Predavanje 08 – Odgovori na vprašanja - Marec 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
```

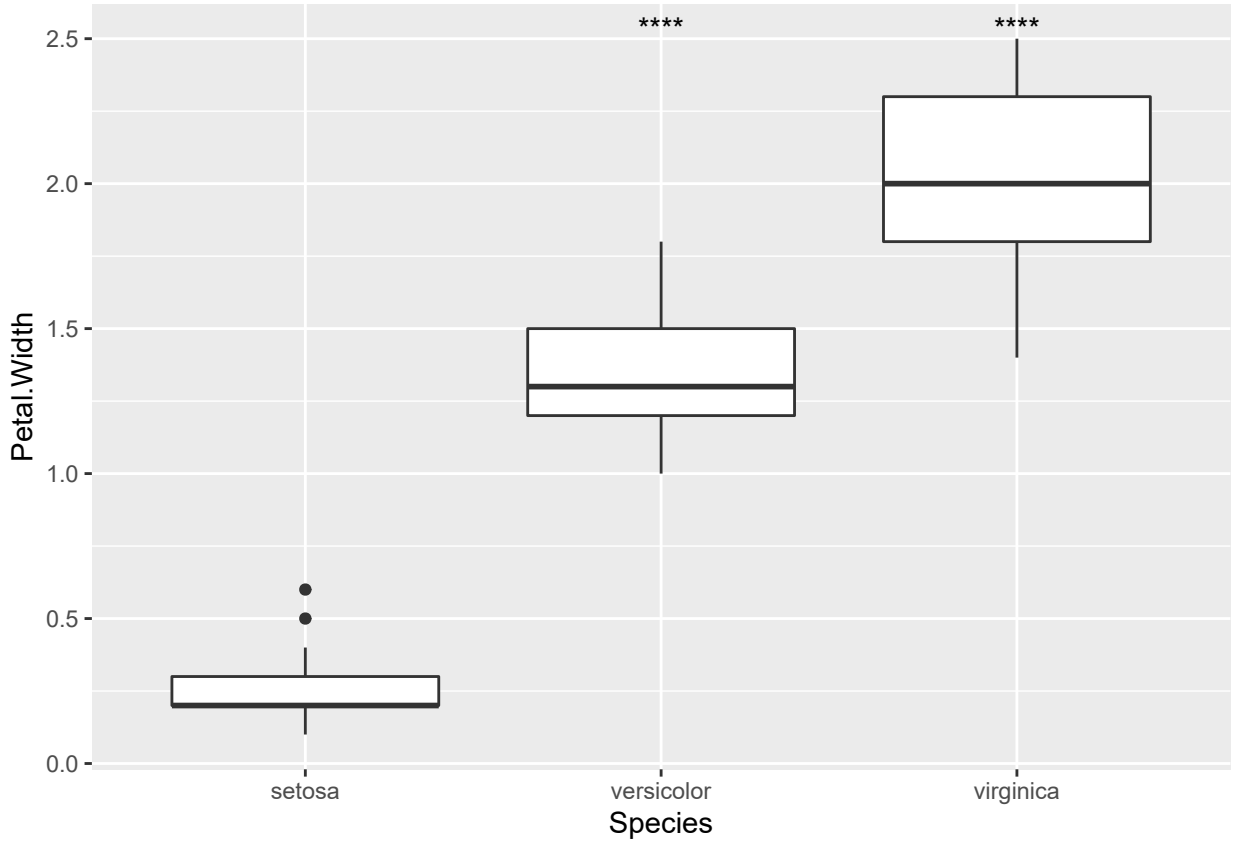
```
# ANOVO uporabimo za statistično primerjavo dolžine listov treh vrst perunike.
# Primerjamo, ali vrsta perunike vpliva na dolžino listov.
my_anova <- aov(Sepal.Length ~ Species, data = iris)
summary(my_anova)
```

```
##              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`.

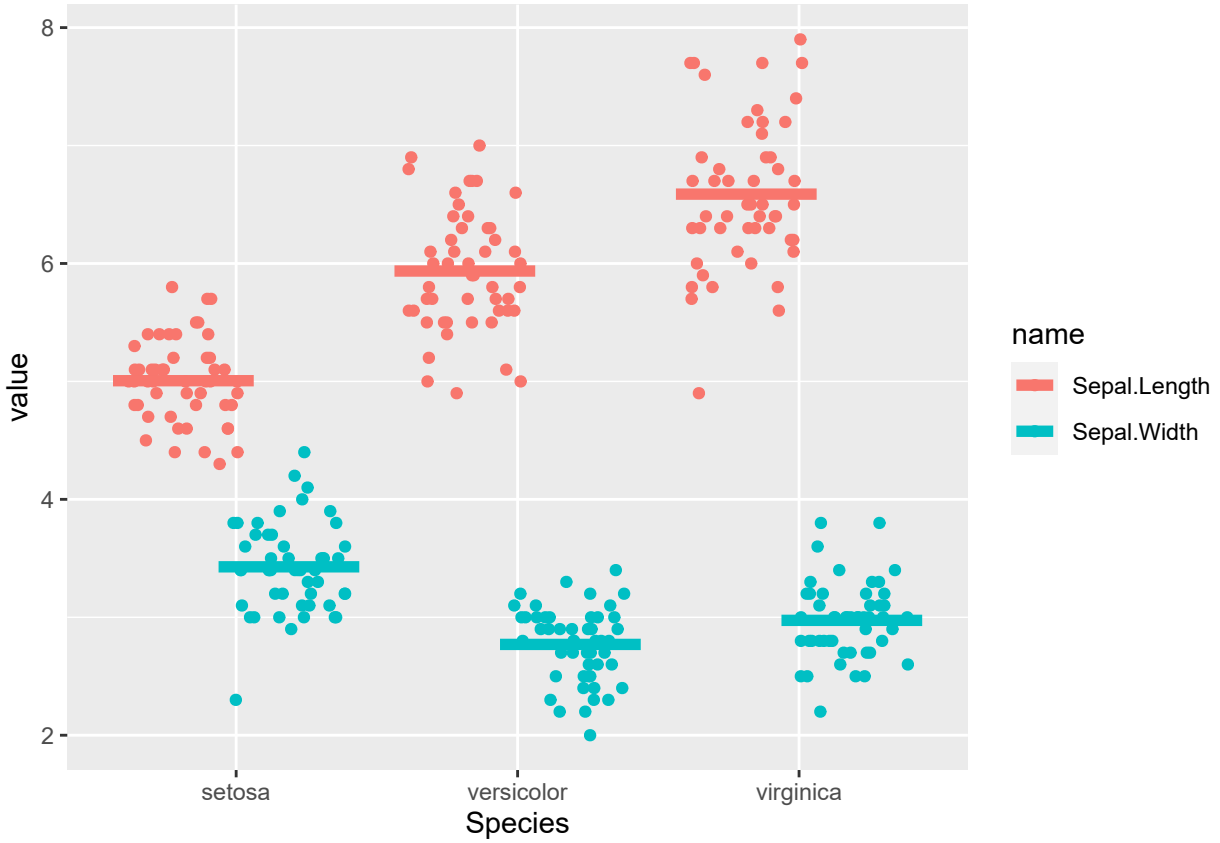
```
library(tidyr)
iris_longer <- iris[ , c("Sepal.Length", "Sepal.Width", "Species")]
iris_longer <- pivot_longer(iris_longer, Sepal.Length:Sepal.Width)
head(iris_longer)
```

```
## # 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.

```
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
```

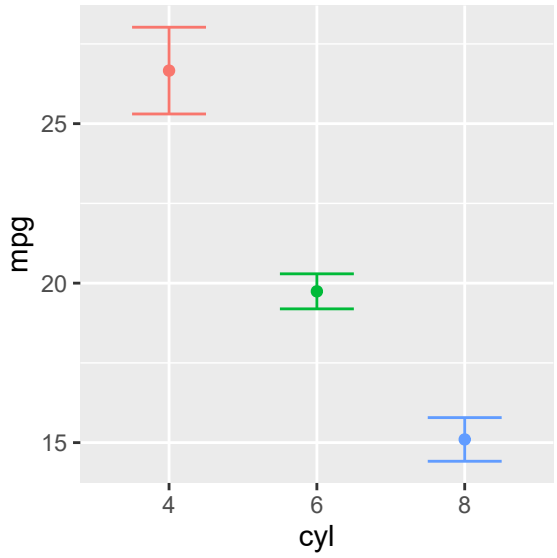
```
mus <- aggregate(mpg ~ cyl, mtcars, FUN = mean)
sds <- aggregate(mpg ~ cyl, mtcars, FUN = function(x) {sd(x) / sqrt(length(x))})
df  <- cbind(mus, SE = sds$mpg)
df$cyl <- as.character(df$cyl)

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")
```



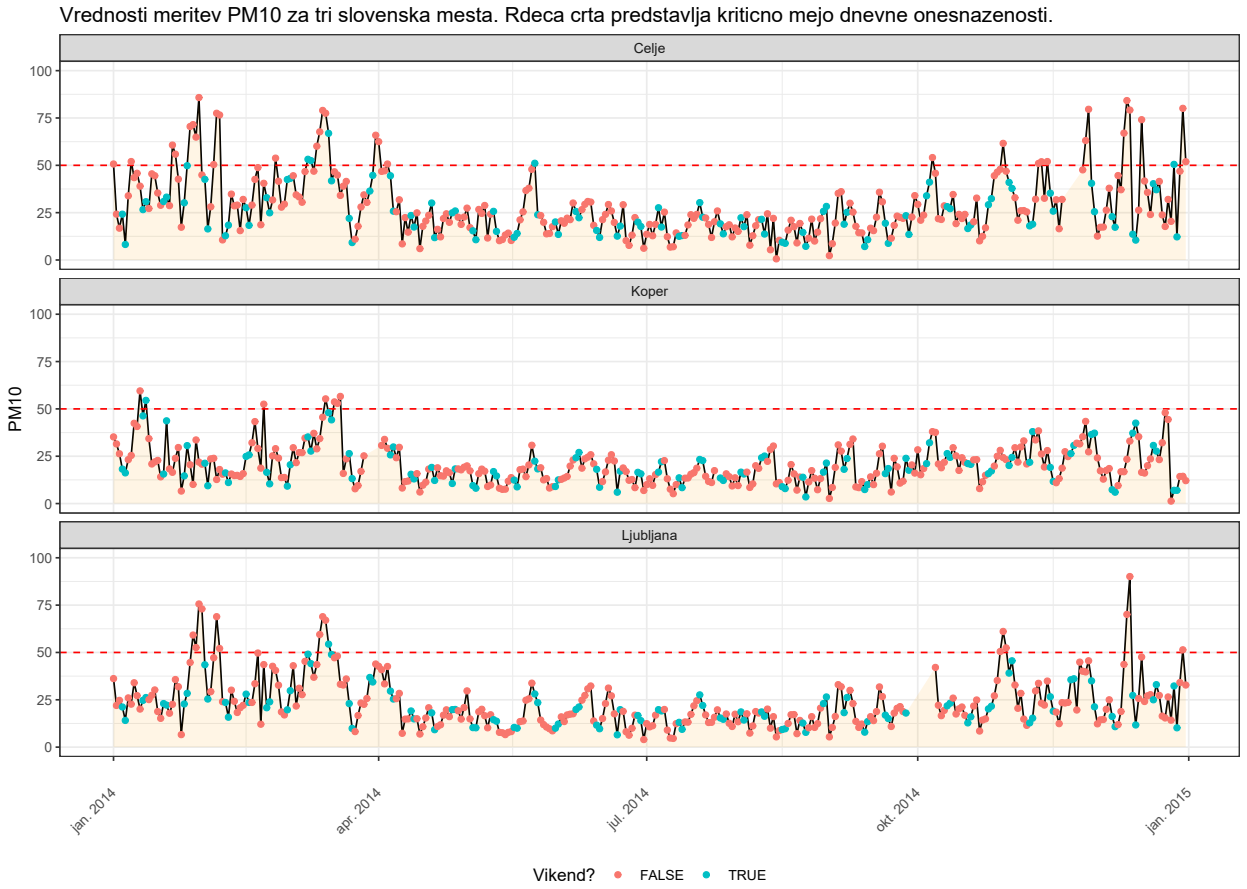
Ponovitev grafov skozi četrto nalogo petega predavanja

Besedilo naloge:

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 možnosti, ki jih ponuja knjižnica ggplot. Še en namig, da se ne bomo ukvarjali še z R: uporabite funkcijo `weekdays()` in podatek, da je vikend dan, ki je sobota ali nedelja.

Za to nalogo bomo naložili podatke o meritvah onesnaženosti iz datoteke *PM10mesta.csv* in izbrali podmnožico meritev za Koper, Celje in Ljubljano. Pred risanjem tudi pretvorimo datum iz niza znakov v podatkovni tip Date.

Narišimo ta graf:



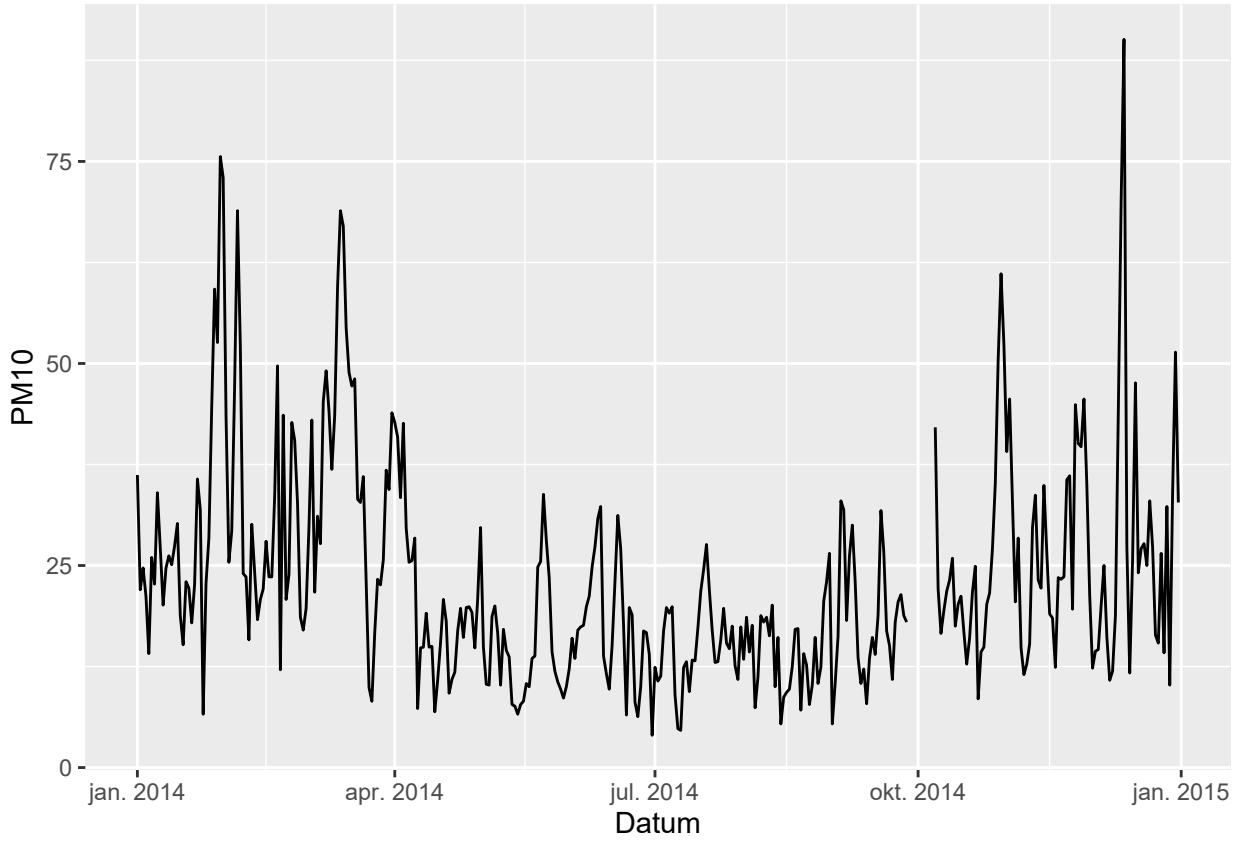
Sedaj poglejmo, kako lahko pridemo do te rešitve korak po koraku.

Najprej preberemo podatke in spremenimo datume v pravilno obliko.

```
df <- read.table("./data_raw/PM10mesta.csv", header = TRUE, sep=",", quote = "\"", dec = '.')
#spremenimo datum
df$Datum <- as.Date(df$Datum)
```

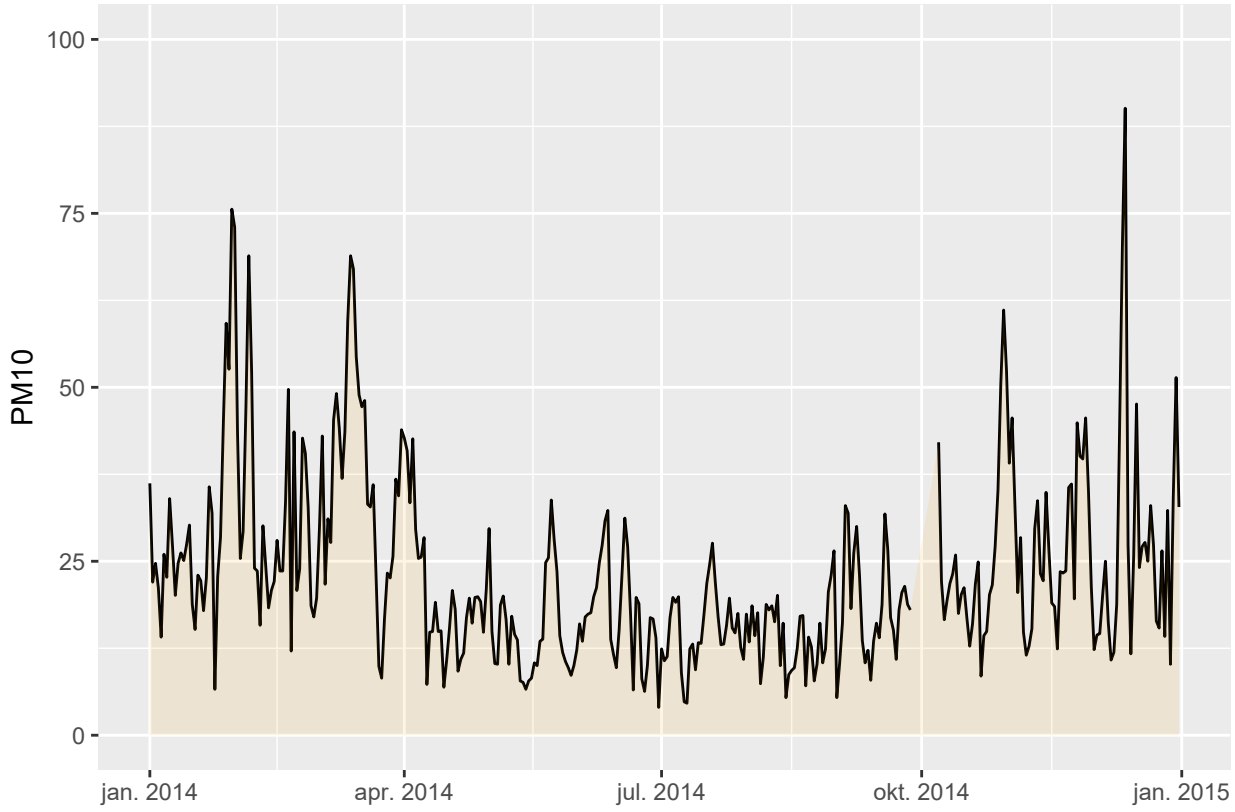
Namesto izrisa treh grafov se bomo za začetek osredotočili za izris vsebnosti delcev PM10 samo za Ljubljano. Izrišimo samo to črto.

```
ggplot(df[df$kraj == "Ljubljana",], aes(x = Datum, y = PM10)) +
  geom_line(colour = "black")
```



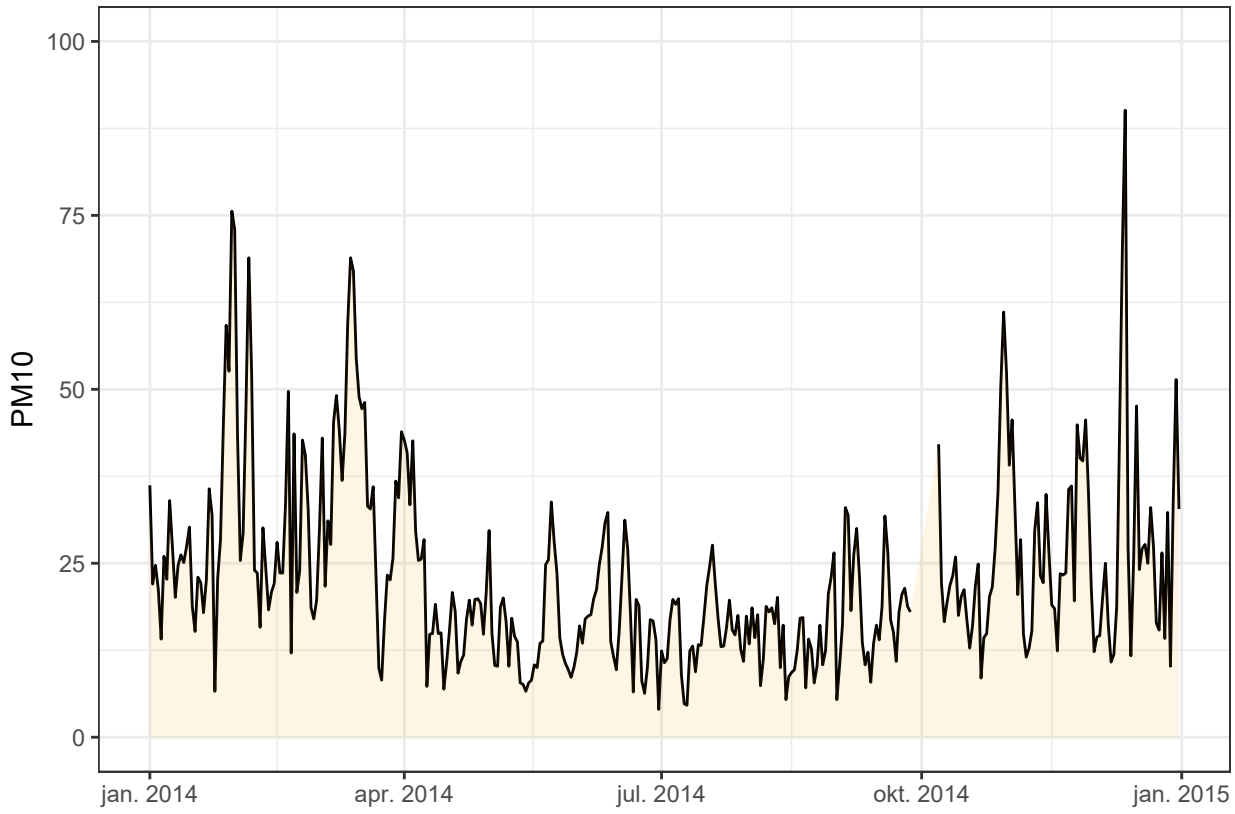
Z ukazom `geom_area` lahko na grafu podamo ploščino. S spodnjim ukazom jo pobarvamo oranžno in naredimo skoraj transparentno. Grafu še omejimo y os med 0 in 100 ter izbrišemo ime osi x.

```
ggplot(df[df$kraj == "Ljubljana",], aes(x = Datum, y = PM10)) +
  geom_line(colour = "black") +
  geom_area(alpha = 0.1, fill = "orange") + ylim(0, 100) +
  xlab("")
```



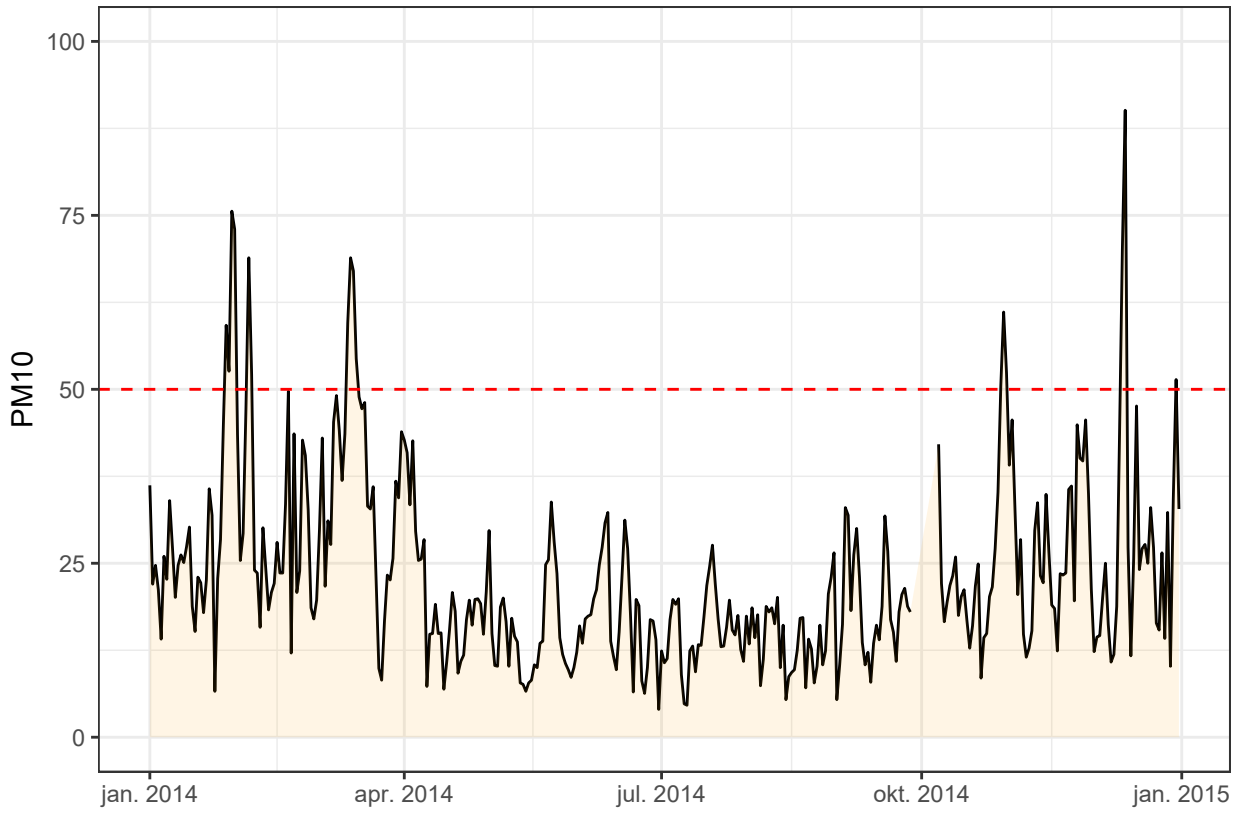
Paket **ggplot2** ima že prednastavljene teme za izris grafov. Ena izmed teh je `theme_bw()`, ki jo bomo uporabili. Z ukazom `?theme_bw()` lahko v zavihku help vidite tudi ostale prednastavljene teme. Z ukazom `theme()` pa jih lahko tudi sami definirate.

```
ggplot(df[df$kraj == "Ljubljana",], aes(x = Datum, y = PM10)) +
  geom_line(colour = "black") +
  geom_area(alpha = 0.1, fill = "orange") + ylim(0, 100) +
  xlab("") +
  theme_bw()
```



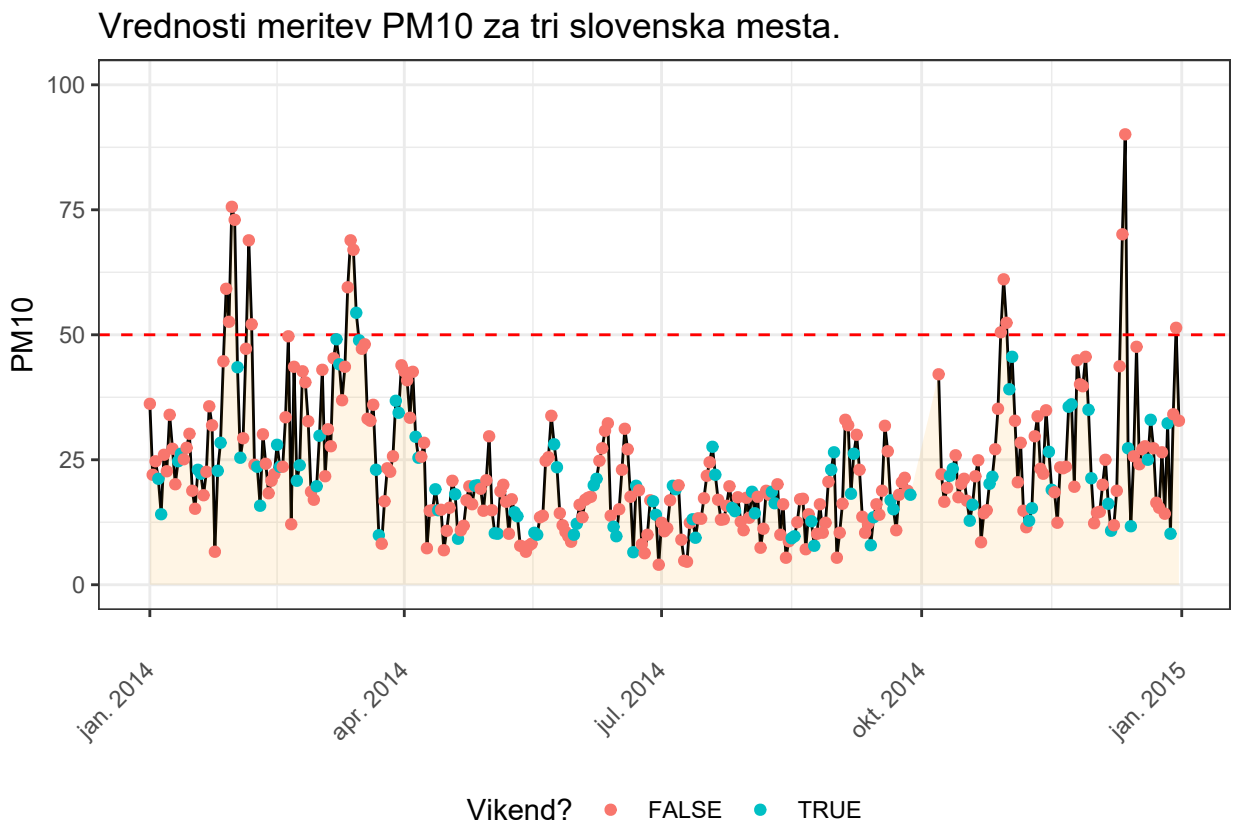
Dodamo še rdečo vodoravno črtkano črto, s katero označimo kritičen prag. Parameter `lty` označuje *line type* oziroma obliko črte, ostalo pa že poznate.

```
ggplot(df[df$kraj == "Ljubljana",], aes(x = Datum, y = PM10)) +
  geom_line(colour = "black") +
  geom_area(alpha = 0.1, fill = "orange") + ylim(0, 100) +
  xlab("") +
  theme_bw() +
  geom_hline(yintercept = 50, lty = "dashed", colour = "red")
```



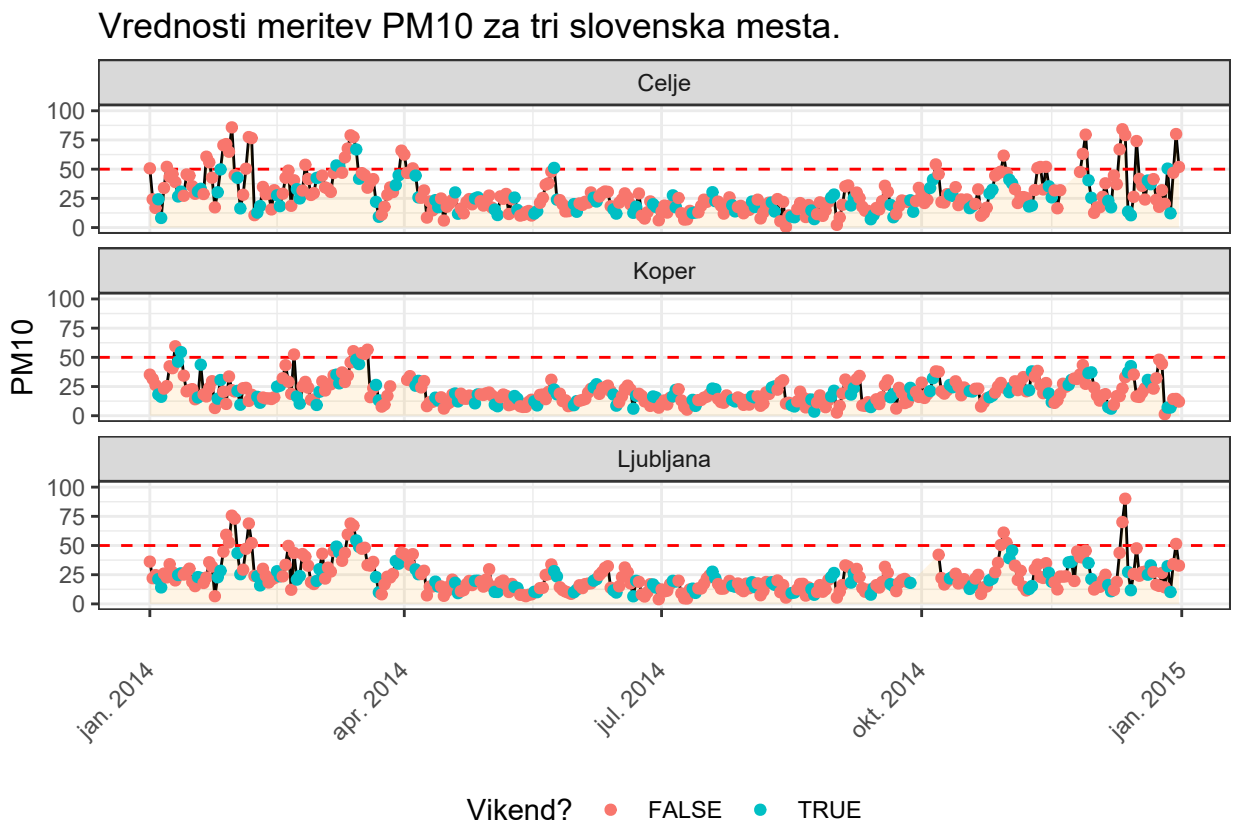
Sedaj bomo naredili nekaj korakov naenkrat, ker so vmesni izrisi dokaj grdi. Z `geom_point` dodamo na črto točke, ki jih obarvamo glede na to, če je dan delavnik ali dan v vikendu. S funkcijo `ggtitle` spremenimo naslov izrisa in s funkcijo `labs` spremenimo nalov legende. Nato legendo, ki se privzeto nahaja na desni strani grafa, premaknemo na spodnjo stran in zarotiramo tekst na x osi za 45 stopinj. To storimo tako, da znotraj funkcije `theme` podamo nove vrednosti parametrom `legend.position` in `axis.text.x`.

```
ggplot(df[df$kraj == "Ljubljana",], aes(x = Datum, y = PM10)) +
  geom_line(colour = "black") +
  geom_area(alpha = 0.1, fill = "orange") + ylim(0, 100) +
  xlab("") +
  theme_bw() +
  geom_hline(yintercept = 50, lty = "dashed", colour = "red") +
  geom_point(data = df[df$kraj == "Ljubljana",],
             aes(colour = weekdays(Datum) == "sobota" |
                   weekdays(Datum) == "nedelja")) +
  ggtitle("Vrednosti meritev PM10 za tri slovenska mesta.") +
  labs(color = "Vikend?") +
  theme(legend.position = "bottom",
          axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1))
```



Na koncu uporabimo še funkcijo `facet_wrap`, kateri podamo skupine v obliki **enačbe**. Enačba je poseben tip spremenljivke, s katerim povemo, da so vrednosti na levi strani tilde '~' odvisne od tistih na desni. Pika '.' v kontekstu spodnje enačbe pomeni *vsi ostali stolpci*. Nekako lahko beremo izriši vse podatke glede na kraj. `facet_wrap` iziriše za vsako skupino svoj graf, `ncol = 1` pa pove, da želimo vse skupne izrisati v samo enem stoplcu.

```
ggplot(df, aes(x = Datum, y = PM10)) +
  geom_line(colour = "black") +
  geom_area(alpha = 0.1, fill = "orange") + ylim(0, 100) +
  xlab("") +
  theme_bw() +
  geom_hline(yintercept = 50, lty = "dashed", colour = "red") +
  geom_point(data = df,
             aes(colour = weekdays(Datum) == "sobota" |
                   weekdays(Datum) == "nedelja")) +
  ggtitle("Vrednosti meritev PM10 za tri slovenska mesta.") +
  labs(color = "Vikend?") +
  theme(legend.position = "bottom",
          axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1)) +
  facet_wrap(. ~ kraj, ncol = 1)
```

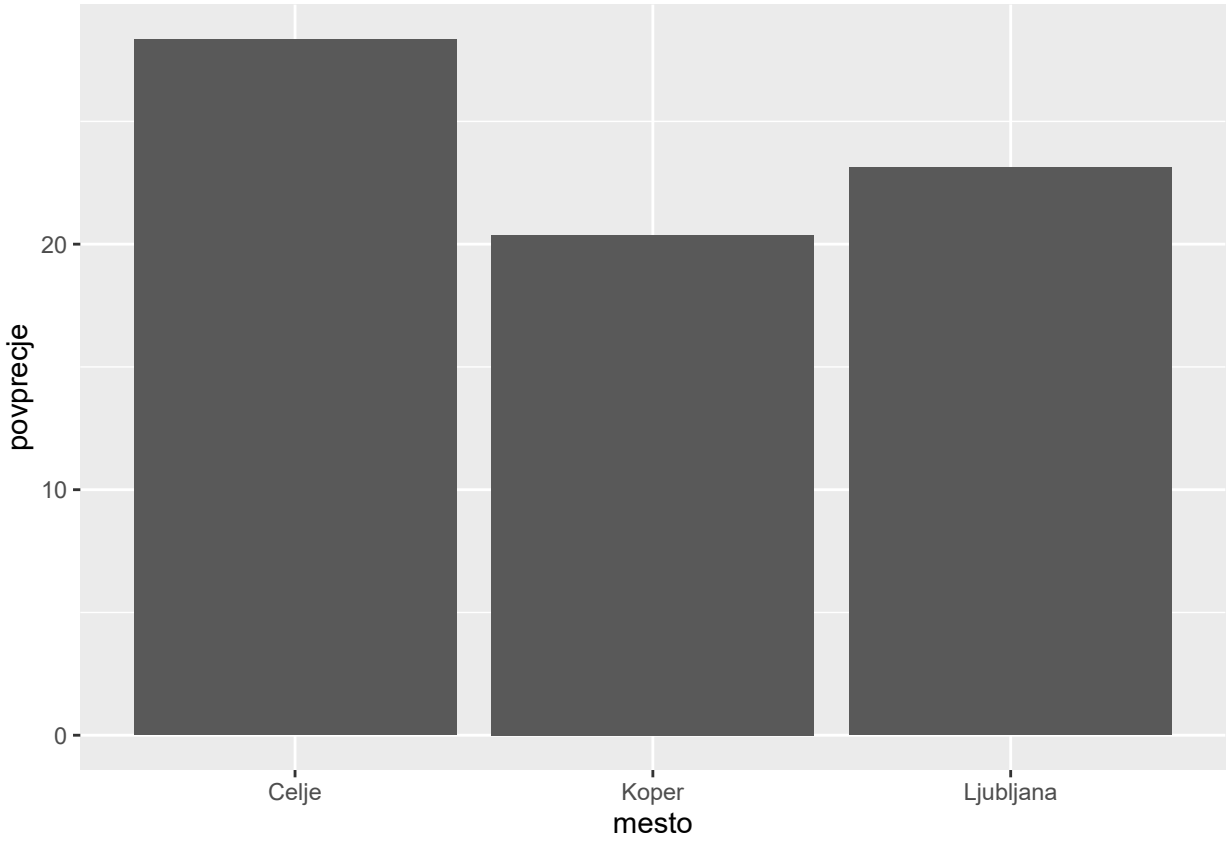


Tako smo prišli do končnega grafa.

Vzorci in slike na grafih

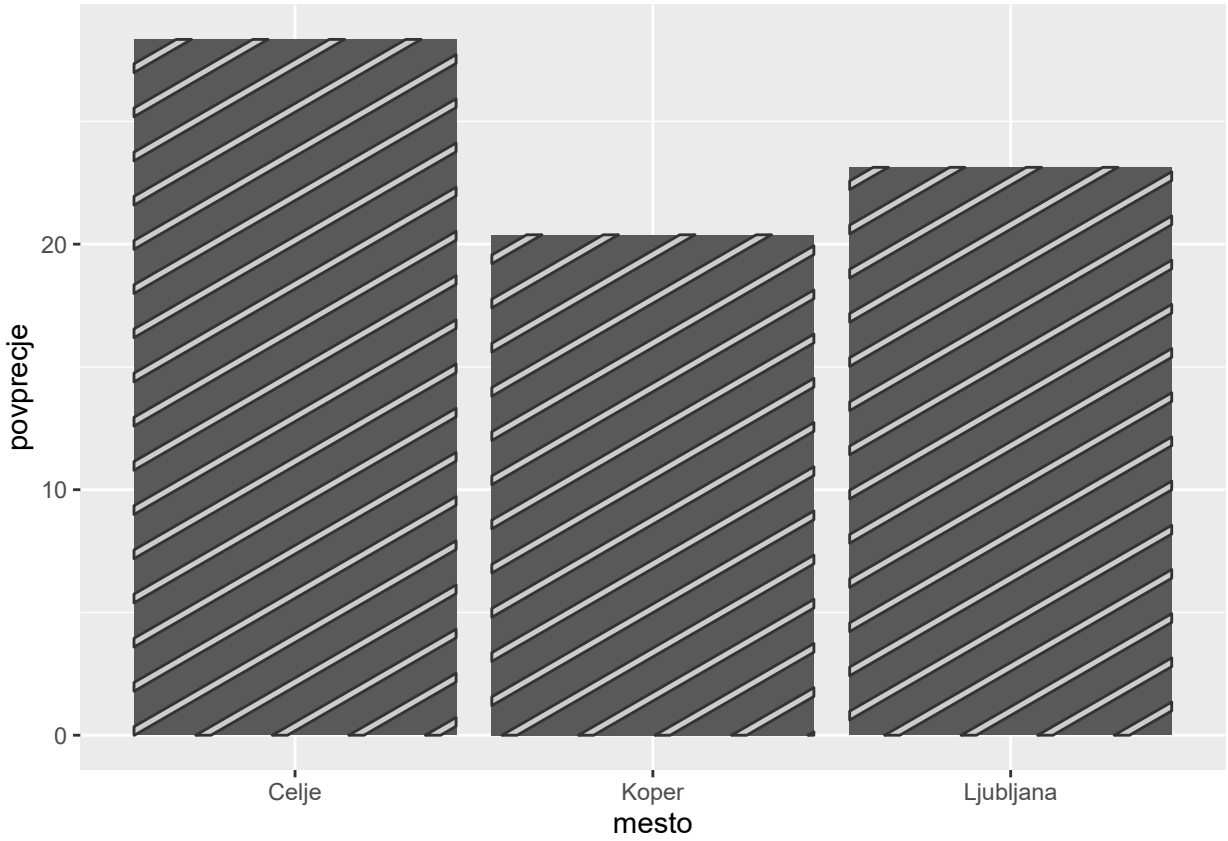
Uporabimo podatke o delcih, da izrišemo stolpični graf, ki prikazuje povprečno onesnaženost glede na mesto.

```
d <- aggregate(x = df$PM10, by = list(df$kraj), FUN = mean, na.rm = TRUE)
names(d) <- c("mesto", "povprecje")
ggplot(d, aes(x = mesto, y = povprecje)) +
  geom_bar(stat = "identity")
```



Za risanje vzorcev na stolpce lahko uporabimo paket **ggpattern**. Ta paket vsebuje skoraj vse geome, ki so v paketu ggplot2, s to razliko, da je imenom dodan še **_pattern**. Poglejmo si, kaj se zgodi, če samo zamenjamo geom.

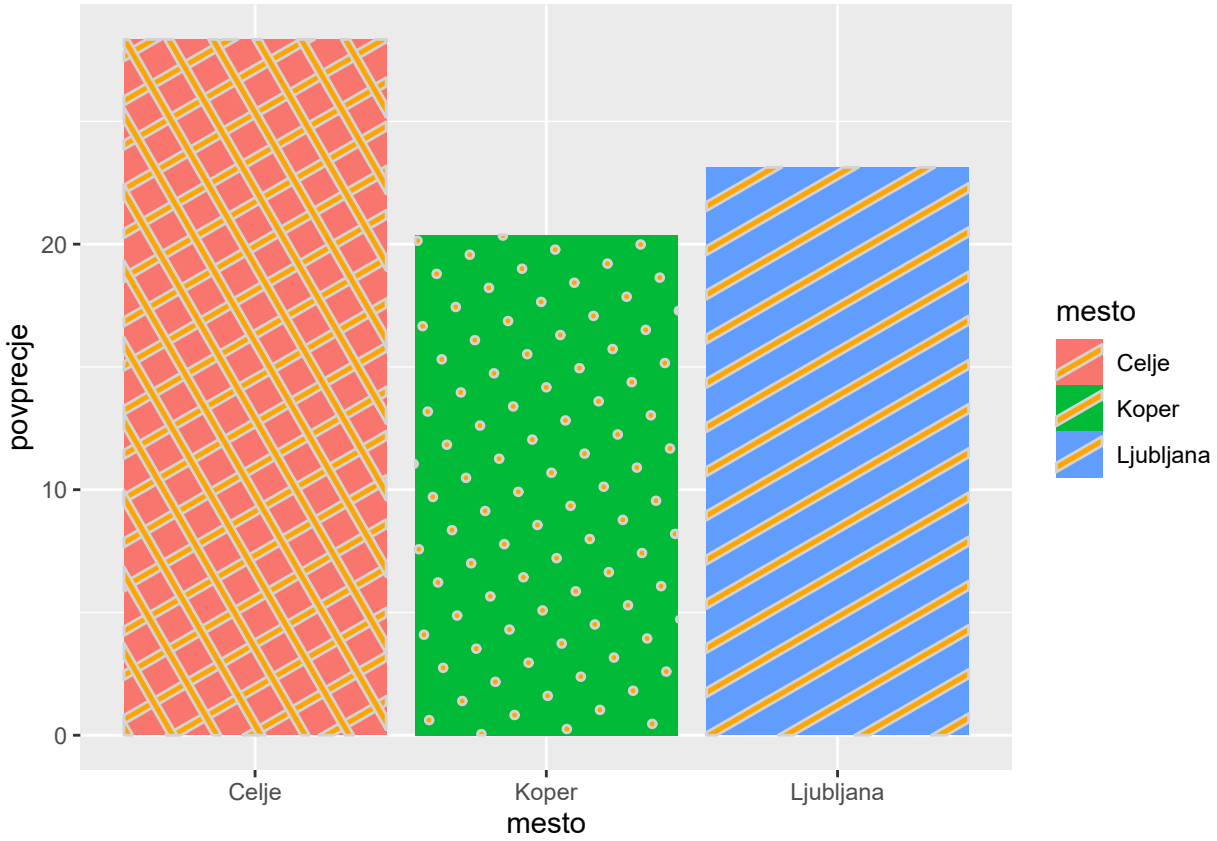
```
library(ggpattern)
ggplot(d, aes(x = mesto, y = povprecje)) +
  geom_bar_pattern(stat = "identity")
```



Opazimo, da avtomatsko izriše črte (stripes).

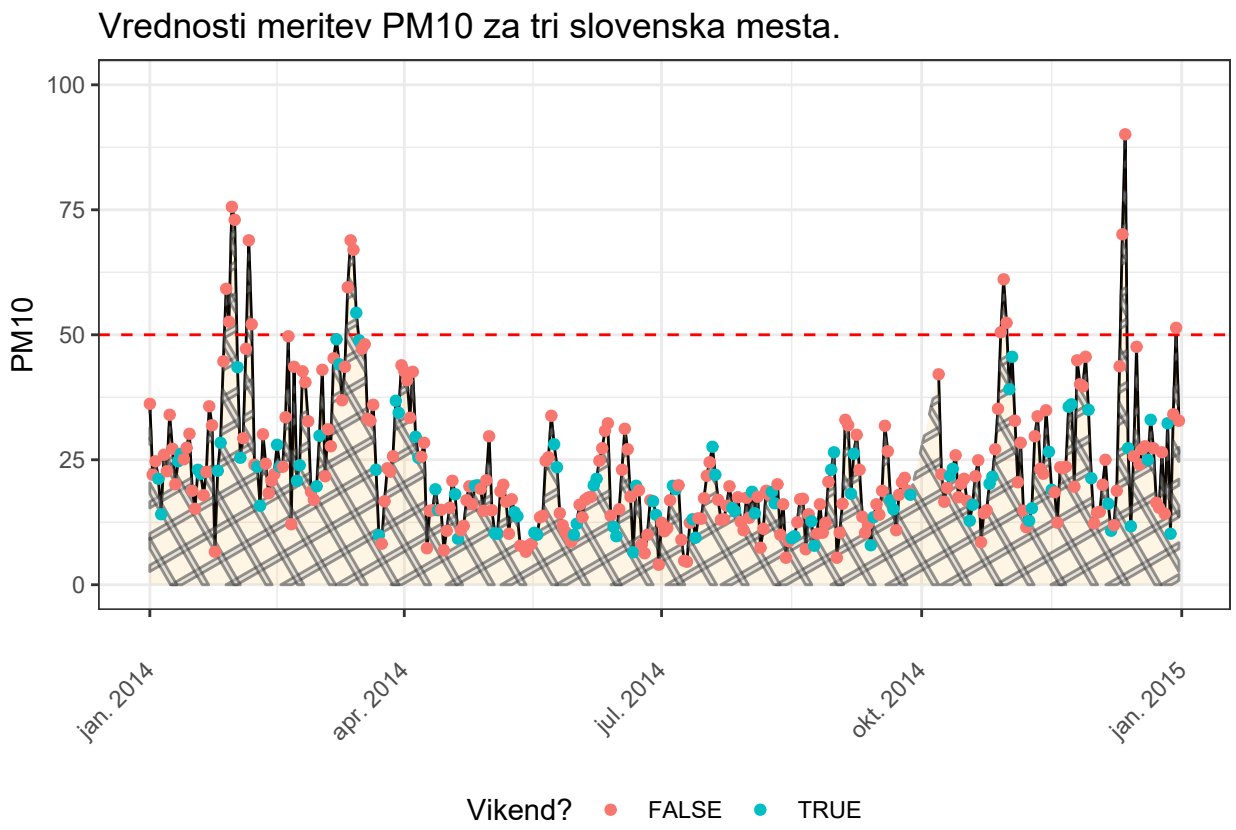
Dodajmo še barve in nekaj drugih privzetih vzorcev.

```
ggplot(d, aes(x = mesto, y = povprecje)) +
  geom_bar_pattern(stat = "identity", aes(fill = mesto, pattern = mesto,
                                          pattern_fill = "green"),
                   pattern = c("crosshatch", "circle", "stripe"),
                   pattern_fill = "orange",
                   pattern_colour = "lightgrey")
```



Vzorce lahko damo na skoraj katerokoli obarvano površino. Poglejmo si na primer koako lahko dodamo vzorec na kompleksen graf iz domače naloge.

```
ggplot(df[df$kraj == "Ljubljana",], aes(x = Datum, y = PM10)) +
  geom_line(colour = "black") +
  geom_area_pattern(alpha = 0.1, fill = "orange", pattern = "crosshatch", pattern_alpha = 0.5) + ylim(0
  xlab("") +
  theme_bw() +
  geom_hline(yintercept = 50, lty = "dashed", colour = "red") +
  geom_point(data = df[df$kraj == "Ljubljana",],
             aes(colour = weekdays(Datum) == "sobota" |
                   weekdays(Datum) == "nedelja")) +
  ggtitle("Vrednosti meritev PM10 za tri slovenska mesta.") +
  labs(color = "Vikend?") +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1))
```



Na grafe lahko s paketom **ggimage** dodamo tudi slike. Paket uporablja drug paket **magick**, ki ga po potrebi predhodno naložite.

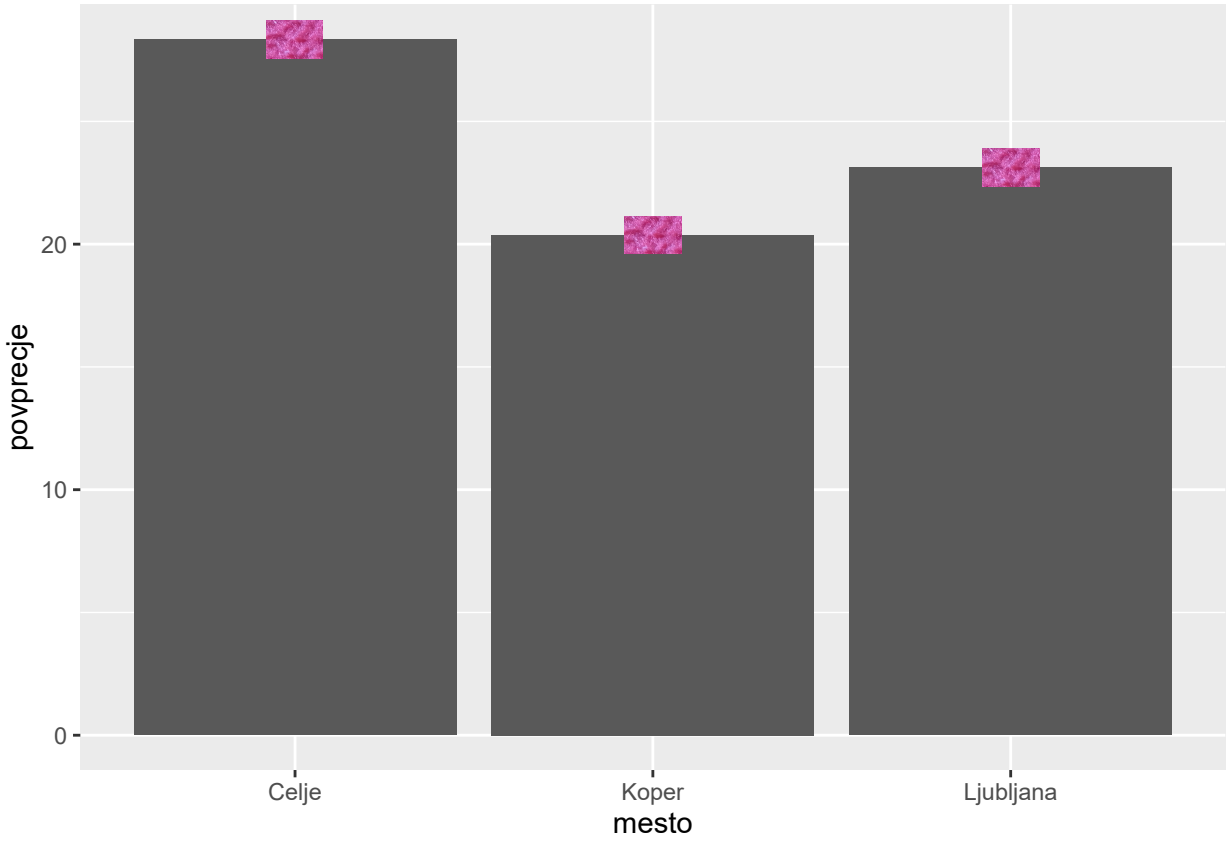
Prikažemo kaj se zgodi, če uporabimo `geom_image` na stolpičnem diagramu.

```
library(ggimage)
```

```
##
## Attaching package: 'ggimage'
```

```
## The following object is masked from 'package:ggpubr':
##
##     theme_transparent
```

```
ggplot(d, aes(x = mesto, y = povprecje, image="./img/small.png")) +
  geom_bar(stat = "identity") + geom_image()
```

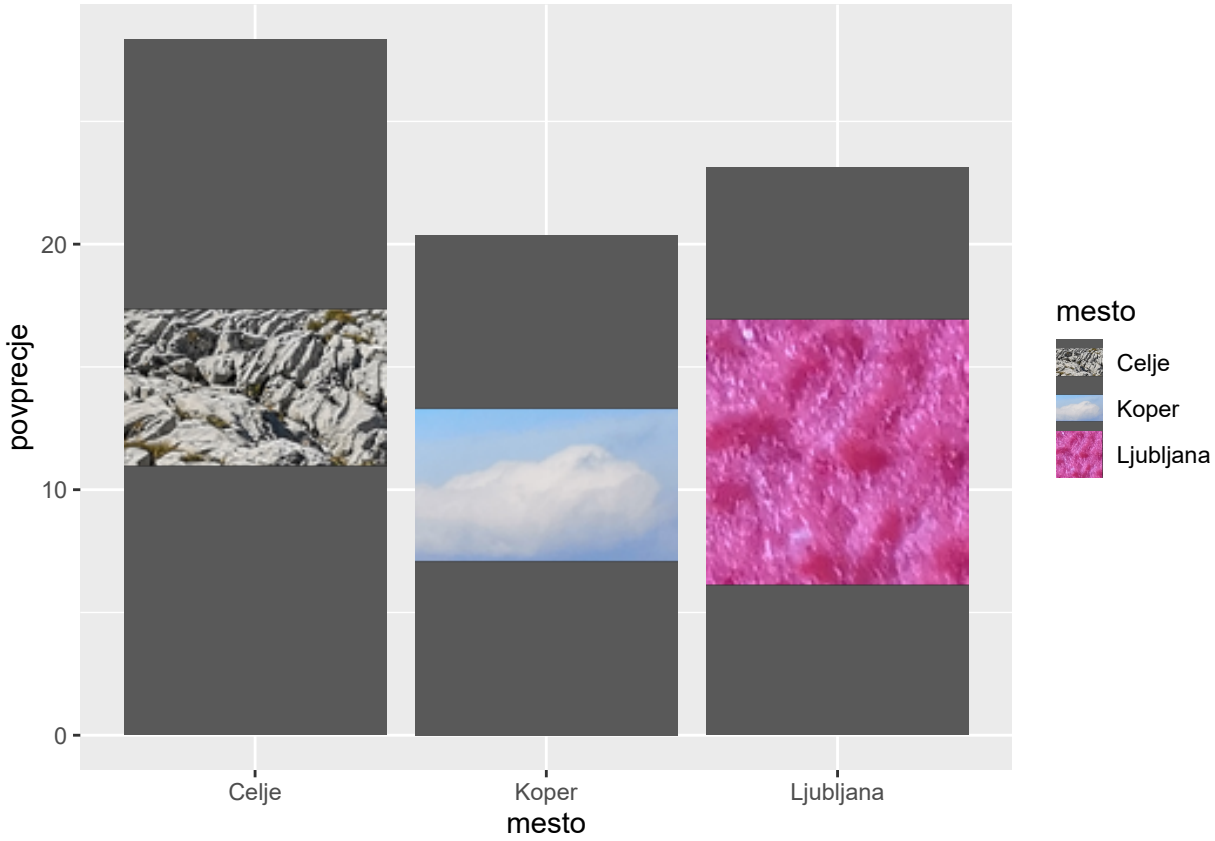


Vidimo, da slikico izriše na vrhu. Podobno lahko tudi namesto točk izrisujemo slike. Poglejmo si, kako bi celoten stolpec zamenjali s poljubim vzorcem (sliko). Najprej si naredimo vektor s potmi do vzorcev na računalniku ali spletu.

```
vzorci <- c("./img/kamen.png",
            "./img/cloud.png",
            "./img/small.png")
```

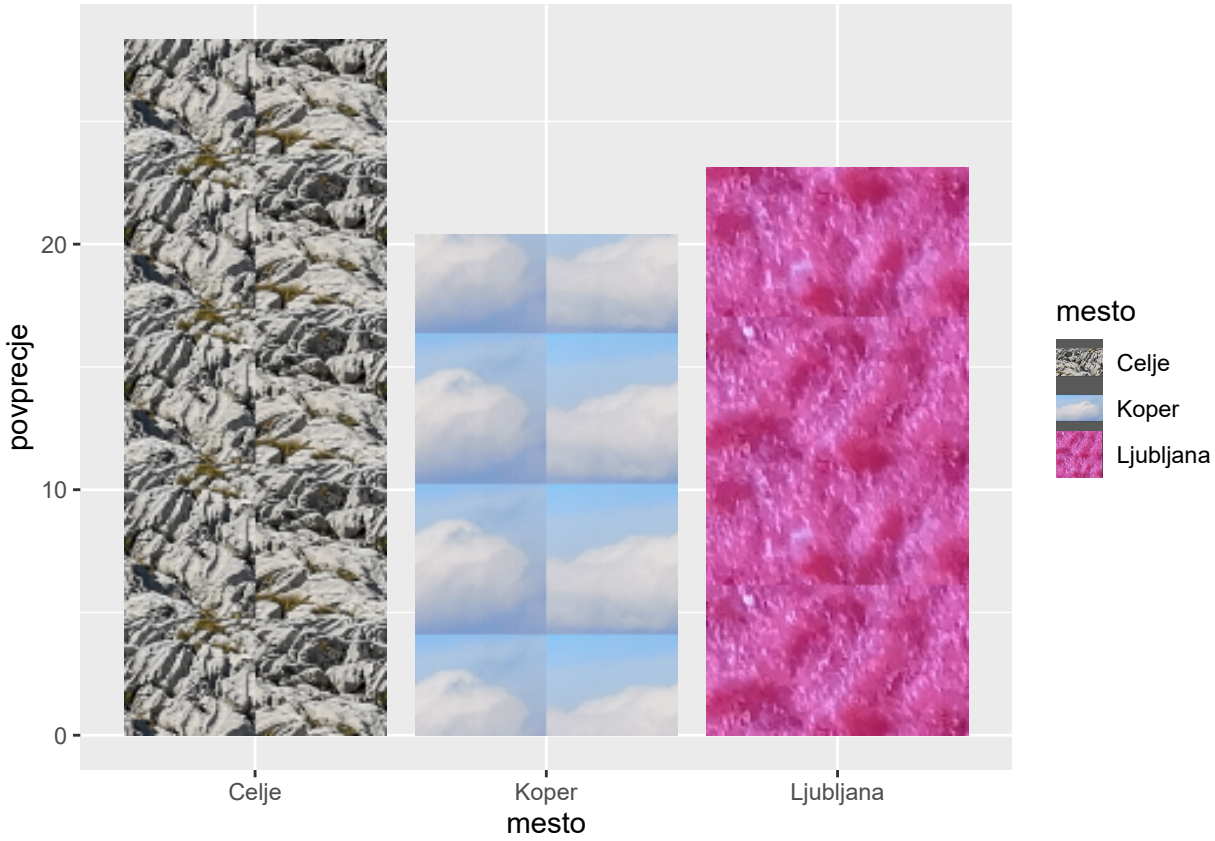
Parameter `pattern_filename` nam omogoča, da izbiramo različne spremenljivke glede na vrednosti stolpca `mesto`. Če uporabimo ta parameter moramo podati poti do slik s funkcijo `scale_pattern_filename_discrete`. Zadnja beseda te funkcije se lahko spreminja, glede na to kateri geom smo predhodno uporabili.

```
ggplot(d, aes(x = mesto, y = povprecje)) +
  geom_bar_pattern(stat = "identity",
                   aes(pattern_filename = mesto),
                   pattern = "image") +
  scale_pattern_filename_discrete(choices = vzorci)
```



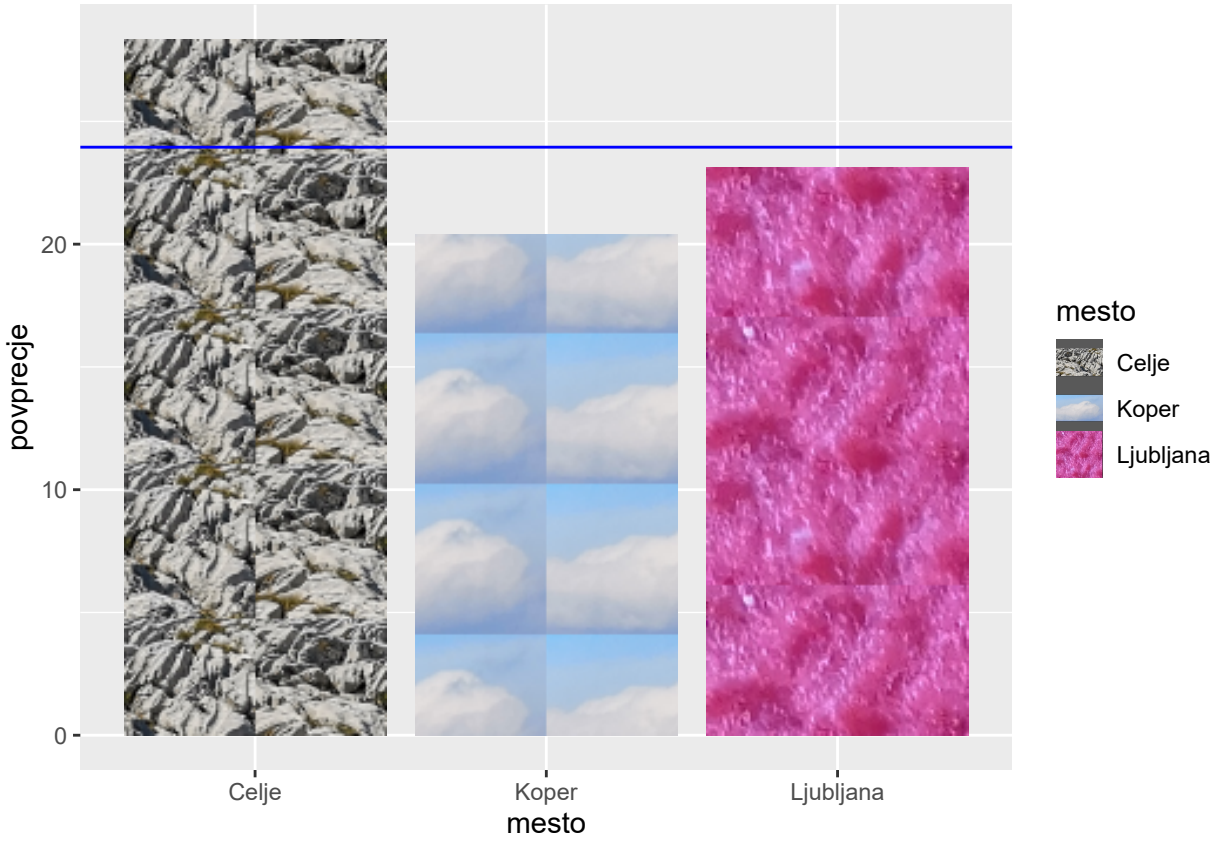
Na sliki vidimo, da je slika centrirana na sredino stoplca. Z dodatnimi parametri jo lahko raztegnemo oziroma razmnožimo cez celotno območje stolpca.

```
ggplot(d, aes(x = mesto, y = povprecje)) +
  geom_bar_pattern(stat = "identity",
                   aes(pattern_filename = mesto),
                   pattern = "image",
                   pattern_type    = 'tile',
                   pattern_filter  = 'box',
                   pattern_scale   = -1) +
  scale_pattern_filename_discrete(choices = vzorci)
```



Dodajanje črte povprečja

```
ggplot(d, aes(x = mesto, y = povprecje)) +
  geom_bar_pattern(stat = "identity",
                   aes(pattern_filename = mesto),
                   pattern = "image",
                   pattern_type    = 'tile',
                   pattern_filter  = 'box',
                   pattern_scale   = -1) +
  scale_pattern_filename_discrete(choices = vzorci)+
  geom_hline(yintercept = mean(d$povprecje), color="blue")
```



Branje poljubnih tekstovnih datotek

Včasih, ko se srečamo s podatki, ki so podani na nek nestandarden način si lahko pomagamo tako, da datoteko preberemo vrstico po vrstico in nato poskušamo zapis razčleniti na razumljivejše dele. V priloženi datoteki *VZOREC ODMERNI SEZNAM DOBIČEK.txt* opazimo, da so podatki ločeni po straneh, ki so med sabo ločene z znaki '=' in '-'. Opazimo lahko tudi, da je zapis nenavaden, oziroma, da se vsaka vrstica tabele raztegne čez dve vrstici dokumenta.

Pri tej nalogi si bomo pomagali s paketom **stringr**, ki je del skupka paketov **tidyverse**, in nam olajša delo z nizi.

Najprej preberemo vse vrstice datotke v spremenljivko `dat`:

```
library(stringr)
fl <- file("./data_raw/VZOREC ODMERNI SEZNAM DOBICEK.txt", "r")
dat <- readLines(fl)
close(fl) #zapremo povezavo do datoteke
head(dat) #izpis prvih par vrstic
```

```
## [1] "  \f"
## [2] " DURS - Izpostava: MARIBOR
## [3] "                                                    O D M E R N I   S E Z N A M
## [4] "                                                     ZA LETO "
## [5] ""
## [6] " ------------------------------------------------------------------------------------------------------------
```

Najprej poskušajmo zaznati v katerih vrsticah se začne nova stran. To je seveda vrstica 1, ter nato vsaka, ki

vsebuje niz '===='. Na koncu dodajmo še indeks zadnje vrstice.

```
strani <- c(1, grep("===", dat), length(dat))
strani
```

```
## [1]  1 31 71
```

Glavna ideja je, da preberemo stran za stranjo in te podatke združimo. Za to lahko uporabimo zanko:

```
for (i in 1:(length(strani)-1)){
  tmp <- preberi_stran(dat[strani[i]:strani[i + 1]])
  dokument <- rbind(dokument, tmp)
}
```

V zgornji zanki smo napisali funkcijo `preberi_stran`, ki ji podamo indeks prve in zadnje vrstice te strani. Nato te podatke združimo z `rbind`.

Poglejmo si sedaj, kako bi sprogramirali to funkcijo. Za nekoliko bolj podrobno razlago glejte komentarje v kodi:

```
preberi_stran <- function(stran){
  glava <- grep(pattern = "---", x = stran)
  # Locimo podvojene vrstce
  vrstice_1 <- stran[seq(from = glava[2]+1, to = length(stran)-2, by = 2)]
  vrstice_2 <- stran[seq(from = glava[2]+2, to = length(stran)-1, by = 2)]

  # Popravimo prve vrstice
  vrstice_1 <- str_squish(vrstice_1) # odstranemo odvecne presledke
  # locimo glede na presledke, pustimo neurejeno zadnjo kolono
  vrstice_1 <- str_split(vrstice_1, " ", simplify = T,n = 4)
  # popravimo zadnjo kolono, locimo glede na vejico in pustimo neurejeno zadnjo
  vrstice_1 <- cbind(vrstice_1[, -4],
                     str_split(vrstice_1[, 4], ",", simplify = T, 2))
  # locimo preostale stolpce  in ohranimo locilo (oprostitev ali izguba)
  tmp <- str_split(vrstice_1[, 5], "Oprostitev|Izguba", simplify = T,  n =2)
  tmp2 <- str_match(vrstice_1[, 5], "Oprostitev|Izguba")
  # zdruzimo skupaj
  vrstice_1 <- cbind(vrstice_1[, -5],
                     tmp[, 1], tmp2, tmp[, -1])
  vrstice_1 <- as.data.frame(vrstice_1)
  colnames(vrstice_1) <- c("St", "st_odmere", "davcna", "naslovnik", "naslovnik_naslov", "odmera", "odm

  vrstice_2 <- str_squish(vrstice_2) # odstranemo odvecne presledke
  # Izvozimo tekstovne stoplce
  #\\d+ je ujemanje s katerokoli števko
  tmp <- str_split(vrstice_2, "\\d+", simplify = T, 4)[, 2:3]
  # Izvozimo numericne stolpce
  #\\D+ je ujemanje s katerikolim znakom razen števk
  tmp2 <- str_split(vrstice_2, "\\D+", simplify = T, 4)[,1:2]
  vrstice_2 <- cbind(tmp2[,1], tmp[,1], tmp2[,2], tmp[,2])
  vrstice_2 <- as.data.frame(vrstice_2)
  colnames(vrstice_2) <- c("st_odlocbe","naslovnik_kraj",
                           "naslovnik_posta_st", "naslovnik_posta")
  return(cbind(vrstice_1, vrstice_2))
}
```

V tej funkciji je uporabljenih nekaj funkcij za delo z nizi, ki jih še nismo obravnavali. Te so:

- `grep`: Vrne indekse, vzorec `pattern` pojavi v besedilu.
- `str_split`: Razdeli niz v več podnizov z podanim vzorcem.
- `str_match`: Izpiše ujemanja z podanim vzorcem.
- `str_squish`: Odstrani vse odvečne presledke, pusti največ enega med besedami.

Vsi vzorci so podani s tako imenovanim regularnim izrazom, ki omogoča iskanje poljubnega niza ali skupino nizov v besedilu. Področje regularnih izrazov je obširno v tem primeru pa uporabljamo le osnovne oblike le tega.

```
dokument <- NULL
for (i in 1:(length(strani)-1)){
  tmp <- preberi_stran(dat[strani[i]:strani[i + 1]])
  dokument <- rbind(dokument, tmp)
}
head(dokument)
```

Po potrebi bi lahko na podoben način dodali tudi stolpca *stran* za številko strani in *datum*, ki ga lahko preberemo iz vsake strani.

Osnovno delo s časovnimi vrstami

Vzemimo podatke *delci2.csv* iz petega predavanja. Ti podatki predstavljajo časovno vrsto. Za vsak dan imamo narejene različne meritve.

Naložimo te podatke in popravimo tip spremenljivke datum.

```
delci <- read.csv("./data_raw/delci2.csv")
delci$Datum <- base::as.Date(delci$Datum, format = c("%m/%e/%Y"))
head(delci)
```

```
##        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
```

Kar veliko uporabnih funkcij za delo z časovnimi vrstami lahko najdete v paketu **zoo** (Z's Ordered Observations). To je paket za delu z regularnimi in iregularnimi časovnimi vrstami.

Recimo, da želimo izračunati povprečje sedem dnevnih oken delcev PM10 v zraku. Za to lahko uporabimo funkcijo `rollmean`. Funkcija sprejme podatke, parameter `k` za izbiro števila sosednjih vrednosti na katereih se računa povprečje in parameter `fill` s katerim lahko zamenjamo začetne, končne in vmesne manjkajoče vrednosti. V tem primeru bomo tem vrednosti priredili kar vrednost NA - not available.

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

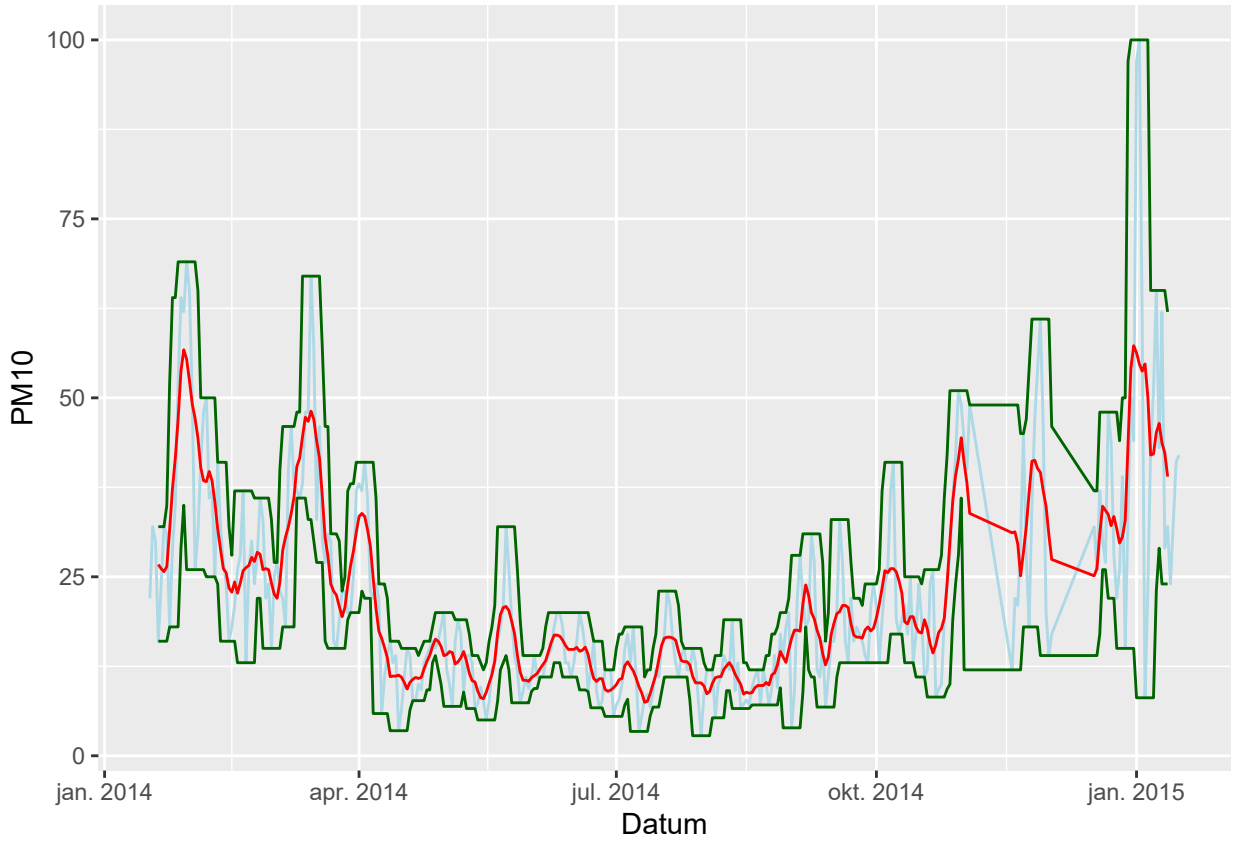
```
okna <- rollmean(delci$PM10, k = 7, fill = 'NA')
head(okna, n = 10)
```

```
##  [1]       NA       NA       NA 26.71429 26.14286 25.71429 26.42857 31.57143
```

```
##  [9] 37.28571 41.57143
```

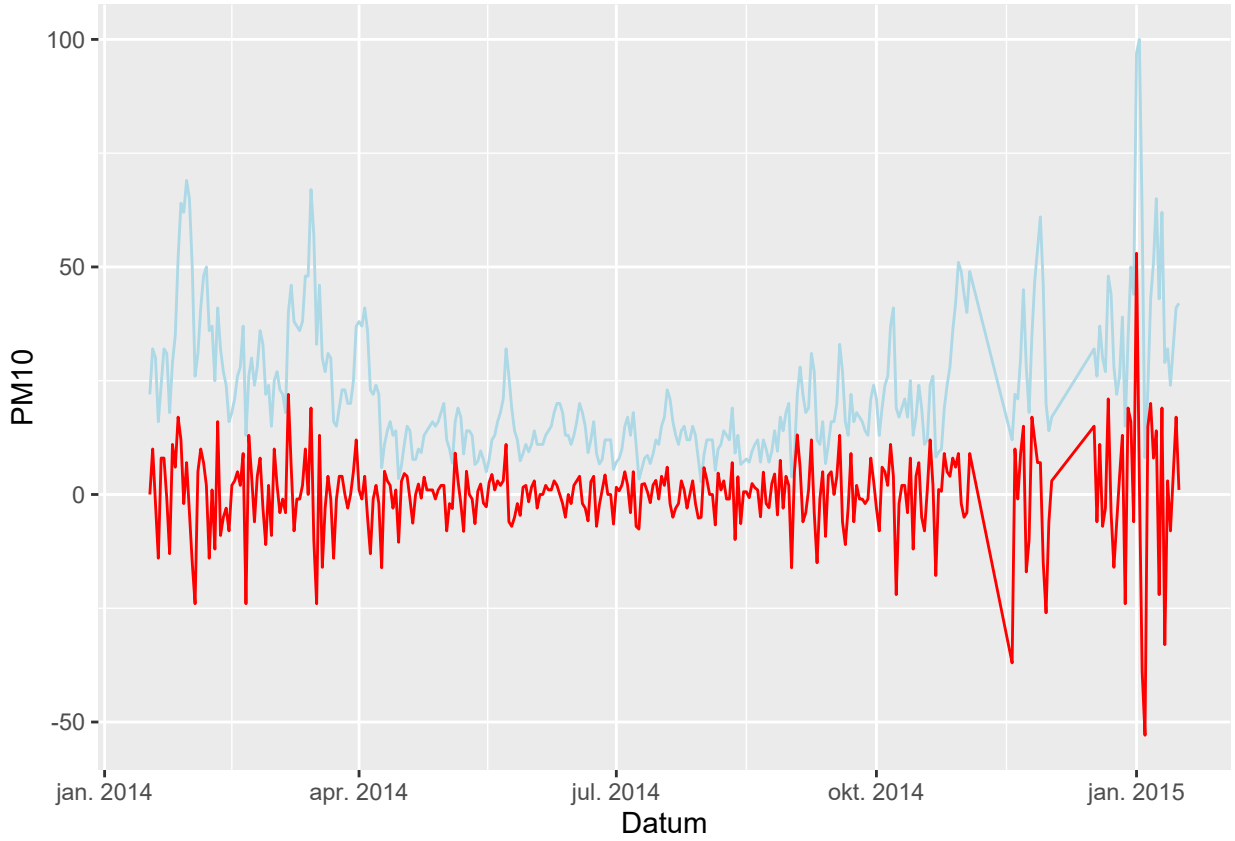
Obstaja še nekaj podobnih funkcij, kot je recimo `rollmax`, ki izračuna maksimalno vrednost okna. V primeru, da želite svoj izračun pa lahko uporabite `rollapply`, ki 'applicira' podano funkcijo na izbrane podatke. Poglejmo si te funkcije na spodnjem grafu.

```
ggplot(delci, aes(x = Datum, y = PM10)) +
  geom_line(color = "lightblue") +
  geom_line(color = "red", aes(y = rollmean(PM10, k = 7, fill = 'NA'))) +
  geom_line(color = "darkgreen", aes(y = rollmax(PM10, k = 7, fill = 'NA'))) +
  geom_line(color = "darkgreen", aes(y = rollapply(PM10, width = 7, FUN = min, fill = 'NA')))
```



Velikokrat uporabna funkcija je tudi `diff`, ki nam vrne razliko med dvema meritvama. Privzeto je ta razlika (lag) ena meritev, lahko pa jo nastavite tudi na večjo. Spodnji primer kaže izris razlik.

```
#zalika od prejšnjega dne
ggplot(delci, aes(x = Datum, y = PM10 )) +
  geom_line(color = "lightblue") +
  geom_line(color = "red", aes(y = c(0,diff(PM10))))
```



Preskočimo sedaj nekoliko naprej v poglavje strojnega učenja in na časovnih podatkih uporabimo znan model **arima** (autoregressive integrated moving average).

Paket, ki vsebuje zmogljivo implementacijo arime za R je **forecast**. Tukaj bomo prikazali samo preprost primer, da vidimo delovanje, podrobnejši opis pa lahko dobite v spletni knjigi.

Paket ima že nekaj vgrajenih funkcij, ki so lahko poznavalcem v pomoč. Poglejmo si razliko med meritvami.

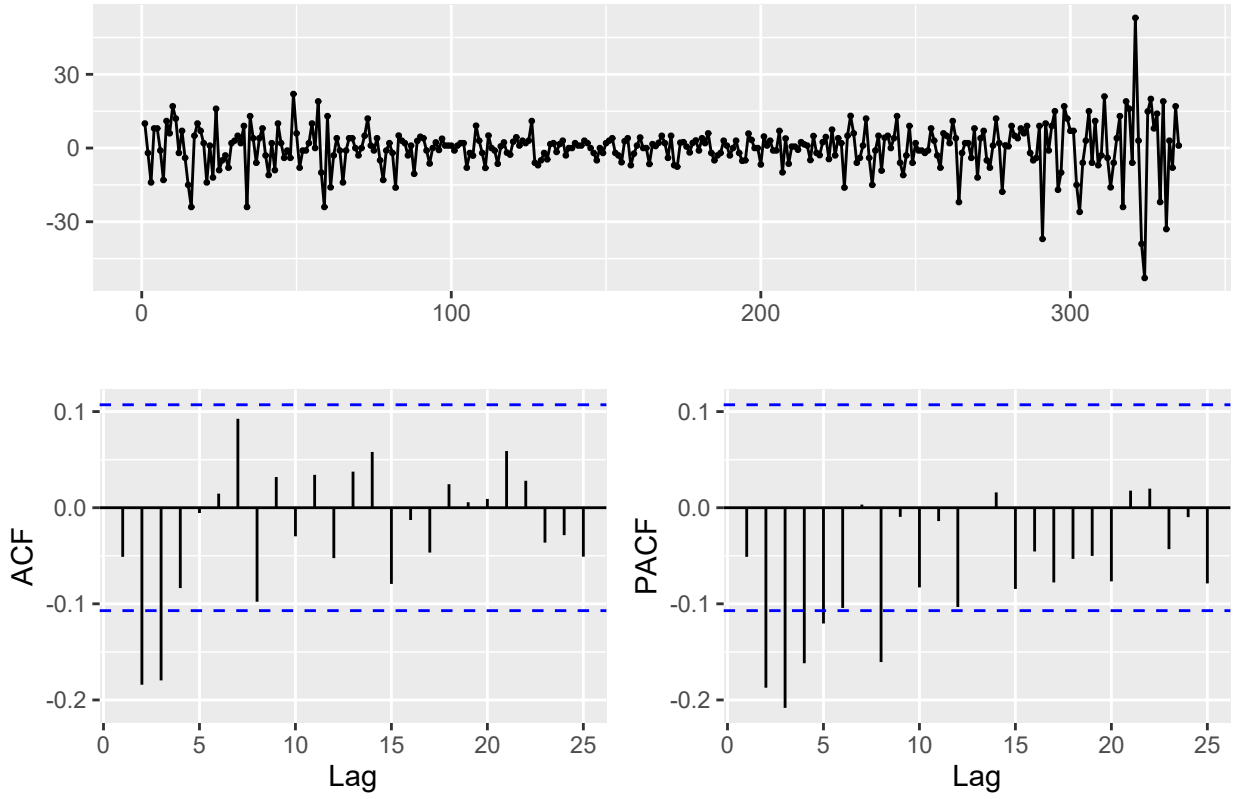
```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##
## Attaching package: 'forecast'
```

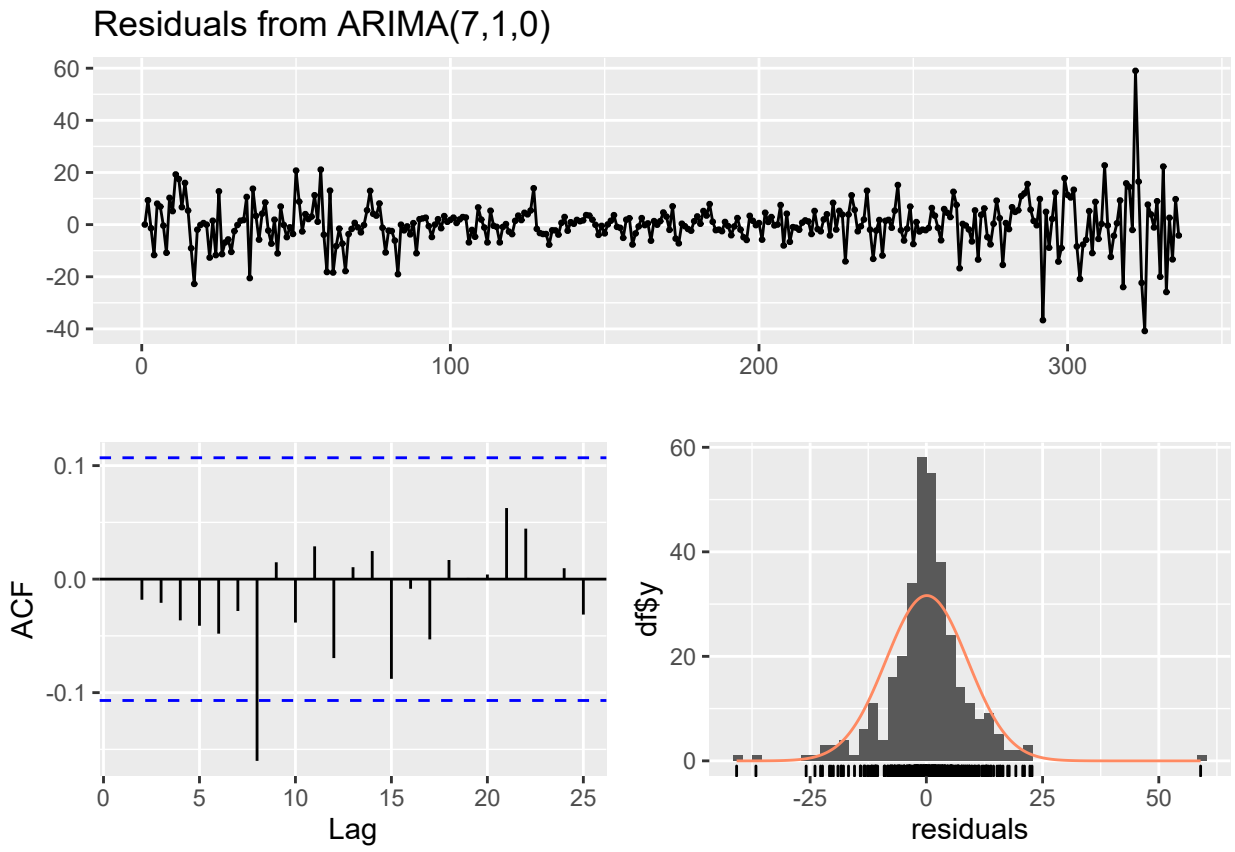
```
## The following object is masked from 'package:ggpubr':
##
##     gghistogram
```

```
ggtsdisplay(diff(delci$PM10), main = "")
```



V našem primeru ni videti očitnega dobrega modela, vendar bomo izbrali za prvi parameter vrednost 7, saj je ACF tam najvišji. Ogledamo si lahko tudi distribucijo residiualov.

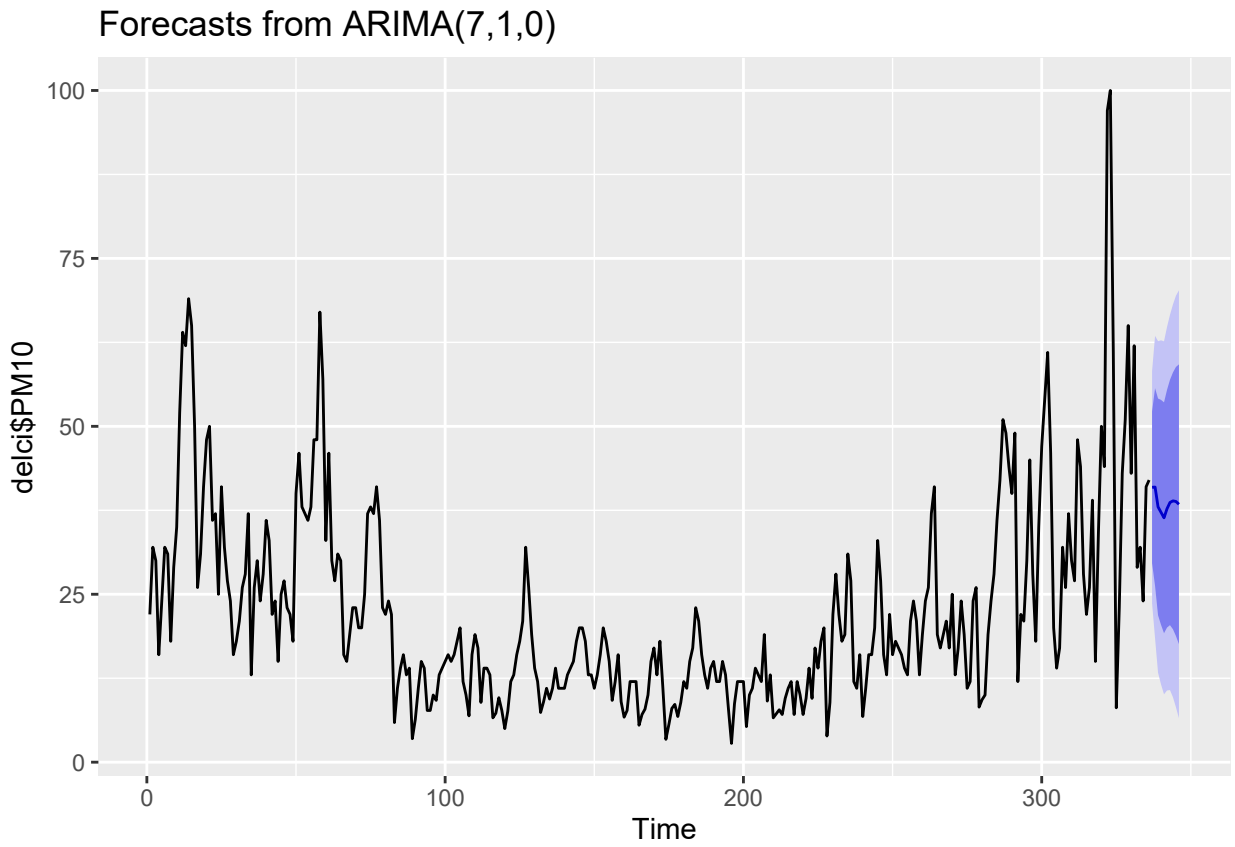
```
fit <- arima(delci$PM10, order = c(7, 1, 0))
checkresiduals(fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(7,1,0)
## Q* = 11.822, df = 3, p-value = 0.008017
##
## Model df: 7.   Total lags used: 10
```

Ustvarjen model lahko tudi uporabimo za napovedovanje prihodnjih vrednosti.

```
autoplot(forecast(fit))
```



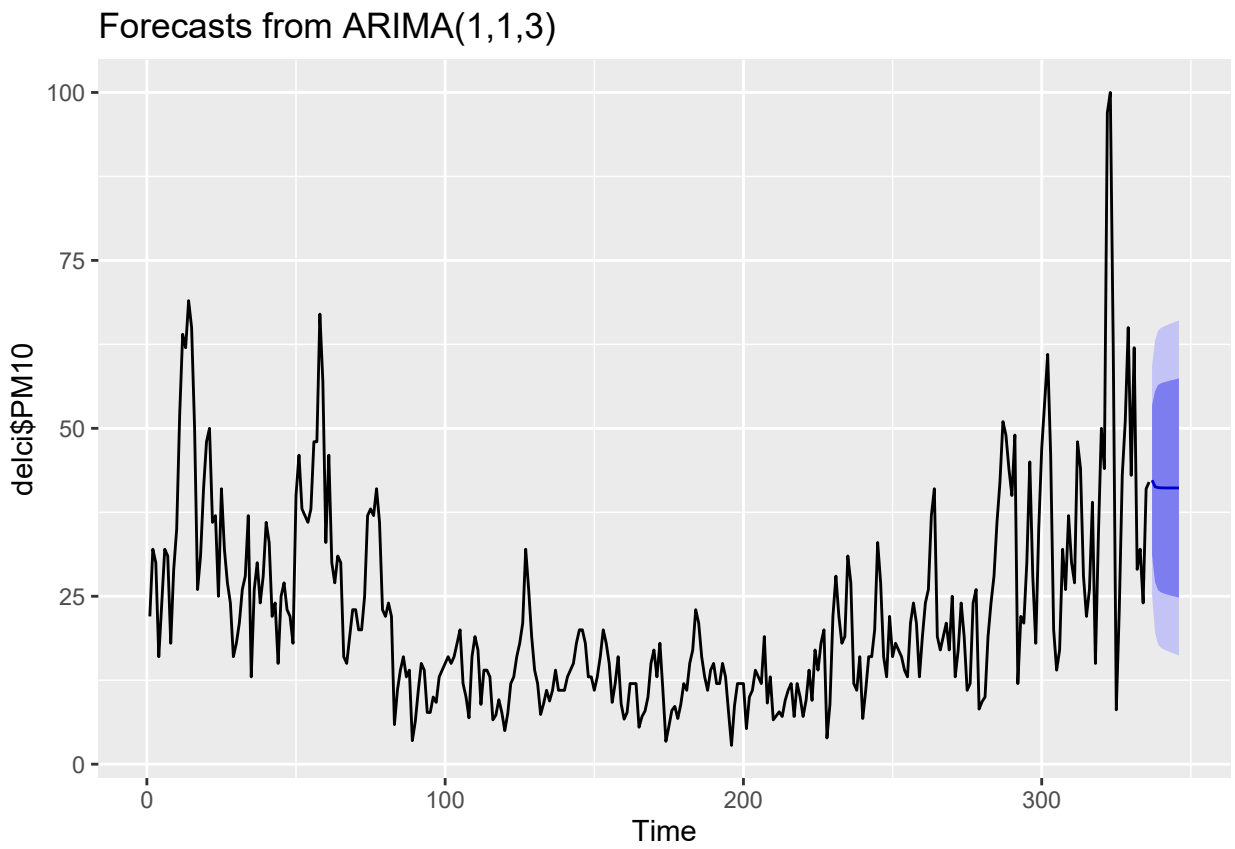
Pogljemo si samo še tabularični izpis:

```
forecast(fit)
```

```
##     Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 337       40.92263 29.62710 52.21815 23.647616 58.19764
## 338       40.95495 26.23763 55.67228 18.446746 63.46316
## 339       38.00371 21.86841 54.13900 13.326905 62.68051
## 340       37.18987 20.44386 53.93588 11.579054 62.80068
## 341       36.38249 19.20228 53.56271 10.107618 62.65736
## 342       37.72362 20.03788 55.40936 10.675613 64.77162
## 343       38.66437 20.40395 56.92478 10.737473 66.59126
## 344       38.90277 19.78670 58.01885  9.667254 68.13829
## 345       38.81210 18.78624 58.83797  8.185184 69.43902
## 346       38.39746 17.56952 59.22541  6.543869 70.25105
```

Paket vsebuje tudi funkcijo `auto.arima`,ki sama poskuša optimizirati parametre metode.

```
fit <- auto.arima(delci$PM10)
autoplot(forecast(fit))
```



Strojno učenje

Metod strojnega učenja je veliko in ponavadi jih dobite že vgrajene v raznih paketih. Tukaj si bomo pogledali en primer linearne regresije in modela z naključnim gozdom (Random Forest).

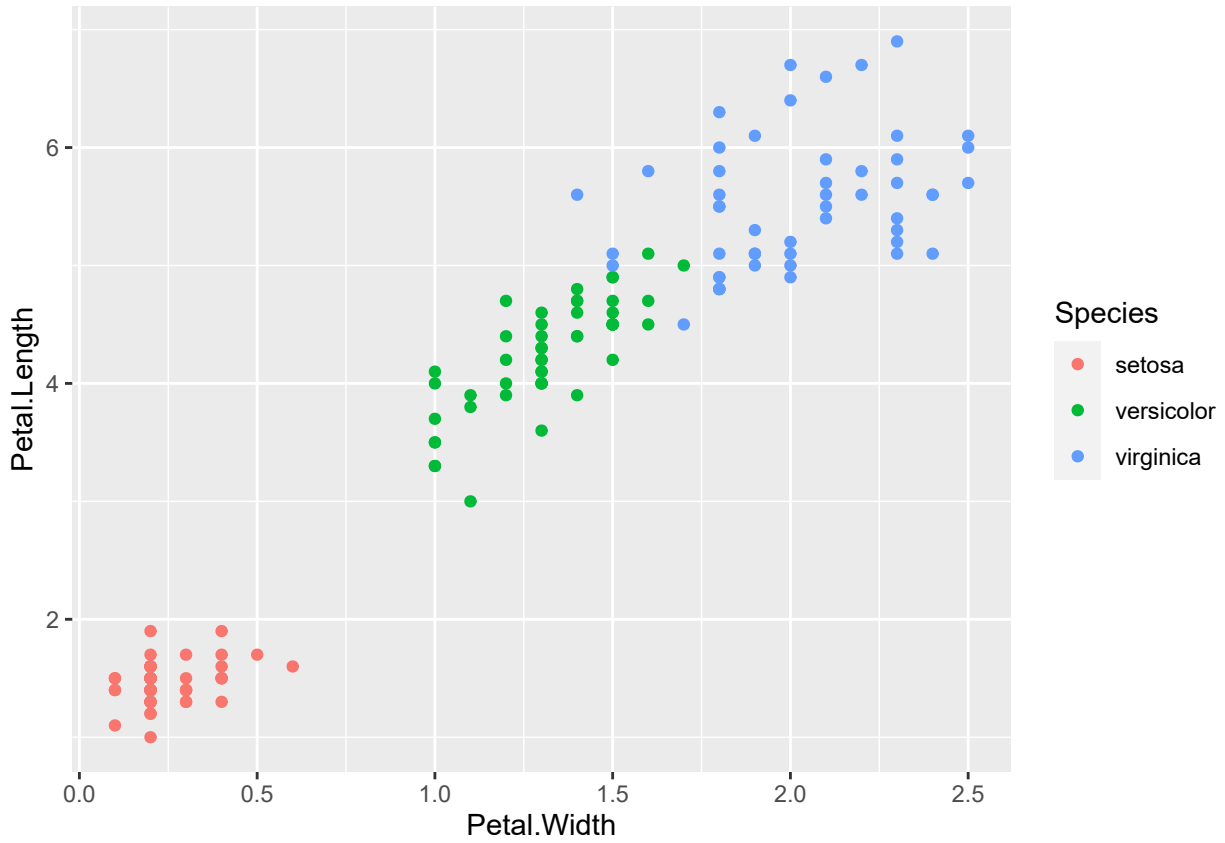
Uporabili bomo podatke `iris`, ki so že priloženi R-ju. Gre za preprost primer, kjer imamo podatke o velikosti čašnih in cvetnih listov preunik, želimo pa napovedati vrsto rože.

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Poglejmo si izris dveh stolpcev in razreda.

```
ggplot(iris, aes(x = Petal.Width, y = Petal.Length, colour = Species)) +
  geom_point()
```



Kot smo videli v uvodnem delu lahko zgradimo linearni model.

```
linearni_model <- lm(Species ~ ., iris)
linearni_model
```

```
##
## Call:
## lm(formula = Species ~ ., data = iris)
##
## Coefficients:
##  (Intercept)  Sepal.Length   Sepal.Width  Petal.Length   Petal.Width
##      1.18650      -0.11191      -0.04008       0.22865       0.60925
```

Kot vidimo, nam je R sporočil opozorilo, da ima težave napovedovati vrednost *Species*, ker je tipa faktor. Poskušajmo zato raje najprej napovedati širino cvetnih listov (Petal.Width) in uporabiti linearni model kot regresijo.

Pri strojnem učenju moramo podatke najprej razdeliti na učno in testno množico, da lahko preverimo ali model dobro deluje. Razdelimo naše podatke na 105 učnih primerov in 45 testnih.

```
set.seed(1234)
sel <- sample(1:nrow(iris), 45, replace =  F)
train <- iris[-sel,]
test <- iris[sel,]
```

Zgradimo linearni model:

```
linearni_model <- lm(Petal.Width ~ Petal.Length + Sepal.Width + Sepal.Length, train)
linearni_model
```

```
##
## Call:
## lm(formula = Petal.Width ~ Petal.Length + Sepal.Width + Sepal.Length,
##     data = train)
##
## Coefficients:
##  (Intercept)  Petal.Length   Sepal.Width  Sepal.Length
##      -0.5506        0.4958        0.2025       -0.1252
```

Skoraj vsi paketi za strojno učenje imajo na voljo tudi funkcijo `predict` s katero lahko napovedujemo neznane vrednosti.

```
napovedi <- predict(linearni_model, test)
head(napovedi)
```

```
##        28        80       101       111       137       133
## 0.2504732 0.9971944 2.3032659 1.8117563 2.1251982 1.9911815
```

Za oceno napovednega modela bomo kar sami spisali nekaj osnovnih funkcij.

```
# srednja absolutna napaka
mae <- function(obs, pred){
  mean(abs(obs - pred))
}

# srednja kvadratna napaka
mse <- function(obs, pred){
  mean((obs - pred)^2)
}

# relativna srednja kvadratna napaka
rmse <- function(obs, pred, mean.val){
  sum((obs - pred)^2)/sum((obs - mean.val)^2)
}
```

Preverimo rezultate.

```
mae(test$Petal.Width, napovedi)
```

```
## [1] 0.1573341
```

```
mse(test$Petal.Width, napovedi)
```

```
## [1] 0.04717747
```

```
rmse(test$Petal.Width, napovedi, mean(train$Petal.Width))
```

```
## [1] 0.07522785
```

Poskušajmo še z metodo naključnega gozda.

```
#naključno drevo
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
#regresija
rf <- randomForest(Petal.Width ~ Petal.Length + Sepal.Width + Sepal.Length,
                   data = train)
napovedi <- predict(rf, test)
head(napovedi)
```

```
##        28        80       101       111       137       133
## 0.2493546 1.2410235 2.0660107 1.9139621 2.0589079 1.9217774
```

Primerjajmo rezultate:

```
mae(test$Petal.Width, napovedi)
```

```
## [1] 0.1650388
```

```
mse(test$Petal.Width, napovedi)
```

```
## [1] 0.04955582
```

```
rmse(test$Petal.Width, napovedi, mean(train$Petal.Width))
```

```
## [1] 0.0790203
```

Metoda naključnega gozda se lahko prilagodi tudi za iskanje faktorjev (klasifikacija).

```
rf <- randomForest(Species ~ .,
                   data = train)
napovedi_r <- predict(rf, test, type = "class")
napovedi_v <- predict(rf, test, type = "prob")
head(napovedi_r) #napovedi razredov
```

```
##         28         80        101        111        137        133
##     setosa versicolor  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```
head(napovedi_v) #verjetnosti napovedi
```

```
##     setosa versicolor virginica
## 28   1.000      0.000     0.000
## 80   0.000      0.998     0.002
## 101  0.000      0.018     0.982
## 111  0.000      0.038     0.962
## 137  0.002      0.012     0.986
## 133  0.000      0.002     0.998
```

Preverimo še odstotek pravilno napovedanih vrst.

```
sum(test$Species == napovedi_r) / length(test$Species)
```

```
## [1] 0.9777778
```

```
rbind(test$Species, napovedi_r)
```

```
##            28 80 101 111 137 133 144 132 98 103 90 70 79 116 14 126 62 4 143 40
##             1  2   3   3   3   3   3   3  2   3  2  2  2   3  1   3  2 1   3  1
## napovedi_r  1  2   3   3   3   3   3   3  2   3  2  2  2   3  1   3  2 1   3  1
##            93 122 5 66 135 47 131 123 84 48 108 3 87 41 115 100 72 32 42 43 2
```

```
##             2   3 1  2   3  1   3   3  2  1   3 1  2  1   3   2  2  1  1  1 1
## napovedi_r  2   3 1  2   3  1   3   3  3  1   3 1  2  1   3   2  2  1  1  1 1
##            138 54 49 102
##              3  2  1   3
## napovedi_r   3  2  1   3
```

V R-ju je še veliko paketov za delo z metodami strojnega učenja. Pogledate si lahko naprimer še pakete **rpart**, **e1071**, **kknn**, **naiveBayes**, **nnet**, **coreLearn**.