>, has_image <lgl>,
#   revenue <dbl>
```

```
korak2 <- korak1 |> mutate(or_pct = open_rate * 100)
korak2 |> select(campaign_id, open_rate, or_pct) |> head(3)
```

```
# A tibble: 3 x 3
  campaign_id open_rate or_pct
```

```
  <chr>           <dbl>  <dbl>
1 NL-002          0.270   27.0
2 NL-007          0.288   28.8
3 NL-009          0.283   28.3
```

```r
# OK, ovo radi. Idemo dalje...

korak3 <- korak2 |>
  summarise(
    n = n(),
    prosjek = round(mean(or_pct), 1)
  )
korak3
```

```
# A tibble: 1 x 2
      n prosjek
  <int>   <dbl>
1    12    28.9
```

Pohranjivanjem svakog koraka u zasebni objekt, možete točno identificirati na kojem koraku nastaje problem. Kad pronađete i ispravite grešku, spojite korake natrag u pipeline.

11.3 print() i glimpse() kao dijagnostika

Unutar funkcija i petlji, dodajte privremene `print()` naredbe da vidite što se događa.

```r
# Debugging s print naredbama
analiziraj_debug <- function(data, tip) {
  podaci <- data |> filter(campaign_type == tip)
  cat("Tip:", tip, "| Redova:", nrow(podaci), "\n")  # Debug ispis

  if (nrow(podaci) == 0) {
    cat("UPOZORENJE: nema podataka za tip", tip, "\n")
    return(NULL)
  }

  podaci |>
    summarise(
      tip = tip,
      n = n(),
      or_M = round(mean(open_rate), 3)
    )
}
```

```
# Testirajte s poznatim i nepoznatim tipom
analiziraj_debug(nl, "weekly_digest")
```

```
Tip: weekly_digest | Redova: 12

# A tibble: 1 x 3
  tip                n  or_M
  <chr>          <int> <dbl>
1 weekly_digest    12 0.289
```

```
analiziraj_debug(nl, "nepostojeci_tip")
```

```
Tip: nepostojeci_tip | Redova: 0
UPOZORENJE: nema podataka za tip nepostojeci_tip

NULL
```

Kad ste riješili problem, uklonite debug ispise. Ostavljanje privremenih `cat()` i `print()` naredbi u gotovom kodu je loša praksa jer zatrpava konzolu nepotrebnim ispisom.

11.4 Česte greške i rješenja

Pogledajmo najčešće greške koje ćete susresti i kako ih riješiti.

```
# 1. "could not find function" -> paket nije učitan
summarise(nl, n = n())
# Rješenje: library(tidyverse) na početku

# 2. "unexpected symbol" -> nedostaje zarez, operator ili zagrada
nl |>
  mutate(x = open_rate y = click_rate)  # Nedostaje zarez
# Rješenje: mutate(x = open_rate, y = click_rate)

# 3. "+ ggplot" umjesto "|> ggplot"
nl |>
  filter(open_rate > 0.2) +  # Krivo: + umjesto |>
  ggplot(aes(x = open_rate))
# Rješenje: koristiti |> do ggplot(), pa + za slojeve

# 4. "object of type 'closure' is not subsettable"
mean[1]  # mean je funkcija, ne vektor
# Rješenje: provjerite jeste li slučajno prepisali ime varijable imenom funkcije
```

> Svaki iskusni programer bio je početnik koji je satima tražio zarez koji nedostaje. Debugging nije znak neznanja, nego sastavni dio posla. Razlika između početnika i iskusnog korisnika nije u tome što iskusni ne griješe, nego u tome da imaju sustavan pristup traženju grešaka.

---

12 Quarto: integracija koda, teksta i rezultata

Do sada ste pisali R kod u skriptama (.R datoteke) koje proizvode tablice i grafove u konzoli. Quarto dokumenti (.qmd datoteke) omogućuju nešto moćnije—integraciju teksta, koda i rezultata u jedan dokument koji se renderira u HTML, PDF ili Word.

Zapravo, svako predavanje na ovom kolegiju je Quarto dokument. Tekst koji čitate, kod koji vidite i grafovi koji se prikazuju nastaju iz jedne .qmd datoteke.

12.1 Struktura Quarto dokumenta

```
# Quarto dokument ima tri dijela:

# 1. YAML zaglavlje (između --- oznaka)
# ---
# title: "Analiza newsletter kampanja"
# author: "Ime Prezime"
# date: today
# format: html
# ---

# 2. Tekst u Markdown formatu
# ## Uvod
# Ova analiza ispituje performanse naših newsletter kampanja...

# 3. R code chunkovi (između ``` oznaka)
# ```{r}
# library(tidyverse)
# nl <- read_csv("newsletter_campaigns.csv")
# ```
```

Kad pokrenete `quarto render`, Quarto izvršava R kod, hvata rezultate (tablice, grafove, ispis) i umeće ih u dokument zajedno s tekstom. Rezultat je profesionalan izvještaj u kojem su analiza i prezentacija neodvojivi.

12.2 Chunk opcije za kontrolu ispisa

Opcije unutar code chunkova kontroliraju što se prikazuje u dokumentu.

```
# echo: true   -> prikaži kod u dokumentu
# echo: false  -> sakrij kod, prikaži samo rezultat
# eval: true   -> izvrši kod
# eval: false  -> ne izvršavaj (samo prikaži kod)
# message: false -> sakrij poruke paketa
# warning: false -> sakrij upozorenja
# fig-width: 8  -> širina grafa u inčima
# fig-height: 5 -> visina grafa u inčima

# Za izvještaj klijentu: echo: false (ne želi vidjeti kod)
# Za kolegicu analitičarku: echo: true (želi vidjeti kako ste to napravili)
```

Ova fleksibilnost je ključna. Isti Quarto dokument možete renderirati s `echo: true` za interni tim (koji želi vidjeti kod) i s `echo: false` za klijenta (koji želi samo rezultate). Mijenjate jednu opciju u YAML zaglavlju i dobivate potpuno drugačiji dokument.

12.3 Inline R kod

Osim code chunkova, R vrijednosti možete umetnuti direktno u tekst.

```
n_kampanja <- nrow(nl)
prosjek_or <- round(mean(nl$open_rate) * 100, 1)
najbolji_tip <- nl |>
  group_by(campaign_type) |>
  summarise(or = mean(open_rate), .groups = "drop") |>
  slice_max(or) |>
  pull(campaign_type)
```

U Quarto dokumentu biste napisali tekst poput: “Analizirali smo 50 kampanja. Prosječni open rate je 25.5%. Najbolji rezultat ima tip weekly_digest.”

Kad se dokument renderira, R vrijednosti se automatski umeću u tekst. Ako se podaci promijene, tekst se automatski ažurira. Nikad više ne morate ručno ažurirati brojke u izvještaju.

12.4 Quarto vs R skripta: kad koristiti što

R skripta (.R) je pravi izbor kad je cilj izračun, transformacija ili generiranje outputa (tablice, grafovi, datoteke). Skripta je brza za izvršavanje i laka za debugging.

Quarto dokument (.qmd) je pravi izbor kad je cilj komunikacija rezultata. Izvještaj za klijenta, akademski rad, interna prezentacija, kolegijalni materijal. Quarto integrira narativ i rezultate u jedinstven dokument.

U praksi, mnogi analitičari koriste oboje. Skriptu koriste za teški posao (čišćenje, modeliranje), a Quarto za prezentaciju rezultata. Skripta generira čiste podatke i grafove, Quarto ih ugrađuje u priču.

---

13 Funkcionali za složenije radne tokove

Vratimo se purrr paketu i pogledajmo naprednije obrasce koji su korisni u praksi.

13.1 walk(): map() bez povratne vrijednosti

Ponekad želite izvršiti nešto za svaki element (na primjer, spremiti graf) ali ne trebate povratnu vrijednost. `walk()` je varijanta `map()` koja izvršava funkciju ali tiho odbacuje rezultat.

```
# Spremi zaseban graf za svaki tip kampanje
tipovi <- unique(nl$campaign_type)

walk(tipovi, \(tip) {
  p <- nl |>
    filter(campaign_type == tip) |>
    ggplot(aes(x = open_rate)) +
    geom_histogram(fill = "steelblue", color = "white", bins = 8) +
    labs(title = paste("Open rate:", tip)) +
    theme_minimal()

  ggsave(paste0("graf_", tip, ".png"), p, width = 7, height = 4)
})
```

`walk()` je idiomatski R način za petlje koje proizvode popratne efekte (side effects) poput spremanja datoteka, ispisa na konzolu ili slanja emailova. Za razliku od `map()`, ne zatrpava konzolu listom NULL vrijednosti.

13.2 map2(): paralelna iteracija preko dva vektora

```r
# Dva vektora: metrike i njihovi naslovi
metrike <- c("open_rate", "click_rate")
naslovi <- c("Open rate kampanja", "Click rate kampanja")

# map2 iterira paralelno: prvi element s prvim, drugi s drugim
rezultati <- map2(metrike, naslovi, \(metrika, naslov) {
  nl |>
    sazetak_metrike(metrika) |>
    mutate(metrika = naslov)
})

bind_rows(rezultati)
```

```
# A tibble: 2 x 6
      M   Med    SD   Min   Max metrika
  <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1 0.255 0.252 0.068 0.115 0.408 Open rate kampanja
2 0.037 0.037 0.019 0.005 0.086 Click rate kampanja
```

`map2()` prima dva vektora i iterira paralelno. Na prvoj iteraciji koristi `metrike[1]` i `naslovi[1]`, na drugoj `metrike[2]` i `naslovi[2]`, i tako dalje. Ovo je korisno kad imate parove ulaznih podataka.

13.3 imap(): iteracija s indeksom

```r
# imap daje i element i njegovo ime/indeks
nl |>
  group_by(campaign_type) |>
  group_split() |>
  set_names(unique(nl$campaign_type) |> sort()) |>
  imap(\(podaci, ime) {
    tibble(
      tip = ime,
      n = nrow(podaci),
      or_M = round(mean(podaci$open_rate), 3)
    )
  }) |>
  bind_rows()
```

```
# A tibble: 5 x 3
  tip                n  or_M
  <chr>          <int> <dbl>
```

```
1 breaking_news      7 0.198
2 event_promo        6 0.25
3 special_report    17 0.259
4 sponsored          8 0.248
5 weekly_digest     12 0.289
```

`imap()` je varijanta `map()` koja automatski proslijeđuje i element i njegovo ime (ili indeks). Korisna je kad trebate znati koji element trenutno obrađujete, na primjer za imenovanje rezultata ili za dijagnostiku.

13.4 possibly(): zaštita od grešaka

Kad primjenjujete funkciju na mnogo elemenata, jedna greška može srušiti cijeli pipeline. `possibly()` omotava funkciju u zaštitni sloj koji hvata greške i vraća default vrijednost umjesto da prekida izvršavanje.

```
# Funkcija koja ponekad pada
opasna_funkcija <- function(tip) {
  podaci <- nl |> filter(campaign_type == tip)
  if (nrow(podaci) < 3) stop("Premalo podataka!")
  mean(podaci$open_rate)
}

# Bez zaštite: jedna greška ruši sve
# map_dbl(c("weekly_digest", "nepostojeci"), opasna_funkcija)  # Error!

# S zaštitom: greška vraća NA, ostali rezultati ostaju
sigurna_funkcija <- possibly(opasna_funkcija, otherwise = NA_real_)

map_dbl(c("weekly_digest", "nepostojeci", "breaking_news"), sigurna_funkcija)
```

```
[1] 0.2886833        NA 0.1978286
```

`possibly(f, otherwise = NA)` kreira novu funkciju koja radi isto kao `f`, ali umjesto da baci grešku, vraća `otherwise` vrijednost. Ovo je neprocjenjivo kad učitavate 50 datoteka i jedna je korumpirana, ili kad analizirate 20 grupa i jedna ima nedovoljno podataka.

---

14 Kompletna analiza: automatizirani izvještaj o kampanjama

Spojimo sve iz ovog predavanja u jednu koherentnu analizu. Cilj je napisati kod koji bi mogao biti tijelo Quarto izvještaja o performansama newsletter kampanja.

```
library(patchwork)

# PARAMETRI
min_kampanja <- 3
decimale <- 3
fokus_metrike <- c("open_rate", "click_rate", "unsubscribe_rate")
```

```
# POMOĆNE FUNKCIJE
sazetak_tipa <- function(data, tip, dec = 3) {
  d <- data |> filter(campaign_type == tip)

  if (nrow(d) < min_kampanja) return(NULL)

  tibble(
    tip = tip,
    n = nrow(d),
    or_M = round(mean(d$open_rate), dec),
    or_SD = round(sd(d$open_rate), dec),
    ctr_M = round(mean(d$click_rate), dec + 1),
    unsub_M = round(mean(d$unsubscribe_rate), dec + 2),
    prosj_rijeci = round(mean(d$word_count), 0),
    udio_sa_slikom = round(mean(d$has_image), 2)
  )
}

graf_usporedba <- function(data, metrika, naslov, boja = "steelblue") {
  data |>
    ggplot(aes(x = fct_reorder(campaign_type, .data[[metrika]]),
               y = .data[[metrika]])) +
    geom_boxplot(fill = boja, alpha = 0.6) +
    coord_flip() +
    labs(title = naslov, x = NULL, y = metrika) +
    theme_minimal()
}
```

```
# ANALIZA
tipovi <- unique(nl$campaign_type)

# Sažetak za sve tipove (s automatskim preskakanjem malih grupa)
tablica_sazetka <- map(tipovi, \(t) sazetak_tipa(nl, t, decimale)) |>
```

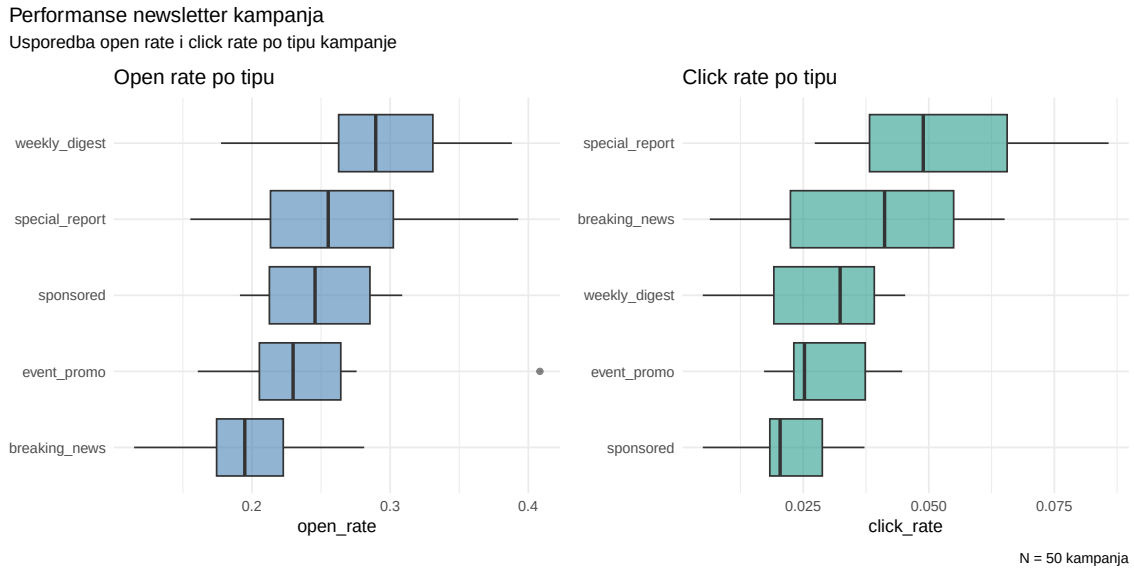
```
  bind_rows() |>
  arrange(desc(or_M))

tablica_sazetka
```

```
# A tibble: 5 x 8
  tip                n  or_M or_SD  ctr_M unsub_M prosj_rijeci udio_sa_slikom
  <chr>          <int> <dbl> <dbl>  <dbl>   <dbl>        <dbl>          <dbl>
1 weekly_digest     12 0.289 0.066 0.029  0.00331          380           0.67
2 special_report    17 0.259 0.067 0.0519 0.00279          486           0.76
3 event_promo        6 0.25  0.086 0.0294 0.00207          210           1
4 sponsored          8 0.248 0.043 0.0223 0.00645          204           0.75
5 breaking_news      7 0.198 0.055 0.0382 0.00405          140           0.71
```

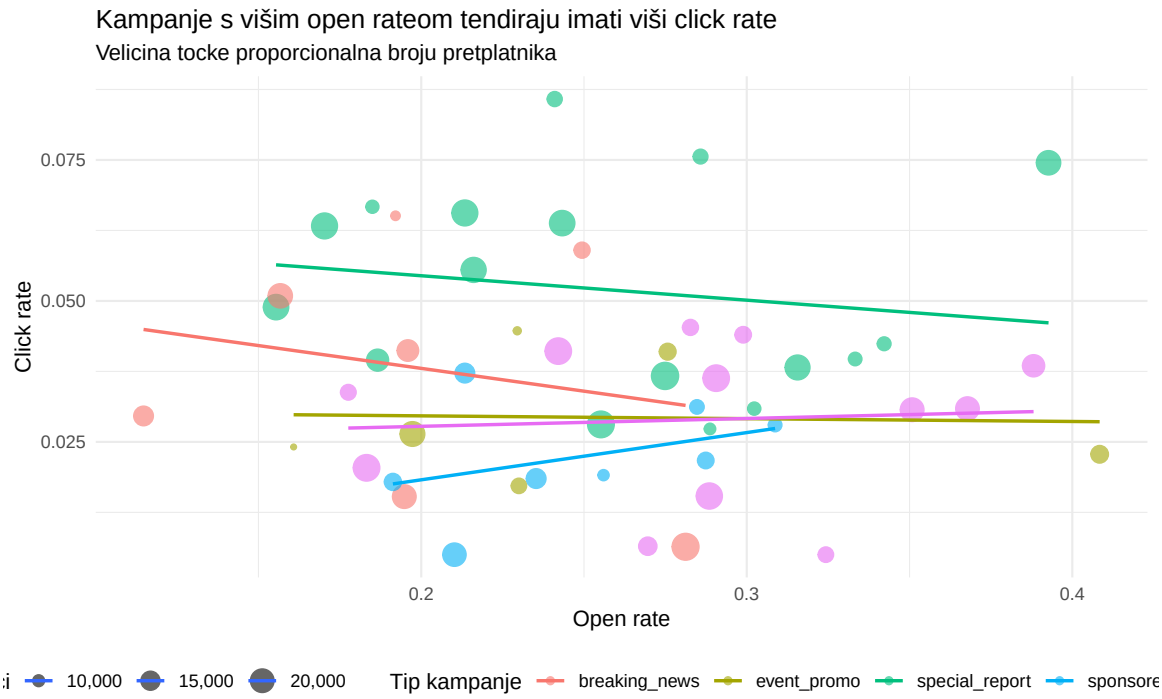
```
# Analiza po stilu naslova (unutar svake kampanje)
nl |>
  group_by(campaign_type, subject_style) |>
  summarise(
    n = n(),
    or_M = round(mean(open_rate), 3),
    .groups = "drop"
  ) |>
  filter(n >= 2) |>
  pivot_wider(
    names_from = subject_style,
    values_from = or_M
  )
```

```
# A tibble: 12 x 7
   campaign_type      n informativni personalizirani upitni brojke  hitno
   <chr>          <int>        <dbl>           <dbl>  <dbl>  <dbl>  <dbl>
 1 breaking_news      5        0.182          NA     NA     NA     NA
 2 event_promo        2       NA               0.285 NA     NA     NA
 3 event_promo        3       NA              NA      0.245 NA     NA
 4 special_report     2       NA               0.242 NA      0.237 NA
 5 special_report     4       NA              NA     NA     NA      0.329
 6 special_report     6        0.2            NA     NA     NA     NA
 7 special_report     3       NA              NA      0.31  NA     NA
 8 sponsored          3       NA              NA     NA      0.252 NA
 9 sponsored          2       NA              NA     NA     NA      0.224
10 weekly_digest      2        0.21           NA     NA      0.254 NA
11 weekly_digest      3       NA              NA     NA     NA      0.312
12 weekly_digest      4       NA               0.328 NA     NA     NA
```

```r
# VIZUALIZACIJA
g1 <- graf_usporedba(nl, "open_rate", "Open rate po tipu")
g2 <- graf_usporedba(nl, "click_rate", "Click rate po tipu", boja = "#2a9d8f")

g1 + g2 +
  plot_annotation(
    title = "Performanse newsletter kampanja",
    subtitle = "Usporedba open rate i click rate po tipu kampanje",
    caption = paste("N =", nrow(nl), "kampanja")
  )
```




```r
# Odnos open rate i click rate
nl |>
  ggplot(aes(x = open_rate, y = click_rate, color = campaign_type, size = subscribers)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, linewidth = 0.8) +
  scale_size_continuous(labels = scales::label_comma()) +
  labs(
    title = "Kampanje s višim open rateom tendiraju imati viši click rate",
    subtitle = "Veličina točke proporcionalna broju pretplatnika",
    x = "Open rate",
    y = "Click rate",
    color = "Tip kampanje",
    size = "Pretplatnici"
  ) +
  theme_minimal() +
  theme(legend.position = "bottom")
```




```
# Kada slati newsletter?
nl |>
  mutate(
    dio_dana = case_when(
      send_hour < 10 ~ "jutro (6-9)",
      send_hour < 14 ~ "prijepodne (10-13)",
      send_hour < 18 ~ "poslijepodne (14-17)",
      .default = "navečer (18+)"
    )
  ) |>
  group_by(dio_dana) |>
  summarise(
    n = n(),
    or_M = round(mean(open_rate), 3),
    ctr_M = round(mean(click_rate), 4),
    .groups = "drop"
  ) |>
  arrange(desc(or_M))
```

```
# A tibble: 4 x 4
  dio_dana                n  or_M  ctr_M
  <chr>               <int> <dbl>  <dbl>
1 navečer (18+)           9 0.277 0.0388
2 poslijepodne (14-17)    15 0.26  0.04
```

```
3 prijepodne (10-13)      10 0.245 0.0307
4 jutro (6-9)             16 0.243 0.0373
```

```r
# KLJUČNI NALAZ: koji faktori predviđaju open rate?
nl |>
  summarise(
    kor_rijeci_or = round(cor(word_count, open_rate), 3),
    kor_linkovi_ctr = round(cor(n_links, click_rate), 3),
    kor_pretplatnici_or = round(cor(subscribers, open_rate), 3)
  )
```

```
# A tibble: 1 x 3
  kor_rijeci_or kor_linkovi_ctr kor_pretplatnici_or
          <dbl>           <dbl>               <dbl>
1         0.127          -0.239              -0.075
```

Analiza otkriva nekoliko nalaza. Breaking news i kampanje s hitnim stilom naslova imaju najviši open rate, ali uz veću varijabilnost. Click rate ne prati savršeno open rate, što sugerira da su čimbenici koji navode ljude da otvore email (naslov, hitnost) različiti od onih koji ih navode da kliknu na sadržaj (relevantnost, format). Korelacija između broja riječi i open ratea govori o preferiranom formatu, dok veličina baze pretplatnika sama po sebi ne predviđa bolji angažman.

Cijela ova analiza, od učitavanja podataka do gotovih grafova i nalaza, koristi principe koje smo naučili ovaj tjedan. Parametri su na jednom mjestu. Pomoćne funkcije enkapsuliraju ponovljivu logiku. map() automatizira iteraciju. Vizualizacija prati principe iz prošlog tjedna. I sve je napisano tako da se može pokrenuti ponovno s novim podacima bez ikakvih promjena u kodu (osim, eventualno, putanje do datoteke).

> Kad pišete analizu, zamislite da ju netko pokreće prvi put, bez ikakvog konteksta. Može li taj netko razumjeti što kod radi, zašto, i kako interpretirati rezultate? Ako da, napisali ste dobru analizu.

! Ključni zaključci

1. Funkcije su alat za izbjegavanje ponavljanja koda. Prema Pravilu tri, ako ste kopirali isti kod tri puta, pretvorite ga u funkciju. Default vrijednosti argumenata čine funkcije fleksibilnima.

2. Klasični `if/else` radi s jednom vrijednošću (za skripte i funkcije). `if_else()` i `case_when()` su vektorizirani (za `mutate()`). Ne miješajte ih.

3. Validacija ulaza u funkcijama sprečava tihe greške. Koristite `warning()` za

upozorenja i `return()` za rano izlaženje.

4. `for` petlje ponavljaju kod za svaki element. Unutar petlje, grafove morate ispisati s `print()`. Za većinu zadataka postoje elegantnije alternative.
5. `map()` iz paketa purrr je moderna alternativa petljama. `map_dbl()`, `map_chr()` i `map_lgl()` vraćaju specifične tipove. `walk()` je za popratne efekte (spremanje datoteka).
6. `map2()` iterira paralelno preko dva vektora. `imap()` daje i element i njegovo ime. `possibly()` štiti od grešaka unutar iteracije.
7. Obrazac `list.files() |> map(read_csv) |> bind_rows()` učitava i spaja više datoteka u jednom koraku.
8. Debugging zahtijeva sustavan pristup, uključujući čitanje poruka, izolaciju problema (korak po korak) i privremene `cat()`/`print()` ispise.
9. DRY princip se primjenjuje na nekoliko načina. Parametri trebaju biti na jednom mjestu, logika u funkcijama, a struktura skripte u jasnim sekcijama.
10. Quarto dokumenti integriraju tekst, kod i rezultate. Koristite ih za izvještaje, radove i prezentacije. R skripte su za teški izračun, Quarto za komunikaciju.
11. Chunk opcije, kao što su `echo`, `eval`, `message`, `warning` i `fig-width`, kontroliraju što se prikazuje u renderiranom dokumentu. Na primjer, `echo: false` sakriva kod za klijente.
12. Cilj ponovljive analize jest taj da netko može pokrenuti vaš kod od početka do kraja s novim podacima i dobiti ažurirane rezultate bez ručnih promjena.

⚠ Priprema za sljedeći tjedan

Sljedeći tjedan ulazimo u **uvod u vjerojatnost**, gdje ćemo naučiti što je vjerojatnost, kako ju računamo, te kakvi su binomna i normalna distribucija. Ovo je konceptualni temelj za sve statističke testove koje ćemo raditi u drugom dijelu kolegija.
Za pripremu:

1. Napišite vlastitu funkciju koja prima tibble i ime kategoričke varijable te vraća tibble s brojem i udjelom (%) svake kategorije. Testirajte je na datasetu `newsletter_campaigns.csv`.
2. Koristeći `map()`, generirajte sažetak open ratea za svaki dan u tjednu (stupac `day_sent`). Spojite rezultate u jedan tibble.
3. Napišite kratki Quarto dokument (.qmd) koji učitava podatke, prikazuje jedan graf i jednu tablicu, s popratnim tekstom. Renderirajte ga u HTML.

4. Pročitajte poglavlje 9 iz Navarro (Learning Statistics with R) o vjerojatnosti. Fokusirajte se na intuiciju, ne na formule.

---

15 Dodatno čitanje

Obavezno

Wickham, H. & Grolemund, G. (2023). *R for Data Science* (2nd edition), Chapters 26, 27 i 29. Besplatno dostupno na r4ds.hadley.nz. Poglavlje 26 pokriva funkcije, poglavlje 27 iteraciju s purrr, poglavlje 29 Quarto dokumente.

Navarro, D. (2018). *Learning Statistics with R*, Chapter 8. Besplatno dostupno na learningstatisticswithr.com. Osnove programiranja u R-u.

Preporučeno

Wickham, H. (2019). *Advanced R* (2nd edition), Chapters 6 i 9. Besplatno dostupno na adv-r.hadley.nz. Poglavlje 6 detaljno pokriva funkcije, poglavlje 9 funkcionalno programiranje (map i prijatelji).

Quarto dokumentacija. Besplatno dostupno na quarto.org. Kompletna dokumentacija za Quarto sustav sa tutorijalima za HTML, PDF i Word dokumente.

Bryan, J. & Hester, J. *What They Forgot to Teach You About R*. Besplatno dostupno na rstats.wtf. Praktični savjeti o organizaciji projekata, debugging-u i radnim tokovima koji se ne uče u udžbenicima statistike.

---

16 Pojmovnik

Pojam	Objašnjenje
Funkcija	Objekt koji prima argumente, izvršava operacije i vraća rezultat. Definira se s `function()`.
Argument	Ulazni podatak funkcije. Navodi se unutar zagrada pri definiciji i pozivu.
Default vrijednost	Podrazumijevana vrijednost argumenta. Definira se s `=` u listi argumenata.
Povratna vrijednost	Rezultat funkcije. Zadnji izraz u tijelu, ili eksplicitno s `return()`.

Pojam	Objašnjenje
`return()`	Eksplicitno vraća vrijednost i izlazi iz funkcije. Korisno za ranu validaciju.
`if/else`	Uvjetna naredba za kontrolu toka. Radi s jednom vrijednošću (nije vektorizirana).
`if_else()`	Vektorizirana uvjetna funkcija za `mutate()`. Radi na cijelom stupcu.
`case_when()`	Vektorizirana funkcija za složeno rekodiranje s više uvjeta.
`for` petlja	Ponavlja blok koda za svaki element u skupu. Sintaksa: `for (x in skup) { ... }`.
`map()`	purrr funkcija koja primjenjuje funkciju na svaki element i vraća listu.
`map_dbl()`	Varijanta `map()` koja vraća numerički vektor.
`map_chr()`	Varijanta `map()` koja vraća tekstualni vektor.
`map_lgl()`	Varijanta `map()` koja vraća logički vektor.
`map2()`	purrr funkcija za paralelnu iteraciju preko dva vektora.
`imap()`	purrr funkcija koja proslijeđuje i element i njegovo ime/indeks.
`walk()`	Varijanta `map()` za popratne efekte (spremanje datoteka). Ne vraća rezultat.
`possibly()`	purrr funkcija koja omotava funkciju u zaštitni sloj. Greška vraća default vrijednost umjesto prekida.
purrr	Paket iz tidyverse za funkcijsko programiranje.
`nest()`	tidyr funkcija koja pakira podatke grupe u ugniježđeni tibble.
`bind_rows()`	Vertikalno spaja listu tibbleova u jedan.
`list.files()`	Base R funkcija za pronalaženje datoteka u direktoriju po uzorku.
`set_names()`	Dodjeljuje imena elementima vektora ili liste.
DRY	Don't Repeat Yourself. Princip da informacija postoji na jednom mjestu u kodu.
Lambda funkcija	Anonimna funkcija. Piše se kao `\(x) x + 1` ili `function(x) x + 1`.
`.data[[var]]`	Pristup stupcu po imenu pohranjenom u varijabli. Za tidyverse funkcije.
`cat()`	Ispis teksta u konzolu. Bez navodnih oznaka i indeksa.

Pojam	Objašnjenje
`warning()`	Ispis upozorenja. Ne zaustavlja program.
`stop()`	Ispis greške i zaustavljanje programa. Za kritične probleme.
Validacija ulaza	Provjera ispravnosti argumenata prije izvršavanja. Sprečava tihe greške.
Skripta (.R)	R datoteka s nizom naredbi. Za izračune i transformacije.
Quarto dokument (.qmd)	Datoteka koja integrira tekst, kod i rezultate. Za izvještaje i komunikaciju.
Chunk opcije	Postavke R code chunka u Quarto dokumentu (echo, eval, message, warning, fig-width).
Inline R kod	R izraz umetnut u tekst Quarto dokumenta. Automatski se evaluira pri renderiranju.
Debugging	Proces pronalaženja i ispravljanja grešaka u kodu.
Side effect	Popratni efekt funkcije (ispis, spremanje datoteke) koji nije povratna vrijednost.
Ponovljiva analiza	Analiza napisana tako da se može pokrenuti od početka do kraja s novim podacima bez ručnih promjena.