Predavanje 08 – Odgovori na vprašanja - November 22

Statistični testi

Večina klasičnih statističnih testov in modelov je vgrajenih že v osnovni R. Poglejmo si uporabo treh izmed najbolj popularnih, t-testa, ANOVE in linearne regresije.

```
# Modelirajmo porabo goriva, pri čemer kot neodvisne spremenljivke uporabimo:
# število cilindrov, konjsko moč in težo.
lr <- lm(mpg ~ cyl + hp + wt, data = mtcars)
summary(lr)
```

```
##
## Call:
## lm(formula = mpg ~ cyl + hp + wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9290 -1.5598 -0.5311  1.1850  5.8986
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 38.75179    1.78686  21.687  < 2e-16 ***
## cyl         -0.94162    0.55092  -1.709 0.098480 .
## hp          -0.01804    0.01188  -1.519 0.140015
## wt          -3.16697    0.74058  -4.276 0.000199 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.512 on 28 degrees of freedom
## Multiple R-squared:  0.8431, Adjusted R-squared:  0.8263
## F-statistic: 50.17 on 3 and 28 DF,  p-value: 2.184e-11
```

```
# t-test uporabimo za statistično primerjavo pričakovane širine listov
# dveh vrst perunike.
x_vir <- iris$Sepal.Width[iris$Species == "virginica"]
x_ver <- iris$Sepal.Width[iris$Species == "versicolor"]

t.test(x_vir, x_ver)
```

```
##
##  Welch Two Sample t-test
##
## data:  x_vir and x_ver
## t = 3.2058, df = 97.927, p-value = 0.001819
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.07771636 0.33028364
## sample estimates:
## mean of x mean of y
```

```
##     2.974     2.770
```

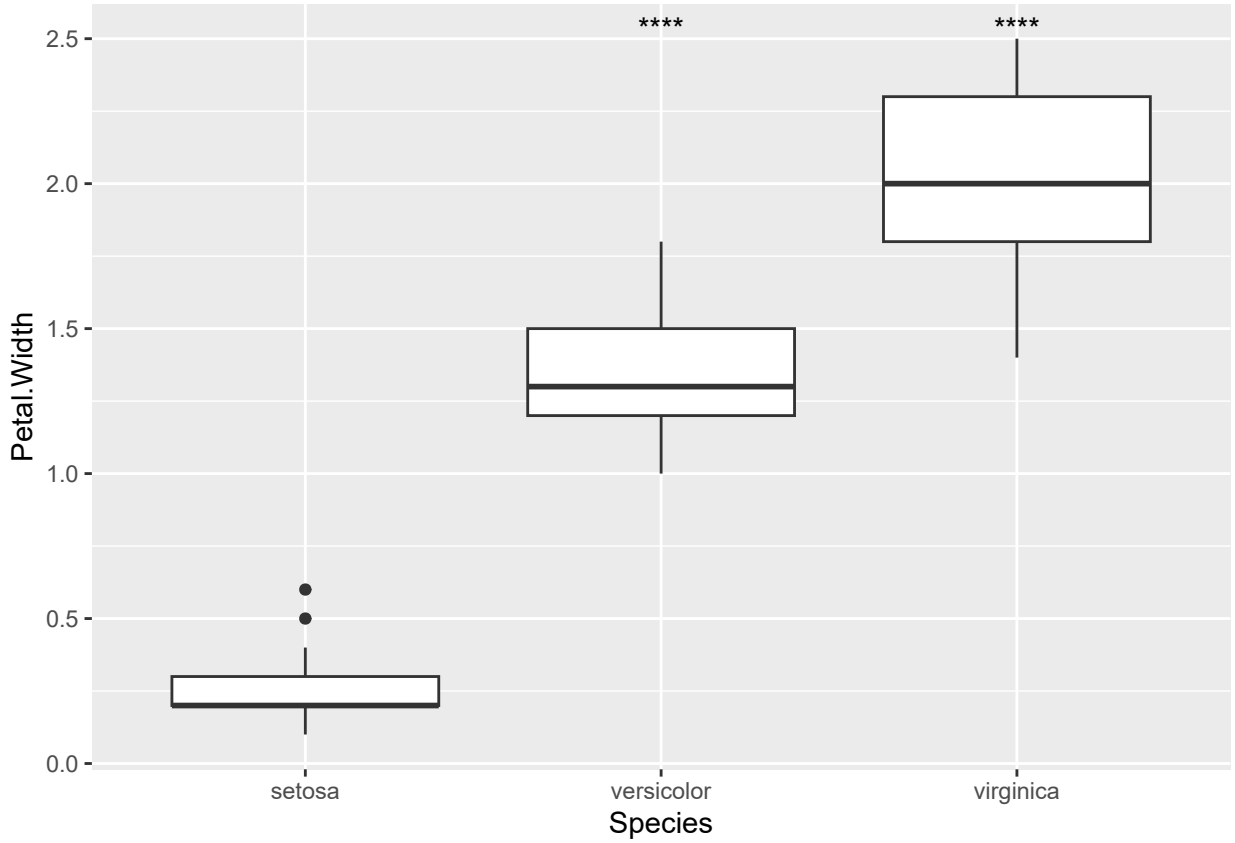
```
# ANOVO uporabimo za statistično primerjavo dolžine listov treh vrst perunike.
# Primerjamo, ali vrsta perunike vpliva na dolžino listov.
my_anova <- aov(Sepal.Length ~ Species, data = iris)
summary(my_anova)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Species       2  63.21  31.606   119.3 <2e-16 ***
## Residuals   147  38.96   0.265
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ggplot2 – statistična značilnost

Včasih želimo rezultate statističnega testa prikazati kar na grafu. Poglejmo si sedaj primer t-testa v ggplot2. Za to bomo potrebovali še en paket **ggpubr** in funkcijo iz tega paketa `stat_compare_means`. Poleg statističnega testa bomo izrisali tudi boxplot (Diagram s *škatlami in brčicami*).

```
library(ggplot2)
library(ggpubr)
ggplot(iris, aes(x = Species, y = Petal.Width)) +
  geom_boxplot() +
  stat_compare_means(label = "p.signif", method = "t.test",
                     ref.group = "setosa")
```



Prikaz točk in povprečja na grafu

Poglejmo si še en zanimiv graf, kjer bomo prikazali točke in povprečja na istem grafu. Pogledali si bomo porazdelitve dolžin in širin čašnih listov različnih perunik. Najprej si pripravimo `data.frame`.

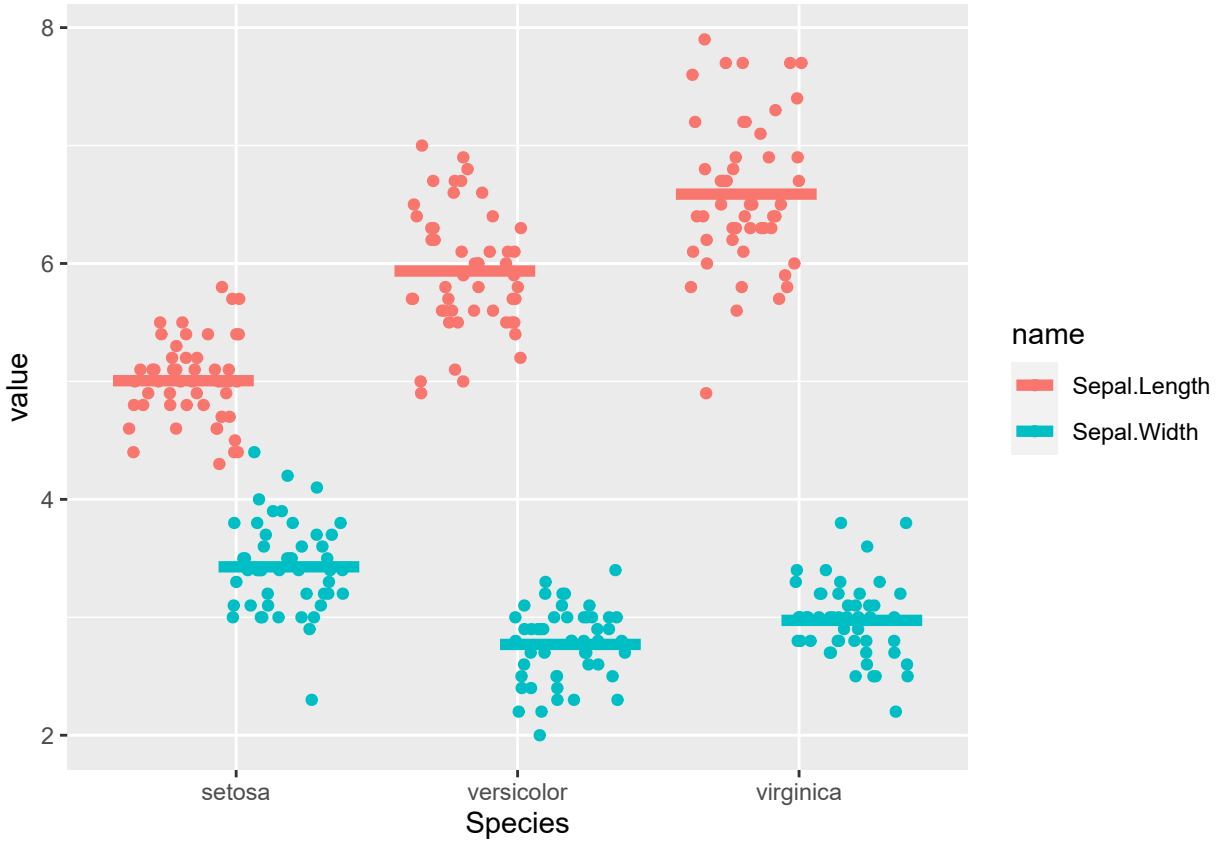
```
library(tidyr)
iris_longer <- iris[ , c("Sepal.Length", "Sepal.Width", "Species")]
iris_longer <- pivot_longer(iris_longer, Sepal.Length:Sepal.Width)
head(iris_longer)
```

```
## # A tibble: 6 x 3
##   Species name         value
##   <fct>   <chr>        <dbl>
## 1 setosa  Sepal.Length   5.1
## 2 setosa  Sepal.Width    3.5
## 3 setosa  Sepal.Length   4.9
## 4 setosa  Sepal.Width    3
## 5 setosa  Sepal.Length   4.7
## 6 setosa  Sepal.Width    3.2
```

Za izris povprečij s črto bomo potrebovali geom `hpline` iz paketa **ungeviz** (https://wilkelab.org/ungeviz/index.html). Za izris točk uporabimo pri `geom_point` argument `position = position_jitterdodge()`. To najprej loči dolžine in širine listov (dodge) in potem še nekoliko raztrosi točke (jitter), da je bolj pregledno, kjer imamo več točk. Če ne bi uporabili tega, bi enostavno dobili prikazane vse točke v isti liniji.

Instalacija devtools::install_github("wilkelab/ungeviz")

```
library(ungeviz)
pl_means <-
  ggplot(iris_longer, aes(x = Species, y = value, color = name)) +
  geom_point(position = position_jitterdodge()) +
  stat_summary(
    fun = "mean",
    position = position_dodge(width = 0.75),

        geom = "hpline"
  )
pl_means
```



ggplot2 – errorbar

Na statističnih grafih, ki vsebujejo opisne statistike, kot je npr. povprečje, pogosto prikažemo še negotovost v obliki standardnih odklonov ali standardnih napak. S knjižnico ggplot2 to storimo z uporabo geom-a `errorbar`. Pred tem moramo ustrezno pripraviti podatke tako, da dodamo še stolpec s spodnjo in zgornjo mejo napake. Če je napaka simetrična, potrebujemo le en stolpec. Poglejmo si odvisnost milj na galono (`mpg`) od števila cilindrov.

```
data("mtcars")
head(mtcars)
```

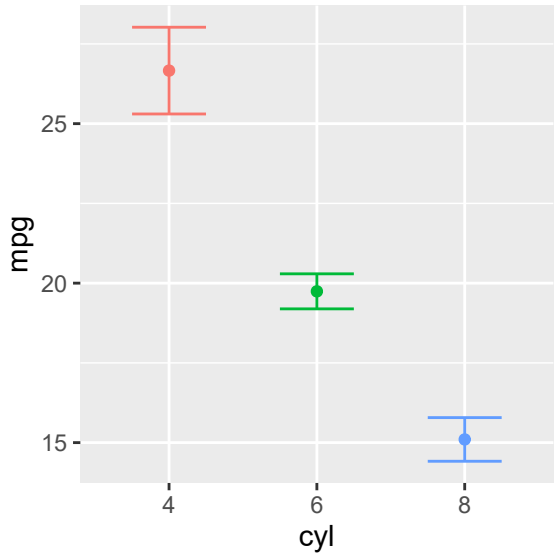
```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
mus <- aggregate(mpg ~ cyl, mtcars, FUN = mean)
sds <- aggregate(mpg ~ cyl, mtcars, FUN = function(x) {sd(x) / sqrt(length(x))})
df  <- cbind(mus, SE = sds$mpg)
df$cyl <- as.character(df$cyl)

head(df)
```

```
##   cyl      mpg        SE
## 1   4 26.66364 1.3597642
## 2   6 19.74286 0.5493967
```

```
## 3   8 15.10000 0.6842016
```

```
library(ggplot2)
ggplot(df, aes(x = cyl, y = mpg, colour = cyl)) +
  geom_point() +
  geom_errorbar(aes(ymin = mpg - SE, ymax = mpg + SE), width = 0.5) +
  theme(legend.position = "none")
```



Risanje stolpičnih diagramov

Želimo izrisati podatke anket, kjer so anketiranci odgovarjali na vprašanja o vzrokih zakaj se rekreirajo manj kot pet krat tedensko. Odgovori so bili ponujeni vnaprej, odgovarjali pa so z "Da", "Delno", "Ne" in "Zame ne velja", glede na to, če menijo, da se odgovor sklada z njihovimi vzroki. Vprašanja so lahko pustili tudi neodgovorjena.

Najprej poglejmo, kako so bili odgovori shranjeni.

```
library(openxlsx)
anketa <- read.xlsx("./data_raw/vpr_predavanje_8.xlsx")
head(anketa)
```

```
##   Q2a_too_far Q2b_no_connections Q2c_not_quality Q2d_disabled Q2e_no_time
## 1           2                  1               2            4           1
## 2           2                  3               1            4           2
## 3          -2                 -2              -2           -2          -2
## 4           4                  4               2            4           1
## 5           3                  3               3            4           3
## 6           2                  2               3            3           1
##   Q2f_finance Q2g_no_company Q2h_too_tired Q2i_no_need_motivation
## 1           3              3             2                      3
## 2           3              2             2                      3
## 3          -2             -2            -2                     -2
## 4           2              2             2                      2
## 5           3              3             3                      2
## 6           3              2             2                      3
```

Iz podatkov lahko razberemo, da vsak stolpec predstavlja vprašanje, vsaka vrstica pa enega anketiranca.

Odgovori so zakodirani s številkami in sicer:

- < 0: **Neodgovorjeno**
- 1: **Da**
- 2: **Delno**
- 3: **Ne**
- 4: **Zame ne velja**

Iz teh podatkov želimo izrisati podoben graf, kot je na spodnji sliki.

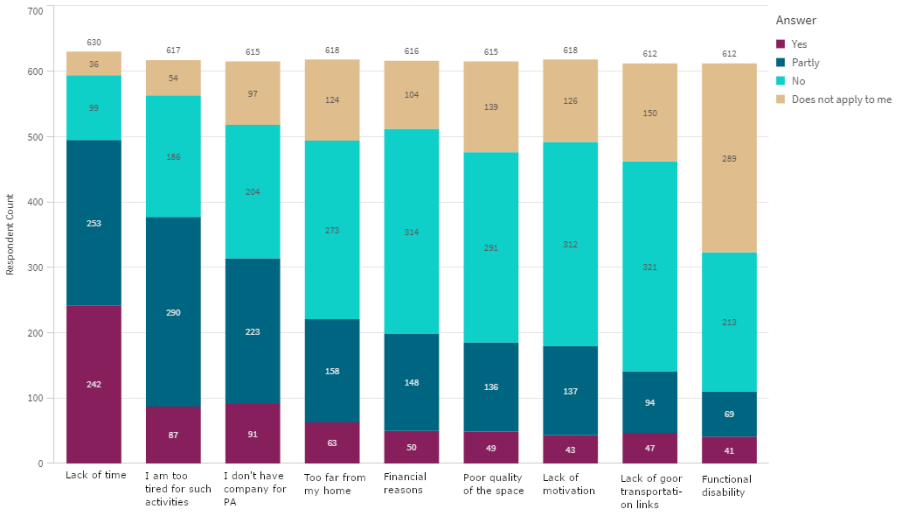


Figure 1: Primer željenega stolpičnega diagrama

```
library(openxlsx)
anketa_raw <- read.xlsx("./data_raw/vpr_predavanje_8.xlsx")
head(anketa_raw)
```

```
##   Q2a_too_far Q2b_no_connections Q2c_not_quality Q2d_disabled Q2e_no_time
## 1           2                  1               2            4           1
## 2           2                  3               1            4           2
## 3          -2                 -2              -2           -2          -2
## 4           4                  4               2            4           1
## 5           3                  3               3            4           3
## 6           2                  2               3            3           1
##   Q2f_finance Q2g_no_company Q2h_too_tired Q2i_no_need_motivation
## 1           3              3             2                     3
## 2           3              2             2                     3
## 3          -2             -2            -2                    -2
## 4           2              2             2                     2
## 5           3              3             3                     2
## 6           3              2             2                     3
```

Poskušajmo sedaj ustvariti čim bolj podoben graf v R-ju.

Najprej nekoliko predelajmo vhodne podatke, saj namesto številk lahko v R-ju uporabimo faktorje, da so podatki bolj razumljivi. Manjkajoče vrednosti (negativne) zamenjamo z *NA*, ostalim pa priredimo podane odgovore.

```
anketa <- anketa_raw
anketa[anketa < 0] <- NA
anketa[anketa == 1] <- "Da"
```

```
anketa[anketa == 2] <- "Delno"
anketa[anketa == 3] <- "Ne"
anketa[anketa == 4] <- "Zame ne velja"
head(anketa)
```

```
##     Q2a_too_far Q2b_no_connections Q2c_not_quality  Q2d_disabled Q2e_no_time
## 1         Delno                 Da           Delno Zame ne velja          Da
## 2         Delno                 Ne              Da Zame ne velja       Delno
## 3          <NA>               <NA>            <NA>          <NA>        <NA>
## 4 Zame ne velja      Zame ne velja           Delno Zame ne velja          Da
## 5            Ne                 Ne              Ne Zame ne velja          Ne
## 6         Delno              Delno              Ne            Ne          Da
##   Q2f_finance Q2g_no_company Q2h_too_tired Q2i_no_need_motivation
## 1          Ne             Ne         Delno                    Ne
## 2          Ne          Delno         Delno                    Ne
## 3        <NA>           <NA>          <NA>                  <NA>
## 4       Delno          Delno         Delno                 Delno
## 5          Ne             Ne            Ne                 Delno
## 6          Ne          Delno         Delno                    Ne
```

Trenutno smo pretvorili odgovore v tip *character* sedaj pa jih lahko še v *faktor*.

```
anketa[] <- lapply(anketa, factor, levels = c("Da", "Delno",
                                              "Ne", "Zame ne velja"))
head(anketa)
```

```
##     Q2a_too_far Q2b_no_connections Q2c_not_quality  Q2d_disabled Q2e_no_time
## 1         Delno                 Da           Delno Zame ne velja          Da
## 2         Delno                 Ne              Da Zame ne velja       Delno
## 3          <NA>               <NA>            <NA>          <NA>        <NA>
## 4 Zame ne velja      Zame ne velja           Delno Zame ne velja          Da
## 5            Ne                 Ne              Ne Zame ne velja          Ne
## 6         Delno              Delno              Ne            Ne          Da
##   Q2f_finance Q2g_no_company Q2h_too_tired Q2i_no_need_motivation
## 1          Ne             Ne         Delno                    Ne
## 2          Ne          Delno         Delno                    Ne
## 3        <NA>           <NA>          <NA>                  <NA>
## 4       Delno          Delno         Delno                 Delno
## 5          Ne             Ne            Ne                 Delno
## 6          Ne          Delno         Delno                    Ne
```

Postopek lahko naredimo tudi direktno. Tukaj moramo paziti, da podamo argumenta *levels* in *labels*, kjer istoležne vrednosti predstavljajo pretvorbo. Primer `1 = "Da", 2 = "Delno"` itd..

```
an2 <- anketa_raw
an2[] <- lapply(anketa_raw, factor, levels = c(1,2,3,4),
                labels = c("Da", "Delno", "Ne", "Zame ne velja"))
head(an2)
```

```
##     Q2a_too_far Q2b_no_connections Q2c_not_quality  Q2d_disabled Q2e_no_time
## 1         Delno                 Da           Delno Zame ne velja          Da
## 2         Delno                 Ne              Da Zame ne velja       Delno
## 3          <NA>               <NA>            <NA>          <NA>        <NA>
## 4 Zame ne velja      Zame ne velja           Delno Zame ne velja          Da
## 5            Ne                 Ne              Ne Zame ne velja          Ne
## 6         Delno              Delno              Ne            Ne          Da
##   Q2f_finance Q2g_no_company Q2h_too_tired Q2i_no_need_motivation
```

```
## 1          Ne             Ne         Delno                    Ne
## 2          Ne          Delno         Delno                    Ne
## 3        <NA>           <NA>          <NA>                  <NA>
## 4       Delno          Delno         Delno                 Delno
## 5          Ne             Ne            Ne                 Delno
## 6          Ne          Delno         Delno                    Ne
```

Za izris podatkov s funkcijo *ggplot* ta oblika ni primerna. Boljša je dolga oblika, kjer bi imeli en stolpec z vprašanjem in drugi z odgovorom. Pretvorimo v podatke z uporabo *pivot_longer*.

```
library(tidyr)
anketa_long <- pivot_longer(anketa, 1:ncol(anketa),
                            names_to = "Vprasanje",
                            values_to = "Odgovor")
anketa_long
```

```
## # A tibble: 261 x 2
##    Vprasanje              Odgovor
##    <chr>                  <fct>
##  1 Q2a_too_far            Delno
##  2 Q2b_no_connections     Da
##  3 Q2c_not_quality        Delno
##  4 Q2d_disabled           Zame ne velja
##  5 Q2e_no_time            Da
##  6 Q2f_finance            Ne
##  7 Q2g_no_company         Ne
##  8 Q2h_too_tired          Delno
##  9 Q2i_no_need_motivation Ne
## 10 Q2a_too_far            Delno
## # ... with 251 more rows
```

Opazimo lahko, da sta prva in deseta vrstica enaki. Nas za izris zanima koliko je bilo takšnih odgovorov, zato moramo za vsako kombinacijo vprašanja in odgovora prešteti število vrstic. To lahko naredimo s funkcijo *table*.

```
table(anketa_long)
```

```
##                         Odgovor
## Vprasanje                Da Delno Ne Zame ne velja
##   Q2a_too_far             0     5  6             2
##   Q2b_no_connections      2     2  7             2
##   Q2c_not_quality         3     3  6             1
##   Q2d_disabled            0     0  4             9
##   Q2e_no_time             5     6  2             0
##   Q2f_finance             0     4  8             1
##   Q2g_no_company          1     5  5             2
##   Q2h_too_tired           1     9  3             0
##   Q2i_no_need_motivation  3     4  6             0
```

Funkcija *table* je našo tabelo spremenila nazaj v širšo obliko. Rezultat je v ozadnju sicer še vedno skrit v dolgi obliki. Poglejmo, kakšen izpis dobimo, če ta rezultat pretvorimo v *data.frame*.

```
head(data.frame(table(anketa_long)))
```

```
##            Vprasanje Odgovor Freq
## 1        Q2a_too_far      Da    0
## 2 Q2b_no_connections      Da    2
## 3    Q2c_not_quality      Da    3
```

```
## 4       Q2d_disabled      Da    0
## 5        Q2e_no_time      Da    5
## 6        Q2f_finance      Da    0
```

Če želimo, rezultat *table* shraniti v enaki obliki kot je v izpisu jo lahko sami pretvorimo v širšo obliko. Spremenljivko `anketa_counts` bomo potrebovali kasneje, zato jo shranimo. Če želite prešeti manjkajoče vrednosti lahko nastavite parameter `useNA = "ifany"` ali `useNA = "always"`.

```
anketa_counts <- pivot_wider(data.frame(table(anketa_long, useNA = "no")),
                             names_from = "Odgovor", values_from = "Freq")
anketa_counts
```

```
## # A tibble: 9 x 5
##   Vprasanje                 Da Delno    Ne `Zame ne velja`
##   <fct>                  <int> <int> <int>           <int>
## 1 Q2a_too_far                0     5     6               2
## 2 Q2b_no_connections         2     2     7               2
## 3 Q2c_not_quality            3     3     6               1
## 4 Q2d_disabled               0     0     4               9
## 5 Q2e_no_time                5     6     2               0
## 6 Q2f_finance                0     4     8               1
## 7 Q2g_no_company             1     5     5               2
## 8 Q2h_too_tired              1     9     3               0
## 9 Q2i_no_need_motivation     3     4     6               0
```

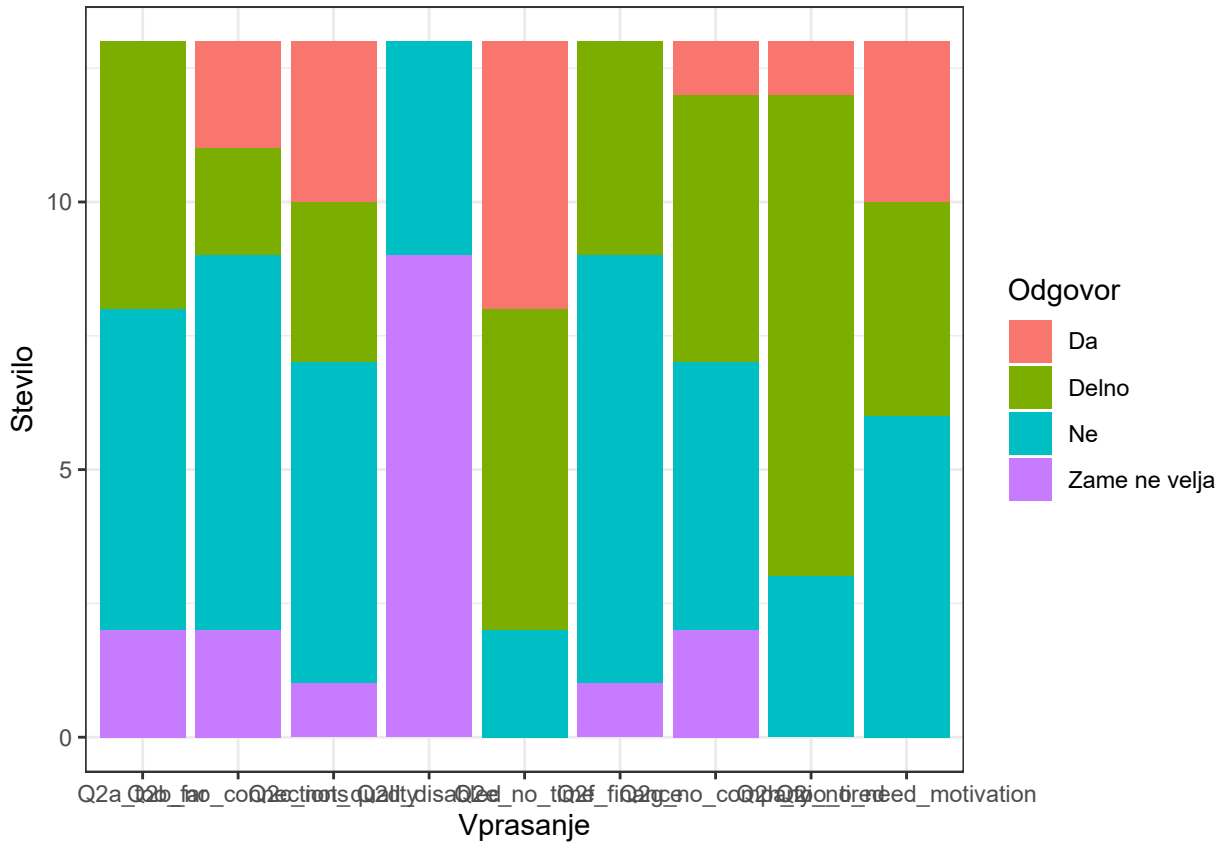
Samo za primer poglejmo, kako bi to tabelo spremenili nazaj v dolgo obliko s funkcijo *pivot_longer*. Če ste pozorni, opazite, da se je sedaj stolpec **Odgovor** spremenil v tip **character**.

```
anketa_fin <- pivot_longer(anketa_counts, 2:5, names_to = "Odgovor", values_to = "Stevilo")
anketa_fin
```

```
## # A tibble: 36 x 3
##    Vprasanje          Odgovor       Stevilo
##    <fct>              <chr>           <int>
##  1 Q2a_too_far        Da                  0
##  2 Q2a_too_far        Delno               5
##  3 Q2a_too_far        Ne                  6
##  4 Q2a_too_far        Zame ne velja       2
##  5 Q2b_no_connections Da                  2
##  6 Q2b_no_connections Delno               2
##  7 Q2b_no_connections Ne                  7
##  8 Q2b_no_connections Zame ne velja       2
##  9 Q2c_not_quality    Da                  3
## 10 Q2c_not_quality    Delno               3
## # ... with 26 more rows
```

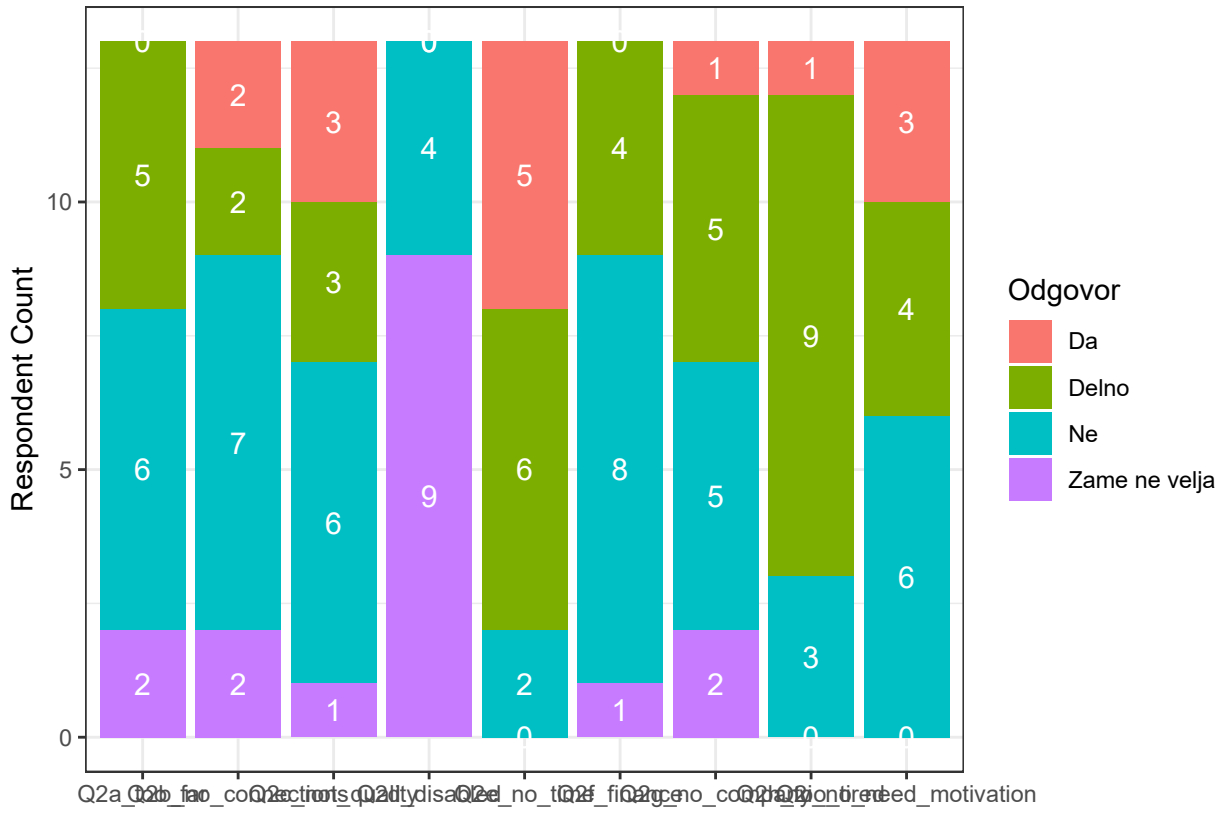
Uporabimo sedaj podatke v `anketa_fin` za izris stolpičnega diagrama. Pri tem določimo parameter `fill` za izbiro barve vsakega dela stolpca. Izbrali bomo `theme_bw`, ki se najbolj ujema z željenim ozadnjem.

```
library(ggplot2)
ggplot(anketa_fin, aes(x = Vprasanje, y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw()
```



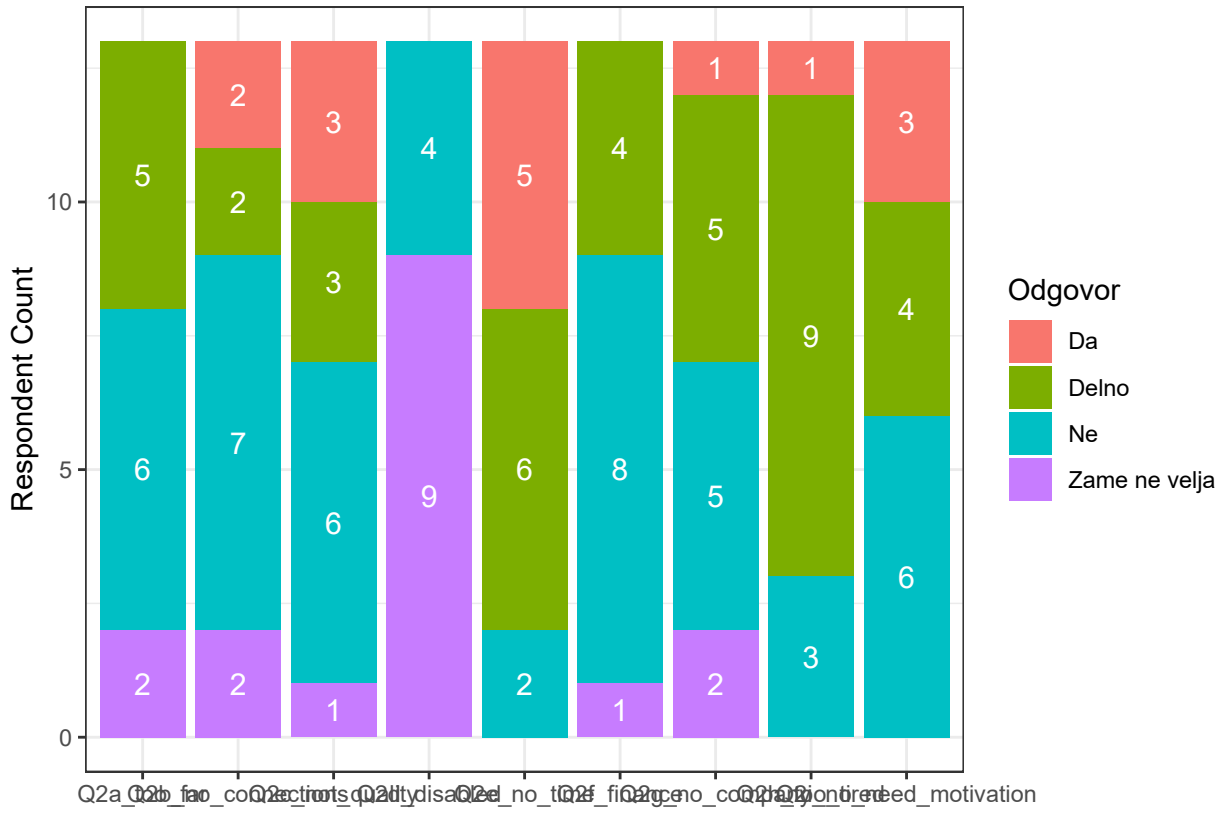
Zgornjemu diagramu od željenih elementov manjka le še izpis številk. Ostale vrednosti pa so prikazane vendar bi želeli diagram še polepšati. Za začetek spremenimo text na x in y osi ter dodajmo število odgovorov na vsak stolpec. To naredim z `geom_text`, kjer povemo naj izpisuje vrednost **Stevilo** z `aes(label=Stevilo)`, z `position_stack` pa povemo, da želimo tekst na stolpcih, pri tem `vjust` vertikalno premakne izpis na sredino, privzeto pa je na vrhu. Določimo še belo barvo in velikost pisave.

```
ggplot(anketa_fin, aes(x = Vprasanje, y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color="white",
            size=4)
```



Opazite lahko, da imamo na diagramu izpisane tudi vrednost 0 za manjkajoče vrednosti. Ker tega ne želimo, je najlažje, da te vrednosti preprosto odstranimo iz vhodnih podatkov.

```
anketa_fin <- anketa_fin[anketa_fin$Stevilo != 0,]
ggplot(anketa_fin, aes(x = Vprasanje, y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color="white",
            size=4)
```



Vrstni red stolpcev je privzeto odvisen kar od abecednega vrstnega reda imen stolpcev. Če želimo izbrati drugi vrstni red lahko uporabimo ´reorder´, vendar moramo podati želeni vrstni red. Vrstni red lahko določimo, glede na število odgovorov z "Da". Za to uporabimo prej shranjeno tablo *anketa_counts*.

```
vrstni_red <- anketa_counts[order(anketa_counts$Da),]$Vprasanje
vrstni_red
```

```
## [1] Q2a_too_far            Q2d_disabled           Q2f_finance
## [4] Q2g_no_company         Q2h_too_tired          Q2b_no_connections
## [7] Q2c_not_quality        Q2i_no_need_motivation Q2e_no_time
## 9 Levels: Q2a_too_far Q2b_no_connections Q2c_not_quality ... Q2i_no_need_motivation
```

Nekatera vprašanja imajo enako število odgovorov z vrednostjo "Da". Če želimo znotraj teh urediti po drugem atributu lahko te preposto dodamo funkciji *order*. Uredimo vrstni red po vseh stolpcih.

```
vrstni_red <- anketa_counts[order(anketa_counts$Da,
                                  anketa_counts$Delno,
                                  anketa_counts$Ne,
                                  anketa_counts$`Zame ne velja`),]$Vprasanje
vrstni_red
```

```
## [1] Q2d_disabled           Q2f_finance            Q2a_too_far
## [4] Q2g_no_company         Q2h_too_tired          Q2b_no_connections
## [7] Q2c_not_quality        Q2i_no_need_motivation Q2e_no_time
## 9 Levels: Q2a_too_far Q2b_no_connections Q2c_not_quality ... Q2i_no_need_motivation
```

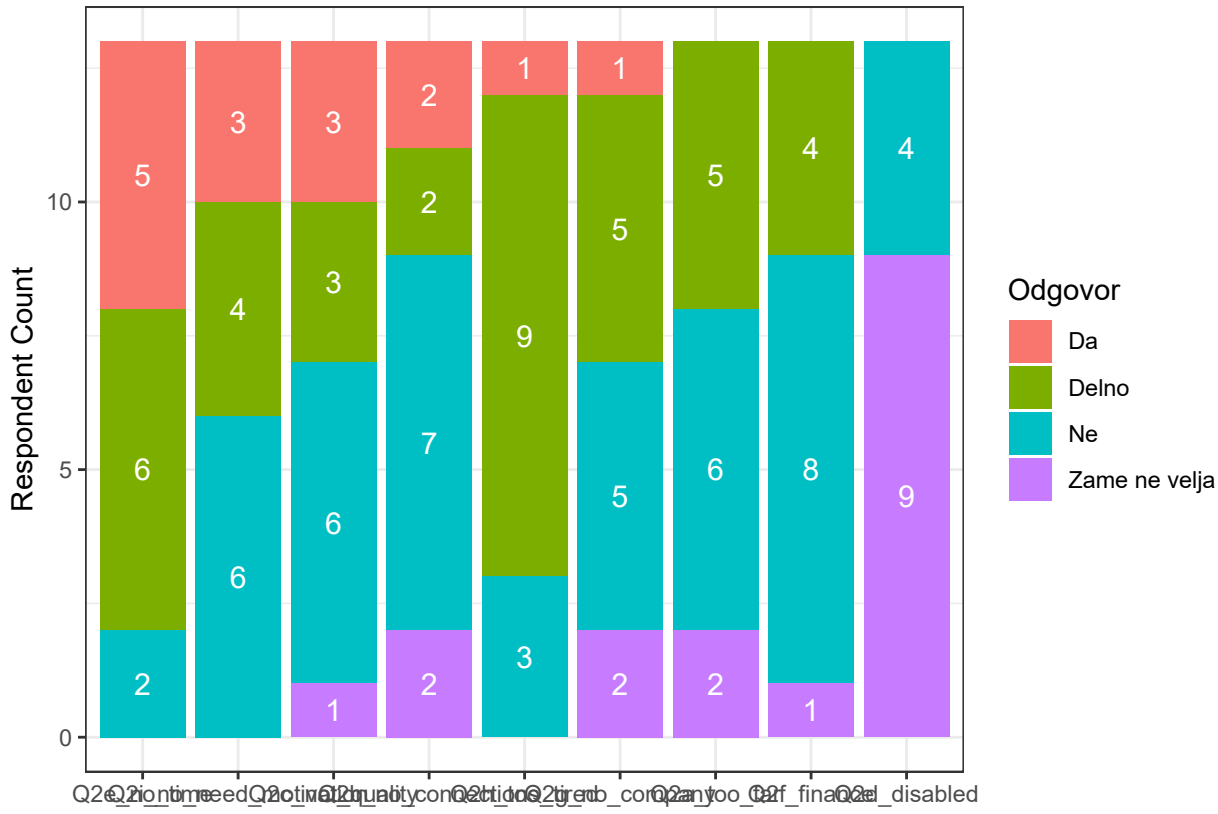
Z for zanko se lahko sprehodimo, čez vektor *vrstni_red* in podatkom, podamo stolpec, ki bo vplival na vrstni red izpisa. Z zadnji vrstici zanke bi lahko vpisovali ´-i´, vendar so pozitivna števila lažje razumljiva.

```
anketa_fin$ord <- 0
for(i in 1:length(vrstni_red)){
  #izberemo vse vrstice, ki pripadajo i-temu vprašanju
  sel <- anketa_fin$Vprasanje == vrstni_red[i]
  #dodamo pravilni indeks v ord
  anketa_fin[sel, "ord"] <- length(vrstni_red) - i
}
anketa_fin
```

```
## # A tibble: 29 x 4
##    Vprasanje          Odgovor       Stevilo   ord
##    <fct>              <chr>           <int> <dbl>
##  1 Q2a_too_far        Delno               5     6
##  2 Q2a_too_far        Ne                  6     6
##  3 Q2a_too_far        Zame ne velja       2     6
##  4 Q2b_no_connections Da                  2     3
##  5 Q2b_no_connections Delno               2     3
##  6 Q2b_no_connections Ne                  7     3
##  7 Q2b_no_connections Zame ne velja       2     3
##  8 Q2c_not_quality    Da                  3     2
##  9 Q2c_not_quality    Delno               3     2
## 10 Q2c_not_quality    Ne                  6     2
## # ... with 19 more rows
```

Izrišimo sedaj stolpce v tem vrstnem redu.

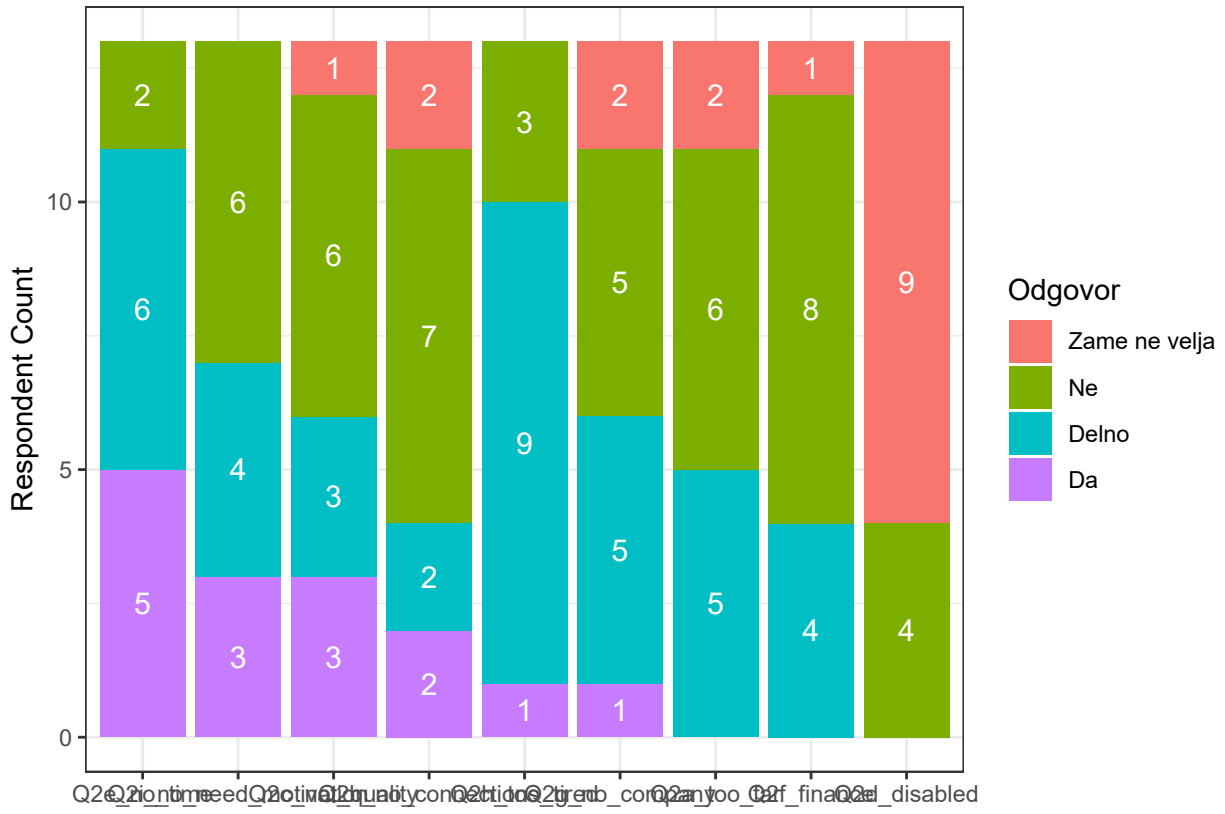
```
ggplot(anketa_fin,
       aes(x = reorder(Vprasanje, ord), y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color="white",
            size=4)
```



Vrstni red izpisa stolpcev smo sedaj določili. Opazimo pa lahko, da je vrednost "Da" sedaj na vrhu namesto spodaj, kot bi si želeli. Vrstni red vresnosti v stolpcu, je ponovno določen z abecednim vrstnim redom, ker uporabljamo spremenljivko tipa **character**. Če uporabimo tip **factor**, pa bo vrstni red določen z vrednostmi v faktorju. Vrednosti moramo v *levels* podati od zgoraj navzdol.

```
anketa_fin$Odgovor <- factor(anketa_fin$Odgovor, levels = c("Zame ne velja", "Ne",
                                                            "Delno", "Da"))
```

```
ggplot(anketa_fin,
       aes(x = reorder(Vprasanje, ord), y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  theme(legend.position = "right") +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color="white",
            size=4)
```



V primeru so namesto stolpcev vzroki dejansko opisani s stavki in so v razporejeni čez več vrstic. Opis lahko zamenjamo kar tako, da popravimo vrednosti v stolpcu **Vprasanje**. Da to storimo moramo spet vsako vrednost zamenjati z drugo! Ta postopek smo že srečali, ko smo menjali 1 z "Da", 2 z "Delno" itd.

V primeru, da se nam postopek ponavlja, ga je smiselno posplošiti in napisati funkcijo. S tem enake kode ne popravljamo na več mestih v skripti ampak le na enem: v funkciji. Napišimo svojo funkcijo, ki bo znala zamenjati eno vrednost z drugo.

```
zamenjaj <- function(vektor, obstojece, nove){
  nov_vektor <- vector("logical", length = length(vektor))
  for(i in 1:length(obstojece)){
    nov_vektor[vektor == obstojece[i]] <- nove[i]
  }
  return(nov_vektor)
}
#primer
zamenjaj(c(1,2,3,1,3,2,2,1,2), c(1,2,3), c("a", "b", "c"))
```

```
## [1] "a" "b" "c" "a" "c" "b" "b" "a" "b"
```

Funkcija sprejme 3 vektorje. Prvi vektor želimo spremeniti, v drugem vektorju moramo podati vrednosti, ki jih želimo menjati in v tretjem nove vrednosti. Drugi vektor lahko programsko določimo z *unique*, vendar je včasih preglednejše, da podamo oba vektroja, kjer lahko spreminjamo vrstni red.

Postopek menjave ene vrednosti z drugo je zelo pogost, zato že obstajajo druge funkcije, ki jih lahko uporabite, kot so naprimer **case_when()**, **recode()**, **gsub()**. Vsaka funkcija ima svoje posebnosti, naša funkcija **zamenjaj** zna zamenjati le eno vrednost z drugo. Ne moremo pa recimo zamenjati vseh negativnih števil z nizom "negativno".

Vseeno jo uporabimo.

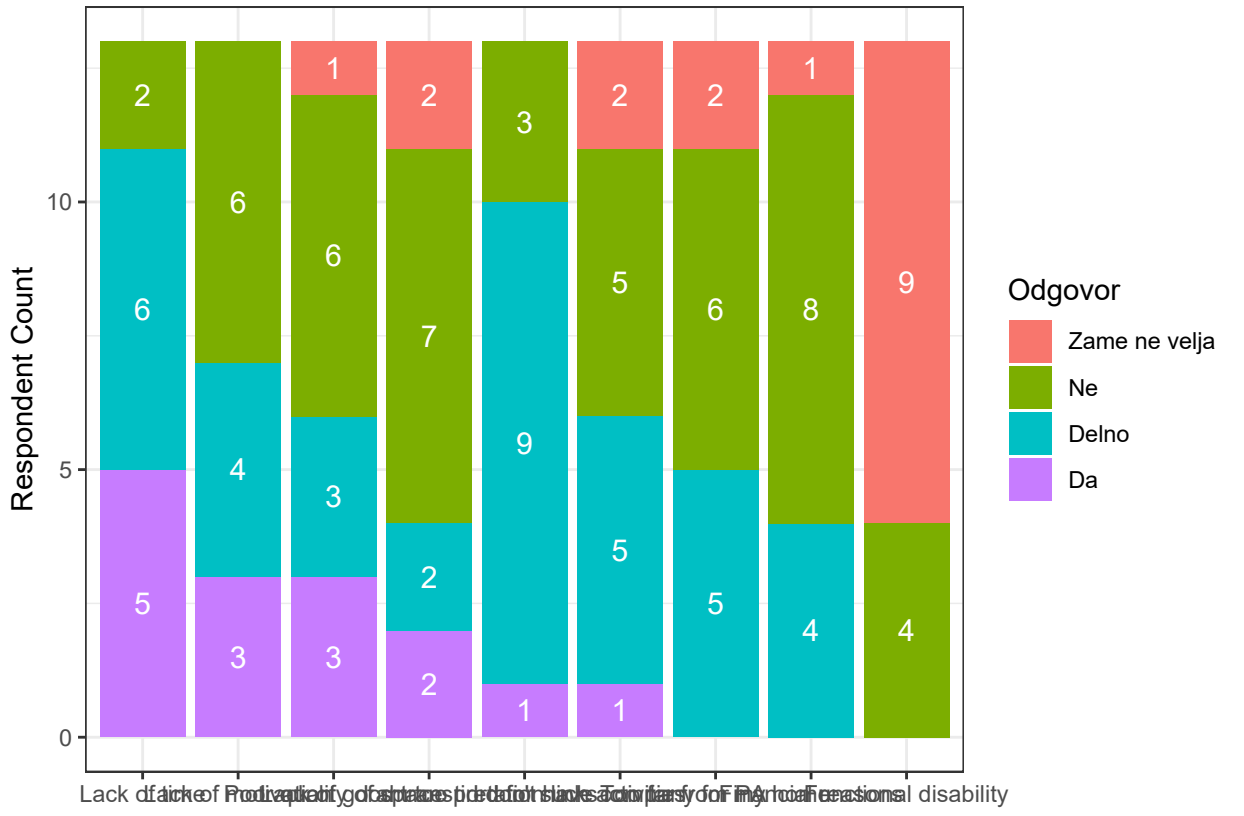
```
unique(anketa_fin$Vprasanje)
```

```
## [1] Q2a_too_far            Q2b_no_connections     Q2c_not_quality
## [4] Q2d_disabled           Q2e_no_time            Q2f_finance
## [7] Q2g_no_company         Q2h_too_tired          Q2i_no_need_motivation
## 9 Levels: Q2a_too_far Q2b_no_connections Q2c_not_quality ... Q2i_no_need_motivation
```

```
opisi <- zamenjaj(anketa_fin$Vprasanje, unique(anketa_fin$Vprasanje),
                  c("Too far from my home",
                    "Lack of good transportation links",
                    "Poor quality of space",
                    "Functional disability",
                    "Lack of time",
                    "Financial reasons",
                    "I don't have company for PA",
                    "I am too tired for such activities",
                    "Lack of motivation"))
anketa_fin$Vprasanje <- opisi
```

Če sedaj izrišemo graf vidimo, da imamo zaradi daljših opisov še več prekrivanj teksta na x osi. To lahko rešimo ročno, tako da vnesemo znak "\n", ki pomeni novo vrstico v naše nize.

```
ggplot(anketa_fin, aes(x = reorder(Vprasanje, ord), y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  theme(legend.position = "right") +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color="white",
            size=4)
```



Uporabimo pa lahko paket **stringr** za delo z nizi, ki že vsebuje funkcijo **str_wrap**, ki nam razdeli nize na več vrstic. Preizkusimo:

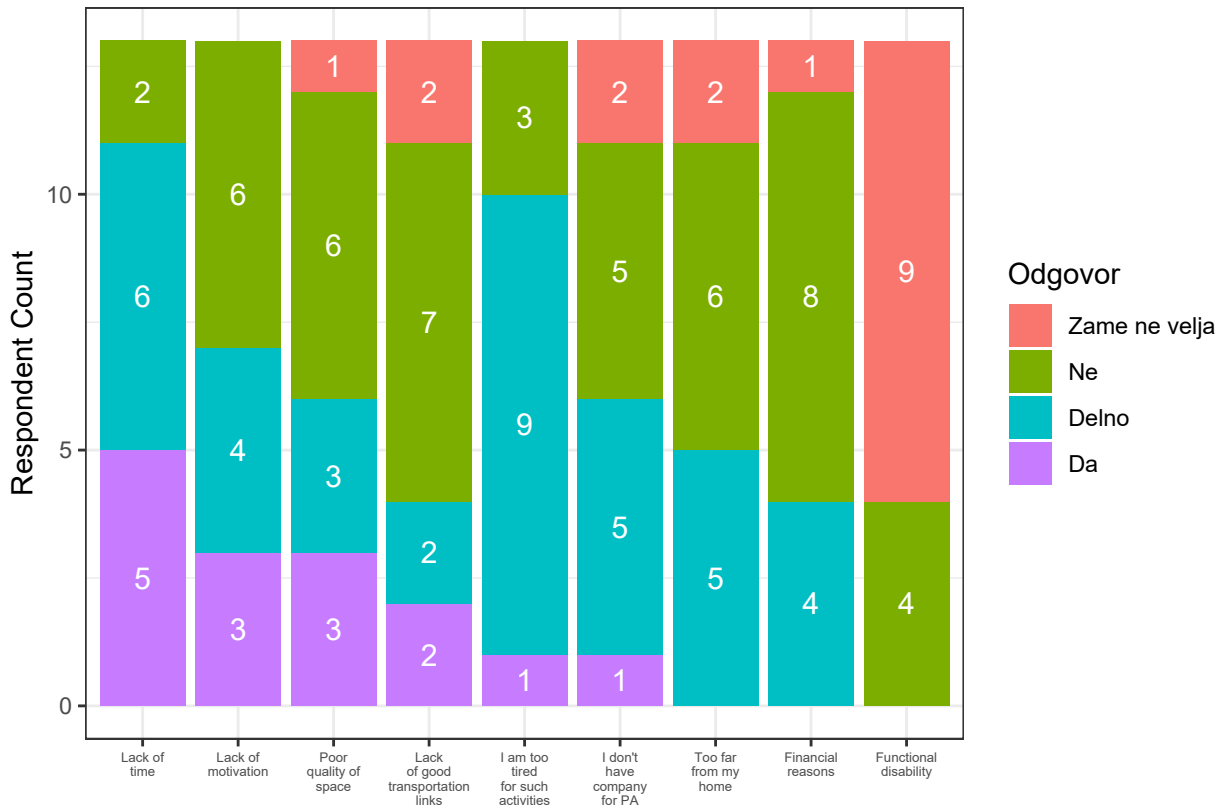
```
library(stringr)

str_wrap("Ta stavek je dokaj dolg!", width = 10)
```

```
## [1] "Ta stavek\nje dokaj\ndolg!"
```

Te spremembe lahko ponovno naredimo, kar v tabeli, lahko pa jih uporabimo le pri izpisu diagrama, tako da definiramo svojo funkcijo kar znotraj klica **ggplot**. Tukaj moramo parameter *width* določiti na roke. Primerna širina je nekoliko odvisna od velikosti izhodne slike, ki pa se razlikuje, če želimo sliko shraniti v ta izroček oziroma, če sliko izrišemo na celoten ekran. Če so stavki zelo dolgi lahko spremenimo še velikost pisave z `theme(axis.text.x = element_text(size = 5))`. Če boste poganjali kodo izven te skripte, lahko nastavite `width = 15` in spustite zmanjševanje teksta na x osi.

```
ggplot(anketa_fin, aes(x = reorder(Vprasanje, ord), y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  theme(legend.position = "right") +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color="white",
            size=4) +
```

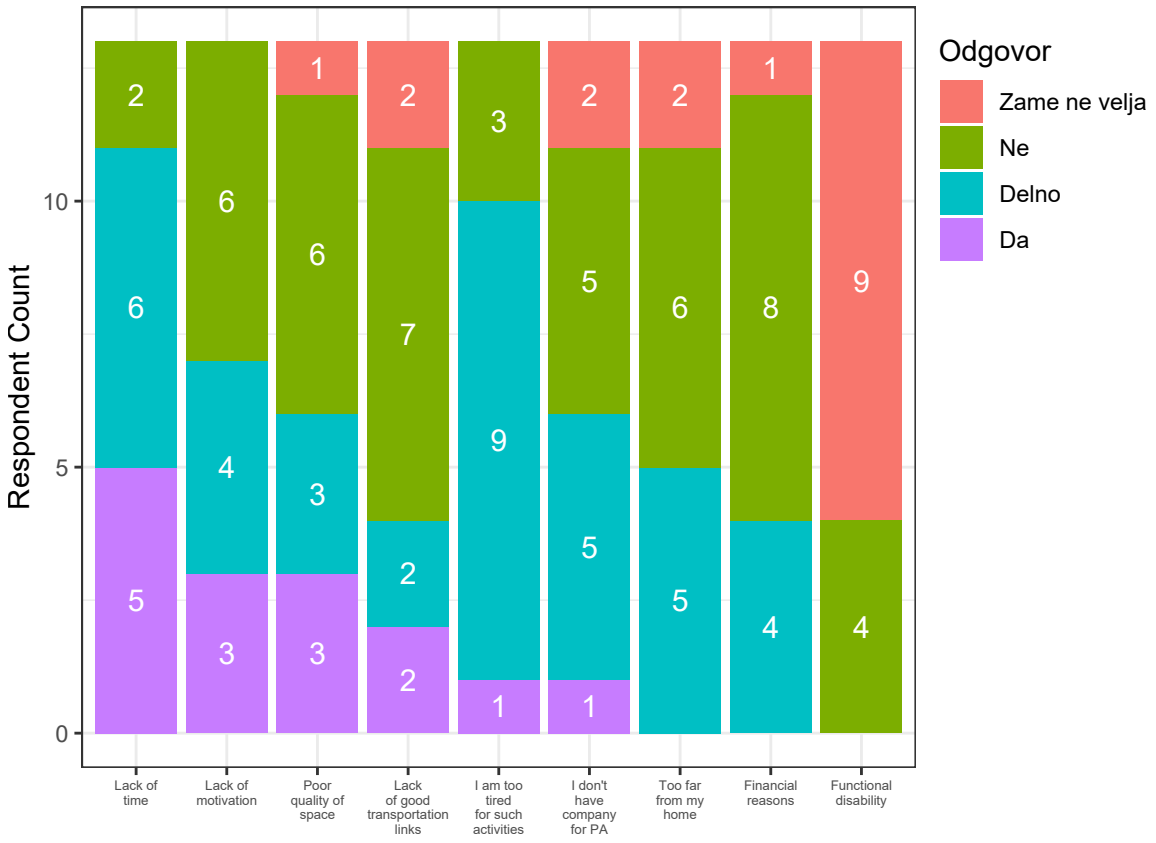
```
  scale_x_discrete(labels = function(x) {str_wrap(x, width = 10)}) +
  theme(axis.text.x = element_text(size = 5))
```



V primeru vidimo, da je legenda desno zgoraj. **ggplot** nam privzeto izriše legendo na desni, imamo pa še možnosti, da jo izrišemo levo, spodaj ali zgoraj s tem, da nastavimo `legend.position` na eno izmed vrenosti "left", "bottom" ali "top". Če želimo legendo podati na poljubno mesto lahko podamo par števil *c(x, y)*, kjer vrednost c(0, 0) predstavlja spodnji levi rob c(1, 1), pa zgornji desni rob. Privzeto bo na izrani poziciji sredina legende, če pa želimo na tej poziciji izbrati levi zgornji kot, pa to podobno nastavimo z `legend.justification = c(0,1)`. V primeru, da legendo ročno premaknemo z `c(x,y)` se izriše le graf brez dodatnega prostora, ki pa ga lahko dodamo na desno z parametrom `plot.margin = margin(top, right, bottom, left, unit)`. S to funkcijo spodaj dodamo diagramu 4.5 cm prosotra na desni.

```
ggplot(anketa_fin, aes(x = reorder(Vprasanje, ord), y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  theme(legend.position = c(1.01, 0.98),
        legend.justification = c(0, 1)) +
  theme(plot.margin = margin(0, 4.5, 0, 0, "cm")) +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color="white",
            size=4) +
```

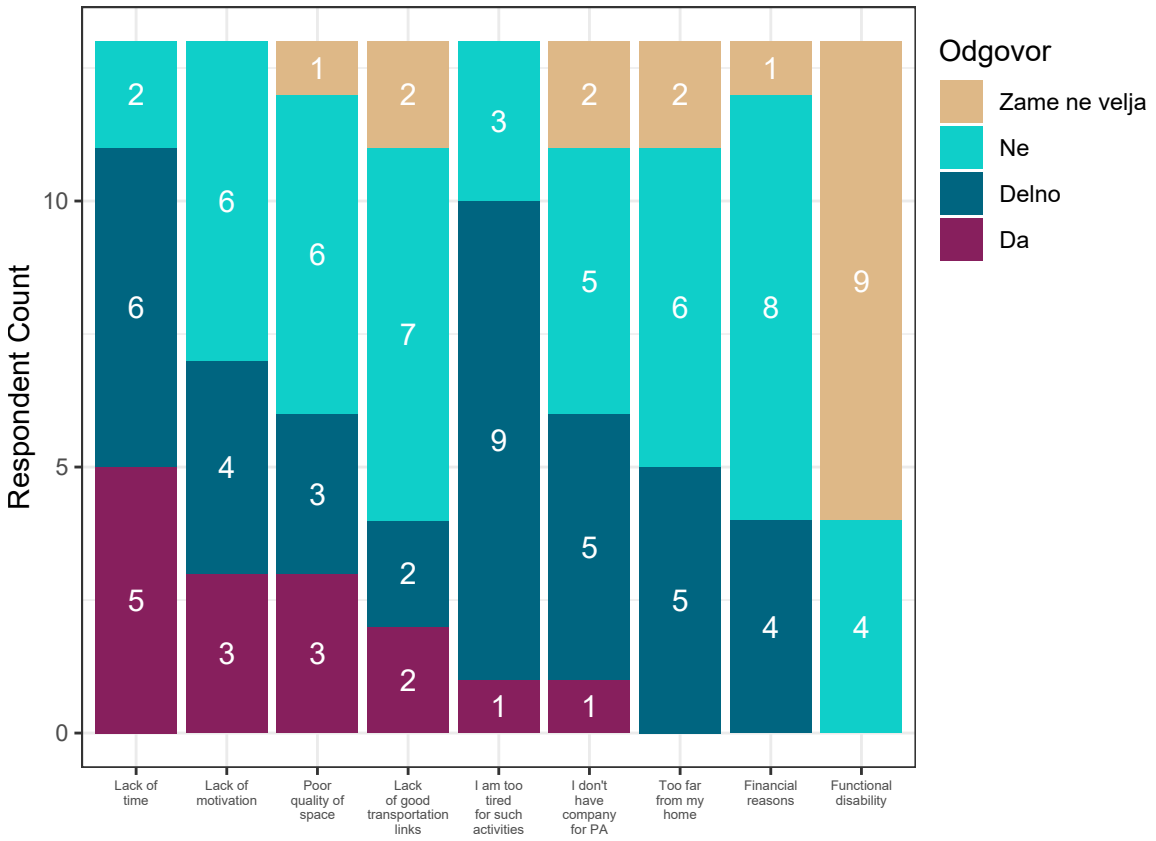
```
  scale_x_discrete(labels = function(x) {str_wrap(x, width = 10)}) +
  theme(axis.text.x = element_text(size = 5))
```



Na koncu zamenjajmo še barve na grafu! Ročno jih lahko izbiramo s funkcijo **scale_fill_manual** ali pa uporabimo že privzete palete z **scale_fill_brewer**. Barve lahko izbiramo z že privetimi imeni, funkcijo **rgb** ali s tem, da napišemo vrednosti rgb v heksadecimalni obliki kot niz. Poglejmo na spodnjem primeru.

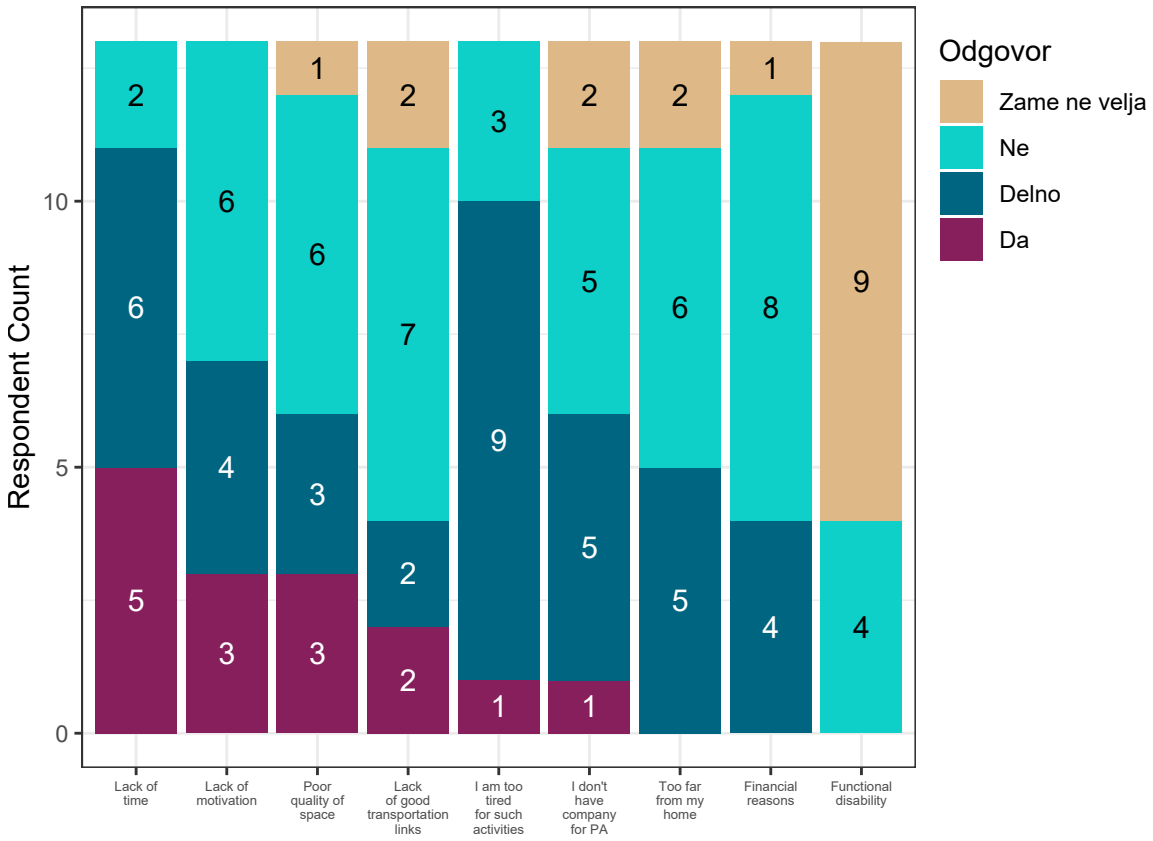
```
ggplot(anketa_fin, aes(x = reorder(Vprasanje, ord), y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  theme(legend.position = c(1.01, 0.98),
        legend.justification = c(0, 1)) +
  theme(plot.margin = margin(0, 4.5, 0, 0, "cm")) +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color="white",
            size=4) +
  scale_x_discrete(labels = function(x) {str_wrap(x, width = 10)}) +
  theme(axis.text.x = element_text(size = 5)) +
  scale_fill_manual(values = c("burlywood",#lahko po imenih(blizu (224, 189, 141))
                               rgb(15, 207, 201, maxColorValue = 255),
                               rgb(0, 101, 128, maxColorValue = 255),
```

```
                                "#871F5D"))
```



Spremenimo še barvo teksta glede na vrednost, ki jo prikazuje. Tukaj lahko preprosto uporabimo našo že napisano funkcijo **zamenjaj**!

```
ggplot(anketa_fin, aes(x = reorder(Vprasanje, ord), y = Stevilo, fill = Odgovor)) +
  geom_bar(stat = "identity") +
  #belo ozadje
  theme_bw() +
  ylab("Respondent Count") +
  xlab("") +
  theme(legend.position = c(1.01, 0.98),
        legend.justification = c(0, 1)) +
  theme(plot.margin = margin(0, 4.5, 0, 0, "cm")) +
  #Dodamo tekst na graf
  geom_text(aes(label=Stevilo),
            position = position_stack(vjust = 0.5),
            color=zamenjaj(anketa_fin$Odgovor, c("Zame ne velja", "Ne", "Delno", "Da"),
                           c("black", "black", "white", "white")),
            size=4) +
  scale_x_discrete(labels = function(x) {str_wrap(x, width = 10)}) +
  theme(axis.text.x = element_text(size = 5)) +
  scale_fill_manual(values = c("burlywood",#lahko po imenih(blizu (224, 189, 141))
                               rgb(15, 207, 201, maxColorValue = 255),
                               rgb(0, 101, 128, maxColorValue = 255),
                               "#871F5D"))
```



Organizacija podatkov in krajša analiza

Poglejmo si dobljene podatke:

```
data <- read.table("C:/delavnice/R-za-neprogramerje/Predavanje_08/data_raw/test_data.csv",
                   header = T,
                   dec = ".",
                   sep = ";",
                   quote = ""
                   )
head(data)
```

```
##   Sample_ID Sample Replicate    Cq   Tm
## 1      01_1      1         1 15.95 81.0
## 2      01_2      1         2 15.93 81.2
## 3      01_3      1         3 15.85 81.2
## 4      02_1      2         1 15.94 81.3
## 5      02_2      2         2 15.97 81.2
## 6      02_3      2         3 15.99 81.2
```

Imamo rezultate bioloških poskusov, kjer je vsak poskus narejen v treh tehničnih replikantih. Radi bi opravljali analizo tako, da računamo statistike (povprečja) vsakega poskusa posebej. Pri analizi bi radi ohranili tudi ohranili vse podatke in okvirno naredili naslednjo analizo:

1. Preveri Tm vrednosti in zavrzi vse podatke, ki imajo Tm vrednost izven območja 81.0-81.7.
2. Za vsak vzorec preveri outlierje: če se Cq vrednost replikata razlikuje za >1 od mediane treh replikatov, jo zavrzi.

3. Če ima vzorec vsaj dva pozitivna replikata, je pozitiven, sicer negativen.
4. Izračunaj povprečno Cq za vse pozitivne vzorce.

Za takšno analizo so podatki že v primerni obliki in nam jih ni potrebno preoblikovati z **pivot_longer** ali **pivot_wider**. Poglejmo si par primerov, kako bi naredili takšno analizo.

Rešitev 1

Najprej poglejmo rešitev z snovjo, ki smo jo že spoznali. Rešitev bo sicer odstranjevala vrstice, za katere med analizo ugotovimo, da so neustrezne.

```
dim(data)
```

```
## [1] 30  5
```

Imejmo v mislih, da imamo na začetku 10 poskusov, vsak pa ima 3 replikante, torej imamo 30 vrstic. Posamezni koraki bodo tudi shranjeni v svoje tabele, kar seveda lahko preskočimo.

Korak 1

Preveri Tm vrednosti in zavrzi vse podatke, ki imajo Tm vrednost izven območja 81.0-81.7.

```
step1 <- data[data$Tm >= 81.0 & data$Tm <= 81.7,]
dim(step1)
```

```
## [1] 22  5
```

Korak 2

Za vsak vzorec preveri outlierje: če se Cq vrednost replikata razlikuje za >1 od mediane treh replikatov, jo zavrzi.

Najprej opazimo, da je vrednost **Cq** tipa **character**, ker vsebuje tudi znak "N". Če pretvorimo te podatke v numerične se bo ta vrednost zamenjala z **NA**. Ker so to manjkajoči podatki jih nato tudi odstranimo.

```
step2 <- step1
step2$Cq <- as.numeric(step2$Cq)
step2 <- step2[!is.na(step2$Cq),]
dim(step2)
```

```
## [1] 21  5
```

Sedaj pa lahko izračunamo mediano za vsako skupino. Tukaj si lahko pomagamo s funkcijo **aggregate**.

```
mediane <- aggregate(step2$Cq, by = list(step2$Sample), FUN = median, na.rm = T)
names(mediane) <- c("skupina", "mediana")
mediane
```

```
##   skupina mediana
## 1       1   15.93
## 2       2   15.97
## 3       3   17.22
## 4       4   28.47
## 5       5   21.34
## 6       6   34.21
## 7       7   19.85
## 8       8   21.31
## 9      10   23.35
```

Pripnimo te mediane k našim podatkom. Ker sta tabeli različni moramo podatke združiti s funkcijo **merge**. Funkciji podamo obe tabeli in z parametroma `by.x` in `by.y` povemo, po katerih vrednostih naj združuje.

```
step2 <- merge(step2, mediane, by.x = "Sample", by.y = "skupina")
step2 <- step2[step2$mediana < step2$Cq + 1 & step2$mediana > step2$Cq - 1, ]
step2
```

```
##    Sample Sample_ID Replicate    Cq   Tm mediana
## 1       1      01_1         1 15.95 81.0   15.93
## 2       1      01_2         2 15.93 81.2   15.93
## 3       1      01_3         3 15.85 81.2   15.93
## 4       2      02_1         1 15.94 81.3   15.97
## 5       2      02_2         2 15.97 81.2   15.97
## 6       2      02_3         3 15.99 81.2   15.97
## 7       3      03_1         1 17.25 81.3   17.22
## 8       3      03_2         2 17.22 81.2   17.22
## 9       3      03_3         3 17.16 81.2   17.22
## 10      4      04_1         1 28.47 81.3   28.47
## 11      5      05_1         1 21.34 81.3   21.34
## 12      5      05_2         2 21.42 81.2   21.34
## 13      5      05_3         3 21.32 81.5   21.34
## 14      6      06_1         1 34.21 81.3   34.21
## 15      7      07_1         1 19.90 81.3   19.85
## 16      7      07_2         2 19.80 81.5   19.85
## 17      8      08_1         1 21.25 81.6   21.31
## 18      8      08_2         2 21.40 81.5   21.31
## 19      8      08_3         3 21.31 81.5   21.31
## 20     10      10_1         1 23.31 81.3   23.35
## 21     10      10_2         2 23.39 81.0   23.35
```

Korak 3

Če ima vzorec vsaj dva pozitivna replikata, je pozitiven, sicer negativen.

```
step3 <- step2
table(step3$Sample)
```

```
##
##  1  2  3  4  5  6  7  8 10
##  3  3  3  1  3  1  2  3  2
```

Z ukazom table lahko hitro vidimo, da imata poskus 4 in 6 le en replikant, poskus 9 pa celo manjka. Predpostavimo, da so pozitivni tisti, ki ostajajo. Dodajmo nov stolpec velikost skupine, k naši originalni tabeli. Uporabimo pa kar isti postopek kot v koraku 2, le da zamenjamo funkcijo **median** z **length**.

```
velikosti <- aggregate(step3$Sample, list(step3$Sample), FUN = length)
names(velikosti) <- c("Skupina", "n")
velikosti
```

```
##   Skupina n
## 1       1 3
## 2       2 3
## 3       3 3
## 4       4 1
## 5       5 3
## 6       6 1
## 7       7 2
## 8       8 3
## 9      10 2
```

```
step3 <- merge(step3, velikosti, by.x = "Sample", by.y = "Skupina")
step3 <- step3[step3$n > 1,]
step3
```

```
##    Sample Sample_ID Replicate    Cq   Tm mediana n
## 1       1      01_1         1 15.95 81.0   15.93 3
## 2       1      01_2         2 15.93 81.2   15.93 3
## 3       1      01_3         3 15.85 81.2   15.93 3
## 4       2      02_1         1 15.94 81.3   15.97 3
## 5       2      02_2         2 15.97 81.2   15.97 3
## 6       2      02_3         3 15.99 81.2   15.97 3
## 7       3      03_1         1 17.25 81.3   17.22 3
## 8       3      03_2         2 17.22 81.2   17.22 3
## 9       3      03_3         3 17.16 81.2   17.22 3
## 11      5      05_1         1 21.34 81.3   21.34 3
## 12      5      05_2         2 21.42 81.2   21.34 3
## 13      5      05_3         3 21.32 81.5   21.34 3
## 15      7      07_1         1 19.90 81.3   19.85 2
## 16      7      07_2         2 19.80 81.5   19.85 2
## 17      8      08_1         1 21.25 81.6   21.31 3
## 18      8      08_2         2 21.40 81.5   21.31 3
## 19      8      08_3         3 21.31 81.5   21.31 3
## 20     10      10_1         1 23.31 81.3   23.35 2
## 21     10      10_2         2 23.39 81.0   23.35 2
```

Korak 4

Izračunaj povprečno Cq za vse pozitivne vzorce.

Spet po istem postopku spet izračunamo povprečje. Postopek se ponavlja... premislite, če bi naredili funkcijo.

```
step4 <- step3
povprecja <- aggregate(step4$Cq, list(step4$Sample), FUN = mean)
names(povprecja) <- c("Skupina", "Povprecje")
povprecja
```

```
##   Skupina Povprecje
## 1       1  15.91000
## 2       2  15.96667
## 3       3  17.21000
## 4       5  21.36000
## 5       7  19.85000
## 6       8  21.32000
## 7      10  23.35000
```

Od tukaj naprej lahko povprečja uporabljamo za nadaljnjo analizo ali pa jih pridružimo začetnim podatkom.

```
step4 <- merge(step3, povprecja, by.x = "Sample", by.y = "Skupina")
step4
```

```
##    Sample Sample_ID Replicate    Cq   Tm mediana n Povprecje
## 1       1      01_1         1 15.95 81.0   15.93 3  15.91000
## 2       1      01_2         2 15.93 81.2   15.93 3  15.91000
## 3       1      01_3         3 15.85 81.2   15.93 3  15.91000
## 4       2      02_1         1 15.94 81.3   15.97 3  15.96667
## 5       2      02_2         2 15.97 81.2   15.97 3  15.96667
```

```
## 6       2      02_3         3 15.99 81.2   15.97 3  15.96667
## 7       3      03_1         1 17.25 81.3   17.22 3  17.21000
## 8       3      03_2         2 17.22 81.2   17.22 3  17.21000
## 9       3      03_3         3 17.16 81.2   17.22 3  17.21000
## 10      5      05_1         1 21.34 81.3   21.34 3  21.36000
## 11      5      05_2         2 21.42 81.2   21.34 3  21.36000
## 12      5      05_3         3 21.32 81.5   21.34 3  21.36000
## 13      7      07_1         1 19.90 81.3   19.85 2  19.85000
## 14      7      07_2         2 19.80 81.5   19.85 2  19.85000
## 15      8      08_1         1 21.25 81.6   21.31 3  21.32000
## 16      8      08_2         2 21.40 81.5   21.31 3  21.32000
## 17      8      08_3         3 21.31 81.5   21.31 3  21.32000
## 18     10      10_1         1 23.31 81.3   23.35 2  23.35000
## 19     10      10_2         2 23.39 81.0   23.35 2  23.35000
```

Rešitev 2

Podobno kot prej, le da bomo ohranjali vse vrstice in si pri vsaki le vodili evidenco ali je aktualna ali ne. V tem postopku tudi ne bomo delali tabel za vsak korak ampak bomo uporabljali le eno *dataNA*.

Korak 1

Preveri Tm vrednosti in zavrzi vse podatke, ki imajo Tm vrednost izven območja 81.0-81.7.

```
dataNA <- data
dataNA$Valid <- TRUE
dataNA$Valid <- dataNA$Tm >= 81.0 & dataNA$Tm <= 81.7
dataNA
```

```
##    Sample_ID Sample Replicate    Cq   Tm Valid
## 1       01_1      1         1 15.95 81.0  TRUE
## 2       01_2      1         2 15.93 81.2  TRUE
## 3       01_3      1         3 15.85 81.2  TRUE
## 4       02_1      2         1 15.94 81.3  TRUE
## 5       02_2      2         2 15.97 81.2  TRUE
## 6       02_3      2         3 15.99 81.2  TRUE
## 7       03_1      3         1 17.25 81.3  TRUE
## 8       03_2      3         2 17.22 81.2  TRUE
## 9       03_3      3         3 17.16 81.2  TRUE
## 10      04_1      4         1 28.47 81.3  TRUE
## 11      04_2      4         2 28.64 79.9 FALSE
## 12      04_3      4         3     N 81.0  TRUE
## 13      05_1      5         1 21.34 81.3  TRUE
## 14      05_2      5         2 21.42 81.2  TRUE
## 15      05_3      5         3 21.32 81.5  TRUE
## 16      06_1      6         1 34.21 81.3  TRUE
## 17      06_2      6         2 33.97 68.8 FALSE
## 18      06_3      6         3     N 69.3 FALSE
## 19      07_1      7         1 19.90 81.3  TRUE
## 20      07_2      7         2 19.80 81.5  TRUE
## 21      07_3      7         3 19.84 80.5 FALSE
## 22      08_1      8         1 21.25 81.6  TRUE
## 23      08_2      8         2 21.40 81.5  TRUE
## 24      08_3      8         3 21.31 81.5  TRUE
## 25      09_1      9         1 36.13 68.8 FALSE
```

```
## 26      09_2      9         2 36.58 74.8 FALSE
## 27      09_3      9         3 35.85 68.3 FALSE
## 28      10_1     10         1 23.31 81.3  TRUE
## 29      10_2     10         2 23.39 81.0  TRUE
## 30      10_3     10         3     N 69.5 FALSE
```

Dodali smo stolpec **Valid**, ki nam pove ali je replikant pozitiven ali ne. Tiste, ki ne ustrezajo prvemu koraku označimo z **FALSE**.

Korak 2

Za vsak vzorec preveri outlierje: če se Cq vrednost replikata razlikuje za >1 od mediane treh replikatov, jo zavrzi.

Vrstice, kjer je Cq = "N" zavrnemo.

```
dataNA$Cq <- as.numeric(dataNA$Cq)
dataNA$Valid <- dataNA$Valid & !is.na(dataNA$Cq)
dataNA
```

```
##    Sample_ID Sample Replicate    Cq   Tm Valid
## 1       01_1      1         1 15.95 81.0  TRUE
## 2       01_2      1         2 15.93 81.2  TRUE
## 3       01_3      1         3 15.85 81.2  TRUE
## 4       02_1      2         1 15.94 81.3  TRUE
## 5       02_2      2         2 15.97 81.2  TRUE
## 6       02_3      2         3 15.99 81.2  TRUE
## 7       03_1      3         1 17.25 81.3  TRUE
## 8       03_2      3         2 17.22 81.2  TRUE
## 9       03_3      3         3 17.16 81.2  TRUE
## 10      04_1      4         1 28.47 81.3  TRUE
## 11      04_2      4         2 28.64 79.9 FALSE
## 12      04_3      4         3    NA 81.0 FALSE
## 13      05_1      5         1 21.34 81.3  TRUE
## 14      05_2      5         2 21.42 81.2  TRUE
## 15      05_3      5         3 21.32 81.5  TRUE
## 16      06_1      6         1 34.21 81.3  TRUE
## 17      06_2      6         2 33.97 68.8 FALSE
## 18      06_3      6         3    NA 69.3 FALSE
## 19      07_1      7         1 19.90 81.3  TRUE
## 20      07_2      7         2 19.80 81.5  TRUE
## 21      07_3      7         3 19.84 80.5 FALSE
## 22      08_1      8         1 21.25 81.6  TRUE
## 23      08_2      8         2 21.40 81.5  TRUE
## 24      08_3      8         3 21.31 81.5  TRUE
## 25      09_1      9         1 36.13 68.8 FALSE
## 26      09_2      9         2 36.58 74.8 FALSE
## 27      09_3      9         3 35.85 68.3 FALSE
## 28      10_1     10         1 23.31 81.3  TRUE
## 29      10_2     10         2 23.39 81.0  TRUE
## 30      10_3     10         3    NA 69.5 FALSE
```

Podobno kot prej izračunamo mediane, vendar pazimo, da uporabljamo le podatke, kjer je $Valid = T$.

```
mediane <- aggregate(dataNA$Cq[dataNA$Valid], by = list(dataNA$Sample[dataNA$Valid]), FUN = median, na.
names(mediane) <- c("skupina", "mediana")
dataNA <- merge(dataNA, mediane, by.x = "Sample", by.y = "skupina", all = TRUE)
```

```
dataNA$Valid <- dataNA$Valid & (dataNA$mediana < dataNA$Cq + 1 & dataNA$mediana > dataNA$Cq - 1)
dataNA
```

```
##    Sample Sample_ID Replicate    Cq   Tm Valid mediana
## 1       1      01_1         1 15.95 81.0  TRUE   15.93
## 2       1      01_2         2 15.93 81.2  TRUE   15.93
## 3       1      01_3         3 15.85 81.2  TRUE   15.93
## 4       2      02_1         1 15.94 81.3  TRUE   15.97
## 5       2      02_2         2 15.97 81.2  TRUE   15.97
## 6       2      02_3         3 15.99 81.2  TRUE   15.97
## 7       3      03_2         2 17.22 81.2  TRUE   17.22
## 8       3      03_1         1 17.25 81.3  TRUE   17.22
## 9       3      03_3         3 17.16 81.2  TRUE   17.22
## 10      4      04_3         3    NA 81.0 FALSE   28.47
## 11      4      04_1         1 28.47 81.3  TRUE   28.47
## 12      4      04_2         2 28.64 79.9 FALSE   28.47
## 13      5      05_1         1 21.34 81.3  TRUE   21.34
## 14      5      05_2         2 21.42 81.2  TRUE   21.34
## 15      5      05_3         3 21.32 81.5  TRUE   21.34
## 16      6      06_1         1 34.21 81.3  TRUE   34.21
## 17      6      06_2         2 33.97 68.8 FALSE   34.21
## 18      6      06_3         3    NA 69.3 FALSE   34.21
## 19      7      07_2         2 19.80 81.5  TRUE   19.85
## 20      7      07_1         1 19.90 81.3  TRUE   19.85
## 21      7      07_3         3 19.84 80.5 FALSE   19.85
## 22      8      08_3         3 21.31 81.5  TRUE   21.31
## 23      8      08_1         1 21.25 81.6  TRUE   21.31
## 24      8      08_2         2 21.40 81.5  TRUE   21.31
## 25      9      09_1         1 36.13 68.8 FALSE      NA
## 26      9      09_2         2 36.58 74.8 FALSE      NA
## 27      9      09_3         3 35.85 68.3 FALSE      NA
## 28     10      10_1         1 23.31 81.3  TRUE   23.35
## 29     10      10_2         2 23.39 81.0  TRUE   23.35
## 30     10      10_3         3    NA 69.5 FALSE   23.35
```

Korak 3

Če ima vzorec vsaj dva pozitivna replikata, je pozitiven, sicer negativen.

```
velikosti <- aggregate(dataNA$Sample[dataNA$Valid], list(dataNA$Sample[dataNA$Valid]), FUN = length)
names(velikosti) <- c("Skupina", "n")
dataNA <- merge(dataNA, velikosti, by.x = "Sample", by.y = "Skupina", all = TRUE)
dataNA$Valid <- dataNA$Valid & dataNA$n > 1
dataNA
```

```
##    Sample Sample_ID Replicate    Cq   Tm Valid mediana  n
## 1       1      01_1         1 15.95 81.0  TRUE   15.93  3
## 2       1      01_2         2 15.93 81.2  TRUE   15.93  3
## 3       1      01_3         3 15.85 81.2  TRUE   15.93  3
## 4       2      02_1         1 15.94 81.3  TRUE   15.97  3
## 5       2      02_2         2 15.97 81.2  TRUE   15.97  3
## 6       2      02_3         3 15.99 81.2  TRUE   15.97  3
## 7       3      03_1         1 17.25 81.3  TRUE   17.22  3
## 8       3      03_2         2 17.22 81.2  TRUE   17.22  3
## 9       3      03_3         3 17.16 81.2  TRUE   17.22  3
```

```
## 10      4      04_2         2 28.64 79.9 FALSE   28.47  1
## 11      4      04_3         3    NA 81.0 FALSE   28.47  1
## 12      4      04_1         1 28.47 81.3 FALSE   28.47  1
## 13      5      05_1         1 21.34 81.3  TRUE   21.34  3
## 14      5      05_2         2 21.42 81.2  TRUE   21.34  3
## 15      5      05_3         3 21.32 81.5  TRUE   21.34  3
## 16      6      06_1         1 34.21 81.3 FALSE   34.21  1
## 17      6      06_2         2 33.97 68.8 FALSE   34.21  1
## 18      6      06_3         3    NA 69.3 FALSE   34.21  1
## 19      7      07_1         1 19.90 81.3  TRUE   19.85  2
## 20      7      07_2         2 19.80 81.5  TRUE   19.85  2
## 21      7      07_3         3 19.84 80.5 FALSE   19.85  2
## 22      8      08_2         2 21.40 81.5  TRUE   21.31  3
## 23      8      08_3         3 21.31 81.5  TRUE   21.31  3
## 24      8      08_1         1 21.25 81.6  TRUE   21.31  3
## 25      9      09_1         1 36.13 68.8 FALSE      NA NA
## 26      9      09_2         2 36.58 74.8 FALSE      NA NA
## 27      9      09_3         3 35.85 68.3 FALSE      NA NA
## 28     10      10_1         1 23.31 81.3  TRUE   23.35  2
## 29     10      10_2         2 23.39 81.0  TRUE   23.35  2
## 30     10      10_3         3    NA 69.5 FALSE   23.35  2
```

Korak 4

Izračunaj povprečno Cq za vse pozitivne vzorce.

```
povprecja <- aggregate(dataNA$Cq[dataNA$Valid], list(dataNA$Sample[dataNA$Valid]), FUN = mean)
names(povprecja) <- c("Skupina", "Povprecje")
povprecja
```

```
##   Skupina Povprecje
## 1       1  15.91000
## 2       2  15.96667
## 3       3  17.21000
## 4       5  21.36000
## 5       7  19.85000
## 6       8  21.32000
## 7      10  23.35000
```

Ponovno lahko združimo podatke.

```
dataNA <- merge(dataNA, povprecja, by.x = "Sample", by.y = "Skupina", all = T)
dataNA
```

```
##    Sample Sample_ID Replicate    Cq   Tm Valid mediana  n Povprecje
## 1       1      01_1         1 15.95 81.0  TRUE   15.93  3  15.91000
## 2       1      01_3         3 15.85 81.2  TRUE   15.93  3  15.91000
## 3       1      01_2         2 15.93 81.2  TRUE   15.93  3  15.91000
## 4       2      02_1         1 15.94 81.3  TRUE   15.97  3  15.96667
## 5       2      02_2         2 15.97 81.2  TRUE   15.97  3  15.96667
## 6       2      02_3         3 15.99 81.2  TRUE   15.97  3  15.96667
## 7       3      03_1         1 17.25 81.3  TRUE   17.22  3  17.21000
## 8       3      03_2         2 17.22 81.2  TRUE   17.22  3  17.21000
## 9       3      03_3         3 17.16 81.2  TRUE   17.22  3  17.21000
## 10      4      04_2         2 28.64 79.9 FALSE   28.47  1        NA
## 11      4      04_3         3    NA 81.0 FALSE   28.47  1        NA
```

```
## 12      4      04_1         1 28.47 81.3 FALSE   28.47  1        NA
## 13      5      05_3         3 21.32 81.5  TRUE   21.34  3  21.36000
## 14      5      05_1         1 21.34 81.3  TRUE   21.34  3  21.36000
## 15      5      05_2         2 21.42 81.2  TRUE   21.34  3  21.36000
## 16      6      06_2         2 33.97 68.8 FALSE   34.21  1        NA
## 17      6      06_3         3    NA 69.3 FALSE   34.21  1        NA
## 18      6      06_1         1 34.21 81.3 FALSE   34.21  1        NA
## 19      7      07_2         2 19.80 81.5  TRUE   19.85  2  19.85000
## 20      7      07_1         1 19.90 81.3  TRUE   19.85  2  19.85000
## 21      7      07_3         3 19.84 80.5 FALSE   19.85  2  19.85000
## 22      8      08_1         1 21.25 81.6  TRUE   21.31  3  21.32000
## 23      8      08_2         2 21.40 81.5  TRUE   21.31  3  21.32000
## 24      8      08_3         3 21.31 81.5  TRUE   21.31  3  21.32000
## 25      9      09_1         1 36.13 68.8 FALSE      NA NA        NA
## 26      9      09_3         3 35.85 68.3 FALSE      NA NA        NA
## 27      9      09_2         2 36.58 74.8 FALSE      NA NA        NA
## 28     10      10_1         1 23.31 81.3  TRUE   23.35  2  23.35000
## 29     10      10_2         2 23.39 81.0  TRUE   23.35  2  23.35000
## 30     10      10_3         3    NA 69.5 FALSE   23.35  2  23.35000
```

Iz celotne množice lahko sedaj preprosto tudi dobimo povprečja, kjer je vidno tudi kateri poskusi so negativni.

```
unique(dataNA[, c("Sample", "Povprecje")])
```

```
##    Sample Povprecje
## 1       1  15.91000
## 4       2  15.96667
## 7       3  17.21000
## 10      4        NA
## 13      5  21.36000
## 16      6        NA
## 19      7  19.85000
## 22      8  21.32000
## 25      9        NA
## 28     10  23.35000
```

Rešitev 3

Ta rešitev prikazuje, da so podatki že v zelo čisti obliki in se jih je preprosto uporabiti za analizo v okolju paketov **tidyverse**, ki naredi delo z R-jem še bolj pregledno.

Iz tega okolja smo že uporabljali funkcije **ggplot**, **pivot_longer** in **pivot_wider**.

Naslednja rešitev uporablja pipe operator %>%, ki le poda podatke iz leve strani na desno kot prvi argument. Primer:

```
sin(c(1,2,3))
```

```
## [1] 0.8414710 0.9092974 0.1411200
```

```
c(1,2,3) %>% sin()
```

```
## [1] 0.8414710 0.9092974 0.1411200
```

Pogljmo našo analizo:

```
library(dplyr)
```

```
##
```

```
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data$Cq <- as.numeric(data$Cq) #tudi to bi lahko dodali v pipe
data %>%
  filter(Tm >= 81.0 & Tm <= 81.7) %>% #Korak 1
  group_by(Sample) %>% #Določimo, da želimo delati glede na Sample
  mutate(mediana = median(Cq, na.rm = T)) %>% #izračun median
  filter(mediana < Cq + 1 & mediana > Cq - 1) %>%#filtiranje outlierjev (Korak2)
  mutate(velikost = n()) %>% #Izračun velikosti
  filter(velikost > 1) %>% #Korak 3
  mutate(povprecje = mean(Cq)) %>% #Izračun povprečji (Korak 4)
  select(Sample, povprecje) %>% #Izberemo le dva stolpca
  unique() #Samo unikatne vrstice
```

```
## # A tibble: 7 x 2
## # Groups:   Sample [7]
##   Sample povprecje
##    <int>     <dbl>
## 1      1      15.9
## 2      2      16.0
## 3      3      17.2
## 4      5      21.4
## 5      7      19.8
## 6      8      21.3
## 7     10      23.4
```

Avtomatsko generiranje kombinacij in shranjevanje slik

Včasih želimo narediti enako analizo za več različnih podmnožic stolpcev. Če želimo našteti vse različne kombinacije lahko to preprosto naredimo s funkcijo **combn**.

Poglejmo primer z uporabo podatkov **mtcars**.

```
data("mtcars")
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Najprej naštejmo vse kombinacije stolpcev. Funkcija **combn** naredi vse kombinacije podanega vektorja, velikost kombinacij pa izberemo z drugim argumentom.

```
combn(names(data), 3)
```

```
##      [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
```

```
## [1,] "Sample_ID" "Sample_ID" "Sample_ID" "Sample_ID" "Sample_ID" "Sample_ID"
## [2,] "Sample"    "Sample"    "Sample"    "Replicate" "Replicate" "Cq"
## [3,] "Replicate" "Cq"        "Tm"        "Cq"        "Tm"        "Tm"
##      [,7]        [,8]        [,9]     [,10]
## [1,] "Sample"    "Sample"    "Sample" "Replicate"
## [2,] "Replicate" "Replicate" "Cq"     "Cq"
## [3,] "Cq"        "Tm"        "Tm"     "Tm"
```

Če želimo, da se vrednosti tudi ponavljajo lahko uporabimo **expand.grid**. Tej funkciji podamo več vektorjev in nam vrne vse kombinacije vrednosti.

```
test <- expand.grid(names(data), names(data), names(data))
head(test)
```

```
##        Var1      Var2      Var3
## 1 Sample_ID Sample_ID Sample_ID
## 2    Sample Sample_ID Sample_ID
## 3 Replicate Sample_ID Sample_ID
## 4        Cq Sample_ID Sample_ID
## 5        Tm Sample_ID Sample_ID
## 6 Sample_ID    Sample Sample_ID
```

Radi bi izrisali ternarni graf za vse možne kombinacije atributov podatkovne množice **mtcars**. Naložimo najprej paket library in si poglejmo primer takega diagrama. Pozor paket **ggtern** po prepisal nekatere funkcije **ggplota**. Če jih želite uporabljati morate ponovno naložiti paket **ggplot** z ´library(ggplot)´.

Diagram za stolpce **mpg**, **qsec** in **wt**.

```
library(ggtern)
```

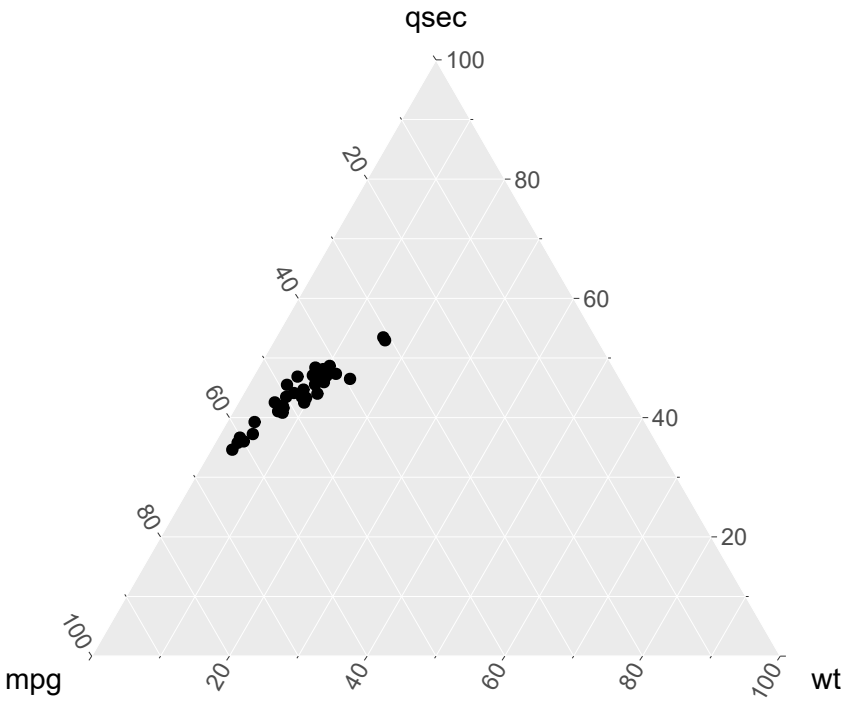
```
## Registered S3 methods overwritten by 'ggtern':
##   method           from
##   grid.draw.ggplot ggplot2
##   plot.ggplot      ggplot2
##   print.ggplot     ggplot2
```

```
## --
## Remember to cite, run citation(package = 'ggtern') for further info.
## --
```

```
##
## Attaching package: 'ggtern'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     aes, annotate, ggplot, ggplot_build, ggplot_gtable, ggplotGrob,
##     ggsave, layer_data, theme_bw, theme_classic, theme_dark,
##     theme_gray, theme_light, theme_linedraw, theme_minimal, theme_void
```

```
ggtern(data=mtcars,aes(x=mpg,y=qsec, z=wt)) +
  geom_point()
```



Za nameščanje paketa **ggtern** potrebujete trenutno najnovejšo verzijo R-ja in sicer 4.2.2. Če imate starejši R vas bo pri nalaganju paketa le-ta opozoril, da imate zastarelo verzijo namespace methods. Ker je to osnovni del R-ja ga morate ponovno namestiti. Vse lahko poženete tudi direktno iz R-ja oziroma je bolje da uporabite RGui.

```
#Če imate težave z namespace (methods)
install.packages(installr)
library("installr")
updateR() #iz RGui - posodobi instalacijo R-ja

install.packages("ggtern") #naložite še paket
```

Shranimo si vse kombinacije, ki jih potrebujemo. Za lažje delo bomo kombinacije tudi transponirali - zamenjali vrstice in stolpce v tabeli z ukazom **t**.

```
kombinacije <- combn(names(mtcars), 3)
dim(kombinacije)
```

```
## [1]   3 165
```

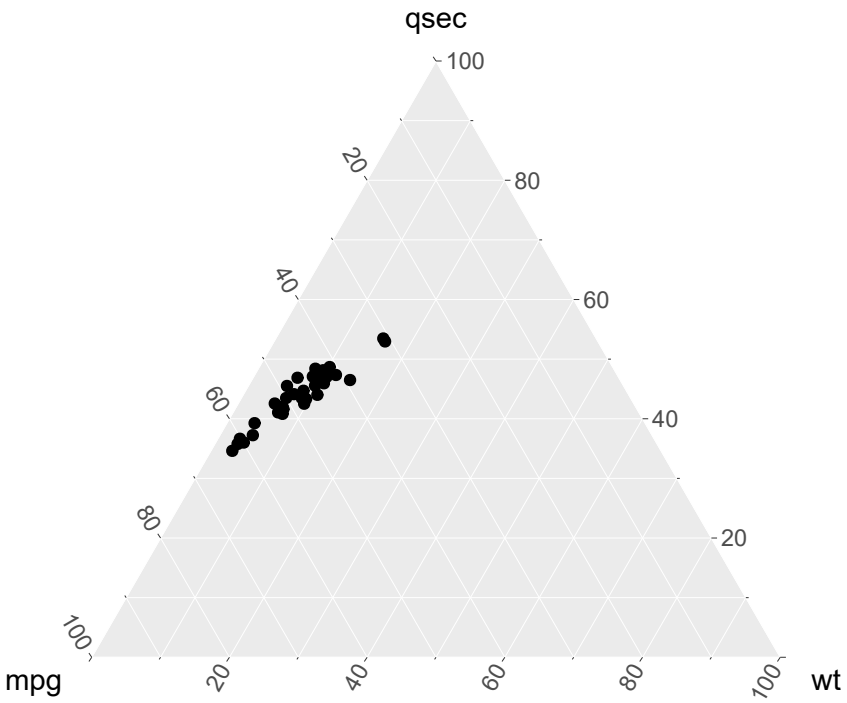
```
kombinacije <- t(kombinacije)
head(kombinacije)
```

```
##      [,1]  [,2]  [,3]
## [1,] "mpg" "cyl" "disp"
## [2,] "mpg" "cyl" "hp"
## [3,] "mpg" "cyl" "drat"
## [4,] "mpg" "cyl" "wt"
```

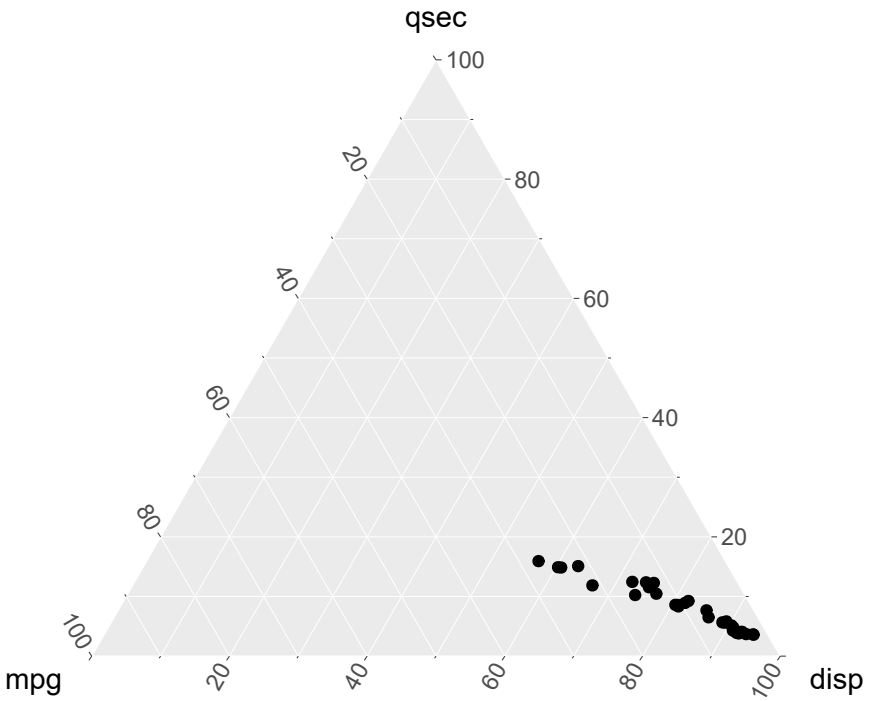
```
## [5,] "mpg" "cyl" "qsec"
## [6,] "mpg" "cyl" "vs"
```

Spišimo sedaj funkcijo, ki nam izriše ternarni diagram za podane stolpce. Pozor: tokrat smo za aestetike namesto **aes** uporabili **aes_string** za katerega so se razvijalci paketa odločili, da ga čez nekaj časa ne bodo več podpirali. Če ste opazili pri uporabi **ggplot** in posledično **ggplot** imen stolpcev ni potrebno navesti v narekovajih, kar privede do težje uporabe, če imamo ime stolpca shranjeno v spremenljivki. Uporaba **aes_string** je v tem primeru najpreprostejša rešitev.

```
izrisi_ternarni_diagram <- function(stolpci, podatki){
  ggtern(data=podatki,aes_string(x = stolpci[1], y = stolpci[2],z = stolpci[3])) +
    geom_point()
}
izrisi_ternarni_diagram(c("mpg", "qsec", "wt"), mtcars)
```




```
izrisi_ternarni_diagram(c("mpg", "qsec", "disp"), mtcars)
```



Zgornja funkcija samo prikazuje rešitev za izbiro stolpcev, napišimo sedaj funkcijo, ki bo naše grafe shranila na disk. Če želite pognati naslednji izsek kode, morate v mapi *data_raw* ustvariti še mapo *ternarni_diagrami*.

```
setwd("C:/delavnice/R-za-neprogramerje/Predavanje_08")
shrani_ternarni_diagram <- function(stolpci, podatki){
  #izrisi graf in ga shrani
  slika <- ggtern(data=podatki,aes_string(x = stolpci[1], y = stolpci[2],z = stolpci[3])) +
    geom_point()
  #ustvari ime datoteke in jo shrani (lahko dodamo parametre)
  pot <- paste(getwd(),
               "./data_raw/ternarni_diagrami/",
               stolpci[1], "_",
               stolpci[2], "_",
               stolpci[3], "_",
               ".png", sep = "")
  print(pot)
  ggsave(pot, slika)
}
#test shranjevanja ene slike
shrani_ternarni_diagram(c("mpg", "qsec", "wt"), mtcars)
#generirano in shranimo vse slike
apply(kombinacije, 1, shrani_ternarni_diagram, podatki = mtcars)
```

Diagrami so v mapi shranjeni z imenom stolpec1_stolpec2_stolpec3.png, kjer je stolpec dejansko ime izbranega stolpca.

Po zagonu bi morali v mapi *ternarni_diagrami* videti shranjene diagrame.

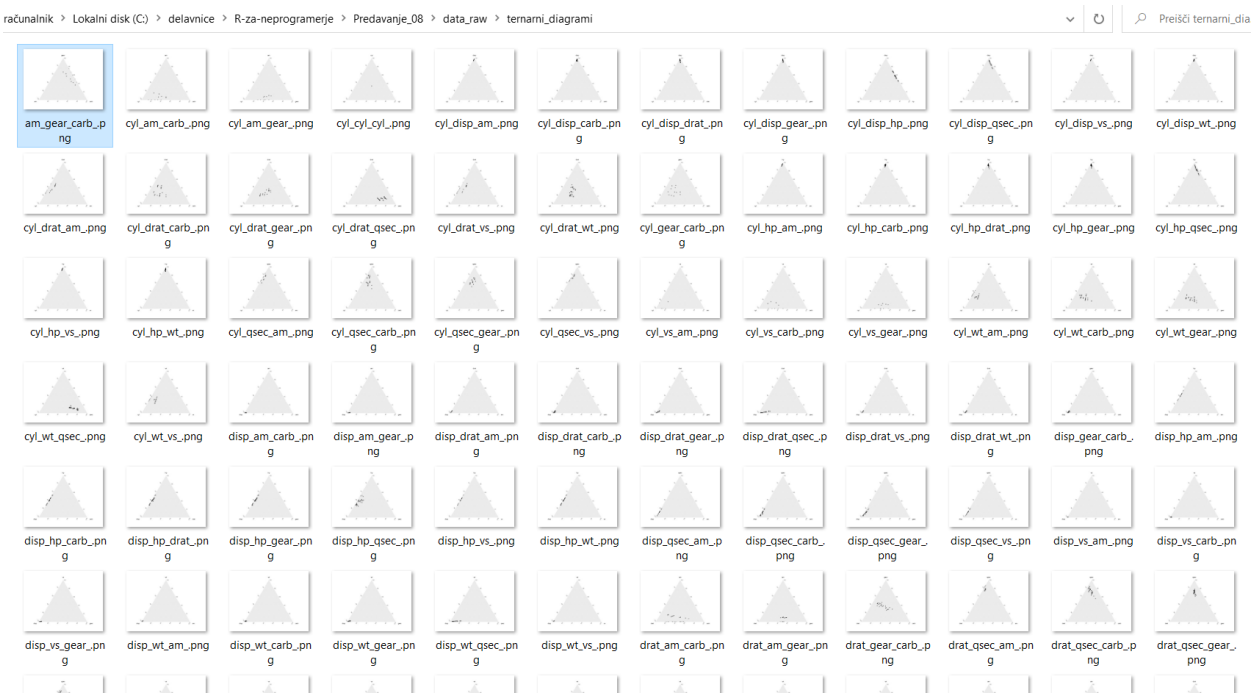


Figure 2: Shranjeni diagrami.

Osamelci - outliers

Definicija **osamelcev** (ang. **outlier**) je široko področje in njihova zaznava je odvisna od primera do primera. V poglavju Organizacija podatkov in krajša analiza smo imeli definicijo osamelcev podano vnaprej.

Nekaj klasičnih pristopov lahko dobite na https://statsandr.com/blog/outliers-detection-in-r/ po katerem je povzet tudi ta del skripte.

Osamelci so v splošnem vrednosti, ki (preveč) odstopajo od naših podatkov. Lahko so posledica napačnih meritev ali pa le redkih dogodkov za katereme moramo premisliti ali jih je sploh smiselno odstraniti.

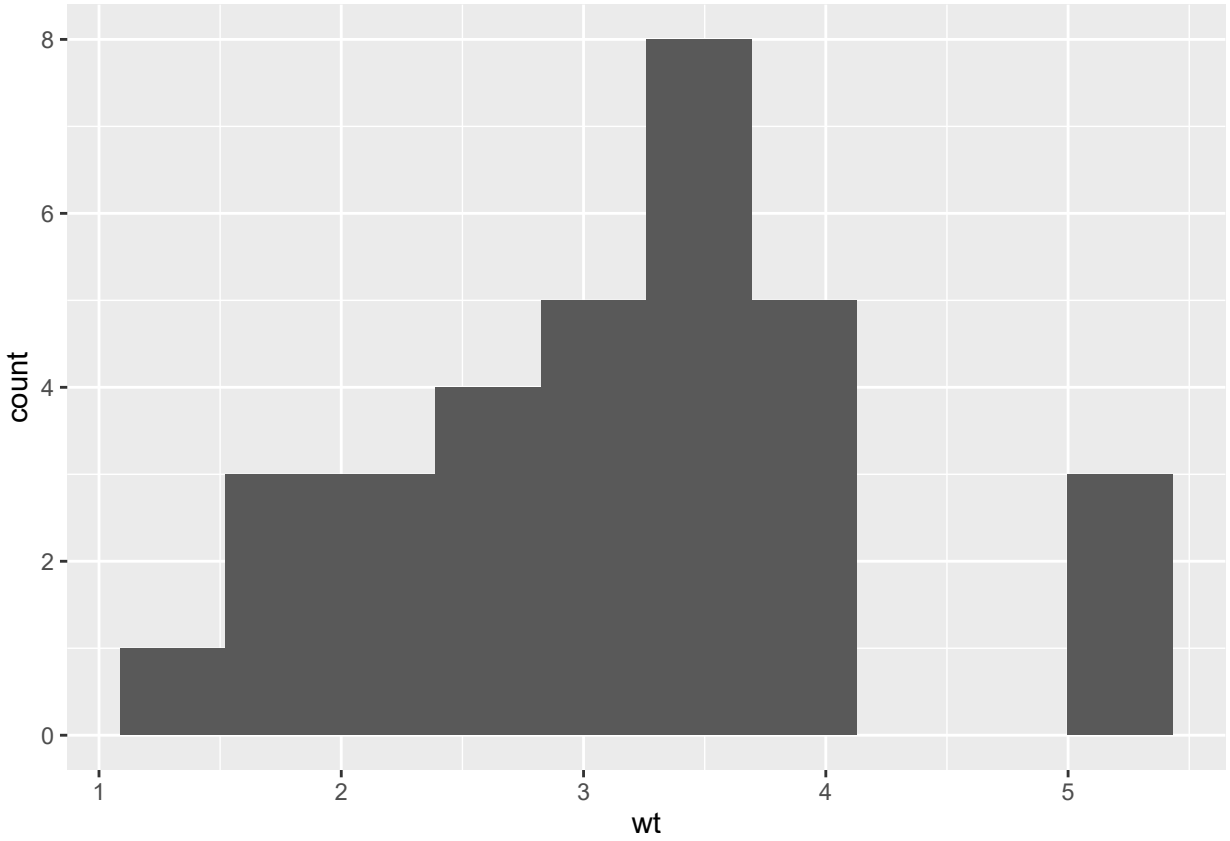
Metode ostrega očesa

Najlažje je včasih podatke le vizualizirati na način, da so nam osamelci jasno vidni.

Poskušajmo detektirati osamelce v množici **mtcars** za atribut **wt**.

Izrišemo lahko histogram vrednosti.

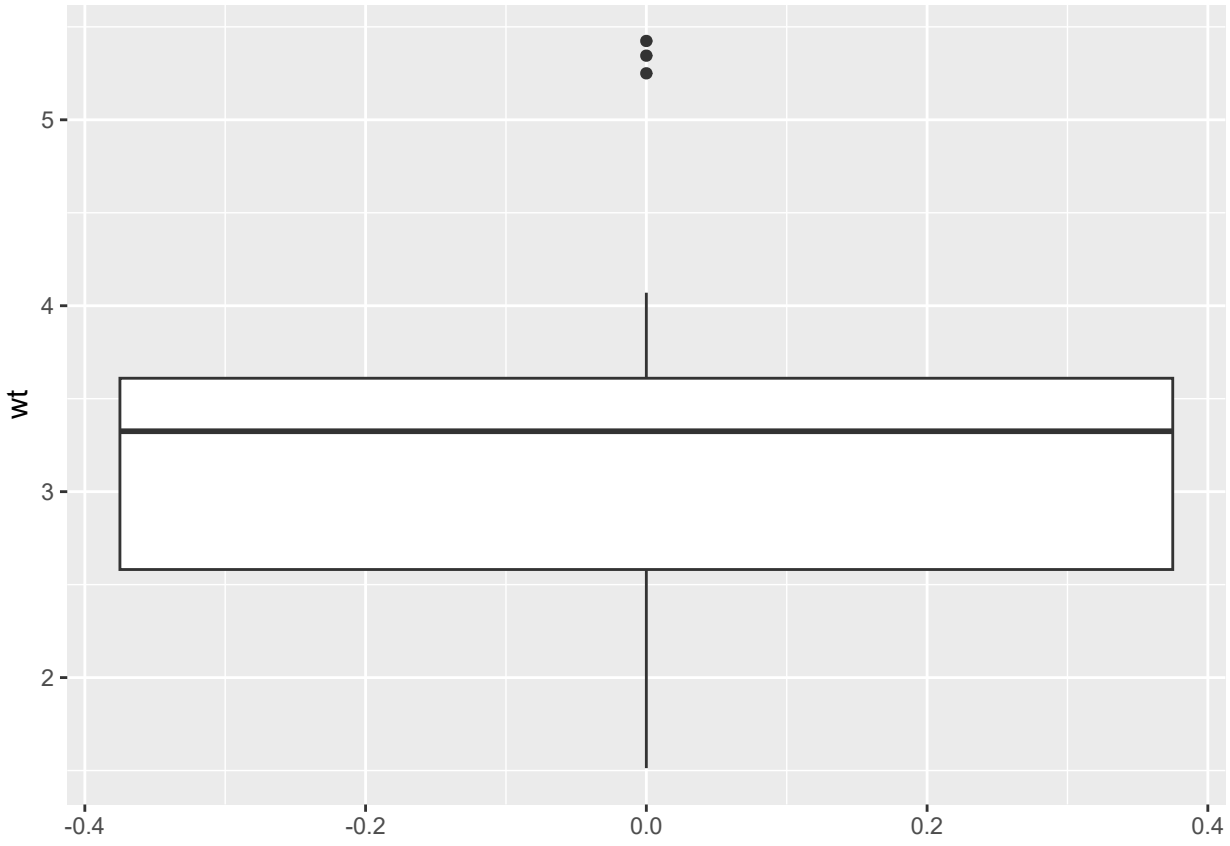
```
ggplot(mtcars, aes(x = wt)) + geom_histogram(bins = 10)
```



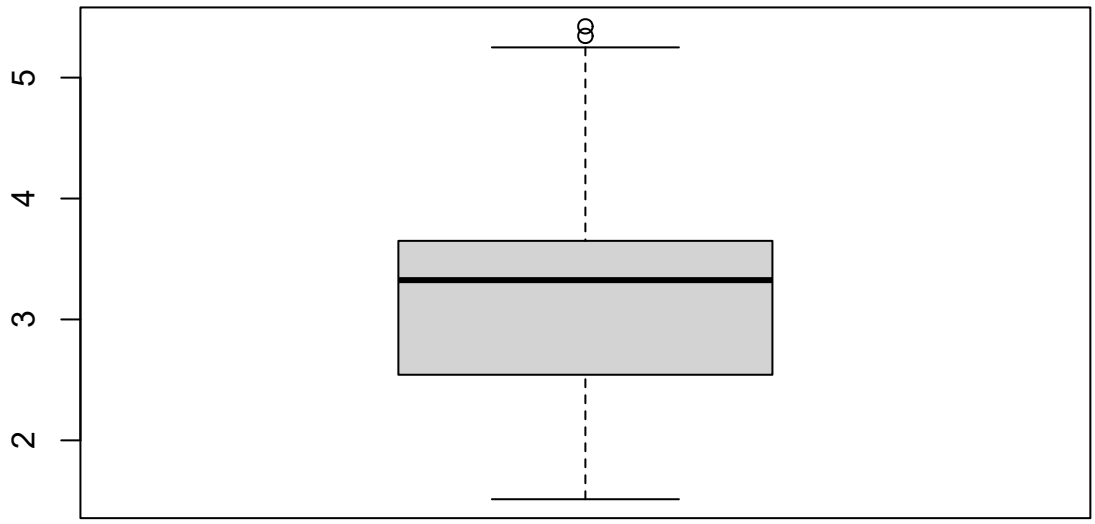
Iz izrisa vidimo, da so 3 vrednosti dokaj visoke in potencialno osamelci.

Drug priročen izris je **boxplot** oziroma škatla z brki.

```
ggplot(mtcars, aes(y = wt)) + geom_boxplot()
```




```
boxplot(mtcars$wt)
```



Pri obeh izrisih so osamelci ozačeni z pikami. Osamelci so pri tem izrisu navadno definirani kot vrednosti, ki so od povprečja oddaljeni za več kot 1.5*(razdalja med prvim in tretjiv kvantilom).

Te vrednosti lahko tudi izpišemo.

```
boxplot.stats(mtcars$wt)$out
```

```
## [1] 5.424 5.345
```

Računske metode

Prve vrednosti na katere pomislimo, da bi lahko bili osamelci sta minimalna in maksimalna.

```
range(mtcars$wt)
```

```
## [1] 1.513 5.424
```

Zelo preprosta metoda je tudi, da za osamelce določimo 2.5% najvišjih in najnižjih vrednosti.

```
spodnja_meja <- quantile(mtcars$wt, 0.025)
zgornja_meja <- quantile(mtcars$wt, 0.975)

mtcars$wt[mtcars$wt < spodnja_meja | mtcars$wt > zgornja_meja]
```

```
## [1] 5.424 1.513
```

Hampfel filter

Hamfel filter določi za osamelce vse vrednosti, ki se od mediane oddaljene več kot 3*MAD. Kjer je MAD povprečno absolutno odstopanje od mediane.

```
median(mtcars$wt)
```

```
## [1] 3.325
```

```
mad(mtcars$wt, constant = 1)
```

```
## [1] 0.5175
```

```
spodnja_meja <- mean(mtcars$wt) - 3*mad(mtcars$wt, constant = 1)
zgornja_meja <- mean(mtcars$wt) + 3*mad(mtcars$wt, constant = 1)

mtcars$wt[mtcars$wt < spodnja_meja | mtcars$wt > zgornja_meja]
```

```
## [1] 5.250 5.424 5.345 1.615 1.513
```

Statistične metode

Poglejmo še par statističnih metod, ki predpostavljajo, da so vhodni podatki normalno porazdeljeni.

Grubbs test

Grubbov test primerja ničelno hipotezo, da najvišja vrednost ni osamelec z hipotezo, da je.

```
library(outliers)
test <- grubbs.test(mtcars$wt) #Dodamo opposite = T, če želimo testirati min
test
```

```
##
##  Grubbs test for one outlier
##
## data:  mtcars$wt
## G = 2.25534, U = 0.83063, p-value = 0.3083
## alternative hypothesis: highest value 5.424 is an outlier
```

P vrednost je 0.3083, kar pomeni, da hipoteze, da 5.424 ni osamelec ne zavržemo. Ponavadi jo zavrnemo, če je $p < 0.05$.

Dixton test

Dixtonov test deluje podobno kot Grubbsov test, vendar je primeren le za majhno število vrednosti.

```
test <- dixon.test(mtcars[1:30,]$wt)
test
```

```
##
##  Dixon test for outliers
##
## data:  mtcars[1:30, ]$wt
## Q = 0.048481, p-value = 0.1197
## alternative hypothesis: highest value 5.424 is an outlier
```

Ponovno za maksimalno vrednost velja enako kot pri Grubbsovem testu.

Rosner test

Z tem testom lahko preverimo več osamelcev hkrati. Iz boxplota smo videli, da sta potencialna dva tako, da bomo preverili oba naenkrat.

```
library(EnvStats)
```

```
##
## Attaching package: 'EnvStats'
## The following objects are masked from 'package:stats':
##
##     predict, predict.lm
## The following object is masked from 'package:base':
##
##     print.default
```

```
test <- rosnerTest(mtcars$wt, k = 2)
test
```

```
## $distribution
## [1] "Normal"
##
## $statistic
##      R.1      R.2
## 2.255336 2.425760
##
## $sample.size
## [1] 32
##
## $parameters
## k
## 2
##
## $alpha
## [1] 0.05
##
## $crit.value
## lambda.1 lambda.2
## 2.938048 2.923571
##
## $n.outliers
## [1] 0
##
## $alternative
## [1] "Up to 2 observations are not\n                                    from the same Distribution."
##
## $method
## [1] "Rosner's Test for Outliers"
##
## $data
##  [1] 2.620 2.875 2.320 3.215 3.440 3.460 3.570 3.190 3.150 3.440 3.440 4.070
## [13] 3.730 3.780 5.250 5.424 5.345 2.200 1.615 1.835 2.465 3.520 3.435 3.840
## [25] 3.845 1.935 2.140 1.513 3.170 2.770 3.570 2.780
##
## $data.name
## [1] "mtcars$wt"
##
## $bad.obs
## [1] 0
##
## $all.stats
```

```
##   i   Mean.i      SD.i Value Obs.Num    R.i+1 lambda.i+1 Outlier
## 1 0 3.217250 0.9784574 5.424      16 2.255336   2.938048   FALSE
## 2 1 3.146065 0.9064935 5.345      17 2.425760   2.923571   FALSE
##
## attr(,"class")
## [1] "gofOutlier"
```

Iz izpisa $n.outliers lahko vidimo, da imamo 0 osamelcev in iz $all.stats, da je pri obeh zadnji stolpec FALSE. Če bi imeli osamelce bi stolpec vseboval vrednost TRUE.