Predavanje 05 – Statistični grafi s knjižnico *ggplot2*

Elementi statističnih grafov

Spretnosti, ki jih zahteva risanje statističnih grafov lahko razdelimo na dve ločeni skupini:

- razumevanje osnovnih konceptov statistične vizualizacije in
- razumevanje orodij, s katerimi te koncepte prenesemo v prakso.

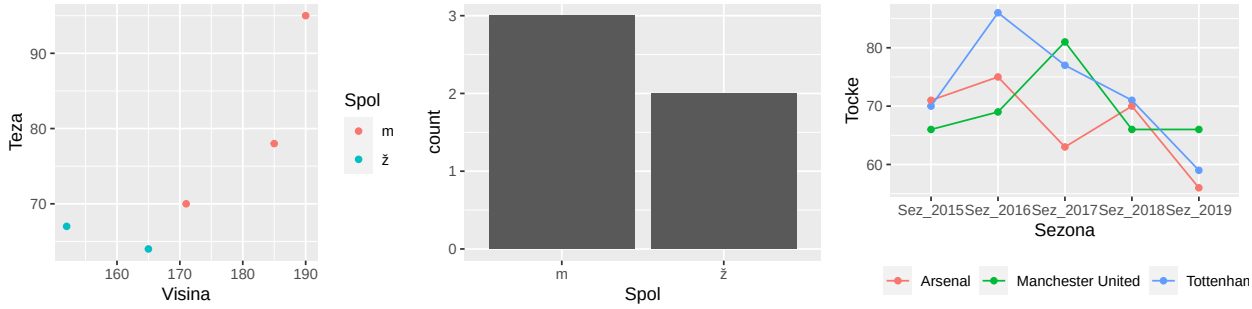
Danes bomo spoznali eno izmed orodij – R v kombinaciji s knjižnico *ggplot2*.

Knjižnica ggplot2 se od večine ostalih orodij za risanje grafov razlikuje v tem, da gre za implementacijo sistematičnega formalnega jezika za opisovanje grafov. Z drugimi besedami, gre za prenos osnovnih konceptov statistične vizualizacije v prakso. Zaradi tega je sicer veliko bolj zmogljiva, a na prvi pogled bolj zahtevna za uporabo v primerjavi z drugimi orodji, na primer risanjem grafov v Excel.

Drznili si bomo trditi, da večina te navidezne zahtevnosti izvira predvsem iz tega, da smo zaradi osredotočenosti na orodja pozabili, kaj je bistvo risanja statističnih grafov. Preden spoznamo knjižnico ggplot bomo torej najprej ponovili nekaj osnovnih konceptov. Pri tem si bomo pomagali s podatki in grafi iz nalog, ki so služile kot priprava na to predavanje:

```
##      Ime Visina Teza Spol Starost
## 1   Alen    171   70    m      41
## 2  Bojan    185   78    m      35
## 3 Cvetka    165   64    ž      28
## 4  Dejan    190   95    m      52
## 5    Eva    152   67    ž      22
```

```
##                Klub      Mesto Sez_2015 Sez_2016 Sez_2017 Sez_2018 Sez_2019
## 1         Tottenham     London       70       86       77       71       59
## 2 Manchester United Manchester       66       69       81       66       66
## 3           Arsenal     London       71       75       63       70       56
```



Osnovne koncepte statističnih grafov bomo poskušali spoznati tako, da se bomo osredotočili na elemente, ki so nujno potrebni, da je graf vsebinsko tak, kot je. Z drugimi besedami, to so elementi, ki jih moramo sporočiti osebi (ali orodju), ki naj bi graf narisala, saj brez njih le-ta grafa ne bo znala narisati. Tem

elementom bomo rekli *ključni elementi*. Elementom, ki niso nujno potrebni ali pa jih je moč samodejno določiti na podlagi dobrih praks, pa bomo rekli *sekundarni elementi*.

Prvi ključen element pri vsakem netrivialnem grafu so **podatki**. Nadaljnja utemeljitev tu ni potrebna – brez podatkov ne moremo narisati grafa.

Kaj so še drugi ključni elementi pri prvem grafu, razsevnemu diagramu višine in teže? Minimalno, kar moramo sporočiti risarju grafa, je, da želimo narisati razsevni diagram. Bolj formalno lahko temu rečemo, da želimo podatek predstaviti z 2D točko. Takoj sledi, da moramo risarju nujno sporočiti , kako naj določi koordinati točke (x in y). Ker želimo točke še ločiti glede na spol in to z barvo, moramo seveda sporočiti tudi, kako je določena barva točke.

A to zadošča, da bo risar narisal vsebinsko enak graf? Menimo, da ja! A bo vsak risar narisal povsem enak graf? Ne! Pri risanju grafa moramo določiti še mnogo sekundarnih elementov, kot so tip in barva ozadja, meje na oseh, lokacije oznak na oseh, barvna paleta, imena osi, lokacija legende. A vendar se vsi ti elementi razlikujejo od ključnih elementov po tem, da obstajajo smiselne privzete vrednosti ali dobre prakse. Npr. os je smiselno poimenovati po spremenljivki iz podatkov, ki določa to koordinato. Dobro orodje nam mora seveda omogočati, da spremenimo tudi te sekundarne elemente, a še bolj pomembno je, da je sposobno narisati lep graf, tudi če sekundarnih elementov ne sporočimo. V tem se sekundarni elementi razlikujejo od ključnih – če npr. ne sporočimo koordinate x točke ali sploh ne sporočimo, da želimo podatke predstaviti s točko, potem risar grafa ne more narisati!

Sedaj na enak način analizirajmo drugi graf, stolpični diagram števila oseb po spolu. Grafična predstavitev je tokrat stolpec ali *škatla* in ne točka, kot je pri razsevnem diagramu. Koordinata x je v tem primeru spol. Kaj pa koordinata y? Na y-os želimo dati število podatkov v skupini, ki jo določa vrednost koordinate x. Na tem mestu velja omeniti, da smo koordinati y priredili neko funkcijo podatkov (višina, teža, spol so v podatkih; število moških/žensk pa ne). Ostali elementi pri tem grafu niso ključni. Sem spada tudi barva, saj navodila niso zahtevala, da se skozi barvo odraža neka lastnost podatkov.

Tretji graf je konceptualno podoben prvemu. Ključni elementi so koordinati x in y (sezona in točke), barva, ki je določena z imenom nogometnega kluba, ter grafična predstavitev, ki je točka. Glavna razlika pa je, da točka ni edina grafična predstavitev! Poleg točke uporabimo tudi črto, ki povezuje točke. Kako pa risar ve, katere točke povezati? Na prvo žogo bi lahko odgovorili, da povežemo točke enake barve. Če pa malo razmislimo, to ni najboljša ideja, saj si bomo včasih želeli med seboj povezati vse točke, a nekatere vseeno pobarvati drugače (npr. vikende in delovnike pri časovnih podatkih). Barvo torej ločimo od še enega ključnega elementa – *skupine*, kateri pripada podatek. V tem primeru sta sicer skupina in barva določeni z imenom kluba in posledično enaki, a v splošnem ne bo vedno tako.

Vse ključne elemente, ki smo jih omenili, lahko razdelimo v tri skupine:

- **Podatki**.
- **Tip grafa** – način, kako naj podatke grafično predstavimo. Npr. točka, črta, škatla.
- **Estetike** – povezave izgleda grafičnih elementov s podatki. Npr. kaj so x-koordinata, y-koordinata, barva, skupina.

Te tri skupine bodo igrale pomembno vlogo pri komuniciranju s knjižnico ggplot2.

Knjižnica *ggplot2*

Sedaj smo pripravljeni, da se spoznamo s knjižnico ggplot2. Najprej jo naložimo:

```
library(ggplot2)
```

Večino dela bomo opravili preko ene same funkcije iz te knjižnice, funkcije **ggplot()**. Ta funkcija bo igrala vlogo našega risarja – mi ji bomo sporočili ključne in včasih tudi sekundarne elemente, funkcija pa nam bo vrnila statistični graf tipa *gg* oziroma *ggplot*.

Spomnimo se, da R, ko mu podamo samo izraz (število, vektor, data.frame...), izpiše vrednost tega izraza (števila, vektorja, data.frame-a...). Enako velja za grafe – ko R podamo graf, ga bo R izrisal. Pokličimo sedaj prvič našega risarja:

```
ggplot()
```


Brez potrebnih ključnih elementov našemu risarju ne preostane drugega, kot da izriše prazen graf.

Poskusimo sedaj narisati razsevni diagram iz prejšnjega poglavja. Najprej naložimo podatke:

```
df.osebe <- read.csv("./data_raw/osebe.csv")
print(df.osebe)
```

```
##      Ime Visina Teza Spol Starost
## 1   Alen    171   70    m      41
## 2  Bojan    185   78    m      35
## 3 Cvetka    165   64    ž      28
## 4  Dejan    190   95    m      52
## 5    Eva    152   67    ž      22
```

Spomnimo se, da so bili ključni elementi tistega grafa **podatki**, **tip grafa** (predstavitev s točko) in nekaj **estetik** (koordinati x in y ter barva). Ta ločitev med tremi tipi ključnih elementov je namerna, saj le-ti sovpadajo s tem, kako ključne elemente podamo funkciji `ggplot()`.

Začnimo s **podatki**. Podatke bomo funkciji `ggplot()` pošiljali kot podatkovne strukture data.frame. Na tem mestu bomo omenili, da podpira tudi druge podatkovne tipe, a o tem ne bomo razpravljali, in da moramo podatke pripraviti v t.i. *dolgi obliki*, o čemer bomo več povedali kasneje.

Ker smo vajeni dela z data.frame-i in smo podatke že prebrali v to obliko, moramo spoznati samo še, kako jih podamo funkciji `ggplot()`. To preprosto storimo tako, da so podatki prvi argument funkcije:

```
ggplot(df.osebe) # lahko tudi ggplot(data = df.osebe)
```


Rezultat je enak kot prej, saj risar nima dovolj informacij, da bi kaj narisal. V zgornjem izrazu opazimo tudi nov element R in to je simbol `#`. S tem simbolom v R označimo komentarje. Pri pisanju programske

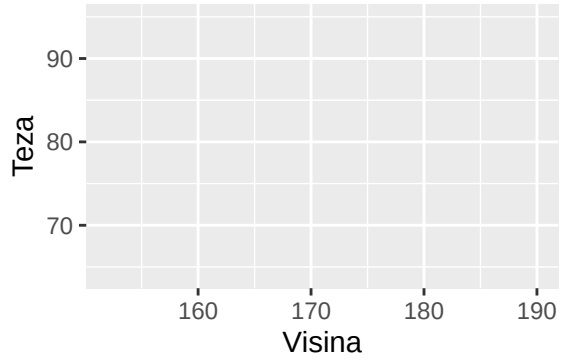
kode se vedno trudimo, da je čim bolj razumljiva. Namen tega je, da bomo kodo razumeli tudi več mesecev po tem ko smo jo napisali, ali pa da jo bodo razumeli drugi ljudje, na primer sodelavci. Vsekakor pa je včasih potrebno podati kakšno dodatno informacijo o nekem delu kode. Temu so namenjeni komentarji. V R označimo začetek komentarja z `#`. Vse kar sledi temu simbolu v vrstici se **ne izvede** ko poženemo kodo. V komentarje lahko torej pišemo poljubno besedilo, ki služi temu, da bralcu poda neke dodatne informacije o kodi. Zgoraj smo na primer želeli bralcu sporočiti, da bi enak rezultat dobil, če bi argument podal z imenom.

V naslednjem koraku bomo risarju podali **estetike**. Kot smo že povedali, so estetike povezave izgleda grafičnih elementov s podatki. Vsaka estetika je tako sestavljena iz imena grafičnega elementa v ggplot2 in podatka. Zaenkrat bomo spoznali tri estetike, ki jih potrebujemo za ta graf: koordinato x, koordinato y in barvo. Njihova imena v ggplot2 so preprosto *x*, *y* in *colour*.

Najbolj pogost način povezovanja estetike s podatki je, da estetiki z enačajem priredimo ime stolpca, ki vsebuje podatek, ki določa njeno vrednost. V našem primeru je to npr. `x = Visina` in `y = Teza`.

Vse estetike funkciji `ggplot` podamo ločene z vejico znotraj funkcije `aes()`.

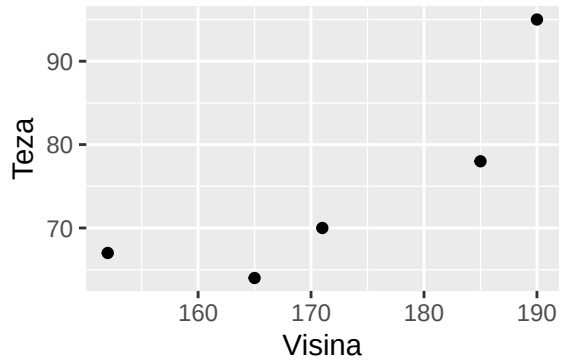
```
ggplot(df.osebe, aes(x = Visina, y = Teza))
```



Naš risar ima končno dovolj informacij, da nekaj nariše. Ker mu še nismo sporočili zadnjega ključnega elementa, načina, kako predstaviti podatke, graf še ni tak, kot bi si želeli, a vseeno je narisal osi. Iz tega lahko razberemo, kakšne so privzete nastavitve glede oblikovanja osi: ggplot minimum, maksimum in razmak med oznakami na osi določi na podlagi samih podatkov v skladu z dobrimi praksami, os pa poimenuje po podatku, ki smo ga priredili estetiki. Vse to lahko kasneje tudi spremenimo, a bistvo je, da nam teh sekundarnih elementov ni potrebno nastavljati. V večini primerov bomo tudi povsem zadovoljnji s privzetimi vrednostmi.

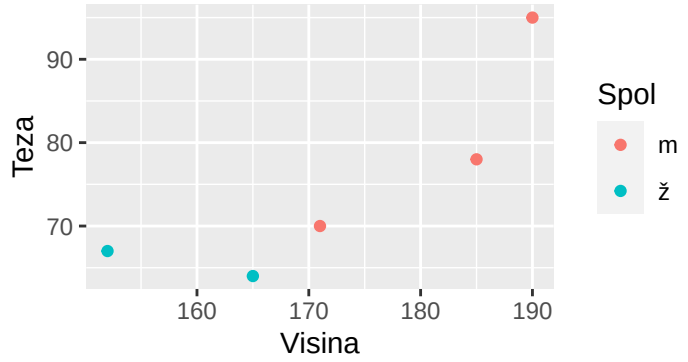
Do našega prvega spodobnega grafa nam manjka le še to, da sporočimo, kako bi radi predstavili podatke. V jeziku knjižnice ggplot2 tem predstavitvam pravimo tudi **geom**-i. Obstaja mnogo različnih geomov in vsak geom ima potencialno drugačen nabor estetik. V tem predavanju bomo spoznali tri geome, *škatlo*, *črto*, in *točko*, začenši s slednjo. Predstavitev s točko se v ggplot2 imenuje `geom_point()`. Novo grafično predstavitev grafu najlažje dodamo tako, da jo prištejemo funkciji `ggplot()`. To se sprva morda zdi nenavadno, ampak je veliko bolj pregledno, ko želimo dodati večje število predstavitev in možnosti:

```
ggplot(df.osebe, aes(x = Visina, y = Teza)) + geom_point()
```



Risar je uspešno narisal graf, ki tudi brez dodatnih možnosti izgleda lepo. Da bi v celoti reproducirali graf iz uvoda, moramo dodati še estetiko *barva* (colour), ki jo v tem primeru določa podatek v stolpcu Spol:

```
ggplot(df.osebe, aes(x = Visina, y = Teza, colour = Spol)) + geom_point()
```

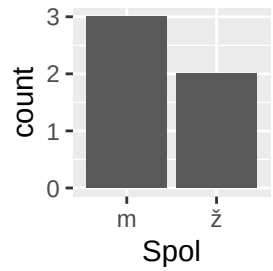


Kot vidimo, risar samodejno dodeli barvo vsaki izmed vrednosti v stolpcu Spol in to informacijo prikaže v legendi na desni strani grafa. Tako barvno paleto kot tudi lastnosti prikaza legende lahko spremenimo, ampak to so sekundarni elementi in v večini primerov nam bo zadoščala privzeta vrednost.

Stolpični diagram

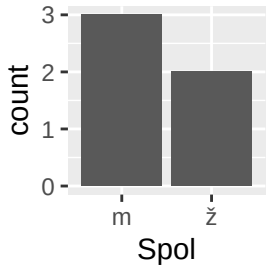
Sedaj bomo poskušali reproducirati še stolpični diagram iz uvoda. Edina nova stvar, ki se jo moramo naučiti, je poimenovanje geoma *škatle* v ggplot2. Ta geom se imenuje `geom_bar()`. Ta geom ima eno samo ključno estetiko – koordinato x. Privzeta vrednost koordinate y pa je točno to, kar želimo, število elementov, ki imajo tisto vrednost koordinate y:

```
ggplot(df.osebe, aes(x = Spol)) + geom_bar()
```



Kot smo že omenili, število elementov ni številka, ki se pojavi v podatkih, ampak je povzetek oz. statistika, ki jo izračunamo iz podatkov. Katero statistiko naj geom uporabi, lahko podamo tudi eksplicitno:

```
ggplot(df.osebe, aes(x = Spol)) + geom_bar(stat = "count")
```



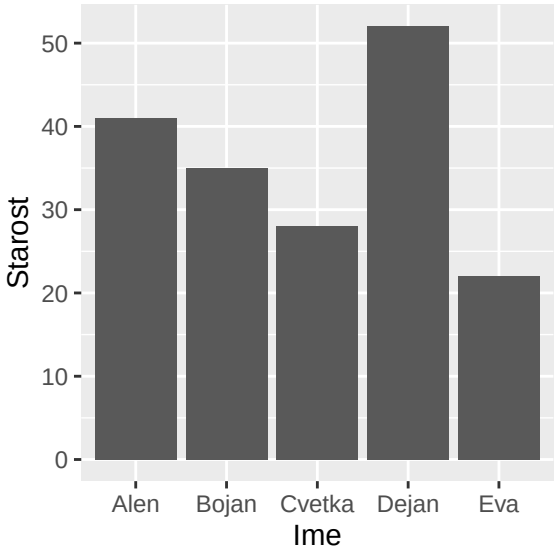
To sicer privede do enakega grafa, a bo koristen ilustrativen vmesni korak, da spoznamo še eno statistiko – identiteto. Pogosto namreč želimo narisati stolpični diagram, kjer eksplicitno določimo tudi koordinato y. Kot primer vzemimo stolpični diagram, kjer s stolpci prikažemo starost oseb:

```
ggplot(df.osebe, aes(x = Ime, y = Starost)) + geom_bar()
```

```
## Error: stat_count() can only have an x or y aesthetic.
```

Kot vidimo, nam R javi napako, saj privzeto delovanje `geom_bar()` zahteva ali x ali y, ne pa oboje. Če želimo podati oboje, moramo eksplicitno spremeniti statistiko iz privzetega števca v identiteto:

```
ggplot(df.osebe, aes(x = Ime, y = Starost)) + geom_bar(stat = "identity")
```



Črtni diagram

Preostane nam še, da reproduciramo črtni diagram iz uvoda. Najprej naložimo podatke:

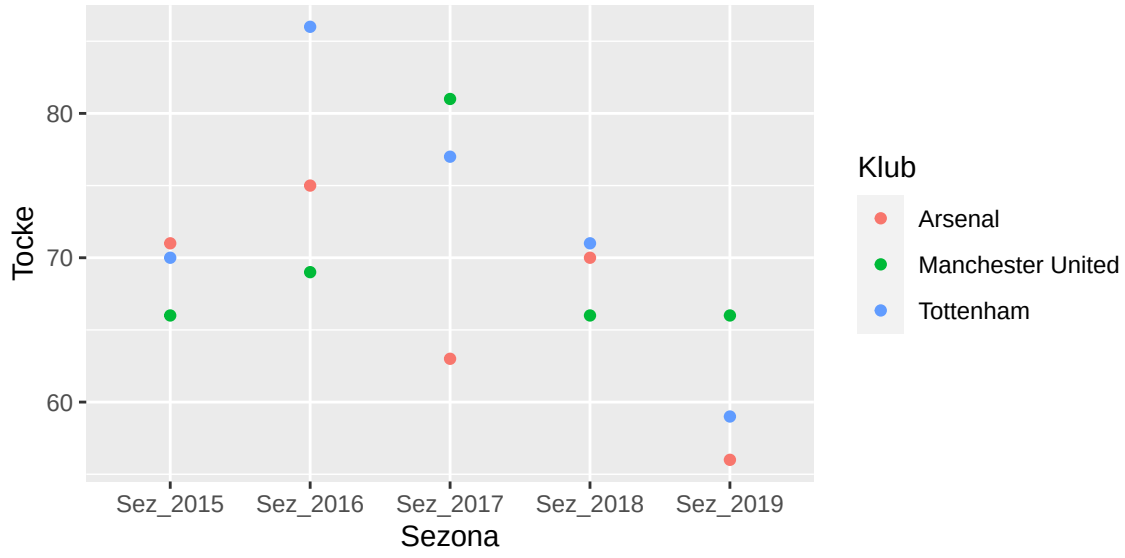
```
df.klubi_dolga <- read.csv("./data_raw/klubi_dolga.csv")
print(df.klubi_dolga)
```

```
##                 Klub      Mesto   Sezona Tocke
## 1          Tottenham     London Sez_2015    70
## 2          Tottenham     London Sez_2016    86
## 3          Tottenham     London Sez_2017    77
## 4          Tottenham     London Sez_2018    71
## 5          Tottenham     London Sez_2019    59
## 6  Manchester United Manchester Sez_2015    66
## 7  Manchester United Manchester Sez_2016    69
## 8  Manchester United Manchester Sez_2017    81
## 9  Manchester United Manchester Sez_2018    66
## 10 Manchester United Manchester Sez_2019    66
## 11           Arsenal     London Sez_2015    71
## 12           Arsenal     London Sez_2016    75
## 13           Arsenal     London Sez_2017    63
## 14           Arsenal     London Sez_2018    70
## 15           Arsenal     London Sez_2019    56
```

Opazimo, da so podatki v nekoliko drugačni obliki. Kot bomo spoznali kasneje, gre za t.i. dolgo obliko, ki je bolj primerna za delo z ggplot2.

Poskusimo najprej z geomom in estetikami, ki smo jih že spoznali (točka; x, y in colour):

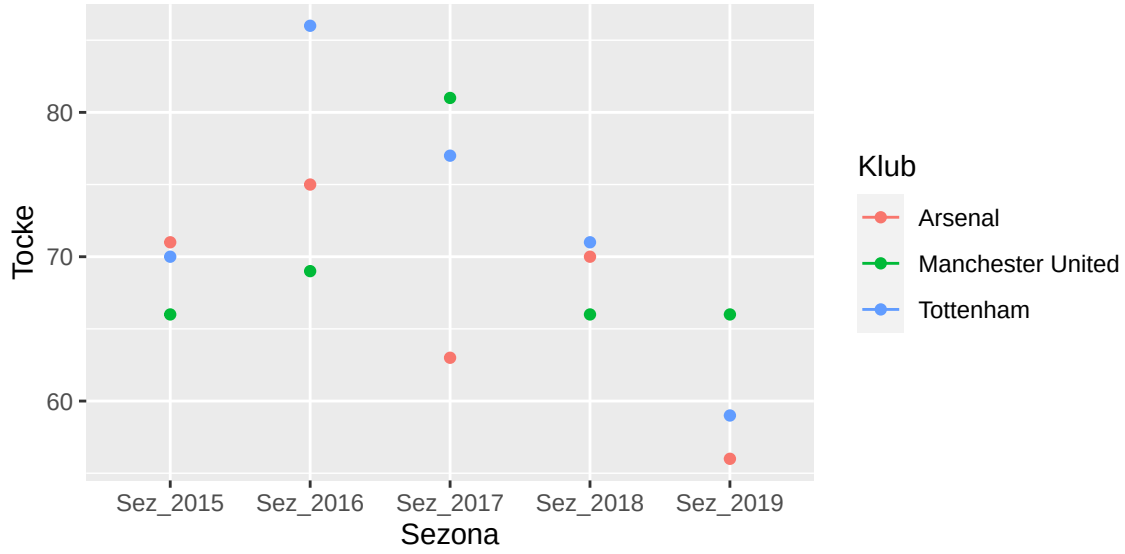
```
ggplot(df.klubi_dolga, aes(x = Sezona, y = Tocke, colour = Klub)) +
  geom_point()
```



Smo že zelo blizu, a manjkajo črte, ki povezujejo točke. Črta, ki povezuje točke od leve proti desni – `geom_line()` – je še zadnji geom, ki ga bomo danes spoznali. Črtni diagram s točkami je graf, ki ima 2 geoma:

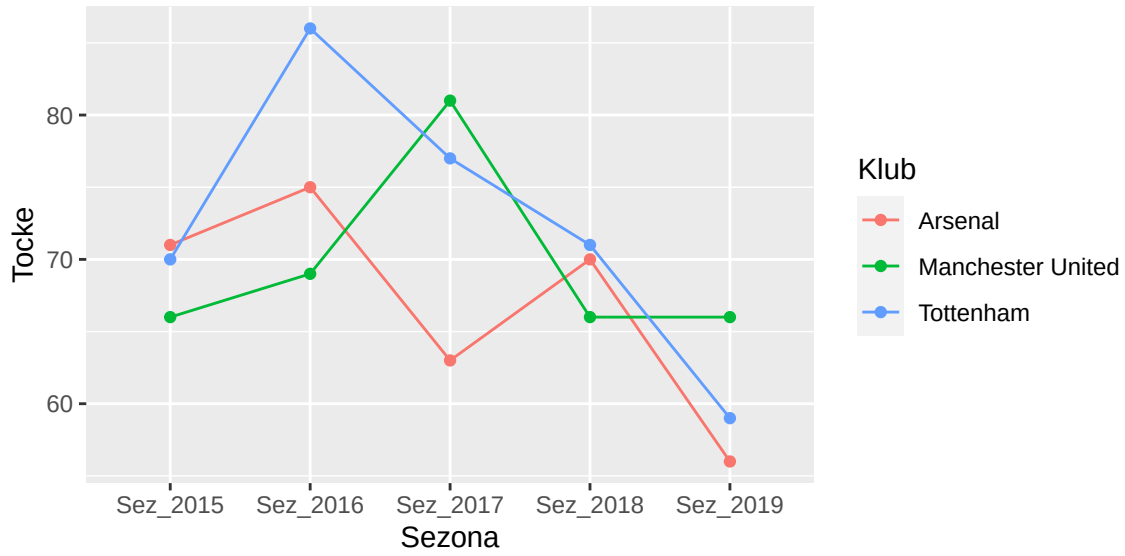
```
ggplot(df.klubi_dolga, aes(x = Sezona, y = Tocke, colour = Klub)) +
  geom_point() + geom_line()
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```



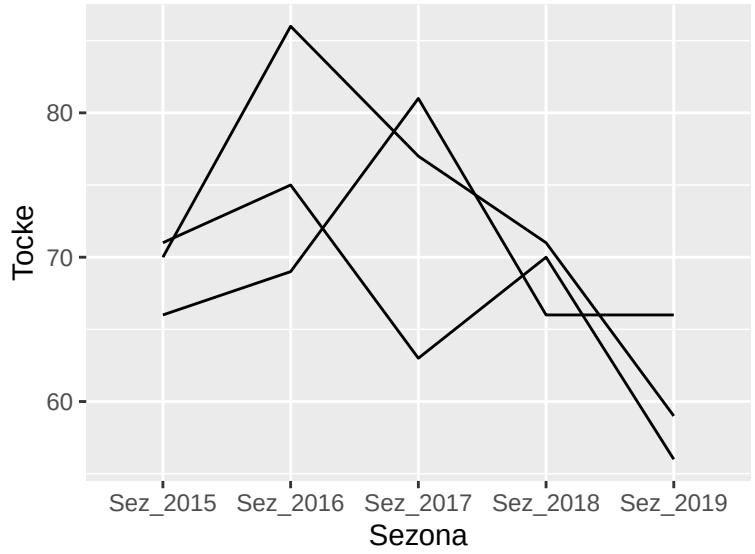
Kot smo omenili že v uvodu, risar ne more vedeti, katere točke povezati s črto, če mu tega ne sporočimo! Sedaj spoznajmo še zadnjo estetiko danes – *group*. V našem primeru sta colour in group povezani z istim podatkom:

```
ggplot(df.klubi_dolga, aes(x = Sezona, y = Tocke, colour = Klub, group = Klub)) +
  geom_point() + geom_line()
```



V razmislek: lahko poskusimo tudi brez estetike barva in geom-a točka:

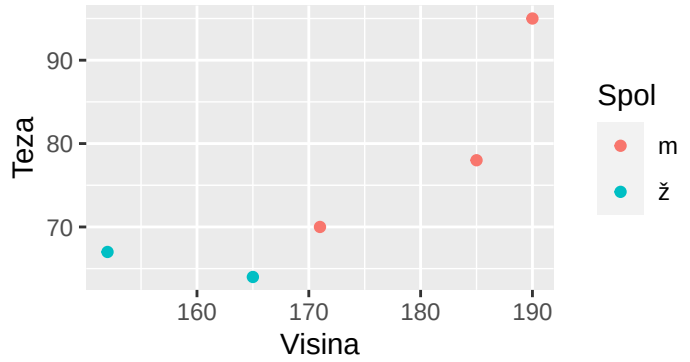
```
ggplot(df.klubi_dolga, aes(x = Sezona, y = Tocke, group = Klub)) +
  geom_line()
```



Shranjevanje grafov

S funkcijo `ggsave()` iz knjižnice ggplot2 lahko grafe tipa ggplot shranimo v enem izmed popularnih formatov. Dobra praksa dela z grafi je, da jih najprej shranimo v spremenljivko. Nato jih lahko izrišemo ali njihovo podobo shranimo v datoteko:

```
g1 <- ggplot(df.osebe, aes(x = Visina, y = Teza, colour = Spol)) + geom_point()
plot(g1)
```




```
ggsave("./plots/graf.jpg", width = 4, height = 3, dpi = 300) # format določi končnica
ggsave("./plots/graf.png", width = 4, height = 3, dpi = 300)
ggsave("./plots/graf.pdf", width = 4, height = 3, dpi = 300)
```

Široka in dolga oblika podatkov

Nekajkrat smo že omenili, da so za uporabo v `ggplot()` bolj primerni podatki v t.i. dolgi obliki. Poglejmo si primera istih podatkov v široki in dolgi obliki:

```
df.klubi <- read.csv("./data_raw/klubi.csv")
print(df.klubi_dolga)
```

```
##                 Klub      Mesto   Sezona Tocke
## 1          Tottenham     London Sez_2015    70
## 2          Tottenham     London Sez_2016    86
## 3          Tottenham     London Sez_2017    77
## 4          Tottenham     London Sez_2018    71
## 5          Tottenham     London Sez_2019    59
## 6  Manchester United Manchester Sez_2015    66
## 7  Manchester United Manchester Sez_2016    69
## 8  Manchester United Manchester Sez_2017    81
## 9  Manchester United Manchester Sez_2018    66
## 10 Manchester United Manchester Sez_2019    66
## 11           Arsenal     London Sez_2015    71
## 12           Arsenal     London Sez_2016    75
## 13           Arsenal     London Sez_2017    63
## 14           Arsenal     London Sez_2018    70
## 15           Arsenal     London Sez_2019    56
```

```
print(df.klubi)
```

```
##                 Klub      Mesto Sez_2015 Sez_2016 Sez_2017 Sez_2018 Sez_2019
## 1          Tottenham     London       70       86       77       71       59
## 2 Manchester United Manchester       66       69       81       66       66
## 3            Arsenal     London       71       75       63       70       56
```

Pri podatkih v dolgi obliki je vsaka meritev/observacija predstavljena v svoji vrstici (V tem primeru vsako število točk). Prednost tega formata je, da lahko stolpce enostavno preslikamo v estetike, saj vsaka vrstica predstavlja en element, ki ga je potrebno grafično prikazati. Slabost tega formata je, da se podatki, ki so skupni večim vrsticam po nepotrebnem ponavljajo (Klub, Mesto). Zaradi tega so podatki tudi težje berljivi za človeka.

Pri podatkih v široki obliki imamo v eni vrstici več oz. vse meritve, ki se nanašajo na neko entiteto v podatkih (v tem primeru klub). Taki podatki so za človeka bolj berljivi, a neprimerni za neposredno delo s knjižnico ggplot2. Težave lahko nastanejo tudi pri poimenovanju stolpcev, saj do stolpcev, ki se začnejo s številkami, ne moremo dostopati brez narekovajev.

Če povzamemo bistvo – za risanje z ggplot2 se je dobro naučiti, kako data.frame pretvoriti iz široke v dolgo obliko. Za to bomo uporabili funkcijo `pivot_longer()` iz knjižnice *tidyr*:

```
library(tidyr)
```

Funkciji podamo data.frame s podatki ter stolpce, kjer so meritve. Stolpce lahko podamo na več načinov, tu si bomo pogledali samo enega, ki pa je dovolj splošen, da pokrije vse primere – podamo indekse stolpcev. V našem primeru se meritve nahajajo v stolpcih od tretjega do sedmega:

```
df.klubi
```

```
##                 Klub      Mesto Sez_2015 Sez_2016 Sez_2017 Sez_2018 Sez_2019
## 1          Tottenham     London       70       86       77       71       59
## 2 Manchester United Manchester       66       69       81       66       66
## 3            Arsenal     London       71       75       63       70       56
```

```
pivot_longer(df.klubi, cols = 3:7)
```

```
## # A tibble: 15 x 4
##    Klub              Mesto      name     value
##    <chr>             <chr>      <chr>    <int>
##  1 Tottenham         London     Sez_2015    70
##  2 Tottenham         London     Sez_2016    86
##  3 Tottenham         London     Sez_2017    77
##  4 Tottenham         London     Sez_2018    71
##  5 Tottenham         London     Sez_2019    59
##  6 Manchester United Manchester Sez_2015    66
##  7 Manchester United Manchester Sez_2016    69
##  8 Manchester United Manchester Sez_2017    81
##  9 Manchester United Manchester Sez_2018    66
## 10 Manchester United Manchester Sez_2019    66
## 11 Arsenal           London     Sez_2015    71
## 12 Arsenal           London     Sez_2016    75
## 13 Arsenal           London     Sez_2017    63
## 14 Arsenal           London     Sez_2018    70
## 15 Arsenal           London     Sez_2019    56
```

Hkrati lahko tudi ustrezno poimenujemo novonastala stolpca, ki hranita imena meritev in vrednosti:

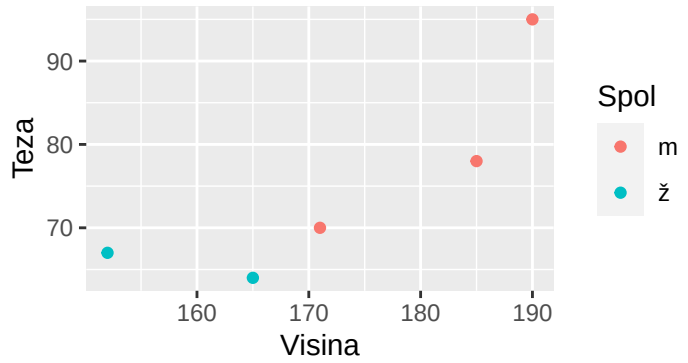
```
pivot_longer(df.klubi, cols = 3:7, names_to = "Sezona", values_to = "Tocke")
```

```
## # A tibble: 15 x 4
##    Klub              Mesto      Sezona   Tocke
##    <chr>             <chr>      <chr>    <int>
##  1 Tottenham         London     Sez_2015    70
##  2 Tottenham         London     Sez_2016    86
##  3 Tottenham         London     Sez_2017    77
##  4 Tottenham         London     Sez_2018    71
##  5 Tottenham         London     Sez_2019    59
##  6 Manchester United Manchester Sez_2015    66
##  7 Manchester United Manchester Sez_2016    69
##  8 Manchester United Manchester Sez_2017    81
##  9 Manchester United Manchester Sez_2018    66
## 10 Manchester United Manchester Sez_2019    66
## 11 Arsenal           London     Sez_2015    71
## 12 Arsenal           London     Sez_2016    75
## 13 Arsenal           London     Sez_2017    63
## 14 Arsenal           London     Sez_2018    70
## 15 Arsenal           London     Sez_2019    56
```

Še nekaj uporabnih ukazov iz knjižnice *ggplot2*

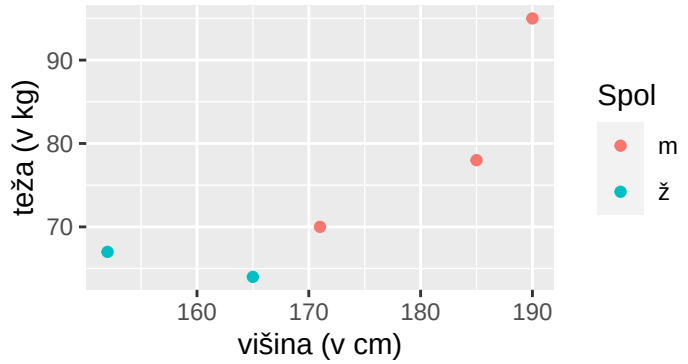
Preden zaključimo, si poglejmo kako lahko nastavimo lastnosti nekaterih sekundarnih elementov, ki jih bomo najpogosteje želeli spremeniti. Kot izhodišče bomo vzeli razsevni diagram iz uvoda:

```
ggplot(df.osebe, aes(x = Visina, y = Teza, colour = Spol)) + geom_point()
```



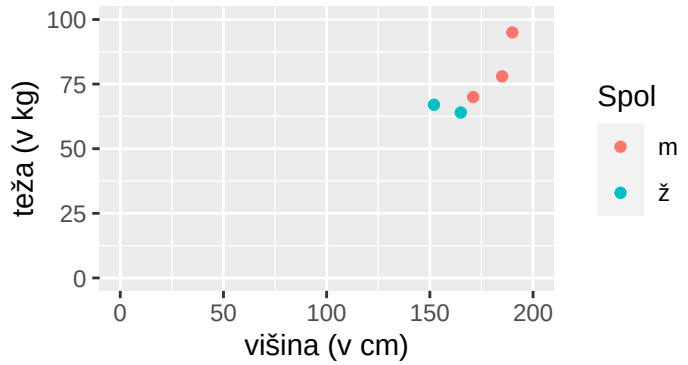
Pogosto si želimo spremeniti imena osi. To storimo s funkcijama `xlab()` in `ylab()`, ki jima podamo novo ime:

```
ggplot(df.osebe, aes(x = Visina, y = Teza, colour = Spol)) + geom_point() +
  xlab("višina (v cm)") + ylab("teža (v kg)")
```



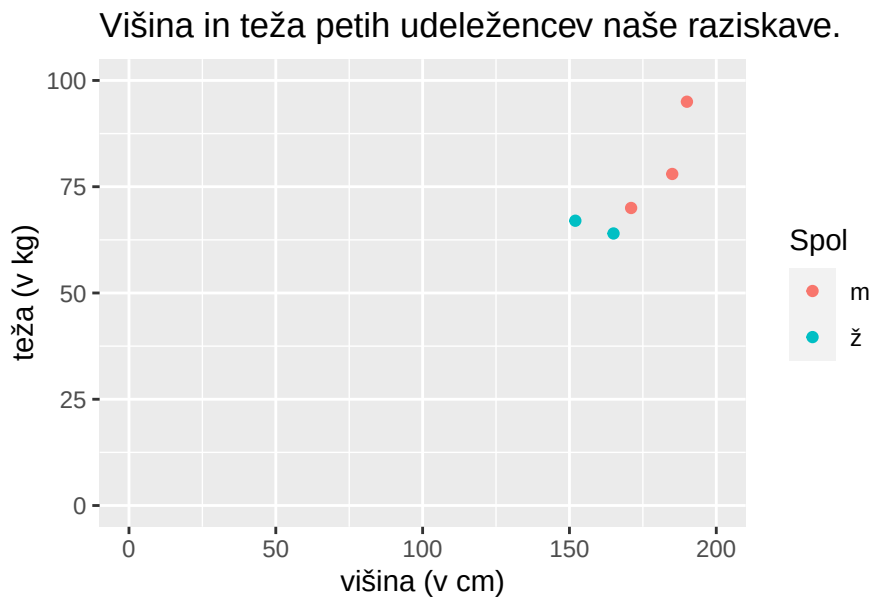
Včasih želimo sami nastaviti meje osi. To storimo s funkcijama `xlim` in `ylim`, v katerih nastavimo zgornjo in spodnjo mejo:

```
ggplot(df.osebe, aes(x = Visina, y = Teza, colour = Spol)) + geom_point() +
  xlab("višina (v cm)") + ylab("teža (v kg)") + xlim(0, 200) + ylim(0, 100)
```



Z uporabo funkcije `ggtitle()` pa lahko nastavimo naslov grafa:

```
ggplot(df.osebe, aes(x = Visina, y = Teza, colour = Spol)) + geom_point() +
  xlab("višina (v cm)") + ylab("teža (v kg)") + xlim(0, 200) + ylim(0, 100) +
  ggtitle("Višina in teža petih udeležencev naše raziskave.")
```



Čeprav razsevni, stolpični in črtni diagram predstavljajo večino grafov, s katerim se srečamo pri tipičnih analizah, obstaja še mnogo drugih tipov grafov. Knjižnica ggplot2 pokriva praktično vse dvorazsežne grafe, ki jih bomo potrebovali, in kopico možnosti, da jih prilagodimo svojim potrebam. Med te možnosti spadajo prilagajanje osi, barvne palete, anotacija grafa. Vse, kar boste verjetno potrebovali (in še več), najdete tudi v zgoščenem povzetku na tej povezavi.

Domača naloga

Vse naloge od nas zahtevajo, da z uporabo knjižnice ggplot2 narišemo enak graf, kot je izrisan v navodilu naloge. Naloge poskusimo rešiti sami, če pa se zatakne, pa lahko iz datoteke *Predavanje_05.Rmd* razberemo, kako je bil narisan graf, ki ga moramo reproducirati.

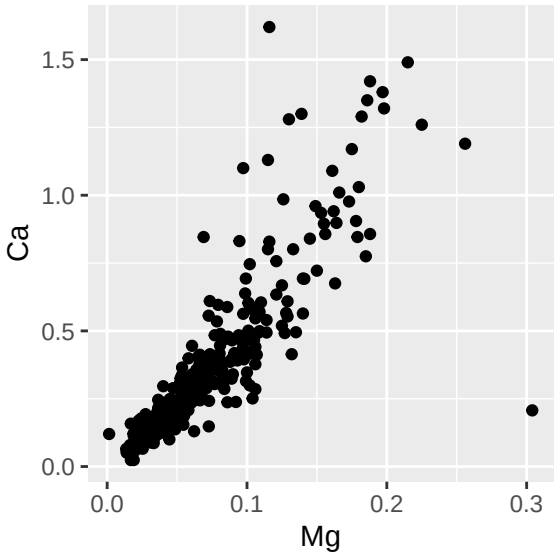
Pri vseh nalogah bomo uporabili podatke o meritvah onesnaženosti iz prejšnjih predavanj. Pred risanjem bomo tudi pretvorili Datum iz niza znakov v primeren podatkovni tip `Date`:

```
df <- read.csv("./data_raw/delci2.csv")
df$Datum <- as.Date(df$Datum, format = "%m/%d/%Y") # pretvorba iz niza v Date
head(df)
```

```
##        Datum PM10    Ca    Cl     K     Mg     Na   NH4  NO3  kraj
## 1 2014-01-17   22 0.186 0.297 0.577 0.0374 0.1450 0.639 1.98 Kranj
## 2 2014-01-18   32 0.132 0.528 0.735 0.0235 0.1090 0.877 2.71 Kranj
## 3 2014-01-19   30 0.145 0.381 0.577 0.0363 0.1590 1.080 2.72 Kranj
## 4 2014-01-20   16 0.127 0.170 0.383 0.0428 0.0608 0.628 2.01 Kranj
## 5 2014-01-21   24 0.202 0.160 0.418 0.0365 0.0346 1.220 3.62 Kranj
## 6 2014-01-22   32 0.610 0.231 0.615 0.0734 0.0468 1.140 3.83 Kranj
```

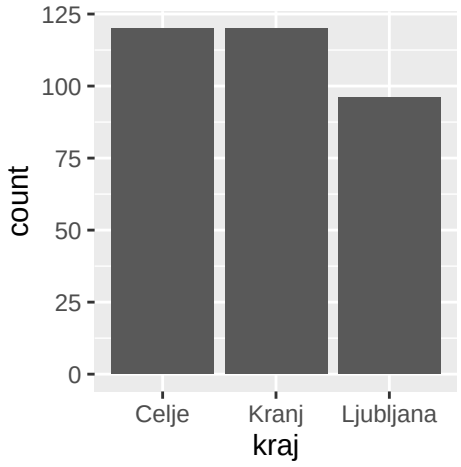
Naloga 1

Narišimo razsevni diagram, ki prikazuje odnos med meritvami magnezija (Mg) in kalcija (Ca):



Naloga 2

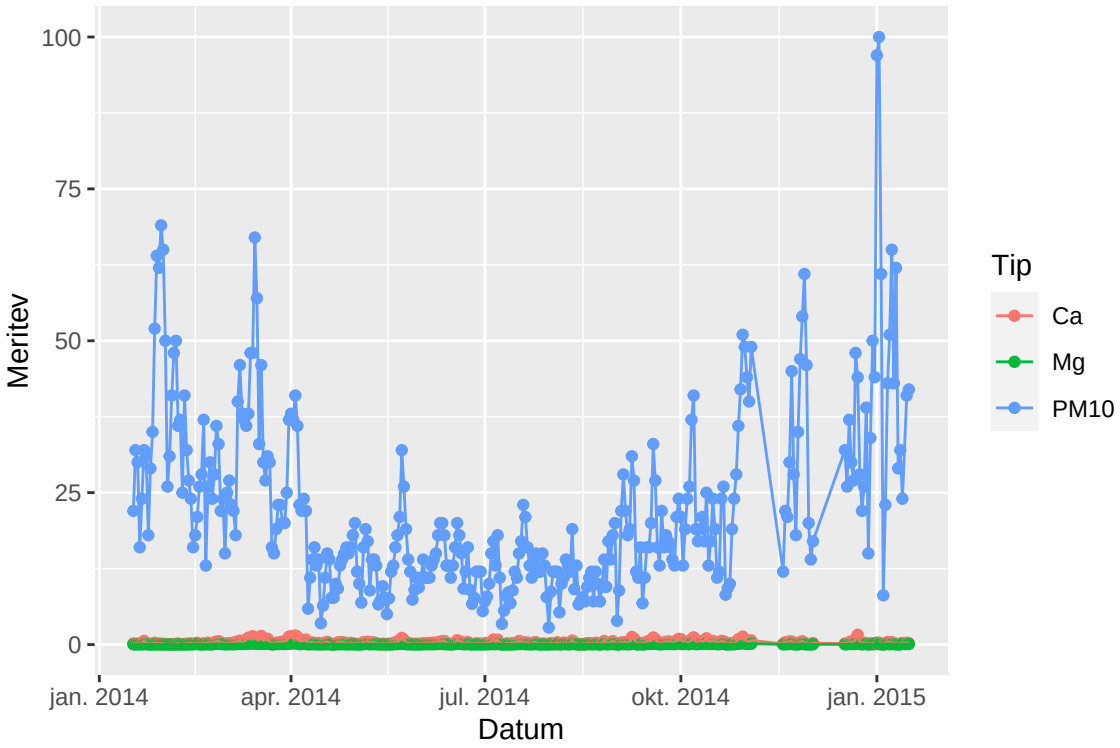
Narišimo stolpični diagram, ki prikazuje število dni meritev glede na kraj:



Naloga 3

Pozor, pri tej nalogi moramo izbrati ustrezno podmnožico stolpcev in podatke pretvoriti v dolgo obliko.

Narišimo črtni diagram, ki prikazuje vrednosti meritev PM10, Ca in Mg skozi čas. Vsako predstavimo z drugo barvo:



Naloga 4

Naloga 4 je težka, saj zahteva uporabo možnosti knjižnice ggplot2, ki jih nismo obravnavali na predavanju. Uporabite prej omenjeni zgoščen povzetek in iskalnik Google in se poskusite čim bolj približati narisanemu grafu. Tudi če vam ne uspe povsem in boste na koncu pogledali rešitev, se boste pri tem zagotovo naučili veliko novih uporabnih množnosti, ki jih ponuja knjižnica ggplot. Še en namig, da se ne bomo ukvarjali še z R: uporabite funckijo `weekdays()` in podatek, da je vikend dan, ki je sobota ali nedelja.

Za to nalogo bomo naložili podatke o meritvah onesnaženosti iz prejšnjih predavanj in izbrali podmnožico meritev za Kranj. Pred risanjem tudi pretvorimo datum iz niza znakov v podatkovni tip Date.

Narišimo ta graf:

Vrednosti meritev PM10 za tri slovenska mesta. Rdeca crta predstavlja kriticno mejo dnevne onesnaženosti.