# Napredni grafički paketi u R-u

**Sedlar, Antonija**

**Master's thesis / Diplomski rad**

**2021**

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

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:126:078478

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku

**Antonija Sedlar**

# Napredni grafički paketi u R-u

Diplomski rad

Osijek, 2021.

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku

**Antonija Sedlar**

# Napredni grafički paketi u R-u

Diplomski rad

Voditelj: doc. dr. sc. Danijel Grahovac

Osijek, 2021.

## Sažetak:

Ovaj rad se bavi grafičkim prikazivanjem podataka u programskom jeziku R-u korištenjem naprednih paketa vizualizacije s naglaskom na paket ggplot2. U radu će se obraditi gramatika grafova te generiranje osnovnih tipova grafova. Također, proći će se kroz sve komponente grafa. Dotaknut će se zanimljivih i korisnih funkcija i njihovih atributa za maninipuliranje svakog segmenta grafa.

**Ključne riječi:**
Napredni grafički paketi u R-u, ggplot2, gramatika grafova, generiranje grafova

## Abstract:

This paper deals with the graphical display of data in the R programming language using advanced visualization packages with an emphasis on the ggplot2 package. The paper also deals with grammar graphs and generate basic types of graphs. Also, we will go through all the components of the graph. We will touch on interesting and useful functions and their attributes for manipulating each segment of the graph.

**Key words:** Advanced graphics packages in R, ggplot2, grammar of graphics, graph generation

# Sadržaj

# Uvod

R je modernija implementacija programskog jezika S koji je nastao 1976. godine, a njegov autor je John Chambers. Osnovni cilj programskog jezika S je, kako sam njegov autor navodi, pretvaranje ideja u software, brzo i vjerodostojno. 1988. u opticaj kreće komercijalna verzija S pod nazivom S-PLUS. 1991. dvojica znanstvenika sa Sveučilišta u Aucklandu (Novi Zeland), statističar Ross Ihaka i statističar i bioinformatičar Robert Gentleman započinju alternativnu implementaciju osnovnog S jezika. Ova implementacija je bila potpuno neovisna o S-PLUS-u. Ihaka i Gentleman s publiciranjem svog rada počinju 1993. godine. 1995. godine se autori R-a odlučuju za njegovo razvijanje u smjeru besplatnog i open source sofwarea. 1997. oformljen je stručni tim za daljnje razvijanje R-a. Prvo službeno izdanje pušteno je 1995. Sveobuhvatna mreža arhiva R-a (The Comprehensive R Archive Network - CRAN) je najavljena 23. travnja 1997. s 3 mirrora i 12 paketa. Prva službena "stabilna beta" verzija (v1.0) objavljena je 29. veljače 2000.

Zanimljivo je da se velik dio koda napisanog za programski jezik S i danas, nepromijenjen, koristi u R-u.

R se navodi kao jezik i okruženje za izvođenje statističkih izračuna, ali i grafičko prikazivanje podataka. Nudi širok spektar statističkih tehnika kao što su linearno i nelinearno modeliranje, klasična statistička ispitivanja, analiza vremenskih nizova, klasifikacija, grupiranje i grafičke tehnike. Jedna od njegovih velikih prednosti je upravo mogućnost generiranja kvalitetnih grafičkih prikaza koje su karakteristične po tome što su pogodne i za stručne publikacije a dobivene su vrlo jednostavno. Velika pažnja je posvećena postavljanju zadanih vrijednosti za "nevažnija" svojstva dizajna grafova, ali sve postavke grafa, odnosno dizajna, je moguće promijeniti. Na taj način se krajnji korisnik ne mora baviti detaljima, ali ima tu mogućnost zbog čega je korisniku ostavljena potpuna autonomija pri generiranju grafova.

Primjetimo da smo R definirali ne samo kao jezik, već i okruženje. Upravo je na "okruženje" stavljen naglasak jer je to potpuno planiran i koherentan sustav. Proširiv je raznim paketima od kojih 8 distribuira R, a mnogo više ih je dostupno na internet stranicama koje obuhvaća CRAN. Na taj način je pokriven vrlo širok spektar moderne statistike i statističkih metoda. Više o samom R-u čitatelj može saznati u [9].

Ovaj rad će biti fokusiran na grafičke pakete, najvećim dijelom na paket ggplot2, ali će se dotaknuti i interaktivnih grafova.

Važno je ne zaboraviti da R omogućuje brojne načine vizualizacije podataka, ali nam ne sugerira koju metodu, odnosno, koji graf crtati s obzirom na podatke i ono što iz njih želimo iščitati. Korisnik mora znati što želi dobiti vizualizacijom i ovakvom analizom, a ovi napredni paketi su moćan alat koji će pomoći pri ostvarivanju cilja.

# 1. O paketu ggplot2

ggplot2 je R paket za naprednu vizualizaciju podataka odnosno stvaranje naprednih grafičkih prikaza. Već smo spomenili da se R pobrinuo za detalje grafa na način da su neke značajke grafa automatski zadane, ali, ukoliko to želi, korisnik može mijenjati gotovo sve na grafu. Ovakav način vizualizacije zapravo omogućava paket ggplot2. Pretvaranje podataka u vizualne prikaze se postiže takozvanom gramatikom grafike. To je princip koji omogućava sastavljanje grafikona kombiniranjem neovisnih komponenti, a sastoji se od skupa temeljnih načela. Graf se gradi iterativno, u slojevima.
Redosljed slojeva je:

- sloj koji prikazuje neobrađene podatke
- sloj napomena
- sloj statističkih sažetaka.

Kako bismo na ovaj način mogli stvarati grafove, s potpunom kontrolom i razumijevanjem, potrebno je usvojiti grafičku gramatiku. Na taj način, ne samo da se razvija sposobnost stvaranja svih vrsta grafova, već se otvara i mogućnost gradnje novih komponenata te proširivanje grafova na taj način.

Između ostalog, ggplot2 je izuzetno koristan paket za svladati ga jer detaljnu vizualizaciju možemo promatrati kao dodatni alat za dublju analizu, a samim tim i za dublje razumijevanje podataka. Podaci na papiru, u numeričkom formatu, često mogu "sakriti" ili "zamutiti" karakteristike koje postaju jasne tek njihovom vizualizacijom.

# 2. Gramatika grafova

Kao što je već napomenuto, kako bismo savladali korištenje ggplot2 paketa te bili sposobni koristiti njegov puni potencijal, potrebno je razumjeti i savladati gramatiku grafova.

Gramatikom grafova se opisuju sve temeljne značajke koje su osnova svih statističkih grafičkih prikaza. Gramatika se može smatrati teoretskom osnovom ovog paketa. Ovdje ćemo proći sve komponete gramatike te naučiti kako ih međusobno povezivati i ispreplitati.

Također, gramatika omogućava iterativnu izgradnju grafa što znači da graf možemo mijenjati u koracima, svojstvo po svojstvo.

Grafovi se sastoje od podataka koje želimo vizualizirati i, onog na čemu je naš fokus ovdje, takozvanog mapiranja. Mapiranje u ovom kontekstu tumačimo kao upute, odnosno opise, koje dajemo R-u, o tome kako želimo da se podaci prikazuju.

Razlikujemo pet komponenti mapiranja:

- sloj
- skaliranje
- koordinatni sustav
- facetiranje
- tema

**Sloj** je skup geometrijskih elemenata i statističkih transformacija. Geometrijske elemente grafa čini sve ono što vidimo na samom crtežu grafa. To uključuje elemente kao što su točke, crte, poligoni i slično. Statističke transformacije su transformacije kojima radimo sažetak promatranih podataka, odnosno dajemo pregled na podacima. To uključuje, primjerice, prebrojavanje opservacije u svrhu kreiranja histograma, fitanje linearnog modela i slične metode.

**Skaliranje** možemo promatrati kao funkciju koja vrijednosti iz domene koju čine podaci preslikava u kodomenu koju čine estetske komponente. Pod estetske komponente smatramo boju, oblik i veličinu. Ova komponenta mapiranja crta legende i koordinatne osi što omogućava iščitavanje vrijednosti izvornih podataka iz grafa. O ovakvom iščitavanju podataka možemo razmišljati kao o inverznoj funkciji mapiranja. Naime, kada mapiramo podatke, mi njih "prevodimo" u vizuale. No, kada očitavamo vrijednosti s grafa, mi uzimamo dijelove grafičkih prikaza i pretvaramo ih u kvantificirane, numeričke podatke.

**Koordinatni sustav** opisuje kako mapiranje podatke preslikava u ravninu našeg grafa. Također, ova komponenta kreira osi i rešetke koje omogućavaju lakše čitanje grafa. Uobičajeno je korištenje Kartezijevog koordinatnog sustava, ali ovaj paket omogućava korištenje i niza drugih koordinatnih sustava, uključujući nelinearne koordinatne sustave, polarne koordinate i projekcijske karte.

**Facet** omogućuje razlaganje podataka na skupove po ključu kojeg sami izabiremo te određuje kako će ti podaci biti prikazani. Ovo se još naziva i kondicioniranje.

Posljednja komponenta mapiranja, **tema**, služi za profinjene grafa u smislu vizualizacije. Bavi se detaljima prikaza kao što su veličina fonta, boja pozadine i slično. Ove postavke grafa su unaprijed postavljene, ali, kao što smo već napomenuli, korisnik zadržava potpunu autonomiju, te može promijeniti bilo koju od ovih i sličnih stavki grafa.

## 2.1. Komponente gramatike

Kako bismo razumjeli izgradnju grafa, proći ćemo kroz komponente grafike. Prva stvar koju odlučujemo kada idemo u generiranje grafa jest tip grafa koji želimo skicirati. To može biti linijski, točkasti, stupčasti itd. Tip grafa određujemo s funkcijama iz familije geom, a kratki pregled funkcija i tipova grafova koje generiraju možemo vidjeti u tablici 1

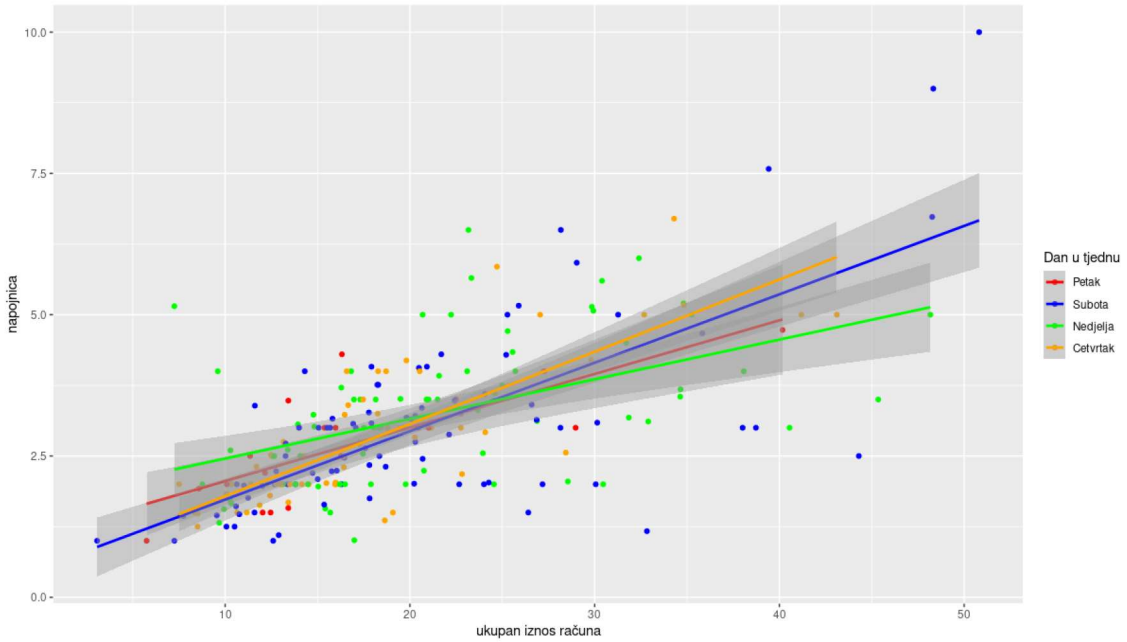
| Vrsta grafa | Funkcija |
|---|---|
| dijagram raspršenosti | geom_point() |
| bubblechart | geom_point() |
| stupčasti dijagram | geom_bar() |
| kutijasti dijagram | geom_boxplot() |
| linijski dijagram | geom_line() |

Tablica 1: Tipovi grafova i pridružene funkcije

Moguće je kombinirati više ovakvih funkcija pri čemu će nastajati kompleksniji i zanimljivi grafovi. Važno je pripaziti da kombinacije imaju smisla. Gramatika paketa ggplot2 dopušta najrazličitije kombinacije, međutim, to ne znači da, samo zato što su naredbe prošle, takvi grafovi imaju smisla.

U nastavku pogledajmo zanimljivu kombinaciju funkcija geom.

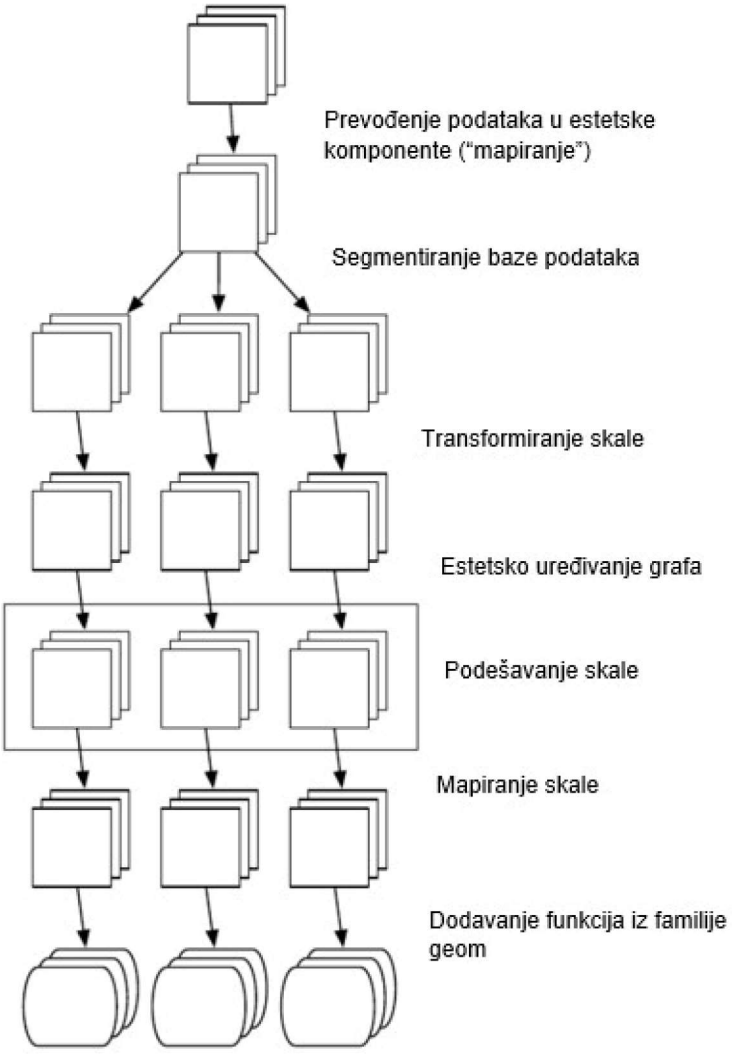
```
ggplot(tips, aes(total_bill, tip, colour = factor(day))) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "ukupan iznos racuna", y = "napojnica", colour ="Dan u tjednu") +
  scale_color_manual(labels = c("Petak", "Subota", "Nedjelja", "Cetvrtak"),
  values = c("red", "blue", "green", "orange"))
```




Slika 1: Kombiniranje više funkcija iz familije geom

## 2.2. Izgradnja grafa

Ovdje ćemo promotriti općenitu strukturu koda za generiranje grafa pomoću paketa ggplot2. Također, razjasnit ćemo slojeve i redoslijed kojim ih slažemo te napraviti pregled svakog od njih. Proučimo sliku 2

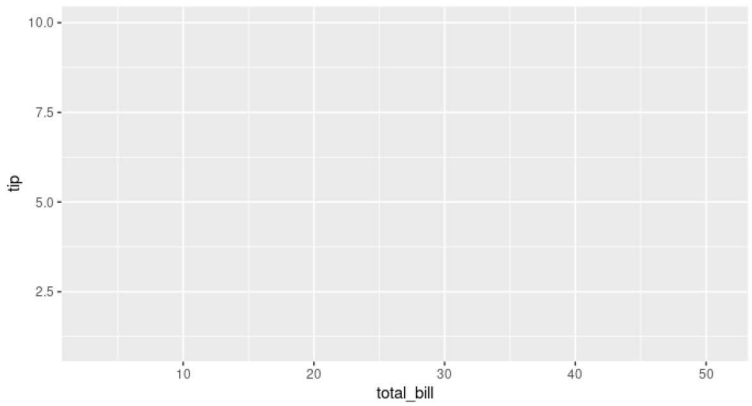



Slika 2: Shematski opis procesa generiranja grafa [10]

Svaka razina na ovoj slici predstavlja jedan sloj grafa, a shema se odnosi na generiranje grafa s tri podsegmenta. U svakom koraku se transformira pojedini data frame, neovisno jedan o drugom. Izuzetak je korak "podešavanje skale" koji istovremeno djeluje na sve segmente baze podataka.

Prvi korak u izgradnji grafa, kako je vidljivo na slici 2 je prevođenje podataka u estetske komponente, tzv. mapiranje. Pri tome navodimo bazu podataka koju analiziramo, a navedeni proces postižemo funkcijom ggplot. Primjer slijedi u nastavku

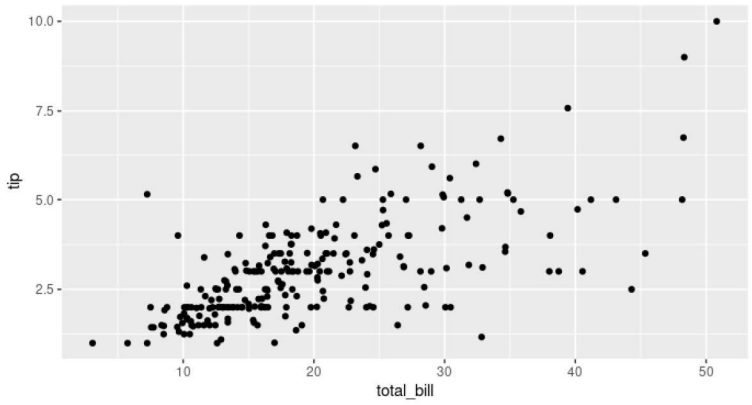
```
graf <- ggplot(tips, aes(total_bill, tip))
graf
```




Slika 3: Prvi korak u izgradnji grafa

Sada dodajemo novi sloj kako bismo dodali elemente na naš graf. To ćemo napraviti funkcijom iz familije geom - geom_point():

```
graf + geom_point()
```




Slika 4: Dodavanje novog sloja

Korištenje funkcije geom_point() je zapravo prečac za korištenje funkcije layer() koja generira novi sloj grafa. S ovom funkcijom je moguće definirati pet komponenti grafa:

- mapping - skup estetskih preslikavanja, specificiranih pomoću funkcije aes() i kombiniranih sa zadanim postavkama grafa, Ukoliko proslijedimo vrijednost "NULL" ovom parametru, koristit će se zadane postavke postavljene s funkcijom ggplot()
- data - skup podataka koji, ukoliko ne postavimo na "NULL", poništava zadani skup podataka grafa.
- geom - naziv geometrijskog objekta koji se koristi za prikazivanje opservacija.
- stat - ime statističke transformacije koju želimo koristiti; treba postaviti samo jedno: geom ili stat.
- position - metoda koja se koristi za podešavanje objekata koji se preklapaju, poput titranja, slaganja ili izbjegavanja

Dakle, gore navedeni kod za generiranje grafa je ekvivalentan sljedećem kodu.

```
graf + layer(
  mapping = NULL,
  data = NULL,
  geom = "point",
  stat = "identity",
  position = "identity"
)
```

Napominjemo da se ovakvo definiranje sloja, pomoću funkcije layer() vrlo rijetko koristi jer za ovakvu detaljnost najčešće nema potrebe.

Ovime smo naveli osnovne komponente svakog grafa: baza podataka iz koje skiciramo podatke, pridruživanje podataka koordinatnim osima i funkcija kojom određujemo tip grafa (kutijasti dijagram, dijagram raspršenosti...). Ovo su funkcije bez kojih nema grafa. Sve ostale funkcije i atributi su opcionalni.

### 2.2.1. Podaci

Svaki sloj zahtjeva podatke koji su povezani s njim. Podaci koje grafički prikazujemo s ggplot2 paketom moraju biti u formatu tidy date frame-a. Pod ovakvim objektom podrazumjevamo bazu podataka u kojoj jedan stupac predstavlja jednu varijablu, a jedan redak jednu opservaciju. Ovo su jaki zahtjevi, ali su logični i opravdani iz više razloga: podaci su izrazito važni pa je najbolje biti eksplicitan u vezi njih, osobito zbog toga što je moguće kombinirati više baza podataka. Naime, na istom grafu svaki sloj može koristiti drugačiju bazu podataka. Također, puno je lakše spremiti i upravljati data frameom nego s mnoštvom vektora.

### 2.2.2. Prevođenje podataka u estetske komponente

Prevođenje podataka u estetske komponente, takozvano mapiranje, provodimo funkcijom aes(). U nastavku slijedi primjer.

```
aes(x = total_bill, y = tip, colour = day)
```

Dakle, ovdje iznos računa preslikavamo na x-os, napojnice y-os, a dane u boje. S funkcijom aes() se mogu raditi i transformacije slučajnih varijabli, npr. mogli smo koristiti log(total_bill) i log(tip), ali preporuka je te transformacije odrađivati funkcijom dplyr::mutate(). Napominjemo da se u funkciji aes() trebaju koristiti samo imena varijabli, bez znaka $ ispred njih.

### 2.2.3. Familija funkcija geom

Familija funkcija geom podrazumjeva funkcije koje generiraju geometrijske elemente grafa. U ovom potpoglavlju ćemo napraviti pregled ovih funkcija. Popis funkcija i njihovo djelovanje vidljivo je u tablici 3

| **Primitivni geomi** | |
|---|---|
| geom_blank() | Ne prikazuje ništa, a najkorisniji je za prilagođavanje raspona koordinatnih osi |
| geom_point() | Točke |
| geom_path() | Put |
| geom_ribbon() | Vrpca - put s vertikalnom debljinom |
| geom_segment() | Linijski segment definiran s početnom i krajnjom točkom |
| geom_rect() | Pravokutnici |
| geom_polygon() | Poligoni |
| geom_text() | Tekst |
| **Jedna varijabla** | |
| Diskretne varijable | |
| geom_bar() | Prikazuje distribuciju diskretne varijable |
| Neprekidne varijable | |
| geom_histogram() | Dijeli varijablu na segmente te tako prebrojava opservacije i prikazuje ih u obliku stupaca |
| geom_density() | Prikazuje procjenjenu zaglađenu gustoću |
| geom_dotplot() | Slaže individualne točke grafa u točkasti dijagram |
| geom_freqpoly() | Dijeli varijablu na segmente te tako prebrojava opservacije i prikazuje ih pomoću linija |
| **Dvije varijable** | |
| Obje neprekidne | |
| geom_point() | Točke u koordinatnom sustavu |
| geom_quantile() | Generira zaglađenu kvantilsku regresiju |
| geom_rug() | Marginalni rug graf |
| geom_smooth() | Zaglađena krivulja koja najbolje odgovara podacima |
| geom_text() | Tekstualne oznake |
| Prikaz distribucije | |
| geom_bin2d() | Segmentiranje varijabli u pravokutnike i prebrojavanje |
| geom_density2d() | Procjenjena 2d zaglađena gustoća |
| geom_hex() | Segmentiranje varijabli u heksagone i prebrojavanje |
| Barem jedna diskretna varijabla | |
| geom_count() | Prebrojavanje broja točaka na različitim lokacijama |
| geom_jitter() | Nasumično treperenje točaka preklapanja |
| Jedna diskretna i jedna neprekidna varijabla | |
| geom_bar(stat = "identity") | Stupčasti dijagram sa stupcima do predefinirane točke |
| geom_boxplot() | Kutijasti dijagram |
| geom_violin() | Violinski dijagram |

Tablica 2

| **Jedna vremenska, jedna neprekidna varijabla** | |
|---|---|
| geom_area() | Područje |
| geom_line() | Linije |
| geom_step() | Stepenasti dijagram |
| Prikazivanje nepravilnosti | |
| geom_crossbar() | Vertikalni stupci sa centrom |
| geom_errorbar() | Prikaz standardne devijacije |
| geom_linerange() | Vertikalne linije |
| geom_pointrange() | Vertikalne linije sa centrom |
| **Prostorno** | |
| geom_map() | Brza verzija funkcije geom_polygon() za map funkcije |
| **Tri varijable** | |
| geom_contour() | Konture |
| geom_tile() | Dodavanje pravokutnika preko cijele ravnine |
| geom_raster() | Brza verzija funkcije geom_tile() pri čemu su svi pravokutnici iste veličine |

Tablica 3: Tipovi grafova i pridružene funkcije

### 2.2.4. Familija funkcija stats

Ovo je familija funkcija koje omogućavaju statističke transformacije. S nekim od njih smo se sreli jer se koriste u pozadini funkcija iz familije geom. Rijetko ih koristimo kao takve, ali je korisno znati kojim geom funkcijama korenspodiraju jer njihova dokumentacija sadrži više detalja o transformacijama koje one čine. U sljedećoj tablici je pregled funkcija stats koje odgovaraju određenim funkcijama iz familije geom.

| **Funkcija iz familije stats** | **Funkcija iz familije geom** |
|---|---|
| stat_bin() | gein_bar(), geom_freqpoly(), geom_histogram() |
| stat_bin2d() | geom_bin2d() |
| stat_bindot() | geom_dotplot() |
| stat_binhex() | geom_hex() |
| stat_boxplot() | geom_boxplot() |
| stat_contour() | geom_contour() |
| stat_quantile() | geom_quantile() |
| stat_smooth() | geom_smooth() |
| stat_sum() | geom_sum() |

Tablica 4: Funkcije iz familije stats i odgovarajuće funkcije iz familije geom

Osim ovih, postoji i niz funkcija iz familije stats koje ne mogu biti zamijenjene funkcijama geom, a njihov pregled dajemo u sljedećoj tablici.

| Funkcija iz familije stats | Opis |
|---|---|
| stat_ecdf() | Graf empirijske funkcije distribucije |
| stat_function() | Računa vrijednost y za funkcije x vrijednosti |
| stat_summary() | Radi sažetak y vrijednosti pri različitim x vrijednostima |
| stat_summary2d() stat_summary_hex() | Sažetak vrijednosti raspoređenih u blokove |
| stat_qq() | Radi izračun za QQ graf |
| stat_spoke() | Konvertira kuteve i radiuse u položaj |
| stat_unique() | Uklanja duplicirane redove |

Tablica 5: Funkcije iz familije stats i njihovi opisi

# 3. Osnovni grafovi

Cilj ovog poglavlja je susresti se s gramatikom grafova na primjerima te naučiti kreirati bazične, ali korisne grafove. Ovdje ćemo se bazirati na osnovnim svojstvima grafa i brzoj i efikasnoj vizualizaciji podataka. Kasnije kroz rad ćemo se dublje upoznati s komponentama gramatike grafova te ćemo postojeće grafove nadograđivati metodama koje ćemo kroz rad upoznati.

## 3.1. Baza podataka

Ovdje predstavljamo bazu podataka koju ćemo koristiti kroz cijelo poglavlje, ali i kasnije kroz rad. Radi se o bazi podataka "tips.csv" koja je dostupna na stranici Kaggle [7]. Baza je skup statističkih podataka koji sadržavaju 244 opservacija 7 varijabli. To su:

- total_bill - numerička (neprekidna) varijabla koja predstavlja iznos cjelokupnog računa (u dolarima)
- tip - numerička (neprekidna) varijabla koja predstavlja iznos napojnice koja je ostavljena konobaru (u dolarima)
- sex - spol platitelja računa: muški ili ženski (kategorijalna varijabla)
- smoker - odnosi se na to jesu li gosti restorana odabrali dio restorana u kojem je pušenje dozvoljeno ili ne (kategorijalna varijabla)
- day - dan u tjednu u kojem je posjeta restoranu zabilježena. Vrijednosti koje ova varijabla poprima su četvrtak, petak, subota i nedjelja (kategorijalna varijabla)
- time - doba dana u kojem je transakcija obavljena, odnosno obrok koji je poslužen - ručak ili večera (kategorijalna varijabla)
- size - broj serviranih i naplaćenih porcija (numerička diskretna varijabla)

Podaci datiraju iz 1990. godine, a dio su istraživanja kojim se htjelo ustanoviti koji su to faktori koji utječu na visinu napojnice. Istraživanje je provedeno u SAD-u, a prikupljanje

podataka je trajalo 2 i pol mjeseca. Naime, napojnice su veliki dio američke kulture. Fiksni dio plaće je vrlo mali te su napojnice glavnina prihoda uslužnih djelatnika. Cilj istraživanja je iskoristiti sve ono što se pokaže utjecajno na visinu napojnice kako bi se konobari pravedno raspodjelili te na taj način nitko ne bi bio zakinut. Kroz primjere u nastavku ćemo pokušati detektirati koji su to faktori važni za djelatnike restorana.

## 3.2. Osnovne komponente grafa

Svaki graf nacrtan pomoću paketa ggplot2 se sastoji od tri ključne komponente. To su:

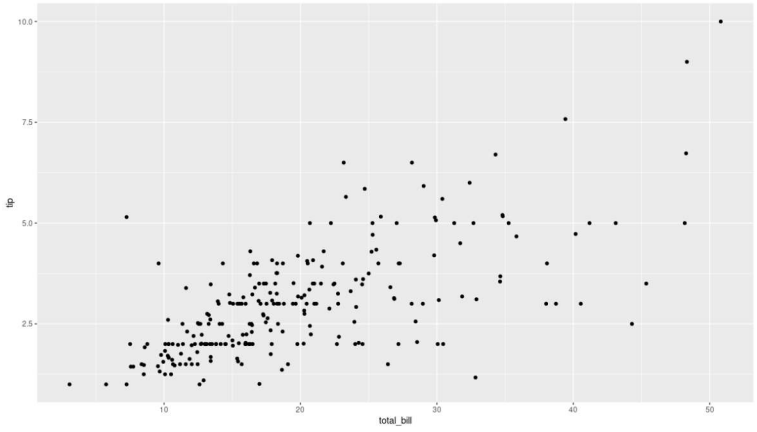
- skup podataka
- skup estetskih značajki kojima određujemo pravilo pridruživanja između varijabli iz naše baze podataka i vizualnih svojstava i
- barem jedan sloj kojim zadajemo kako prikazati svaku opservaciju. Slojeve je uobičajeno kreirati koristeći funkcije iz familije funkcija geom.

U nastavku slijedi primjer:

```
ggplot(tips, aes(x = total_bill, y = tip)) +
  geom_point()
```

Dakle, u ovom primjeru, tips je baza podataka iz koje "izvlačimo" varijable, funkcija aes (skraćeno od Aesthetic) predstavlja estetsku značajku, a njome smo koordinatnim osima pridružili skupove podataka koje želimo prikazati. Treću osnovnu komponentu, sloj, ovdje predstavlja funkcija geom_point, a njom smo odredili prikazivanje podataka u obliku točaka.

Ovakvim kodom dolazimo do grafa prikazanog na Slici 5.




Slika 5: Osnovne komponente grafa

Primjetimo još da su baza podataka i estetika dani funkcijom ggplot, dok je komponenta koja govori kako će podaci biti prikazani (tzv. sloj) integrirana na način da se ggplot-u doda znak "+", Još jedna napomena: kako gotovo svaki graf pridružuje određene podatke koordinatnim osima, prvi argumenti koji su proslijeđeni funkciji aes() se automatski pridružuju apscisi i ordinati, respektivno. Uzevši to u obzir, naš prethodni kod je identičan (u smislu krajnjeg rezultata) sljedećem:

```
ggplot(tips, aes(total_bill, tip)) +
  geom_point()
```

U nastavku ćemo koristiti posljednju inačicu koda zbog jednostavnosti i preglednosti. Također, zbog preglednosti koda, preporuka je svaku naredbu pisati u novi red.

Osvrnimo se još na sam sadržaj grafa sa slike 5. Vidimo da postoji korelacija između iznosa računa i napojnica. To je očekivano jer Amerikanci napojnicu često računaju kao određeni postotak iznosa računa. Vidimo nekoliko stršećih vrijednosti gdje je za manji račun ostavljena nešto veća napojnica, ali i obrnutu situaciju, gdje je za veći račun ostavljena manja napojnica.

### 3.2.1. Ostali estetski atributi

U ovom potpoglavlju razmotrit ćemo još estetskih atributa kao što su boja, veličina i oblik. Ove značajke, kao i pridruživanje podataka koordinatnim osima, dodajemo kao argument funkciji aes(). U nastavku dajemo primjere korištenja ovih atributa:

```
aes(total_bill, tip, color = day)
```

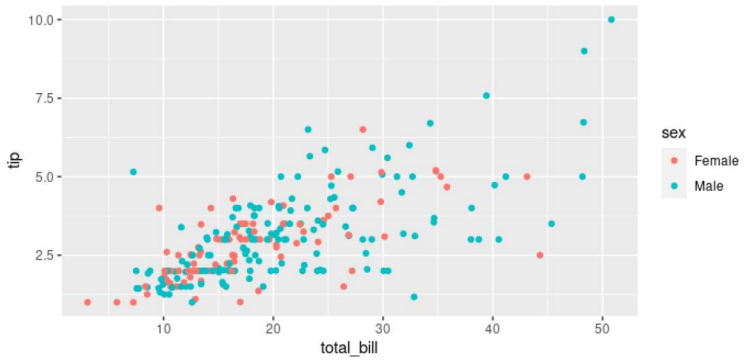
```
aes(total_bill, tip, shape = drv)
```

```
aes(total_bill, tip, size = cyl)
```

Kao što je već spomenuto, ggplot2 se brine za brojne detalje kao što su oznake koordinatnih osi, legende itd. Svaka skala se brine za jednu od tih značajki. Za sad ćemo koristiti zadane vrijednosti ovih karakteristika. Promotrimo sada graf sa Slike 5, ali ovog puta (bojom) grupirajmo podatke po spolu platitelja računa i to na način da podatke mapiramo bojama koje će korespondirati spolu platitelja računa. To postižemo sljedećim kodom:

```
ggplot(tips, aes(total_bill, tip, colour = sex)) +
  geom_point()
```

Ovako kreiran graf dan je u nastavku.




Slika 6: Iznos računa i napojnica po spolu

Na ovaj način detektirali smo da su stršeće vrijednosti napojnice koje su ostavile osobe muškog spola.

Ono što je još moguće ovdje jest cijeli graf, odnosno sve točke grafa, prikazati jednom bojom (umjesto da "skaliramo" po grupama, ovdje spolu). To možemo učiniti izvan funkcije

aes(), tako da ovaj argument dodamo u novi sloj grafa. Dva su načina na koja to možemo učiniti, a oba su dana u nastavku.

```
ggplot(tips, aes(total_bill, tip))
+ geom_point(aes(colour = "red"))
```

```
ggplot(tips, aes(total_bill, tip))
+ geom_point(colour = "red")
```

Od ostalih estetskih svojstava spomenut ćemo

- unutarnje obojenje likova i mogućnost odabira transparentne pozadine
- manipuliranje linijama: odabir debljine linije te vrste (istočkano, isrtano, točka-crta...)
- prilagođavanje poligona bojanjem, izborom linija te veličinom
- manipuliranje točkama u smislu izbora oblika točke (ovdje se na točku referiramo u kontekstu oznake u koordinatnom sustavu) pri čemu ovaj paket nudi 25 različitih predefiniranih oblika. Također je moguće koristiti proizvoljno slovo ili znak kao oznaku te je moguće podesiti veličinu točke kao i boju obojenja točke.
- prilagođavanje teksta izborom fonta te veličine. Moguće je i podcrtavanje, podebljavanje te nakošavanje teksta kao i horizontalno i vertikalno prilagođavanje.


Slika 7: Mogućnosti linija


Slika 8: Mogućnosti točaka

Za više detalja o ovim atributima upućujem čitatelja na vignette("ggplot2-specs") [2].

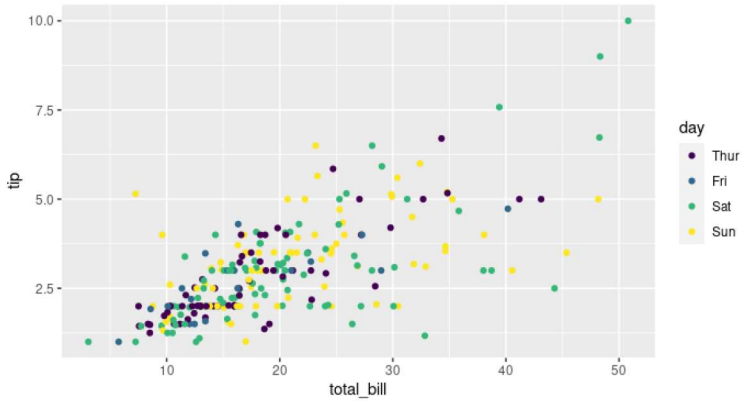
Pri izradi grafova važno je imati na umu koji estetski atributi dobro funkcioniraju s kojim tipom prikazivanja. Savjetuje se umjereno korištenje ovih atributa. Bolja je praksa grafove razložiti na više prikaza kojima će se naglasak staviti na određene komponente. Takav način rada nam omogućuje bolje razumijevanje podataka i donošenje zaključaka.

### 3.2.2. Fasetiranje

U ovom potpoglavlju objasnit ćemo fasetiranje (eng. *faceting*). Ovaj pojam označava razlaganje grafa na više podgrafova bazirajući se na kategorije odabrane kategorijalne varijable. Na primjer, ukoliko imamo puno kategorija određene varijable u odnosu na koju promatramo podatke, ako se odlučimo za bojanje točaka grafa po grupama, graf će biti nepregledan i nećemo puno toga vidjeti iz takvog prikaza te nećemo doći do nikakvih zaključaka o podacima. Nasuprot tome, ukoliko iskoristimo fasetiranje i graf podijelimo na više podgrafova,

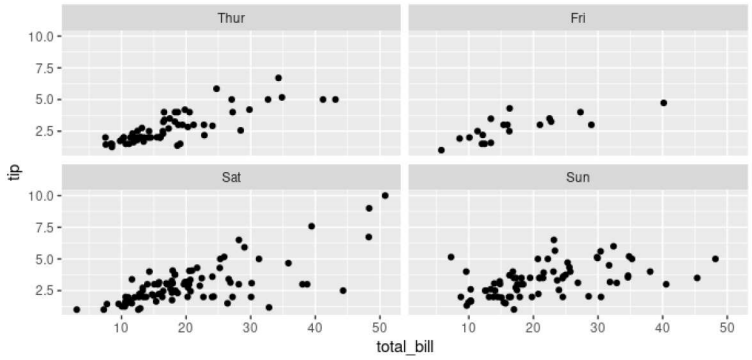
analiza će nam biti puno jednostavnija i iz ovakvog prikaza će nam biti moguće donositi zaključke o svakoj pojedinoj kategoriji. Postoje tri vrste fasetiranja: obavijeno (eng. *wrapped*), rešetkasto (eng. *grid*) i zadano, koje zapravo ne predstavlja fasetiranje, već samo jedan graf (eng. *null*). Obavijeno je korisnije te ćemo se za sada fokusirati na njega.

Pogledajmo sada primjer. Razmatrat ćemo dosadašnji graf iznosa računa i napojnica, ali s obzirom na dan u tjednu. Uočimo da na grafu 9 ne možemo doći do nekakvih konkretnih zaključaka.




Slika 9: Iznos računa i napojnica u odnosu na dane

S druge strane, ako se odlučimo za razlaganje grafa na više podgrafova (Slika 16), dobivamo puno pregledniji prikaz i jasniju predodžbu o podacima. Sada uočavamo da su subotom veći računi i veće napojnice. Nedjeljom su također zabilježeni veći iznosi računa, ali napojnice ne prate to povećanje. Uočavamo da je petkom najmanje posjeta restoranu itd.




Slika 10: Iznos računa i napojnica u odnosu na dane korištenjem funkcije facet_wrap

Ovakav graf smo dobili sljedećim kodom:

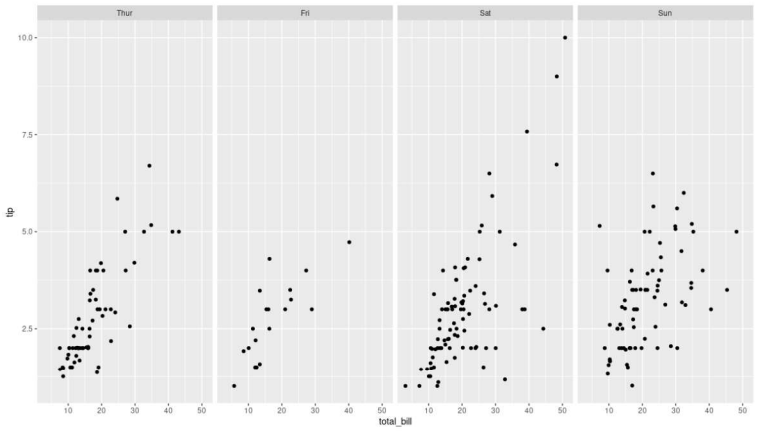
```
ggplot(tips, aes(total_bill, tip)) +
  geom_point() +
  facet_wrap(~day)
```

Dakle, ovakvo raščlanjivanje grafa dobili smo dodavši sloj *facet_wrap()* kojoj smo proslijedili ključ po kojem smo htjeli grupirati podatke, odnosno kategorijalnu varijablu (u ovom slučaju *day*) sa znakom $\sim$ ispred nje. S atributima ncol i nrow određujemo u koliko stupaca,

odnosno redaka raspoređujemo podgrafove. Pri tome određujemo samo jedno. S atributom dir određujemo hoće li raspoređivadnje podgrafova biti horizontalno ili vodoravno. Atribut as.table regulira raspoređivanje podgrafova u tablicu pri čemu se onaj s najvećim vrijednostima smješta u donji desni kut ukoliko proslijedimo vrijednost TRUE i u gornji desni kut ukoliko proslijedimo vrijednost FALSE.

Funkcija facet_grid podgrafove smješta u rešetku, a gornji graf s ovom funkcijom možemo vidjeti u nastavku.

```
ggplot(tips, aes(total_bill, tip)) +
  geom_point() +
  facet_grid(~day)
```




Slika 11: Iznos računa i napojnica u odnosu na dane korištenjem funkcije facet_grid

Ukoliko umjesto facet_grid( varijabla) koristimo facet_grid(varijabla ), podgrafovi će se poredati u stupac. To implicira da će se x-osi poravnati što je posebno korisno za uspoređivanje distribucija.

## 3.3. Osnovne funkcije iz familije geom

Do sada smo u svakom grafu koristili funkciju geom_point(). Ona je uglavnom bila jedini dodatni sloj grafa, a njome smo zadavali crtanje podataka u obliku geometrijskih točaka. Iako smo već napravili pregled funkcija iz familije geom, u ovom potpoglavlju ćemo ponoviti najvažnije funkcije koje ćemo koristiti kroz rad, a potrebne su nam za skiciranje osnovnih grafova. To su

- `geom_smooth()`

  provlači krivulju kroz podatke, tj zaglađuje ih i prikazuje standardnu grešku takve krivulje

- `geom_boxplot()`

  skicira kutijasti dijagram u svrhu sažetka distribucije skupa točaka, odnosno grafičkog prikazivanja karakteristične petorke (minimum, donji kvartil, medijan, gornji kvartil i maksimum)

- `geom_histogram()` i `geom_freqpoly()`

  generira histogram, a koristi se za skiciranje distribucije neprekidnih slučajnih varijabli

- `geom_bar()`

  generira stupčasti dijagram, a koristan je za prikazivanje distribucije diskretne slučajne varijable

- `geom_path()` i `geom_line()`

  se koriste za dobivanje linija između točaka (kako bismo uočili kako su se vrijednosti mijenjale tokom vremena). Razlika između ove dvije funkcije je u tome što funkcija geom_line() točke povezuje redom, slijeva, na desno, dok geom_path() može povezivati točke u bilo kojem smjeru.

## 3.4. Manipuliranje koordinatnim osima

U daljnjim poglavljima rada ćemo detaljnije proći mogućnosti manipuliranja i prilagođavanja koordinatnih osi. U ovom potpoglavlju ćemo se dotaknuti samo osnovnih značajki, a to su imenovanje koordinatnih osi te određivanje njihovog raspona. Navedeni elementi grafa se modificiraju funkcijama xlab(), ylab(), ylim() i ylim(). U nastavku slijedi primjer.

```
ggplot(tips, aes(x = total_bill, y = tip)) +
  geom_point() +
  xlab("iznos racuna") +
  ylab("iznos napojnice") +
  xlim(2,20) +
  ylim(0,6)
```

Dakle, na ovaj način smo promijenili naziv apscise u "iznos racuna", ordinate u "iznos napojnice" te smo zadali da x-os prikazuje vrijednosti od 2 do 20, a y-os od 0 do 6.

## 3.5. Slojevi

Već smo spomenili da su slojevi osnovni dio izrade grafova te da omogućuju njihovu strukturiranu izgradnju. Slojeve koristimo iz tri razloga, a to su:

- kako bismo prikazali "sirove" podatke. Ovakvim prikazivanjem podataka uočavamo njihovu općenitu strukturu, lokalnu strukturu te stršeće vrijednosti. Ovaj sloj je dio gotovo svakog grafa, služi za dobivanje generalne slike i često je u najranijim fazama izgradnje grafova jedini sloj.
- kako bismo prikazali statističke sažetke podataka. Kako naš model napreduje i što više otkrivamo o njemu, korisno je graditi i prikazati predikcije. Njih grafički prikazujemo kako bismo otkrili više o modelu i samim podacima te kako bi nas to dovelo do daljnjih zaključaka.
- za dodavanje dodatnih metapodataka. To se odnosi na kontekst, napomene i reference. Ovaj sloj nam daje pozadinu, bilješke koje pomažu u razumjevanju i shvaćanju značenja sirovih podataka ili fiksne reference.

## 3.6. Osnovni tipovi grafova

Ovdje ćemo napraviti pregled funkcija iz familije funkcija geom. Ove funkcije mogu same tvoriti grafove koji će služiti kao cjeloviti prikaz, ali mogu biti i samo jedan dio izgradnje kompleksnijih grafova. Sve funkcije geom su dvodimenzionalne što znači da im moramo proslijediti dva argumenta, podatke koje želimo prikazati na x-osi i podatke koje želimo na y-osi. Svi geomi (funkcije iz familije geom) podrazumjevaju estetske značajke pod koje spadaju boja i veličina, a kod funkcija koji produciraju oblike koji imaju površine (funkcije geom_bar() - stupčasti dijagram, geom_style() i geom_polygon() - poligon) automatski se generira boja površine.

Kako budemo navodili funkcije, uz njih ćemo navesti i praktičan primjer. U svrhu toga, kreiramo data frame sljedećim kodom:

```
df <- data.frame(
  x = c(1, 2, 3),
  y = c(3, 6, 4),
  label = c("a","b","c")
)
```

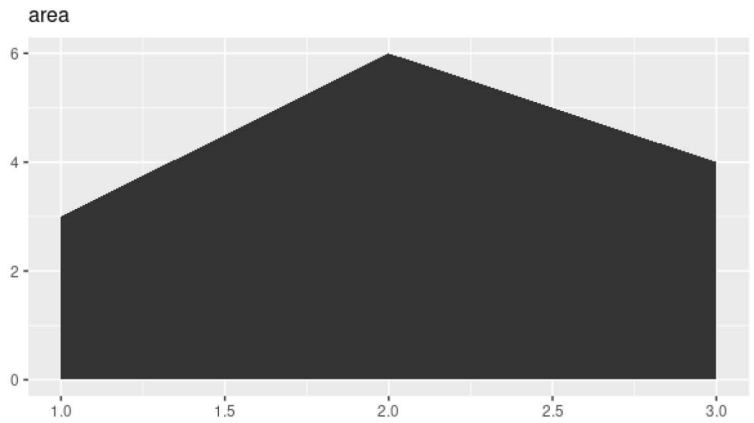
Ovu malu bazu podataka ćemo za početak nacrtati u obliku osnovnog grafa na koji ćemo nadograđivati naše primjere. Graf je dan sljedećim kodom.

```
graf <- ggplot(df, aes(x, y, label = label)) +
  labs(x = NULL, y = NULL) + #uklanjanje naziva koordinatnih osi
  theme(plot.title = element_text(size = 12))
  #smanjenje velicine naslova grafa
```

Navedimo sada funkcije koje spadaju u ovu grupu:

- geom_area() konstruira linijski dijagram kojem je ispunjena površina ispod grafa - do x-osi. Ukoliko pokušamo nacrtati više grupa, one će se slagati jedna iznad druge. Promotrimo primjer:

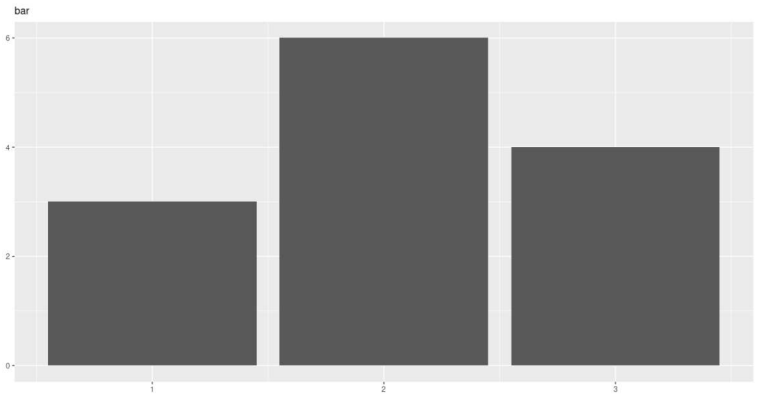
```
graf + geom_area() + ggtitle("area")
```




Slika 12: Primjer funkcije geom_area()

- geom_bar(stat = "identity") služi za kreiranje stupčastog dijagrama pri čemu ne mislimo na prebrojavanje neke pojave, već na crtanje stupaca do određene točke. "stat = "identity"" je ovdje potreban kako bismo naznačili da ne prebrojava, odnosno da podatke ostavi nepromijenjenima. I ovdje, u slučaju više grupa, one se slažu jedna iznad druge. Slijedi primjer:

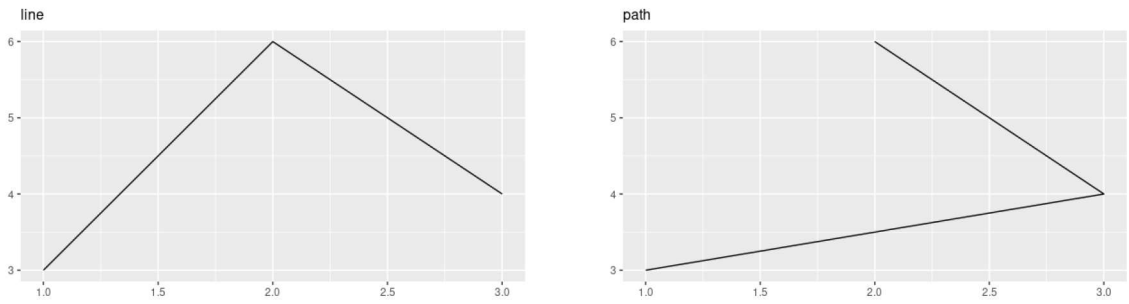
```
graf + geom_bar(stat = "identity") + ggtitle("bar")
```




Slika 13: Primjer funkcije geom_bar()

- geom_line() crta linijski dijagram. Atribut "group" definira koje opservacije će biti spojene. Ova funkcija povezuje točke slijeva nadesno dok funkcija geom_path() povezuje točke redom kojim se koordinate koje pripadaju tim točkama pojavljuju u podacima. Obje funkcije podrazumjevaju atribut linetype kojim manipuliramo izgledom same linije, a biramo između pune, istočkane ili iscrtkane. Promotrimo na primjeru:

```
graf + geom_line() + ggtitle("line")
graf + geom_path() + ggtitle("path")
```

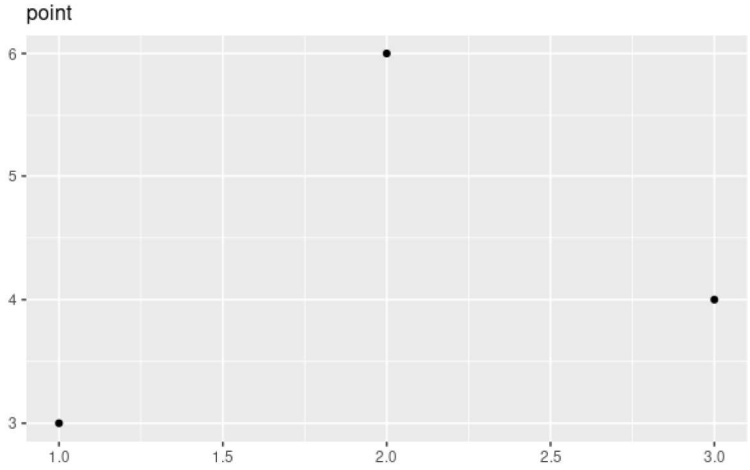



(b)

Slika 14: Razlika između funkcija geom_line() i geom_path()

- geom_point() omogućava crtanje dijagrama raspršenosti, a uključuje atribut "shape", odnosno oblik.

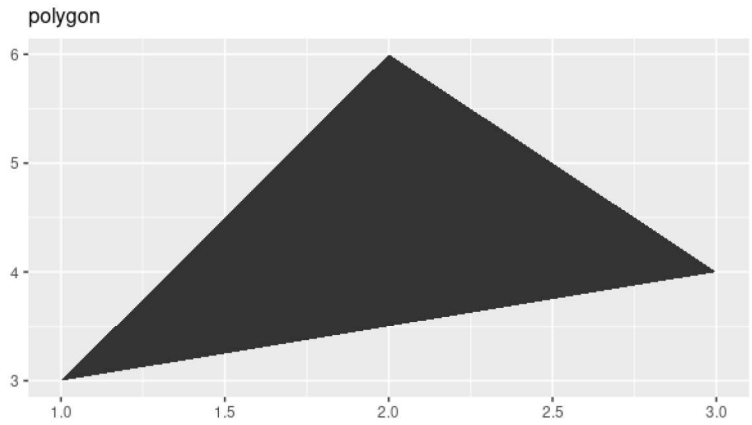
```
graf + geom_point() + ggtitle("point")
```




Slika 15: Primjer funkcije geom_point()

- geom_polygon() crta poligone koji su zapravo ispunjeni putovi. Svaki vrh poligona zahtjeva poseban red u bazi podataka koju analiziramo. Primjer grafa:

```
graf + geom_polygon() + ggtitle("polygon")
```




Slika 16: Primjer funkcije geom_polygon()

geom_rect(), geom_tile() i geom_raster() crtaju pravokutnike. geom_rect() crta pravokutnik čiji su vrhovi zadani s minimalnom vrijednošću x koordinate, maksimalnom vrijednošću x koordinate, minimalnom vrijednošću y koordinate i maksimalnom vrijednošću y koordinate.

Primjetimo da u primjerima nismo svaki sloj pisali u novi red, što smo naveli kao najbolju praksu, već smo sve pisali u jedan redak. Ovdje, i u nastavku, ukoliko se koristimo ovakvim zapisom, to je isključivo s ciljem preglednosti i jasnoće koda u radu.

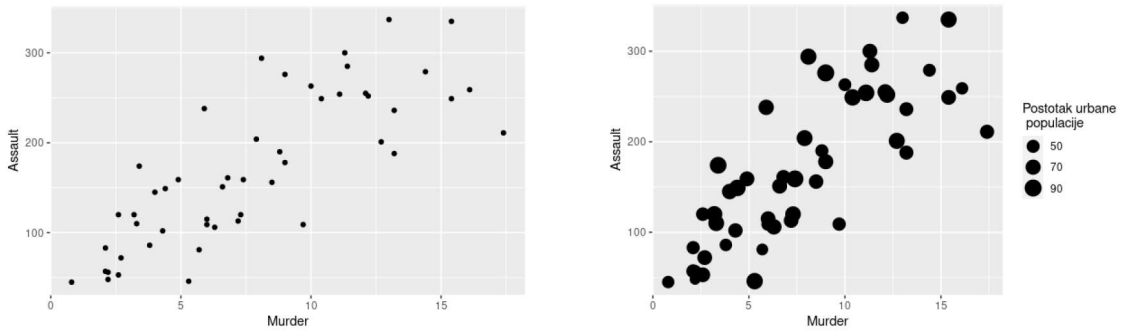
## 3.7. Grupni geomi

Na najvišoj razini, funkcije geom možemo podijeliti na pojedinačne i grupne. Funkcije geom koje karakteriziramo kao pojedinačne za svaki red iz baze podataka, odnosno za svaku opservaciju, crtaju zaseban grafički objekt. Primjerice, geom_point() uzima red po red i za svaki red crta jednu točku. Grupni geom nekoliko opservacija prikazuje jednim geometrijskim objektom koji može biti rezultat statističkih analiza i sažetaka. Primjerice, jedan takav objekt je kutijasti dijagram. Ukoliko promatramo linije i putove, uočit ćemo da su oni nešto između pojedinačnih i grupnih geoma. Naime, svaka linija se sastoji od skupa ravnih segmenata pri čemu svaki taj segment označava dvije točke (početak segmenta je jedna točka, kraj druga). Dodjeljivanje grafičkih elemenata opservacijama kontroliramo estetskim atributima.

### 3.7.1. Ponderirani podaci

U slučajevima kada analiziramo bazu podatka u kojoj svaki red sadrži više informacija, nije dovoljno bazirati se samo na dva podatka i zasebno ih analizirati, već je potrebno u obzir uzeti i kontekst, odnosno ostale podatke. To je moguće na način da koristimo ponderiranje, odnosno promatranim varijablama pridružimo težine. Primjer ćemo prikazati na bazi podataka USArrest koja je ugrađena u R. Baza se sastoji od 50 opservacija koje predstavljaju savezne države Sjedinjenih Američkih Država. O svakoj državi zabilježena su 4 podatka: broj počinjenih napada, ubojstava, silovanja te postotak urbanog stanovništva u toj državi. Možemo, na primjer, analizirati broj i odnos fizičkih napada i ubojstava, ali bi bilo korisno u obzir uzeti postotak urbanog stanovništva te ponderirati podatke u odnosu na tu varijablu. Slijedi primjer.

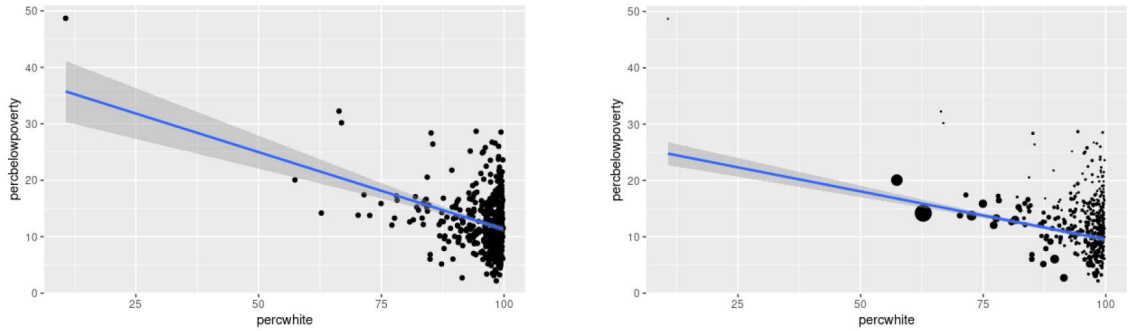
```r
#neponderirani podaci
ggplot(USArrests, aes(Murder, Assault)) +
    geom_point()

# Podaci ponderirani postotkom urbanog stanovnistva
ggplot(USArrests, aes(Murder, Assault)) +
  geom_point(aes(size = UrbanPop)) +
  scale_size_area("Postotak urbane \n populacije", breaks = c(50,  70,  90))
```




Slika 17: Ponderiranje podataka

Razmotrimo slučajeve koji uključuju statističke transformacije poput kutijastog dijagrama, histograma, prikazivanja gustoće, zaglađivanja i kvartilske regresije. Ovdje ponderiramo na način da težine uključimo atributom weight. Taj atribut će biti proslijeđen funkciji statističkog sažetka. Varijabla ponderiranja nije eksplicitno naglašena te ne generira legendu, ali mijenja konačni rezultat grafa, odnosno krajnji ishod statističke analize. Slijedi primjer na kojem ćemo vidjeti kako pridruživanje težina s obzirom na veličinu populacije utječe na modeliranje veze između postotka bijelog stanovništva i postotka ljudi koji se nalaze ispod granice siromaštva. Baza nad kojima radimo analizu se nalazi unutar R-a, a radi se o bazi pod nazivom midwest. Jedna opservacija predstavlja jedan okrug u srednjem zapadu SAD-a, a baza bilježi 28 varijabli koje sadrže niz informacija o pojedinom okrugu.

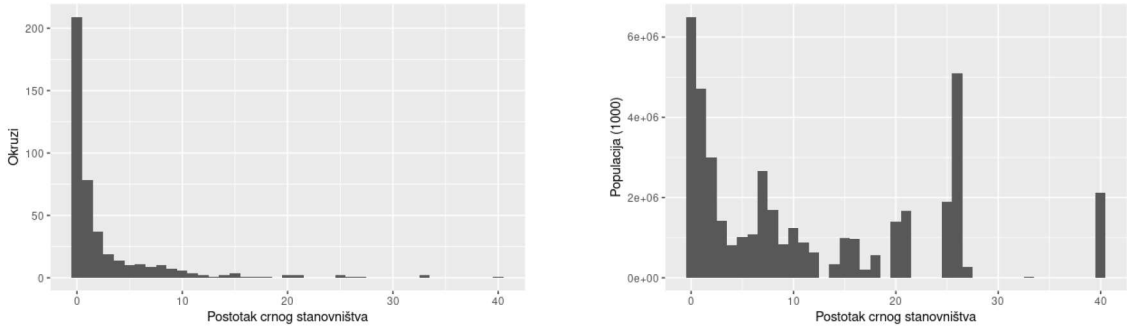
```r
# neponderirano
ggplot(midwest, aes(percwhite, percbelowpoverty)) +
  geom_point() +
  geom_smooth(method = lm, size = 1)

# ponderirano
ggplot(midwest, aes(percwhite, percbelowpoverty)) +
  geom_point(aes(size = poptotal / 1e6)) +
  geom_smooth(aes(weight = poptotal), method = lm, size = 1) +
  scale_size_area(guide = "none")
```




Slika 18: Modeliranje linearne veze bez i sa ponderiranjem podataka

Promotrimo sad utjecaj ponderiranja na prikaz histograma. Vizualizirat ćemo histograme postotka crnog stanovništva. Prvi histogram nam daje uvid u distribuciju okruga dok drugi, ponderirani, pri čemu su težine dodijeljene brojem stanovnika pojedinog okruga, daje uvid u distribuciju broja ljudi.

```
#neponderirano
ggplot(midwest, aes(percblack)) +
  geom_histogram(binwidth = 1) +
  xlab("Postotak crnog stanovnistva") +
  ylab("Okruzi")

#ponderirano
ggplot(midwest, aes(percblack)) +
  geom_histogram(aes(weight = poptotal), binwidth = 1) +
  xlab("Postotak crnog stanovnistva") +
  ylab("Populacija (1000)")
```



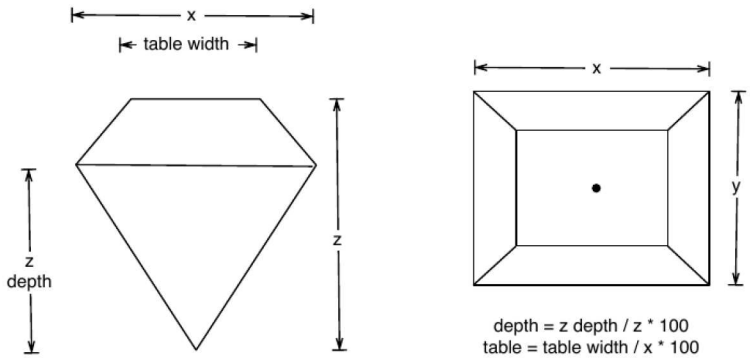

Slika 19: Histogram postotka crnog stanovništva u srednjem zapadu SAD-a

## 3.8. Prikazivanje distribucije

Postoji niz geoma za procjenu distribucije varijable. Izbor funkcije geoma ovisi o:

- radi li se o neprekidnoj ili diskretnoj distribuciji
- dimenziji distribucije
- radi li se o uvjetnoj ili zajedničkoj distribuciji.

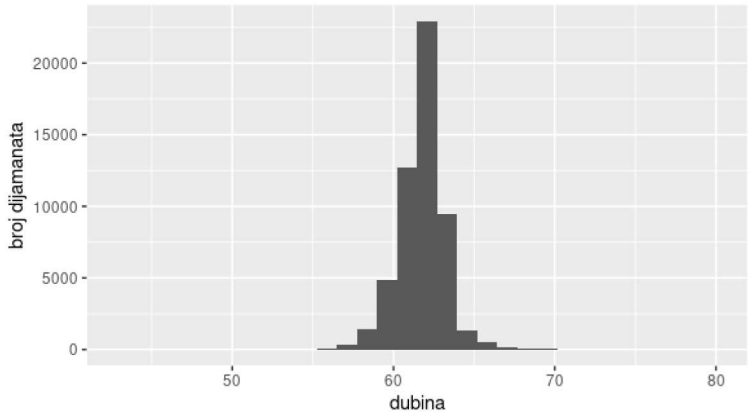
Kako bismo funkcije koje budemo navodili, a koje su prikladnije za velike baze podataka, potkrijepili primjerima, koristit ćemo bazu koja se nalazi u R-u, pod nazivom diamonds. U njoj se nalaze podaci o cijeni, kvaliteti i fizičkim značajkama 54 000 dijamanata. Značajke koje se odnose na kvalitetu dijamanta su: rez, broj karata, boja i čistoća. Fizičke značajke su: dubina, stol te x, y i z. Što su točno fizičke značajke te kako se mjere prikazano je na Slici 20.




Slika 20: Fizičke karakteristike dijamanta [10]

Prijeđimo sad na funkcije i primjere. Za grafički prikaz jednodimenzionalne neprekidne distribucije varijable najvažnija funkcija je geom_histogram(). Pogledajmo primjer.

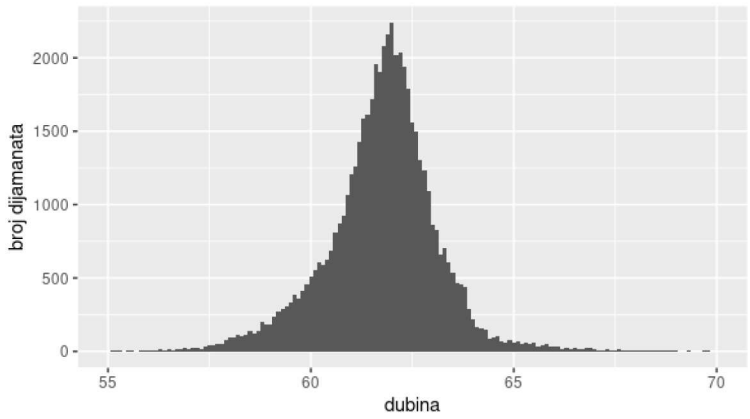
```
ggplot(diamonds, aes(depth)) +
  geom_histogram() +
  xlab("dubina") +
  ylab("broj dijamanata")
```




Slika 21: Histogram dubine dijamanata

Ovo je najjednostavniji histogram bez dodatnih atributa. Međutim, mi iz ovakvog prikaza ne vidimo puno toga. To je zato što su nam stupci grafa preširoki. Važno je eksperimentirati sa širinom stupaca kako bismo dobili grafički prikaz iz kojeg ćemo dobiti što jasniju predodžbu o podacima, odnosno iz kojeg ćemo moći izvesti konkretne zaključke. Pogledajmo na primjeru što se dogodi kada značajno smanjimo širinu stupaca.

```
ggplot(diamonds, aes(depth)) +
  geom_histogram(binwidth = 0.1) +
  xlim(55, 70) +
  xlab("dubina") +
  ylab("broj dijamanata")
```




Slika 22: Histogram dubine dijamanata sa smanjenom širinom stupaca

Ovdje smo dobili puno jasniji prikaz distribucije dubine dijamanata iz kojeg možemo naslutiti da ova fizička karakteristika dijamanta ima normalnu distribuciju što se iz prethodnog histograma nije dalo iščitati.

Osim atributom binwidth, na širinu stupaca možemo utjecati na način da definiramo broj stupaca (bins) ili da specificiramo točna mjesta na kojima želimo da se podaci dijele (breaks). Općenito se ne treba oslanjati na zadane vrijednosti ove funkcije jer se iz tako nacrtanog grafa uglavnom ne mogu iščitati zaključci o distribuciji. Primjetimo još da, osim što smo na histogramu na Slici 22 smanjili širinu stupaca, smanjili smo i raspon x osi. To je također dobra praksa kada želimo detaljnije sagledati distribuciju.

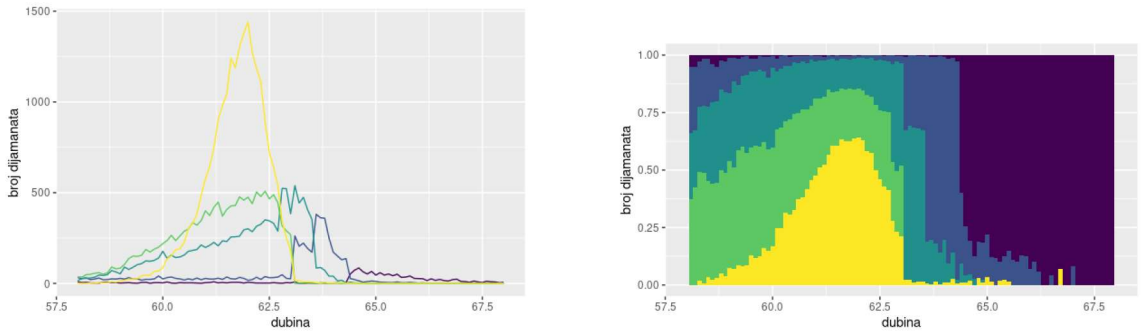
Ukoliko želimo usporediti distribucije više grupa slučajne varijable, to možemo učiniti na nekoliko načina:

- varijablu podijeliti na podgrupe te prikazati histograme tih podgrupa koristeći funkciju facet_wrap($\sim$ var)
- koristeći boju i frekvenciju poligona s funkcijom geom_freqpoly()
- koristeći graf uvjetne gustoće s funkcijom geom_histogram(position = "fill")

U nastavku možemo vidjeti primjere grafova koji koriste funkcije geom_freqpoly() i geom_histogram().

```
ggplot(diamonds, aes(depth)) +
  geom_freqpoly(aes(colour = cut), binwidth = 0.1, na.rm = TRUE) +
  xlim(58, 68) +
  theme(legend.position = "none") +
  xlab("dubina") +
  ylab("broj dijamanata")
```

```r
ggplot(diamonds, aes(depth)) +
  geom_histogram(aes(fill = cut), binwidth = 0.1,
                 position = "fill", na.rm = TRUE) +
  xlim(58, 68) +
  theme(legend.position = "none") +
  xlab("dubina") +
  ylab("broj dijamanata")
```

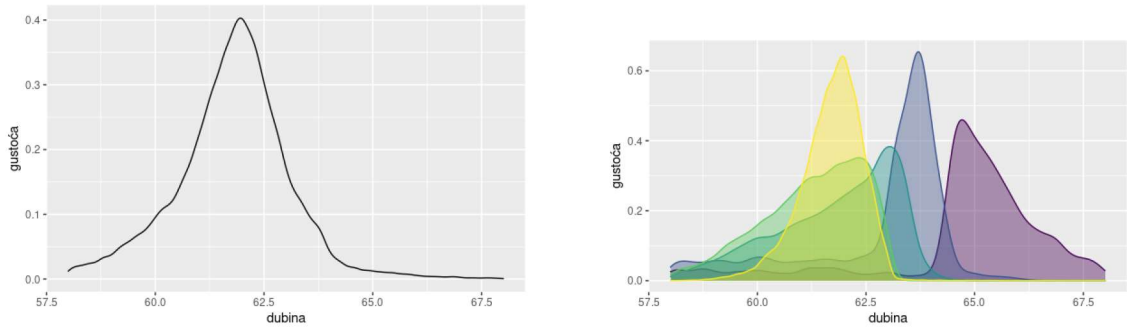



Slika 23: Prikaz distribucije slučajne varijable po grupama

Obje funkcije koriste istu temeljnu statističku transformaciju: stat = bin što rezultira dvjema izlaznim varijablama: prebrojavanjem i gustoćom. Varijabla koja bilježi prebrojavanje se, prema zadanim postavkama, pridružuje y osi jer je, takvim pridruživanjem, graf najintuitivniji. Gustoća je dobivena na način da se broj pojedinih vrijednosti podijeli s ukupnim brojem opservacija i pomnoži sa širinom stupaca. Ovakvi prikazi su korisni kada želimo usporediti oblik distribucije, a ne samo vrijednosti i veličinu.

Vizualizacija alternativna prikazu baziranom na stupcima je vizualizacija procjenom gustoće. Ovakav prikaz se postiže funkcijom geom_density(). Ono što je prednost kod ovakvog prikaza je to što ima poželjna teorijska svojstva, ali je problem što je teže poveziv s podacima. Preporuka je koristiti prikaz gustoće kada je gustoća glatka i neprekidna. Moguće je i manipulirati parametrima podešavanja kako bismo gustoću učinili manje ili više glatkom. Slijede primjeri.

```r
ggplot(diamonds, aes(depth)) +
  geom_density(na.rm = TRUE) +
  xlim(58, 68) +
  xlab("dubina") +
  ylab("gustoća") +
  theme(legend.position = "none")

ggplot(diamonds, aes(depth, fill = cut, colour = cut)) +
  geom_density(alpha = 0.4, na.rm = TRUE) +
  xlab("dubina") +
  ylab("gustoća") +
  xlim(58, 68) +
  theme(legend.position = "none")
```




Slika 24: Primjeri korištenja funkcije geom_density()

Uočimo da smo na drugom grafu distribucije segmentirali u odnosu na rez dijamanta. Parametar alpha se odnosi na razinu prozirnosti boje površine ispod krivulje gustoće. Također, legenda je uklonjena kako bi fokus bio na samom grafu. Važno je još primjetiti da su gustoće na drugom grafu standardizirane zbog čega se informacija o relativnoj veličini pojedine grupe gubi.

Sve funkcije koje smo do sad naveli u ovom potpoglavlju su generirale grafove koji su davali detaljne informacije o distribuciji promatrane slučajne varijable. Međutim, ponekad je potrebno usporediti više distribucija pri čemu moramo biti spremni žrvovati kvalitetu i detaljnost takvih grafova u zamjenu za kvantitetu.

Vizualizacije ovakvog tipa nam omogućavaju:

- geom_boxplot() koja služi za crtanje kutijastog dijagrama. Kutijasti dijagram prikazuje deskriptivnu statistiku što uključuje minimum, donji kvartil, median, gornji kvartil i maksimum podataka. Također, vizualizira stršeće vrijednosti. Ovakav prikaz nam daje puno manje informacija od histograma, ali zauzima puno manje prostora i dovoljan je za kratak pregled i usporedbu. Kutijasti dijagram možemo koristiti i za kategorijalne i za neprekidne varijable. Ukoliko ga koristimo za neprekidne varijable, potrebno je definirati segmente na koje dijelimo tu varijablu (smještamo ju na x-os). Funkcija koju koristimo za takvu podijelu je cut_width().

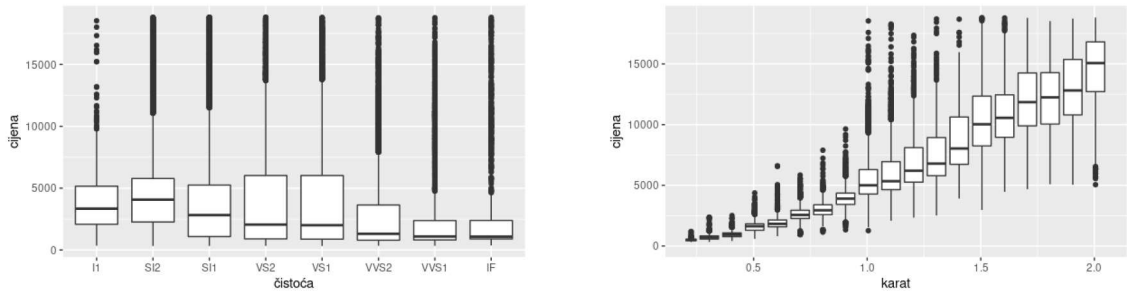
- geom_violin() je kompaktna verzija prikaza distribucije. Temeljno računanje je isto kao kod kutijastog dijagrama, ali se konačni rezultat prikazuje na nešto drugačiji način.

- geom_dotplot() za svaku opservaciju vizualizira jednu točku. Ovako skicirana točka je pažljivo smještena u prostoru s ciljem izbjegavanja preklapanja te prikaza distribucije. Ova funkcija je korisna za manje skupove podataka.

U nastavku slijede primjeri kutijastog i violinskog dijagrama.

```
#kutijasti dijagram u odnosu na kategorijalnu varijablu
ggplot(diamonds, aes(clarity, price)) +
  geom_boxplot()+
  xlab("cistoca") +
  ylab("cijena")

#kutijasti dijagram u odnosu na neprekidnu varijablu
ggplot(diamonds, aes(carat, price)) +
  geom_boxplot(aes(group = cut_width(carat, 0.1))) +
```

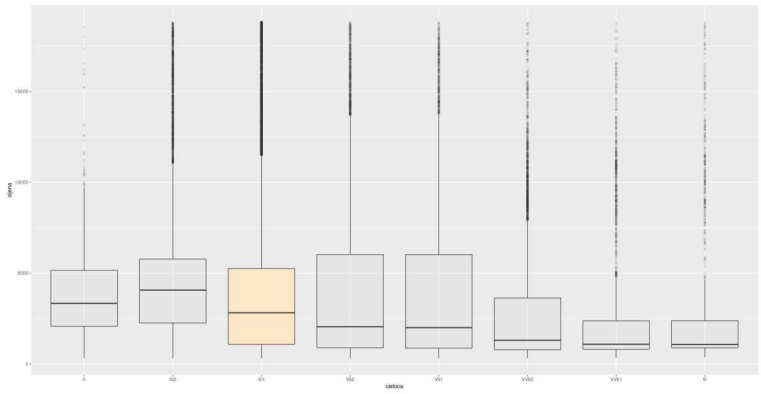
```
  xlim(NA, 2.05) +
  ylim(0, 18850) +
  xlab("karat") +
  ylab("cijena")
```




Slika 25: Primjeri kutijastih dijagrama

Jasno uočavamo rast cijene s porastom karata. Dotaknut ćemo se još jedne korisne opcije kod skiciranja kutijastih dijagrama. Možemo imati potrebu naglasiti jednu od kategorija, odnosno jedan kutijasti dijagram. Ukoliko je to slučaj, to postižemo sljedećim kodom.

```
library(dplyr)

diamonds %>%
  mutate(type=ifelse(clarity=="SI1","Highlighted","Normal")) %>%
ggplot(aes(clarity, price, fill=type, alpha=type)) +
  geom_boxplot()+
  scale_fill_manual(values=c("#FBE7C6", "grey")) +
  scale_alpha_manual(values=c(1,0.1)) +
  theme(legend.position = "none") +
  xlab("cistoca") +
  ylab("cijena")
```

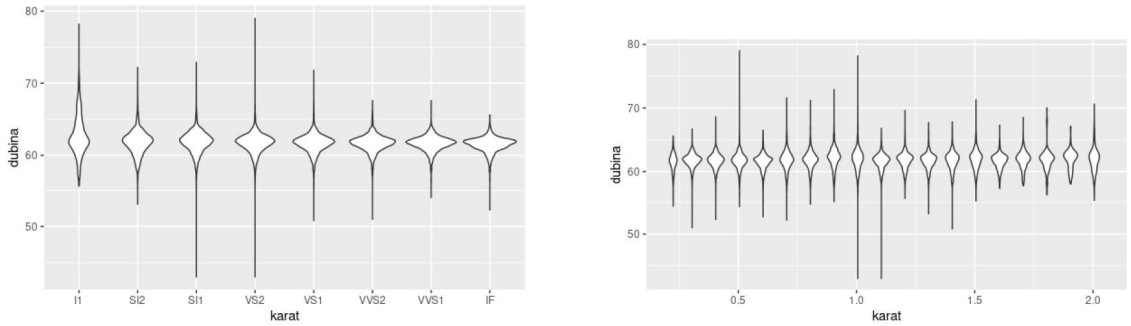


Slika 26: Kutijasti dijagram s naglašenom kategorijom

Dakle, ovdje smo rabili nešto drugačiji pristup nego do sad. Prije svega, moramo učitati biblioteku dplyr kako bismo mogli koristiti operator% > %. Koristimo funkciju mutate

unutar koje koristimo kondicioniranje. Ukoliko je oznaka jasnoće "SI1", kutijasti dijagram će biti naglašen. U suprotnom, ostat će nepromijenjen.

```
library(dplyr)
#u odnosu na kategorijalnu varijablu
ggplot(diamonds, aes(clarity, depth)) +
  geom_violin()+
  xlab("karat") +
  ylab("dubina")

#u odnosu na neprekidnu varijablu
ggplot(diamonds, aes(carat, depth)) +
  geom_violin(aes(group = cut_width(carat, 0.1))) +
  xlim(NA, 2.05)+
  xlab("karat") +
  ylab("dubina")
```




Slika 27: Primjer violinskih grafova

## 3.9. Rješavanje problema preklapanja na grafu

Dijagram rasrpšenosti je vrlo važan alat za uočavanje i procjenu odnosa dviju neprekidnih varijabli. Međutim, ukoliko se suočavamo s velikom bazom podataka, točke u takvom grafu će često biti vizualizirane jedna preko druge zbog čega će takav graf biti izuzetno nepregledan te nas neće dovesti do konkretnih zaključaka o odnosu među promatranim varijablama. U ekstremnim slučajevima, jedino vidljivo iz takvog grafa bit će opseg podataka.

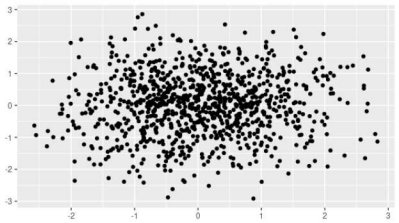
S ovakvim problemom možemo se nositi na nekoliko načina, a izbor ovisi o veličini podataka i količini preklapanja točaka na grafu.

Prvi skup tehnika koji ćemo opisati odnosi se na slučaj kada imamo manji skup podataka. Ove tehnike se fokusiraju na podešavanje estetskih svojstva grafa. Ukoliko smo suočeni s malom količinom preklapanja točaka, možemo ga ublažiti smanjenjem točaka ili uporabom šupljih grafičkih znakova umjesto točaka.

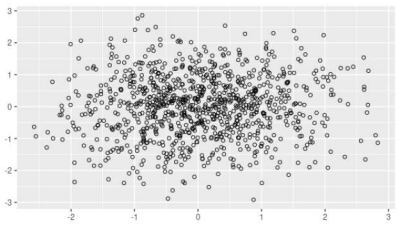
U nastavku slijedi primjer rješavanja preklapanja manipuliranjem grafičkim svojstvima grafa. Graf je dobiven uzorkovanjem 1000 točaka iz dvodimenzionalne normalne distribucije.

```
podaci <- data.frame(x = rnorm(1000), y = rnorm(1000))
norm <- ggplot(podaci, aes(x, y)) + xlab(NULL) + ylab(NULL)
norm + geom_point()
```

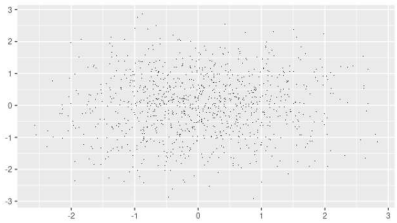
```
norm + geom_point(shape = 1) # suplji krugovi
norm + geom_point(shape = ".") # tocke velicine piksela
```



Slika 28: Graf s velikom količinom preklapanja



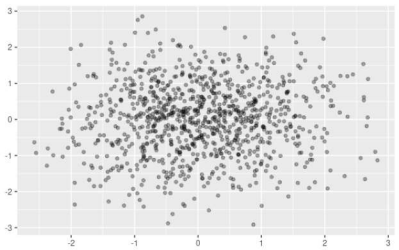
Slika 29: Graf s krugovima



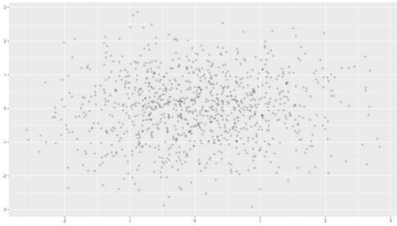
Slika 30: Graf sa smanjenim točkama

Za velike baze podataka i grafove sa značajnom razinom preklapanja točaka možemo se poslužiti tzv. alfa stapanjem kako bismo točke učinili transparentnima. Alfa shvaćamo kao omjer, pri čemu nazivnik predstavlja broj točaka koje se moraju precrtati kako bi se dobila puna boja. Dakle, što je veći nazivnik, to je potrebno više točaka da bi se dobila puna boja što znači da se smanjenjem alfe povećava prozirnost točaka. Vrijednosti manje od $\frac{1}{500}$ se tretiraju kao nula i daju potpuno transparentne točke.

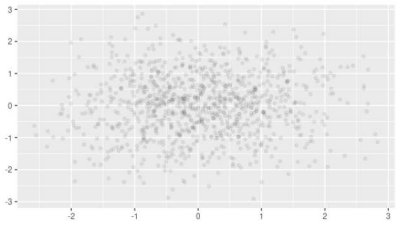
```
norm + geom_point(alpha = 1 / 3)
norm + geom_point(alpha = 1 / 6)
norm + geom_point(alpha = 1 / 15)
```



Slika 31: alpha = $\frac{1}{3}$



Slika 32: alpha = $\frac{1}{6}$



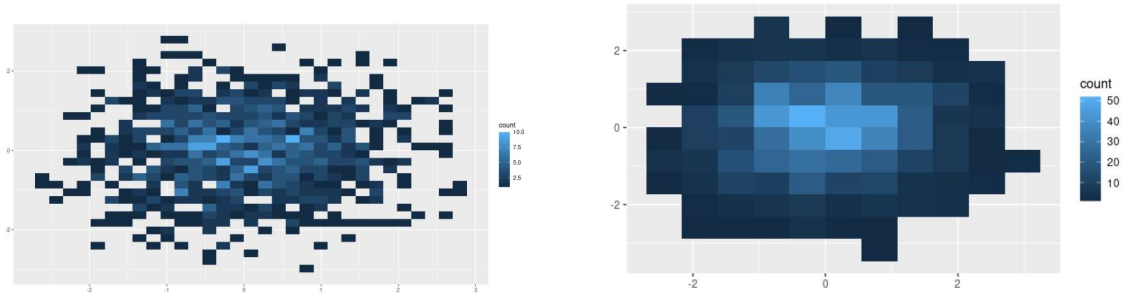
Slika 33: alpha = $\frac{1}{15}$

Ako su podaci barem donekle udaljeni jedni od drugih, preklapanje točaka na grafu možemo riješiti na način da točke nasumično obojimo funkcijom geom_jitter. Ukoliko ovu funkciju koristimo zajedno s manipuliranjem transparentnošću, problem s preklapanjem će se vrlo dobro riješiti. Prema zadanim postavkama, količina nasumičnog obojenja točaka ("titranja" točaka) iznosi 40%. Zadane postavke možemo promijeniti ukoliko mijenjamo širinu i visinu argumenata.

Osim ovog, postoje još dva pristupa ovom problemu.

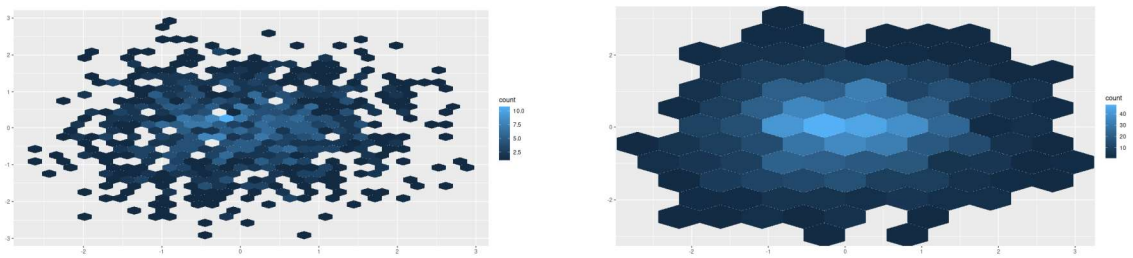
Točke pridružimo poljima i brojimo koliko je točaka u pojedinom polju. Zatim vizualiziramo dobivena prebrojavanja funkcijom geom_bin2d(). Osim vizualizacije ovog broja kvadratima, isto se može dobiti vizualizacijom heksagonima. Preporuka je upravo vizualizacija heksagonima, funkcijom geom_hex() implementiranom u paketu hexbin, obzirom da vizualizacija kvadratima može biti vrlo "nečitka". U nastavku slijede obje vrste prikaza, pri čemu ćemo pokazati i utjecaj broja polja (bin) na sam dijagram.

```
norm + geom_bin2d()
norm + geom_bin2d(bins = 10)
```




Slika 34: Primjer korištenja funkcije geom_bin2d()

```
norm + geom_hex()
norm + geom_hex(bins = 10)
```

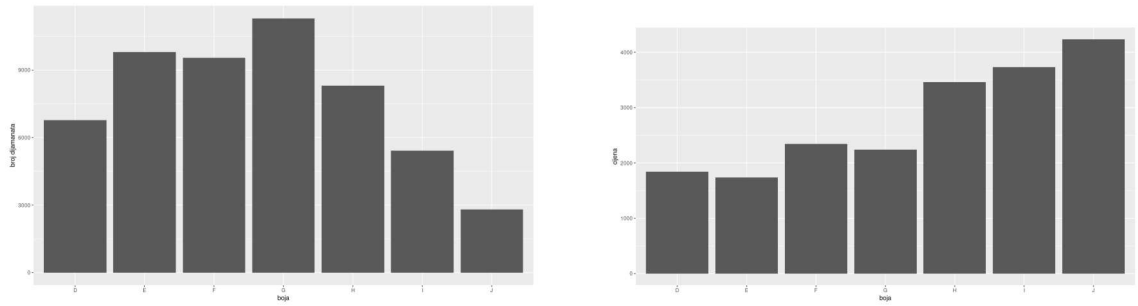


Slika 35: Primjer korištenja funkcije geom_hex()

## 3.10. Statistički sažetci

Već smo vidjeli kako grafički prikazati prebrojavanja podataka, ali ponekad želimo prikazati i razne transformacije i sažetke. Ovdje ćemo suočiti dva grafa izrađena na bazi podataka o dijamantima. Dakle, na prvoj slici vidimo već spomenuto prebrojavanje, dok smo na drugom prikazali medijan cijena dijamanata gledano u odnosu na boju dijamanata.

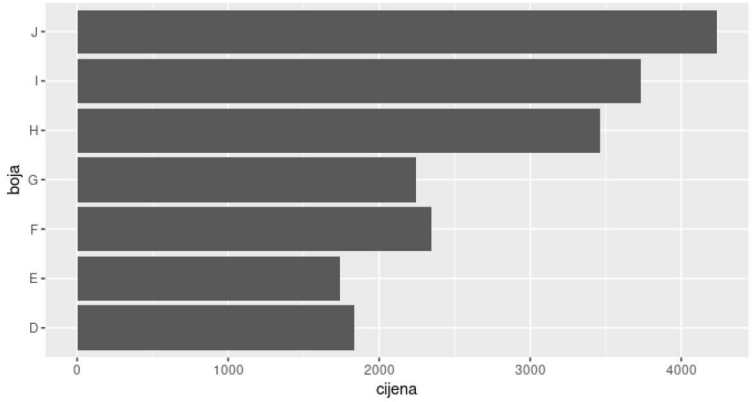
```
ggplot(diamonds, aes(color)) +
  geom_bar() +
  xlab("boja") +
  ylab("broj dijamanata")

ggplot(diamonds, aes(color, price)) +
  geom_bar(stat = "summary_bin", fun = median) +
  xlab("boja") +
  ylab("cijena")
```



Slika 36: Statistički sažetci: prebrojavanje i transformacija

Osim ovakvog prikaza, korisno je stupčaste dijagrame okrenuti horizontalno jer na taj način možemo producirati grafove koji sumiraju različite podatke za iste kategorijalne varijable, a zatim ih postaviti jedan pored drugog i na taj način ih lakše uspoređivati i analizirati podatke. Do takvog grafa dolazimo sljedećim kodom:

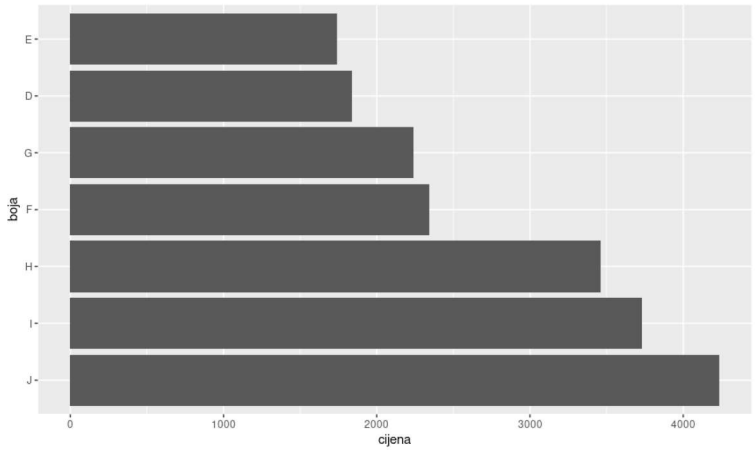
```
ggplot(diamonds, aes(color, price)) +
  geom_bar(stat = "summary_bin", fun = median) +
  xlab("boja") +
  ylab("cijena") +
  coord_flip()
```




Slika 37: Horizontalni stupčasti dijagram

Osim toga, od koristi bi bilo i poredati stupce po veličini. To činimo pomoću funkcije fct_reorder() iz biblioteke forcats na sljedeći način.

```
library(forcats)

diamonds %>%
  count(color) %>%
  mutate(name = fct_reorder(color, desc(price))) %>%
  ggplot( aes(x=color, y=price)) +
  geom_bar(stat="summary_bin", fun = median) +
  coord_flip() +
  xlab("cijena") +
  ylab("boja")
```




Slika 38: Mijenjanje poretka stupaca

Ukoliko želimo poredati u padajućem poretku, samo izostavimo desc(). Više o pravom izboru grafa za prikaz podataka čitatelj može pronaći u [1].

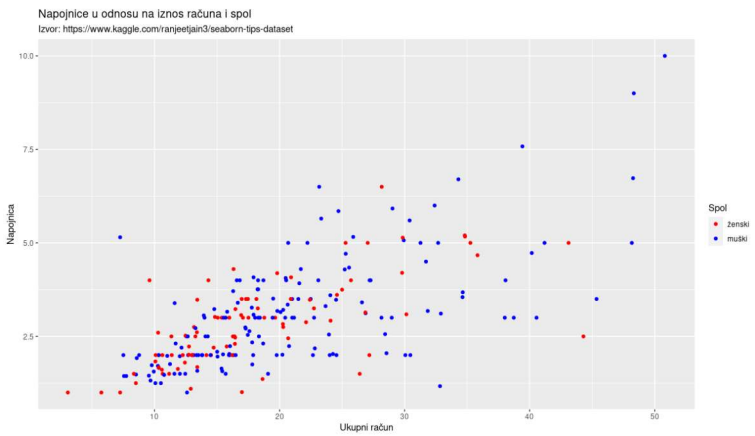
# 4. Dodavanje oznaka i bilježaka

Dodavanje raznih oznaka i bilješki na grafove je često nužno kako bi grafovi bili jasni, a dodajemo ih i sa ciljem pružanja dodatnih informacija. U ovom poglavlju ćemo proći kroz glavne funkcije koje nam omogućuju manipuliranje s ovim elementima grafa.

## 4.1. Naslov grafa i nazivi koordinatnih osi

Počnimo s osnovnim. U prošlim poglavljima je već korišteno preimenovanje koordinatnih osi. To smo učinili kako bismo imali oznake na hrvatskom jeziku. Tu već učavamo koliko su ove oznake važne i elementarne. Ovdje ćemo označavanje koordinatnih osi napraviti na drugačiji način, ali će biti prikazano i dodavanje naslova grafu. Funkcija koja nam sve ovo omogućava je funkcija labs(). Pogledajmo na primjeru kako ona točno funkcionira. Za primjer ćemo koristiti graf koji smo spominjali na početku: iznos računa i napojnica po spolu, ali ovaj put ćemo prilagoditi oznake hrvatskom jeziku.

```
ggplot(tips, aes(x = total_bill, y = tip, colour = sex)) +
  geom_point() +
  labs(
    x = "Ukupni racun",
    y = "Napojnica",
    colour = "Spol",
    labels = c("zenski", "muski"),
    title = "Napojnice u odnosu na iznos racuna i spol",
    subtitle = "Izvor:
    https://www.kaggle.com/ranjeetjain3/seaborn-tips-dataset"
  ) +
  scale_color_manual(labels = c("zenski", "muski"), values = c("red", "blue"))
```




Slika 39: Iznos računa i napojnica po spolu: sa svim oznakama

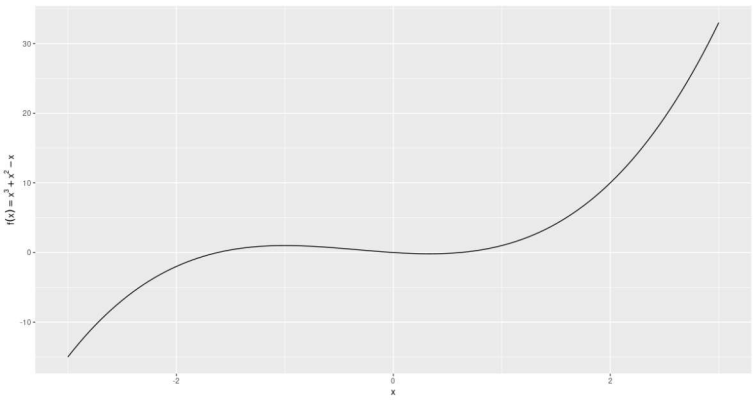
Ovdje smo koristili funkciju scale_color_manual() kojom smo promijenili boju točaka na grafu, ali i oznake vrijednosti na legendi. Vrijednosti koje proslijeđujemo funkciji labs() su uglavnom stringovi (tekstualni podaci). S

\n

označavamo gdje želimo da tekst prijeđe u novi red. Ukoliko želimo matematički izraz dodati na graf, u obliku teksta, to činimo funkcijom quote().

Pogledajmo sad primjer gdje će oznaka y-osi biti matematički izraz, a raspon x-osi ćemo definirati sami.

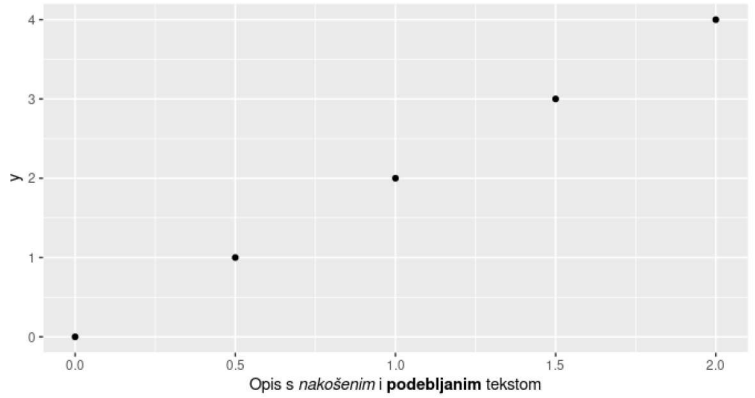
```
vrijednosti <- seq(from = -3, to = 3, by = .05)
df <- data.frame(x = vrijednosti, y = vrijednosti ^ 3
        + vrijednosti ^ 2 - vrijednosti)
ggplot(df, aes(x, y)) +
  geom_path() +
  labs(y = quote(f(x) == x^3 + x^2 - x))
```




Slika 40: Primjer uporabe funkcije quote()

Osim toga, moguće je tekst u opisu slika uređivati na način da ga podebljamo ili nakosimo. Kako bismo to postigli, potrebno je koristiti posebnu temu teksta, a to se radi na sljedeći način:

```
vrijednosti <- seq(from = 0, to = 2, by = .5)
df <- data.frame(x = vrijednosti, y = 2*vrijednosti)
ggplot(df, aes(x, y)) +
  geom_point() +
  labs(x = "Opis s *nakosenim* i **podebljanim** tekstom") +
  theme(axis.title.x = ggtext::element_markdown())
```




Slika 41: Primjer nakošenja i podebljanja teksta u opisu grafa

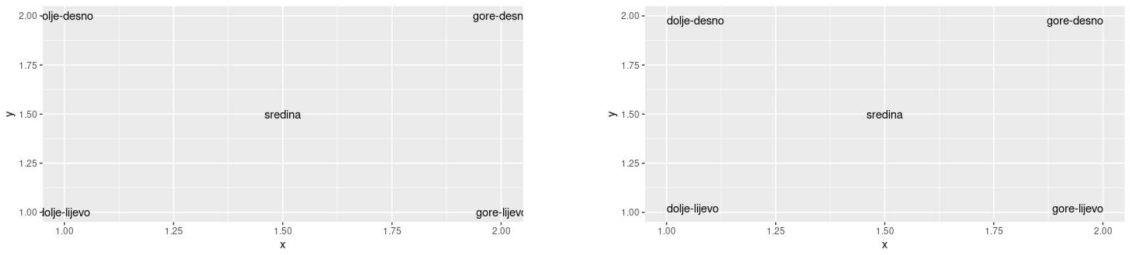
Primjetimo da ono što želimo nakositi ide u jednostruke zvijezdice, dok ono što želimo podebljati ide u dvostruke. Također, za ovakvo uređenje teksta potrebno je instalirati paket "ggtext".

Ukoliko ne želimo prikaz oznake koordinatnih osi, to možemo postići na dva načina i to pomoću funkcije labs(). Možemo joj proslijediti argument x="" ili x = NULL. Optimalnije je koristiti drugu opciju, jer iako x="" ne prikazuje oznaku x-osi, alocira mjesto u memoriji za nju.

## 4.2. Tekstualne oznake

Tekstualne oznake se uglavnom ne dodaju svakoj opservaciji na grafu, ali je vrlo korisno označiti stršeće vrijednosti i ostale važne opservacije. Funkcija koju ćemo koristiti u ovu svrhu je geom_text(). Ona dodaje oznaku točkama na određenom mjestu u koordinatnom sustavu. Ovo je funkcija s najviše atributa jer postoji širok spektar mogućnosti manipulacije tekstom u R-u. U nastavku ćemo se dotaknuti nekih atributa:

- family - ovom atributu proslijeđujemo ime fonta koji želimo upotrijebiti. R upotrebljava klasične fontove, ali može postojati problem u prikazivanju, ovisno o tome na kojem uređaju/operativnom sustavu se grafovi prikazuju. Samo se tri fonta prikazuju pravilno uvijek i svugdje, a to su sans (zadani), serif i mono.
- fontface - određuje značajke teksta, a prima samo tri argumenta - plain (obično, zadano), bold (podebljano) i italic (nakošeno).
- hjust - s ovim atributom određujemo horizontalno poravnanje teksta, a mogućnosti su: "left", "center", "right", "inward", "outward".
- vjust - služi za vertikalno poravnanje teksta, a mogućnosti su: "bottom", "middle", "top", "inward", "outward" U oba poravnanja najkorištenija opcija je "inward". Ona osigurava da tekst ostane unutar grafa, odnosno da ne prelazi granice prikaza.
- size - ovime određujemo veličinu teksta. Veličina se zadaje u milimetrima.
- angle - određuje kut pod kojim se tekst prikazuje (u stupnjevima).




Slika 42: Poravnanje bez i s korištenjem "inward" atributa

Osim navedenih atributa, funkcija geom_text() ima još tri atributa koja je važno koristiti kako bismo manipulirali označavanjem podataka.

- Kada označavamo točke, ne želimo da se tekst međusobno preklapa ili da se prikazuje preko ostalih elemenata grafa. Za izbjegavanje toga nam služe parametri nudge_x i nudge_y.

- Koristan parametar je check_overlap. Kada zadamo check_overlap = TRUE, ukoliko postoji preklapanje, uklonit će se. Ovaj parametar je posebno zanimljiv jer, ukoliko poslažemo podatke prema prioritetu, dobit ćemo graf koji obilježava samo važne podatke.

Funkcija geom_label() je varijacija funkcije geom_text(), a razlika leži u tome što ova funkcija tekstu dodaje pozadinu. Ovo je posebno korisno ukoliko oznaku dodajemo grafu koji ima šarenu ili ispunjenu pozadinu na kojoj se tekst bez dodane pozadine ne bi vidio.

Iako se čini da ggplot2 nudi pregršt opcija za upravljanje tekstom i oznakama, ipak postoje stvari koje nisu najbolje riješene, odnosno ne mogu se kontrolirati:

- Tekst ne utječe na raspon koordinatnih osi. To je zato što se tekstu prosljeđuje apsolutna veličina (npr 25mm), a ne npr u postotku. Zato, kako bismo dobili vizualno privlačan graf s vidljivim oznakama, često je umjesto modificiranja teksta potrebno funkcijama xlim() i ylim() promijeniti raspon x i y-osi.

- Problem preklapanja teksta također nije najbolje riješen. U velikim grafovima (s puno podataka) uglavnom će uvijek doći do preklapanja. Već spomenuta opcija za rješavanje ove situacije, check_overlap = TRUE, je korisna i učinkovita, ali nam ne ostavlja puno autonomije oko toga koje oznake će ostati, odnosno koje će biti uklonjene. Paket koji je prikladniji za ovakav problem je ggrepel [4] čiji je autor Kamil Slowikowski. Ovdje je posebno zannimljiva funkcija geom_text_repel koja vrlo dobro optimizira pozicioniranje tekstualnih oznaka.

- Spomenut ćemo još problem uklapanja teksta na specifična mjesta koja mi želimo. Paket ggfittext [3] (autor Claus Wilke) nudi puno mogućnosti u ovom kontekstu.

## 4.3. Prilagođene oznake

Osim navedenih načina označavanja, često nam dobro dođu i prilagođene oznake. Ovakvo označavanje se dobiva istim funkcijama iz familije geom koje smo koristili za prikazivanje podataka. Neke od tih funkcija su:

- geom_text() i geom_label() koje smo objasnili u prethodnom potpoglavlju.

- geom_rect() koje označavaju zanimljiva područja grafa koja bi nam mogla biti predmetom interesa. Atributi ove funkcije su xmin, xmax, ymin i ymax.

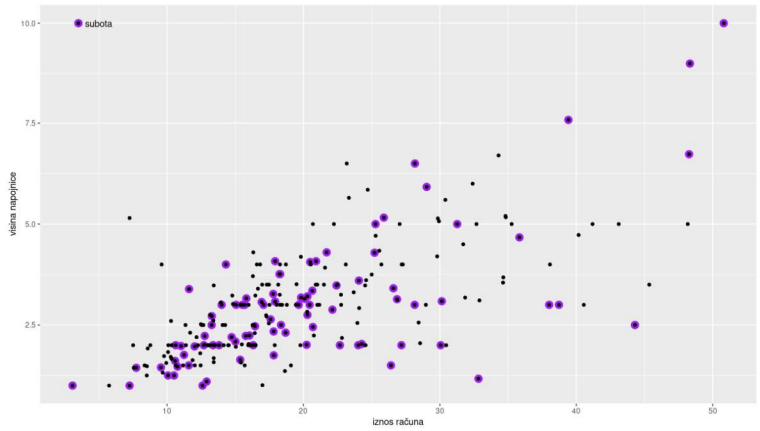
- geom_line(), geom_path() i geom_segment() su funkcije koje koristimo za dodavanje linija. Sve ove funkcije imaju parametar arrow koji nam omogućava dodavanje strelice na liniju na grafu. strelice možemo dobiti i funkcijom arrow() koja ima parametre angle (kut strelica, manji broj producira uži vršak), length (duljina), ends (kraj linije na kojem se crta strelica - početak, kraj ili obje strane) i type (otvoren ili ispunjen vršak strelice).

- geom_vline, geom_hline i geom_abline omogućavaju dodavanje referentnih linija na graf.

Oznake možemo stavljati u prvi plan te s parametrom alpha regulirati vidljivost odnosno transparentnost, ili možemo jednostavno staviti u pozadinu. U nastavku ćemo promotriti nekoliko vrlo korisnih i primjenjivih slučajeva označavanja.

Vraćamo se na bazu podataka tips i graf na kojem prikazujemo odnos visine računa i napojnica. Zamislimo sada da nas eksplicitno zanimaju napojnice koje su ostavljene subotom. To ćemo prikazati na način da podatke koji se vežu uz subotu označimo drugačijom bojom i povećamo veličinu točaka. Osim toga, moramo dodati i oznaku kako bismo znali što naglašeni podaci predstavljaju.

```
ggplot(tips, aes(total_bill, tip)) +
  geom_point(
    data = filter(tips, day == "Sat"),
    colour = "purple",
    size = 4
  ) +
  geom_point() +
  annotate(geom = "point", x = 3.5, y = 10, colour = "purple", size = 4) +
  annotate(geom = "point", x = 3.5, y = 10) +
  annotate(geom = "text", x = 4, y = 10, label = "subota", hjust = "left") +
  xlab("iznos racuna") +
  ylab("visina napojnice")
```



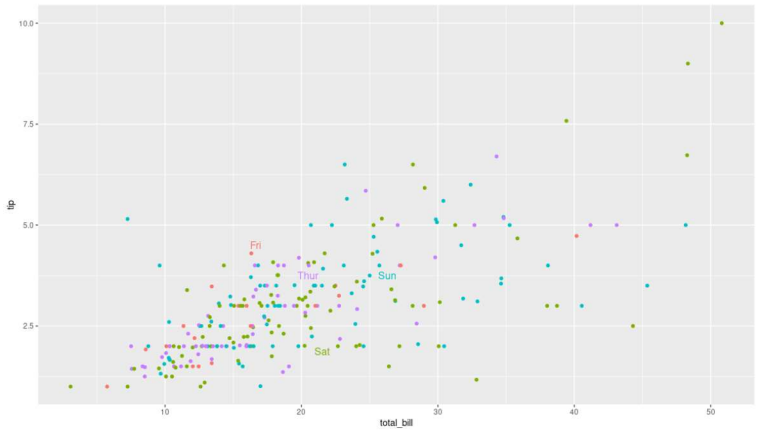

Slika 43: Primjer korištenja prilagođenih oznaka

Dakle, s funkcijom filter() smo regulirali koju kategoriju varijable želimo prikazati, dok smo s annotate() dodali "legendu" na graf. Osim ovakvog dodavanja oznake, u slučaju da su podaci na grafu grupirani, ima smisla koristiti strelice ili linije koje upućuju na naznačenu kategoriju, a to se postiže funkcijama geom_curve() i geom_segment(), respektivno.

## 4.4. Izravno označavanje

U prethodnom poglavlju smo vidjeli primjer izravnog označavanja - postavljanja oznake na sam graf umjesto dodavanja legende sa strane. Ovo je dobro koristiti jer je na taj način lakše tumačiti skicirani graf. Naime, oznake su bliže podacima, a legenda ne zauzima prostor čime dobivamo veće, jasnije grafove. Za korištenje ovih mogućnosti potrebno je učitavanje biblioteke "directlabels". Slijedi primjer.

```
library(directlabels)
ggplot(tips, aes(total_bill, tip, colour = day)) +
  geom_point(show.legend = FALSE) +
  directlabels::geom_dl(aes(label = day), method = "smart.grid")
```




Slika 44: Primjer korištenja izravnog označavanja

Osim ove opcije, zanimljive opcija označavanja se nalaze u biblioteci "ggforce". Od funkcija izdvajamo geom_mark_ellipse() pomoću koje elipsom označavamo grupe podataka te facet_zoom() pomoću koje grafu dodajemo uvećani prikaz dijela na kojem se nalazi kategorija varijable koja je predmet našeg interesa.

# 5. Slaganje grafova

U ovom poglavlju ćemo proučiti kako poslagati grafove na način koji želimo. To se može učiniti i vanjskim programima međutim, ide se prema tome da se sve obavlja u R-u u svrhu optimizacije vremena i zbog preglednosti. Različita slaganja su nam često potrebna kako bismo iz brojeva i podataka dobili i ispričali cjelovitu priču. Razmotrit ćemo slaganje grafova, jedan pored drugog, bez preklapanja i slaganje grafova jedan na drugi.

## 5.1. Slaganje grafova jednog pored drugog

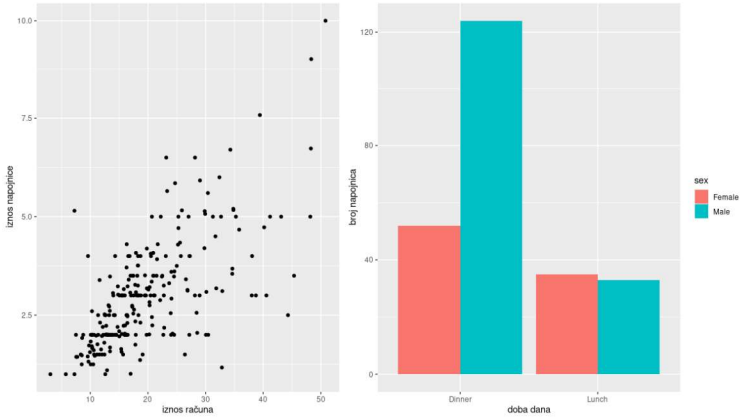
Kako bismo postigli ovakvo slaganje, koristit ćemo biblioteku patchwork (zakrpa) koji će u ovom slučaju proširiti funkciju operatora +. Aranžiranje grafova će na ovaj način biti vrlo jednostavno. Izrađivat ćemo vizualizacije zasebno i jednostavno ih postaviti zajedno pomoću operatora +. Pogledajmo to na primjeru baze s početka rada: "tips". Koristit ćemo sljedeće grafove:

```
graf1 <- ggplot(tips) +
  geom_point(aes(x = total_bill, y = tip)) +
  labs(x = "iznos racuna", y = "iznos napojnice")

graf2 <- ggplot(tips) +
  geom_bar(aes(x = time, fill = sex), position = "dodge") +
```

```
  labs(x = "doba dana", y = "broj napojnica")

graf3 <- ggplot(tips) +
  geom_density(aes(x = tip, fill = sex), colour = NA) +
  facet_grid(rows = vars(sex)) +
  labs(x = "napojnica", y = "gustoca")

graf4 <- ggplot(tips) +
  stat_summary(aes(x = sex, y = tip, fill = sex), geom = "col",
  fun.data = mean_se) +
  stat_summary(aes(x = sex, y = tip), geom = "errorbar",
  fun.data = mean_se, width = 0.5) +
  labs(x = "spol", y = "napojnica")
```

Pogledajmo sada kako radi operator +.

```
library(patchwork)

graf1+graf2
```

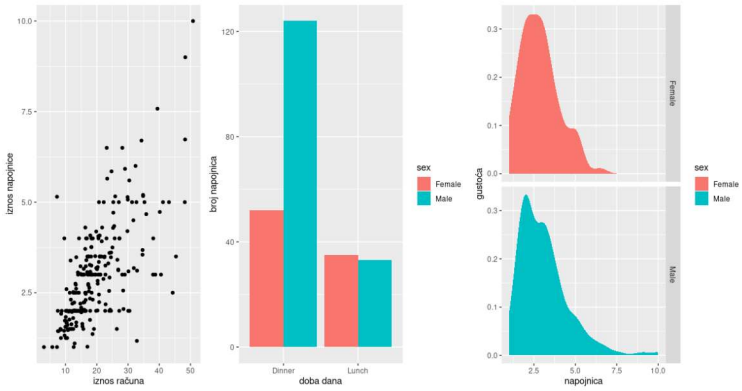



Slika 45: Korištenje operatora + iz biblioteke patchwork

Međutim, ovaj operator ne specificira raspored. Mi proslijeđujemo samo grafove za koje želimo da se prikažu zajedno, a algoritam sam određuje raspored prikaza. Tako će se tri grafa prikazati u jednom redu, a 4 grafa će rasporediti u dva reda po dva grafa. Međutim, i ovdje je zadržana autonomija korisnika te se on može odlučiti za drugačiji razmještaj. Pogledajmo primjere:

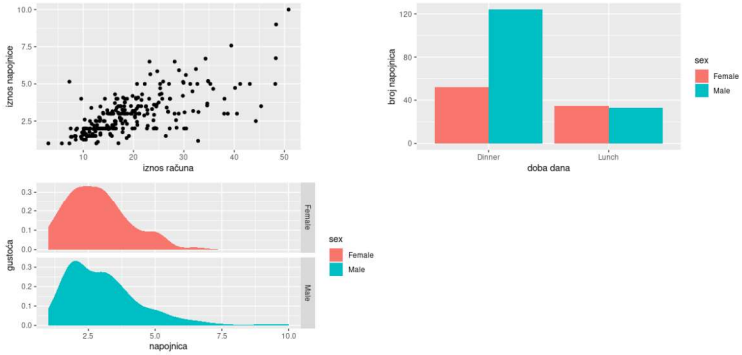
Najizravniji način za manipuliranje rasporedom grafova je funkcija plot_layout() kojoj proslijeđujemo broj redaka i(li) stupaca u koje želimo rasporediti grafove.

```
graf1 + graf2 + graf3
```




Slika 46: Algoritam grafove raspoređuje u jedan redak

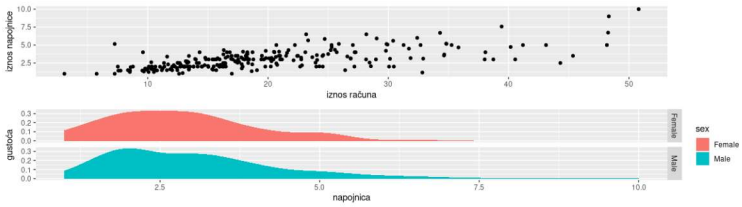
```
graf1 + graf2 + graf3 + plot_layout(nrow = 2) #za broj stupaca upisujemo ncol
```




Slika 47: Korištenje operatora + iz biblioteke patchwork

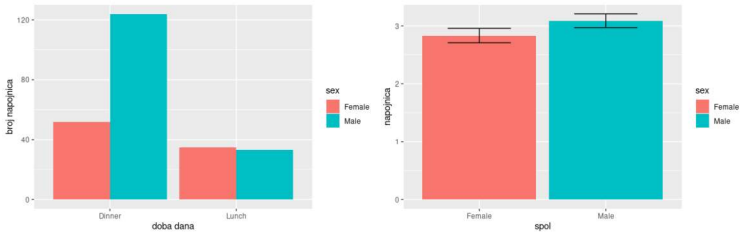
Također, možemo odrediti da nam svi grafovi budu u jednom stupcu, pomoću operatora /. Možemo zahtjevati da svi grafovi budu u jednom retku, a to činimo pomoću operatora —.

```
graf1 / graf3
```




Slika 48: Korištenje operatora /

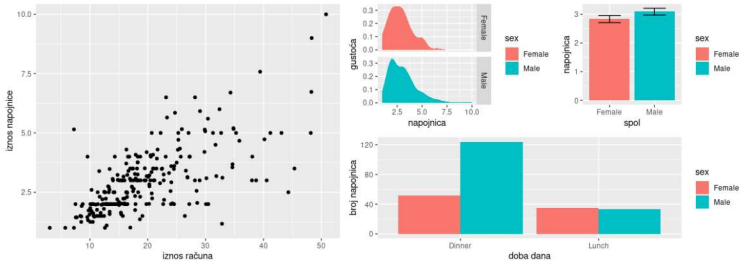
```
graf2 | graf4
```




Slika 49: Korištenje operatora —

Zanimljivo je i da se operatori / i — mogu kombinirati te na takav način dobivamo zaista širok spektar mogućnosti preslagivanja grafova.

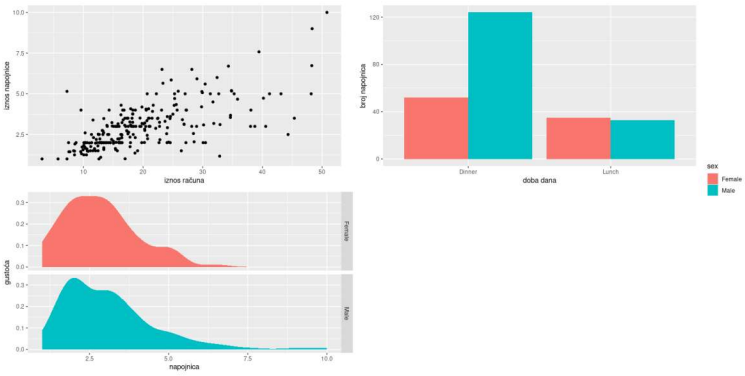
```
graf1 | ( (graf3 | graf4) /  graf2)
```




Slika 50: Kombiniranje operatora / i —

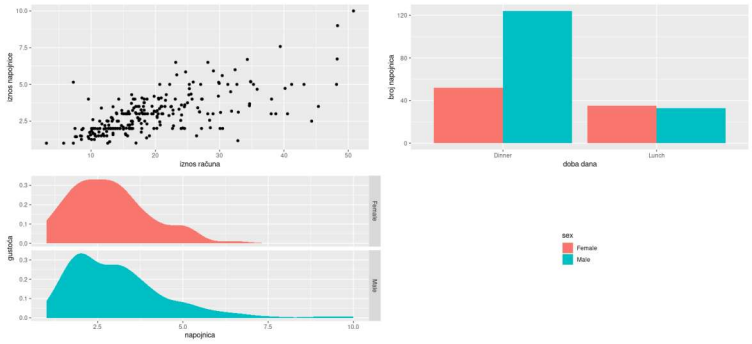
Još jedno izrazito korisno svojstvo kod slaganja grafova je to što nam R omogućava da na ovakvom kolažu grafova koristimo zajedničku legendu. To možemo učiniti na dva načina. Legendu možemo ostaviti sa strane, ali joj možemo dati i poseban prostor, ekvivalentan onom kojeg imaju grafovi. Ukoliko se odlučimo za drugu opciju, koristimo funkciju guide_area().

```
graf1 + graf2 + graf3  + plot_layout(ncol = 2, guides = "collect")
```




Slika 51: Dodavanje zajedničke legende postrani

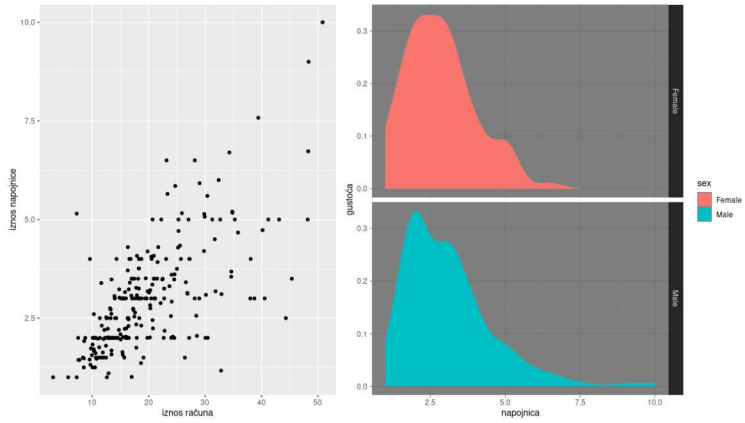
```
graf1 + graf2 + graf3 + guide_area() + plot_layout(ncol = 2, guides = "collect")
```




Slika 52: Dodavanje zajedničke legende među grafove

Jedna od značajki biblioteke patchwork je ta da se grafovi mogu mijenjati i nakon što ih poslažemo zajedno. To se može učiniti s indeksiranjem pomoću znakova [[]]. Pogledajmo na primjeru.

```
graf13 <- graf1 + graf3
graf13[[2]] <- graf13[[2]] + theme_dark()
graf13
```

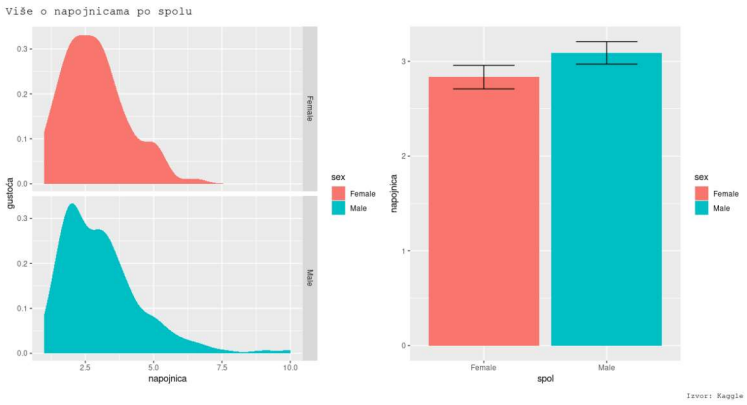



Slika 53: Promjena pozadine nakon što su grafovi posloženi

Primjetimo da smo u ovom primjeru pomoću zagrada [[]] dohvaćali element velikog grafa - podgraf pa ga zatim modificirali. Ukoliko želimo modificirati sve podgrafove nekog grafa, to ćemo učiniti operatorom &. Dakle, umjesto + theme_dark() pisat ćemo & theme_dark() i pozadina će se promijeniti na svim grafovima. Na isti način, korištenjem operatora &, moguće je i prilagoditi skalu y-osi na način da bude jednaka na svim grafovima. To činimo pomoću funkcije scale_y_continuous() kojoj proslijeđujemo parametar limits.

Ovdje ćemo se još dotaknuti dodavanja naslova ovakvom grafu sačinjenom od podgrafova te dodavanja fusnote i mijenjanja fonta slova navedene notacije. Sve navedeno ćemo najbolje proučiti kroz primjer.

```
graf34 <- graf3 + graf4 +
  plot_annotation(
  title = "Vise o napojnicama po spolu",
  caption = "Izvor: Kaggle",
  theme = theme_linedraw(base_family = "mono")
)
graf34
```

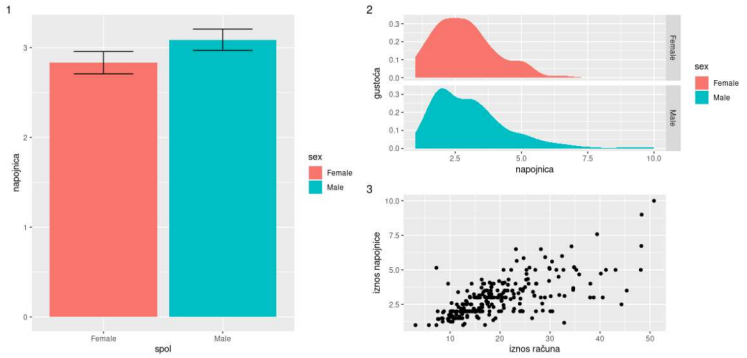



Slika 54: Promjena pozadine nakon što su grafovi posloženi

U gore navedenom primjeru bi čitatelju u ovom trenutku sve trebalo biti jasno i razumljivo.

Posljednja notacija koje ćemo se dotaknuti je posebno zanimljiva pri pisanju znanstvenih radova, a radi se o označavanju svakog podgrafa (na taj način se kasnije lakše referirati na njih). Pokažimo na primjeru kako to postići.

```
graf431 <- graf4 | (graf3 / graf1)
graf431
graf431 + plot_annotation(tag_levels = "1")
```



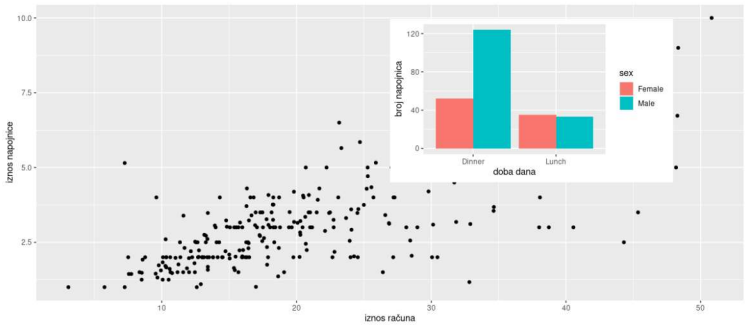

Slika 55: Numeriranje podgrafova

Ovdje je umjesto 1, moguće dodati i "I", "i", "a"...

## 5.2. Lijepljenje grafova jedan preko drugog

Jedna od zanimljivosti biblioteke patchwork je ta što omogućava lijepljenje grafova jedan na drugi. Podgraf lijepimo na veći graf na način da ga pozicioniramo funkcijom inset_element() kojoj prosljeđujemo parametre left, right, top i botom. Ovi parametri primaju vrijednosti od 0 do 1 izražene u jedinicama npc. Pogledajmo primjer.

```
graf1 + inset_element(graf2, left = 0.5, bottom = 0.4, right = 0.9, top = 0.95)
```




Slika 56: Pozicioniranje grafova

Osim na ovaj način, podgraf možemo pozicionirati tako da mu proslijedimo fiksno smještanje, unutar grafa, tako da vrijednosti zadamo u jedinicama koje mi želimo. Tako, ukoliko zadamo npr right = unit(1, "npc") - unit(15, "mm"), odredili smo da će podgraf od desnog ruba grafa biti udaljen 15 milimetara. Osim ovakvih podgrafova, možemo dodavati i elemente koji se sastoje od skupa više grafova koje smo prikazali u prošlom potpoglavlju. Također, moguće je korištenje i operatora & koji funkcionira na isti način kao u prošlom potpoglavlju.

# 6. Skala

## 6.1. Uvod

Ova familija funkcija u paketu ggplot2 pretvara podatke u vizuale. Na ovaj način upravljamo veličinom, bojom, pozicijom i oblikom podataka. Također, ovdje se nalaze i alati kojima kreiramo one dijelove grafa koji nam služe za njegovo razumijevanje, a to su koordinatne osi i legende. Ova familija se nadograđuje na slojeve obrađene u prethodnom poglavlju. Možemo primjetiti da se grafovi mogu izgraditi bez ove familije jer ggplot2 ove elemente automatski producira, ali poznavanje ovih funkcija nam omogućava veću kontrolu nad vizualizacijom podataka i uređenjem grafova.

Ovu familiju funkcija možemo podijeliti u tri skupine, a svaka skupina će biti analizirana u zasebnom potpoglavlju u nastavku. One su:

- pozicija skala i koordinatnih osi
- boja skala i legendi
- ostale estetske komponente

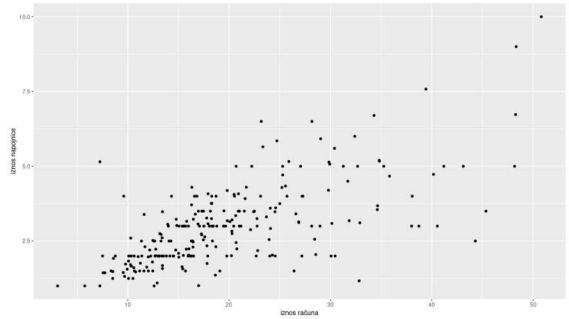
## 6.2. Pozicija skala i koordinatnih osi

Svaki graf ima dvije skale koje određuju koordinatni sustav grafa. Jedna koja se nalazi na apscisi i jedna koje se nalazi na ordinati. Uobičajeno je da korisnik određuje koje varijable se prikazuju u ovakvom koordinatnom sustavu, ali to nije uvijek slučaj. Primjer koji svjedoči drugom slučaju je prikazivanje histograma funkcijom geom_histogram().

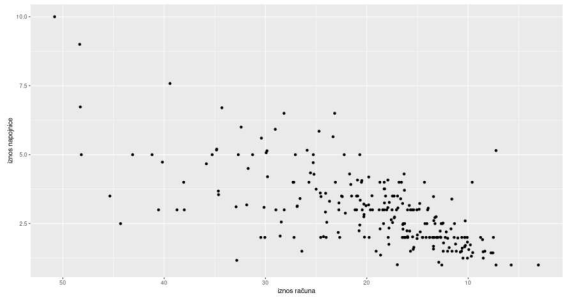
## 6.3. Numerička skala

Najčešće se koriste numeričke neprekidne skale koje su već zadane, a određene su funkcijama scale_x_continuous() i scale_y_continuous(). U najjednostavnijem slučaju one linearno pridružuju podatke koordinatama, odnosno poziciji na grafu. Međutim, postoje i druge opcije ove funkcije koje se koriste kako bi brzo i efikasno prikazale različite transformacije. Primjer funkcija koje daju ovakvo skaliranje su scale_x_log10() i scale_x_reverse(), a njihovu primjenu ćemo proučiti u nastavku nad podatcima iz već poznate baze podataka o napojnicama.

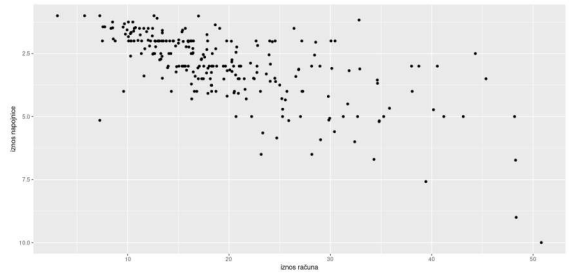
```
baza <- ggplot(tips) +
  geom_point(aes(x = total_bill, y = tip)) +
  labs(x = "iznos racuna", y = "iznos napojnice")

baza
baza + scale_x_reverse()
baza + scale_y_reverse()
baza + scale_x_log10()
```



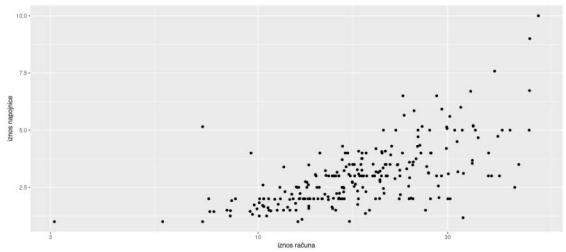
(a) Zadana skala



(b) Obrnuta skala na x osi



(c) Obrnuta skala na y osi



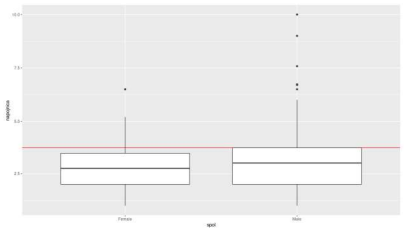
(d) Logaritam po bazi 10 na y osi

Slika 57: Različito skaliranje koordinatnih osi

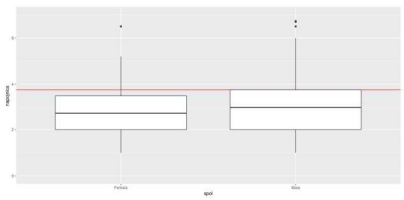
## 6.4. Ograničenja

Prema zadanim postavkama ggplot2 paketa sve vrijednosti koje se nalaze izvan raspona koordinatnih osi (odnosno skale) se konvertiraju u NA vrijednosti (not available - vrijednosti koje nedostaju). Zato je, pri uvećanju grafa, važno biti oprezan kako ne bismo kontaminirali vrijednosti prikazane na grafu. U tu svrhu, preporuka je koristiti funkciju coord_cartesian() i njene argumente xlim i ylim. Primjer korištenja te funkcije te narušavanje grafa u slučaju da graf ne uvećavamo pažljivo promotrit ćemo na sljedećem primjeru.

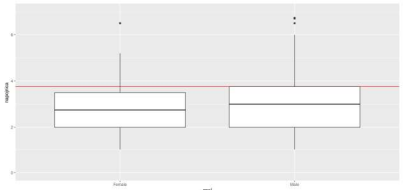
```
baza <- ggplot(tips, aes(sex, tip)) +
  geom_hline(yintercept = 3.76, colour = "red") +
  geom_boxplot() +
  labs(x ="spol",  y = "napojnica")

baza
baza + coord_cartesian(ylim = c(0, 7))
baza + ylim(0, 7)
```



Slika 58: Originalni kutijasti dijagrami



Slika 59: Uvećano



Slika 60: Uvećano na pogrešan način

Primjetimo: u trećem slučaju nije korištena funkcija coord_cartesian, već samo y_lim() zbog čega su vrijednosti izvan danog ograničenja poprimile NA vrijednosti te se vrijednost gornjeg kvartila promijenila. Zbog toga je važno biti posebno oprezan pri ograničavanju koordinatnih osi.

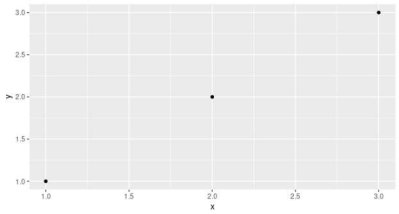
Ovo se može izbjeći na način da se umjesto zadanog svojstva da funkcija automatski vrijednosti koje "ne stanu" u koordinatni sustav pretvori u NA, postavi opcija koja vrijednosti sabije na način da stanu u zadani raspon. To se čini na način da se unutar funkcije scale_fill_gradient doda argument oob = scales::squish() (oob znači out of bounds - izvan granica).

Nasuprot tome, možemo imati potrebu za proširivanjem koordinatnih osi kako bismo uskladili više grafova ili jednostavno, kako bismo os prilagodili prirodi podataka koje obrađujemo (npr. ukoliko vizualiziramo postotke, prirodno je da je raspon koordinatne osi 0-100). Osim toga, važno je uskladiti raspone koordinatnih osi ukoliko koristimo bojanje ili skaliranje koje ovisi o veličinama kako bi oznake bile dosljedne.

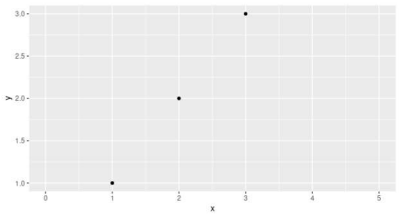
U nastavku dajemo bazični primjer određivanja granica koordinatnih osi gdje ćemo proučiti graf bez ograničenja te prošireni i uvećani graf.

```
baza <- data.frame(x = 1:3, y = 1:3)
graf <- ggplot(baza, aes(x, y)) + geom_point()
```

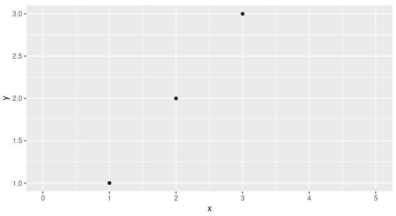
```
graf
graf + scale_x_continuous(limits = c(0, 5))
graf + scale_y_continuous(limits = c(0, 1.25))
```



Slika 61: Graf sa zadanim postavkama



Slika 62: Proširen raspon



Slika 63: Uvećan graf

Još jedno zanimljivo svojstvo je određivanje duljine koordinatnih osi. Osim raspona kojeg zadamo, na grafu se doda još mali odmak. To se čini na način da se u već spomenute funkcije scale_x_continuous i scale_y_continuous doda argument expand = c(0, 0).

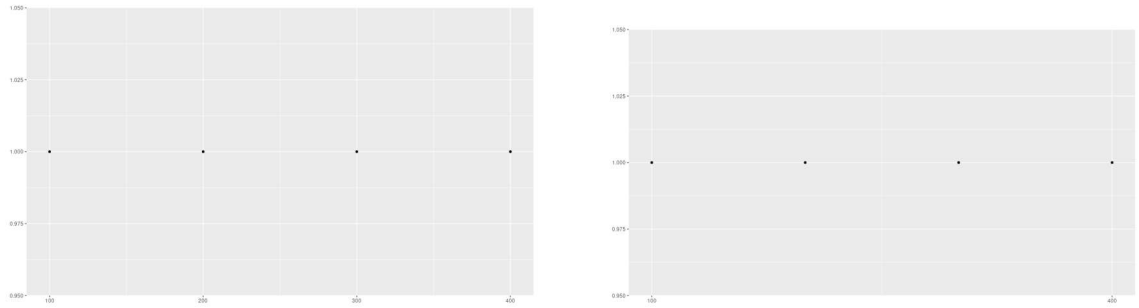
## 6.5. Razmaci

U ggplot2 paketu, moguće je koristiti zadane vrijednosti razmaka, ali ih i ručno odrediti. Potrebno je proslijediti argument koji specificira duljinu skale (vektor duljine dva), a vraćeni rezultat bi trebao biti vektor razmaka. Funkcije razmaka možemo pisati sami, ali možemo pronaći i gotove funkcije u paketu scales [5]. Najkorisnije funkcije u ovom paketu su:

- break_extended() stvara automatske razmake numeričkih osi
- break_log() stvara razmake prikladne za logaritamske osi
- break_pretty() stvara "lijepe" razmake za datum/vrijeme
- break_width() stvara jednake razmake

U nastavku slijedi primjer korištenja razmaka.

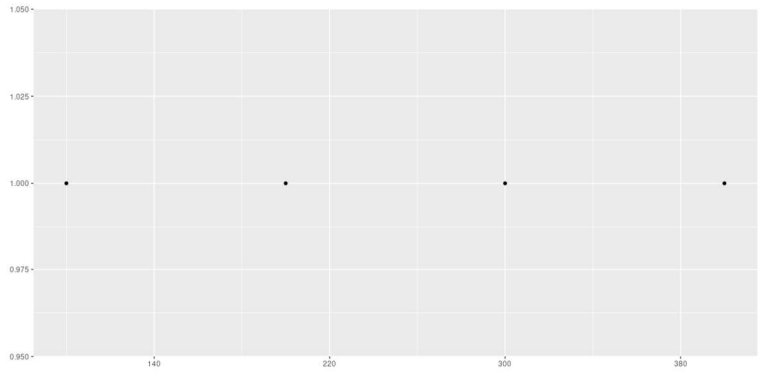
```
baza <- data.frame(
  x_os = c(100, 200, 300, 400),
  y_os = c(1, 1, 1, 1)
)

primjer <- ggplot(baza, aes(x_os, y_os)) +
  geom_point() +
  labs(x = NULL, y = NULL)

primjer
primjer + scale_x_continuous(breaks = scales::breaks_extended(n = 2))
```



Slika 64: Primjer korištenja razmaka na grafu

Razmake je moguće koristiti i na način da specificiramo njihovu širinu te odredimo vrijednost od koje počinju. Pogledajmo primjer.

```
primjer + scale_x_continuous(breaks = scales::breaks_width(80, offset = 20))
```




Slika 65: Primjer korištenja zadane širine razmaka i pomicanja granica

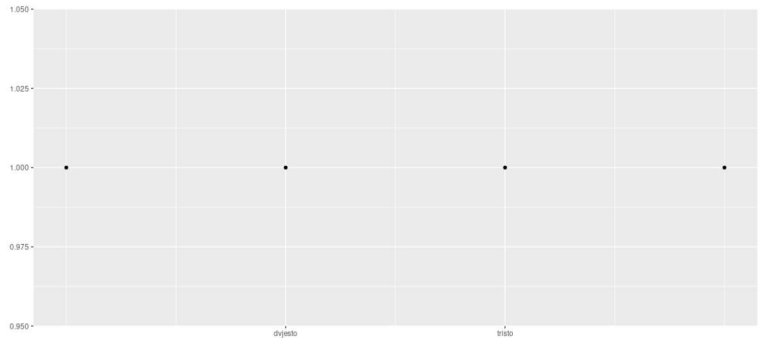
Granice smo pomakli s argumentom offset. U slučaju da ih želimo pomaknuti ulijevo, ispred broja za koji želimo pomjeriti stavljamo -. Ukoliko uopće ne želimo razmake, definiramo argument breaks = NULL.

Spomenut ćemo još i mogućnost dodavanja manjih razmaka koje dobivamo argumentom minor= mb, a posebno su korisne za korištenje kod logaritamske skale jer jasno upućuju na nelinearnost iste.

## 6.6. Označavanje

Svaki razmak je na neki način definiran oznakom. To se može promijeniti na način da samo neki razmaci budu označeni. Pokažimo kako na sljedećem primjeru.

```
primjer
primjer + scale_x_continuous(breaks = c(200, 300),
labels = c("dvjesto", "tristo"))
```



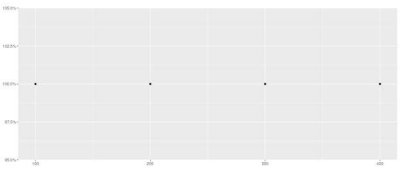

Slika 66: Primjer označavanja razmaka

I u ovom slučaju paket scales omogućava niz alata koji se sam brine za određene elemente grafa, ovdje automatski stvara oznake grafa. Najkorisnije funkcije navodimo u nastavku:

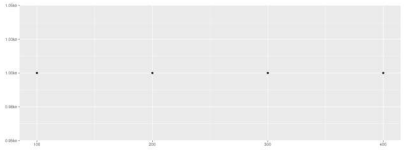
- scales::label_bytes() formatira brojeve kao kilobajte, terabajte itd
- scales::label_comma() formatira brojeve kao decimalne, s decimalnim zarezima
- scales::label_percent() formatira brojeve kao postotke
- scales::label_pvalue() formatira brojeve kao p-vrijednosti. Npr ¡.05, ¡.01, .34 itd.
- scales::label_dollar() prikazuje brojeve u formatu valute (ne nužno dolara)

Pogledajmo primjenu nekih od ovih funkcija.

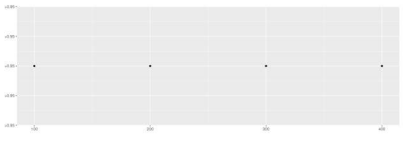
```
primjer + scale_y_continuous(labels = scales::label_percent())
primjer + scale_y_continuous(labels = scales::label_dollar(prefix = "", suffix = "kn")
primjer + scale_y_continuous(labels = scales::label_pvalue(accuracy = 0.05))
```



Slika 67: Format postotka



Slika 68: Format valute



Slika 69: p-vrijednost

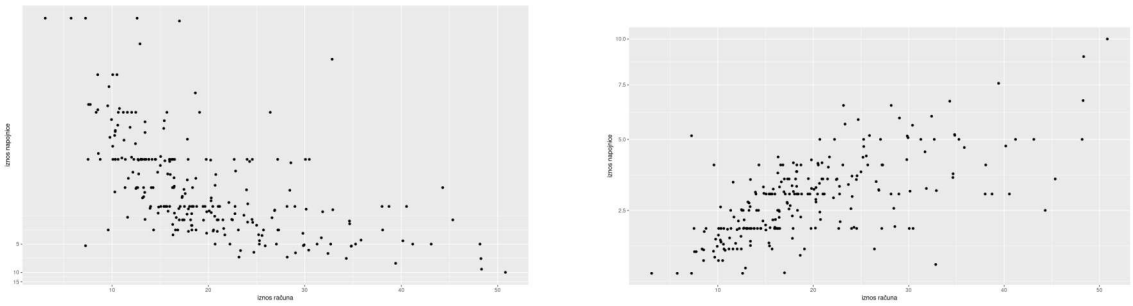
Ukoliko želimo ukloniti sve oznake, koristimo argument labels = NULL.

## 6.7. Transformacije

Neprekidne varijable se iz baza na graf preslikavaju linearno. Međutim, to se može promijeniti, a na primjeru ćemo vidjeti kako

```
ggplot(tips, aes(total_bill, tip)) +
  geom_point() +
  scale_y_continuous(trans = "reciprocal") +
  labs(x = "iznos računa", y = "iznos napojnice")
```

```
ggplot(tips, aes(total_bill, tip)) +
  geom_point() +
  scale_y_continuous(trans = "sqrt") +
  labs(x = "iznos računa", y = "iznos napojnice")
```



Slika 70: Primjer korištenja transformacije

Transformacija je definirana formulom, inverzom i oznakom. Paket scales nudi veliki izbor gotovih transformacija, ali korisniku je omogućeno kreiranje vlastite transformacije, pomoću funkcije scales::trans_new().

Gornje transformacije su se mogle dobiti i na sljedeći način:

```
ggplot(tips, aes(total_bill, tip)) +
  geom_point() +
  scale_y_reverse()

ggplot(tips. aes(total_bill, tip)) +
    geom_point() +
    scale_y_sqrt()
```

Ovo je učinjeno kako bi se korisniku maksimalno olakšalo korištenje najčešćih transformacija.

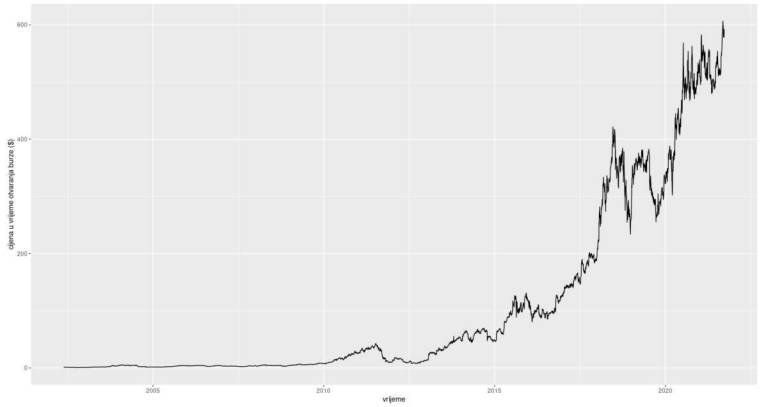
## 6.8. Vremenski nizovi

Vremenski nizovi su vrsta podataka kojoj ćemo pridodati posebnu pažnju pri vizualizaciji. Za početak, važno je da su ovakvi podaci u formatu "Date" (za datume), "POSIXct" (za datum i vrijeme) ili hms za doba dana. Ukoliko vremenski podatak nije u jednom od danih formata, potrebno ga je konvertirati u jedan od navedenih formata. Kada su podaci u jednom od navedenih formata, ggplot2 koristi funkcije scale_x_date() i scale_x_datetime() za podatke u formatu datuma i datum-vrijeme, respektivno. Odgovarajuće skale za druge atribute slijede uobičajena pravila označavanja spomenuta kod neprekidnih podataka. Ono što je ovdje karakteristično je postojanje argumenata koji omogućuju rad s vremenskim jedinicama. Primjerice, date_breaks = "4 weeks" naznačuje svaka 4 tjedna. U ovom poglavlju ćemo promotriti osnovno crtanje vremenskih nizova, ali i naprednu vizualizaciju pomoću paketa dygraphs.

U ovu svrhu, promotrimo bazu podataka koja sadrži podatke o povijesnim cijenama dionica Netflix-a. Baza podataka je dostupna na stranici Kaggel-a pod nazivom "Netflix_stock_history.csv" [6]. Za početak je potrebno transformirati varijablu "Date" u vre-

mensku varijablu, a zatim ćemo prikazati varijablu "Open" koja označava cijenu dionice u dolarima u vrijeme otvaranja burze na određeni dan. Promotrimo kod.

```
netflix <- read.csv("Netflix_stock_history.csv")
netflix$Date <- as.Date(netflix$Date, format= "%Y-%m-%d")

ggplot(netflix, aes(Date, Open, group = 1)) +
  geom_line() +
  xlab("vrijeme") +
  ylab("cijena u vrijeme otvaranja burze ($)")
```




Slika 71: Osnovna vizualizacija vremenskog niza

Ako bismo, primjerice, htjeli na istom grafu prikazati cijenu u vrijeme otvaranja burze i cijenu u vrijeme zatvaranja kroz vrijeme, te dvije linije će se gotovo preklopiti ukoliko se promatra veliki vremenski raspon jer su razlike u vrijednostima vrlo male. Zbog toga iz takvog grafa nećemo puno toga vidjeti. Bilo bi korisno generirati graf koji možemo povećavati kako bismo lakše očitali određene podatke. Tako dolazimo do paketa dygraphs koji omogućava crtanje interaktivnih grafova. Za promatranu bazu podataka, graf produciran funkcijama iz ovog paketa bi izgledao ovako:

```
library(dygraphs)
library(xts)

cijene <- data.frame(
  vrijeme = netflix$Date,
  otvaranje = netflix$Open,
  zatvaranje = netflix$Close
)

prilagodeno <- xts(x = cijene[,-1], order.by = cijene$vrijeme)
graf <- dygraph(prilagodeno)
graf
```

U slučaju da želimo dodati manje razmake na graf, to radimo argumentom date_minor_breaks. Na primjer, možemo imati mjesečne razmake, a unutar njih i tjedne. Na ovaj način možemo

dobiti neuniformnu mrežu na grafu jer broj dana u mjesecu varira pa udaljenost između linija neće svugdje biti jednaka.

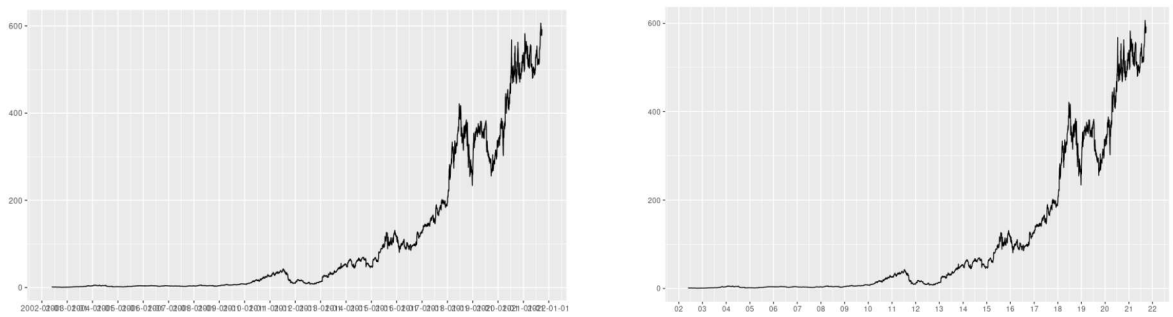
Označavanje datuma kontroliramo argumentom date_labels. Za označavanje koristimo strptime() i format(). Naprimjer, ukoliko želimo pravilno očitati datum 09/30/2021, koristimo string "%m/%d%Y. U nastavku slijedi tablica oznaka za formatiranje datuma.

| Simbol | Značenje |
|---|---|
| **%S** | sekunde (00-59) |
| **%M** | minute (00-59) |
| **%l** | vrijeme, 12-satno (1-12) |
| **%I** | vrijeme, 12-satno (01-12) |
| **%p** | am/pm |
| **%H** | vrijeme, 24-satno (00-23) |
| **%a** | dani u tjednu, skraćeni (Mon-Sun) |
| **%A** | dani u tjednu, puni naziv (Monday-Sunday) |
| **%e** | dan u mjesecu (1-31) |
| **%d** | dan u tjednu (01-31) |
| **%m** | mjesec, numerički(01-12) |
| **%b** | mjesec, skraćeno (Jan-Dec) |
| **%B** | mjesec, puni naziv (January-December) |
| **%y** | godina, bez stoljeća (00-99) |
| **%Y** | godina, sa stoljećem (0000-9999) |

Tablica 6: Simboli za formatiranje datuma

Ovdje ističemo %y. Ovom opcijom se ispisuju samo zadnje dvije znamenke godine, umjesto punog formata. To je korisno kako na grafu ne bi nastala gužva, a sve oznake bi ostale zadržane i jasne. Ovo ćemo oprimjeriti grafovima generiranim nad podacima o cijeni dionica Netflixa.

```
baza <- ggplot(netflix, aes(Date, Open)) +
  geom_line(na.rm = TRUE) +
  labs(x = NULL, y = NULL)

baza + scale_x_date(date_breaks = "1 years")
baza + scale_x_date(date_breaks = "1 years", date_labels = "%y")
```

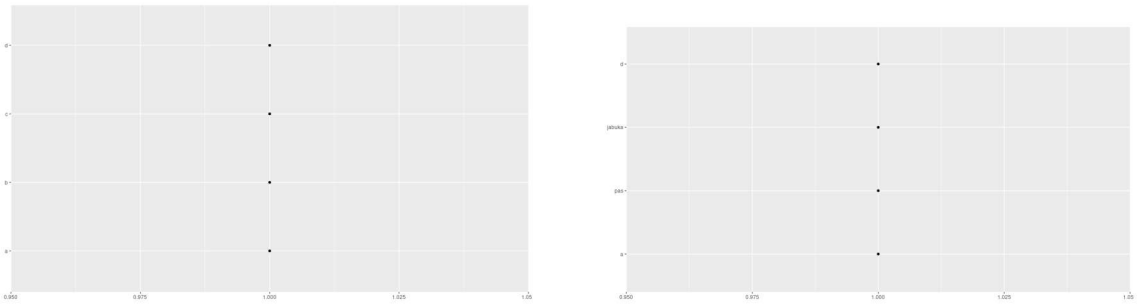


Slika 72: Primjer korištenja formata datuma %y

## 6.9. Označavanje diskretnih varijabli

Diskretne varijable označavamo pomoću funkcija scale_x_discrete() i scale_y_discrete(). ggplot2 upravlja diskretnom skalom na način da svaku kategoriju preslika na cjelobrojnu vrijednost. Također, pri označavanju kategorijalnih podataka postoji mogućnost kreiranja vektora kojim možemo promijeniti oznake povezane s određenim vrijednostima. Na ovaj način možemo promijeniti samo određene oznake, bez promijene redoslijeda ili razmaka. Slijedi primjer.

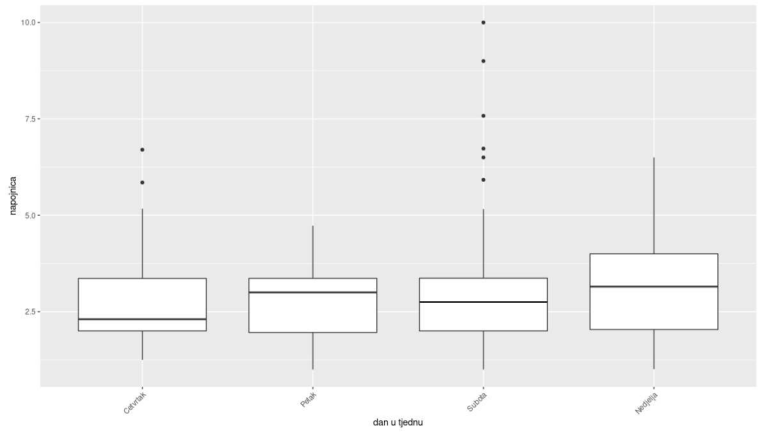
```
baza <- data.frame(
  broj = 1,
  oznake = c("a", "b", "c", "d")
)

graf <- ggplot(baza, aes(broj, oznake)) +
  geom_point() +
  labs(x = NULL, y = NULL)

graf
graf + scale_y_discrete(labels = c(c = "jabuka", b = "pas"))
```



Slika 73: Primjer promijene oznaka

Spomenut ćemo još funkciju guide_axis() koja je korisna pri sprječavanju nastanka preklapanja oznaka na koordinatnim osima. Ukoliko imamo puno oznaka, pomoću ove funkcije ih možemo rasporediti u više redova ili tekstualne oznake ispisati pod određenim kutom te na taj način osigurati preglednost i jasnoću grafa. Pogledajmo primjer korištenja ove funkcije.

```
graf <- ggplot(tips, aes(day, tip)) + geom_boxplot() +
        scale_x_discrete(labels = c(Thur = "Cetvrtak", Fri = "Petak",
        Sat = "Subota", Sun = "Nedjelja")) +
        labs(x = "dan u tjednu", y = "napojnica") +
        guides(x = guide_axis(angle = 45))
```



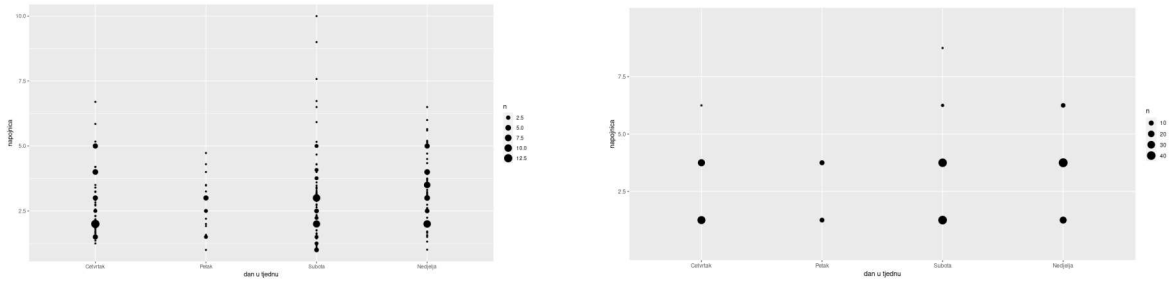

Slika 74: Primjer korištenja funkcije guide_axis()

Ukoliko želimo oznake rasporediti u više redova, funkciji guide_axis() proslijeđujemo argument n.dodge i broj redaka koji želimo.

## 6.10. Grupiranje

Ovdje promatramo kako neprekidnu varijablu podijeliti u određene kategorije. Tako primjerice možemo prikazati broj napojnica određenih iznosa koje su određenim danima ostavljene. To ćemo postići funkcijom geom_count() pri čemu će veličina točke odgovarati broju napojnica. Kao što vidimo na slici 75a, ovakav prikaz je vrlo nepregledan. Zbog toga je bolje neprekidne varijable podijeliti unutar nekoliko kategorija.

```
graf <- ggplot(tips, aes(day, tip)) + geom_count() +
  scale_x_discrete(labels = c(Thur = "Cetvrtak", Fri = "Petak",
  Sat = "Subota", Sun = "Nedjelja")) +
  labs(x = "dan u tjednu", y = "napojnica")
graf
graf + scale_y_binned(n.breaks = 5)
```

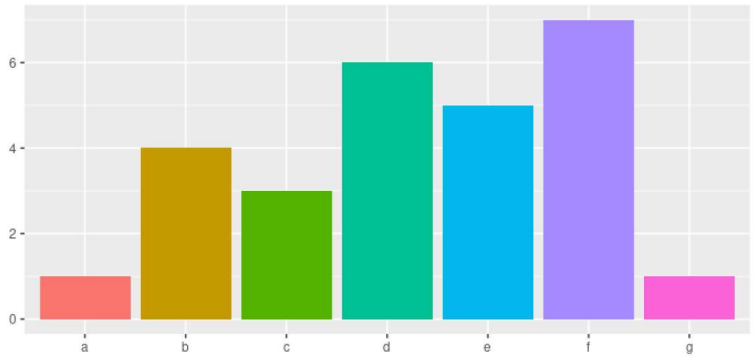


Slika 75: Grupiranje podataka

## 6.11. Bojanje prikaza diskretnih varijabli

Boje su važne pri stvaranju grafova zbog preglednosti samih podataka, ali i zato da graf bude vizualno ugodan. U nastavku ćemo navesti jednostavan primjer dodavanja boja na graf pomoću funkcije scale_fill_discrete().

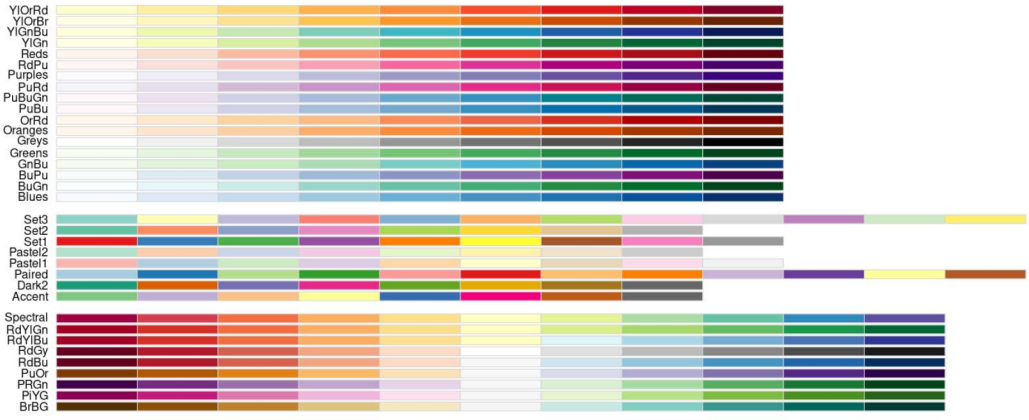
```
baza <- data.frame(x = c("a", "b", "c", "d", "e", "f", "g"),
        y = c(1, 4, 3, 6, 5, 7, 1))
ggplot(baza, aes(x, y, fill = x)) +
  geom_bar(stat = "identity") +
  labs(x = NULL, y = NULL) +
  theme(legend.position = "none") +
  scale_fill_discrete()
```




Slika 76: Korištenje boja na stupčastom dijagramu

Obzirom da je paleta boja prikazana na grafu zadana, ovdje nismo morali dodavati ovu funkciju, a isto bismo postigli koristeći funkciju scale_fill_hue(). Za veći izbor paleta koristimo funkciju scale_colour_brewer, a sve dostupne boje možemo vidjeti sa sljedećom naredbom:

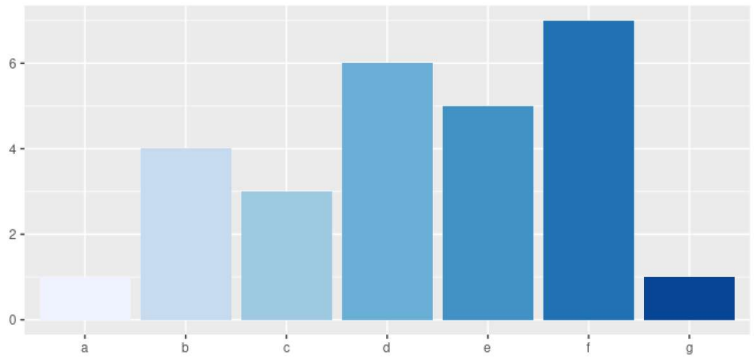
```
RColorBrewer::display.brewer.all()
```




Slika 77: Palete boja dostupne za korištenje s funkcijom scale_color_brewer

Pogledajmo sad primjer korištenja jedne od gore navedenih paleta na našem stupčastom dijagramu.

```
gplot(baza, aes(x, y, fill = x)) +
  geom_bar(stat = "identity") +
  labs(x = NULL, y = NULL) +
  theme(legend.position = "none") +
  scale_fill_brewer("Spectral")
```



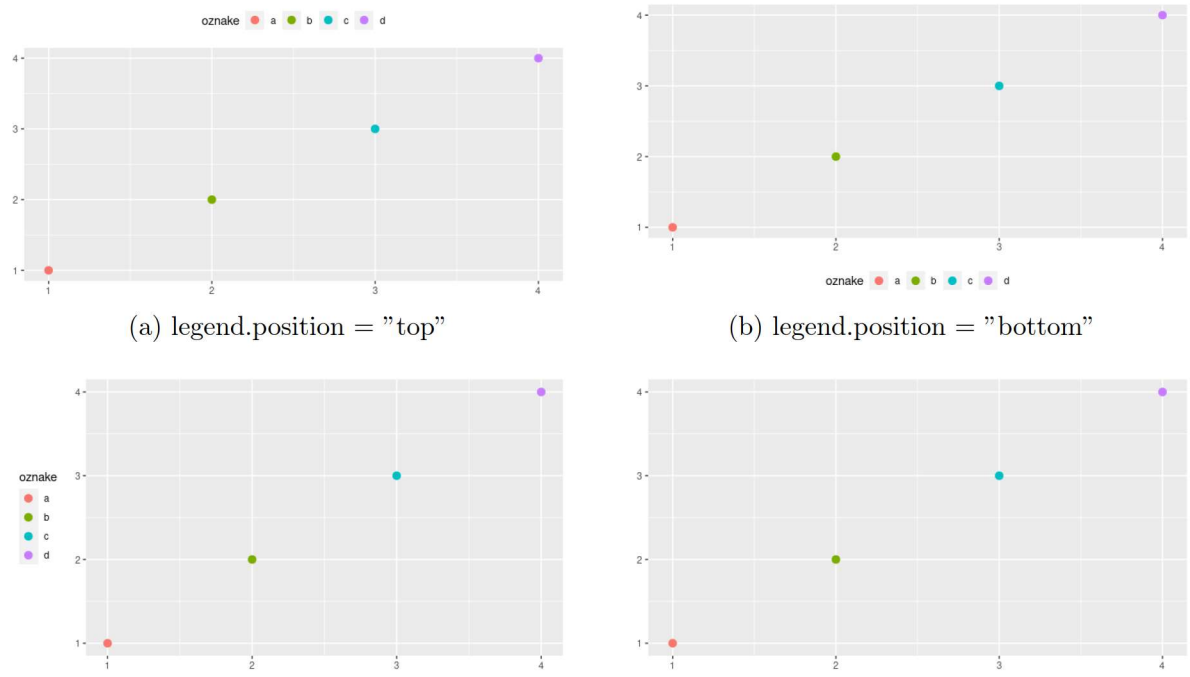

Slika 78: Korištenje drugačije palete na stupčastom dijagramu

Osim za prikaz stupčastih dijagrama, bojanje grafa je moguće koristiti i na drugim vizualizacijama.

## 6.12. Legende

Važan element u tumačenju grafa su legende. Počet ćemo od njihovog najjednostavnijeg atributa, a to je smještanje. Ono se određuje s legend.position, a s vrijednostima left (lijevo), right (desno), top (gore), bottom (dolje) i none (bez) definiramo gdje ju, u odnosu na graf, želimo smjestiti. Na jednostavnom primjeru ćemo pokazati načine smještanja legende.

```
graf + theme(legend.position = "top")
graf + theme(legend.position = "bottom")
graf + theme(legend.position = "left")
graf + theme(legend.position = "none")
```




(a) legend.position = "top"

(b) legend.position = "bottom"

(c) legend.position = "left"

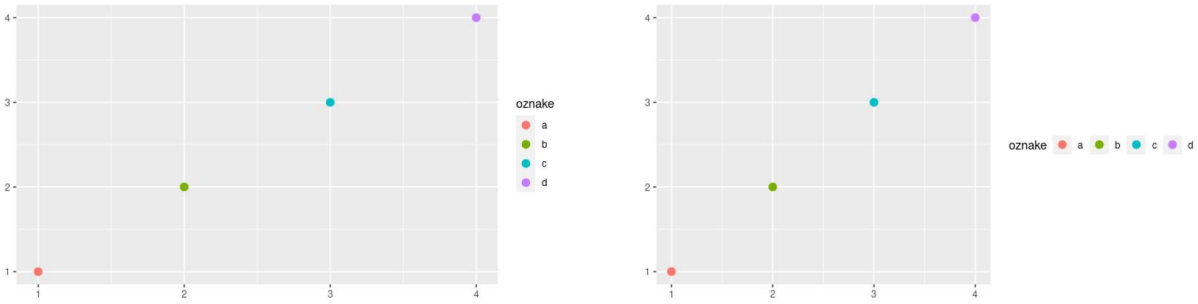
(d) legend.position = "none"

Slika 79: Različito smještanje legende

Osim ovih postavki legende, moguće je i određivanje smjera legende s legend.direction, smještanje više legendi s legend.box te prilagođavanje svake legende unutar zadanog okvira s legend.box.justification.

Na primjeru ćemo pokazati korištenje opcije legend.direction. Primjetimo još da je smještanje legende desno od grafa zadana postavka.

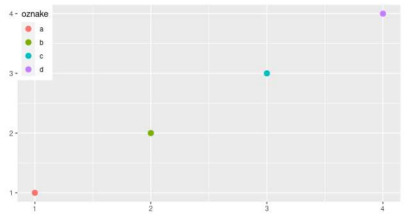
```
graf
graf + theme(legend.position = "right", legend.direction = "horizontal")
```



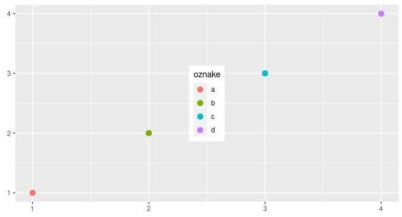

Slika 80: Korištenje opcije legend.direction

Ukoliko na grafu ima dovoljno mjesta, legendu ne moramo prikazivati sa strane, već ju možemo nalijepiti na sam graf. To činimo pomoću legend.position i legend.justification. Obje stavke zahtjevaju proslijeđivanje vektora duljine dva. Vektor c(0,1) smješta legendu u gornji lijevi kut, c(1,0) dolje desno itd. Slijedi primjer.

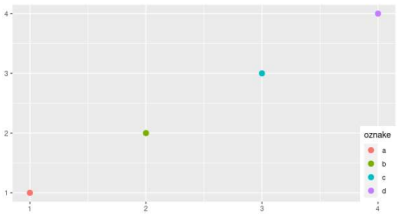
```
graf + theme(legend.position = c(0, 1), legend.justification = c(0, 1))
graf + theme(legend.position = c(0.5, 0.5), legend.justification = c(0.5, 0.5))
graf + theme(legend.position = c(1, 0), legend.justification = c(1, 0))
```




Slika 81: c(0,1)




Slika 82: c(0.5, 0.5)



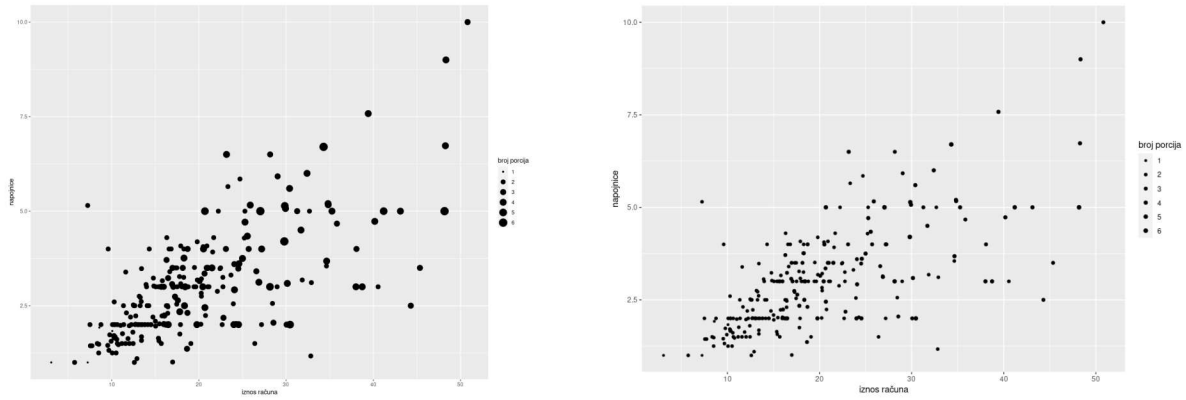

Slika 83: c(1, 0)

Uočimo da je ovakvo određivanje pozicije legende relativno, u odnosu na graf. Točno određivanje pozicije legende je puno zahtjevnije i zahtjeva metodu pokušaja i promašaja.

## 6.13. Ostale estetske komponente

Od ostalih estetskih komponenti, posebno je zanimljiva mogućnost manipuliranja rasponom veličine točaka. Prema zadanim postavkama, točki koja označuje najmanju vrijednost u podacima se pridružuje veličina 1, a najvećoj 6. Prema tome se skaliraju ostale točke. Međutim, na ovaj način možemo dobiti dosta nepregledan graf. Kako bismo to spriječili, možemo promijeniti raspon točaka s atributom range u funkciji scale_size(). Pogledajmo na primjeru.

```
graf <- ggplot(tips, aes(total_bill, tip, size = size)) +
  geom_point() +
  labs(x = "iznos računa", y = "napojnice", size = "broj porcija")

graf
graf + scale_size(range = c(1, 2))
```




Slika 84: Korištenje funkcije scale_size() u kombinaciji s atributom range

Osim ove funkcije, spomenut ćemo još nekoliko korisnih funkcija za skaliranje točaka:

- scale_size_binned() se ponaša kao scale_size(). Razlika je u tome što vrijednosti neprekidne varijable dijeli u kategorije i prema tome skalira točke
- scale_size_date() i scale_size_datetime() skaliraju vremenske varijable

Što se tiče oblika, jedina opcija je solid, a za vrstu linije, jedina opcija je na.value.

# 7. Životopis

Zovem se Antonija Sedlar. Rođena sam 14.05.1997. u Osijeku. Pohađala sam Osnovnu školu Miroslava Krleže u Čepinu i III. gimnaziju Osijek. 2016. godine sam upisala Preddiplomski studij Matematike na Odjelu za matematiku u Osijeku. Završavam ga 2019. godine s temom završnog rada Fermatovi brojevi pod mentorstvom doc. dr. sc. Mirela Jukic Bokun. Iste godine upisujem diplomski studij, smjer Financijska matematika i računarstvo.

# Literatura


[1] S. Berinato, *Good Charts: The HBR Guide to Making Smarter, More Persuasive Data Visualizations*, Harvard Business Review Press, 2016.

[2] *CRAN - vignettes*,
URL: https://cran.r-project.org/web/packages/ggplot2/vignettes/ggplot2-specs.html

[3] *CRAN - paket ggfittext*,
URL: https://cran.r-project.org/web/packages/ggrepel/index.html

[4] *CRAN - paket ggrepel*,
URL: https://cran.r-project.org/web/packages/ggrepel/index.html

[5] *CRAN - paket scales*,
URL: https://cran.r-project.org/web/packages/scales/

[6] *Kaggle - baza podataka Netflix stock data live and latest*,
URL: https://www.kaggle.com/kalilurrahman/netflix-stock-data-live-and-latest

[7] *Kaggle - baza podataka tips*,
URL: https://www.kaggle.com/ranjeetjain3/seaborn-tips-dataset

[8] *R graph gallery*,
URL: https://www.r-graph-gallery.com/index.html

[9] *R - project*,
URL: https://www.r-project.org/

[10] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*, Springer, 2016.