

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
```

```
# ANOVA 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)
```

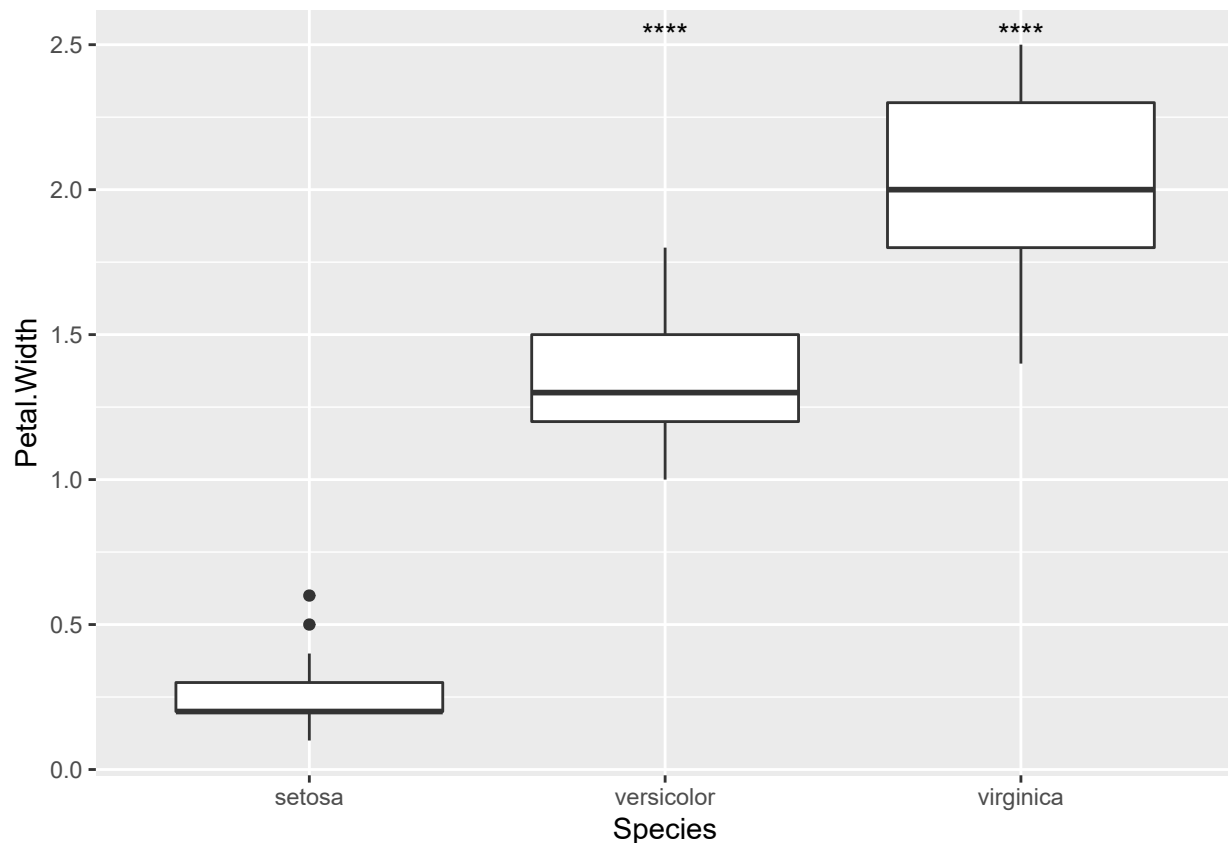
```
library(ggpubr)
```

```
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

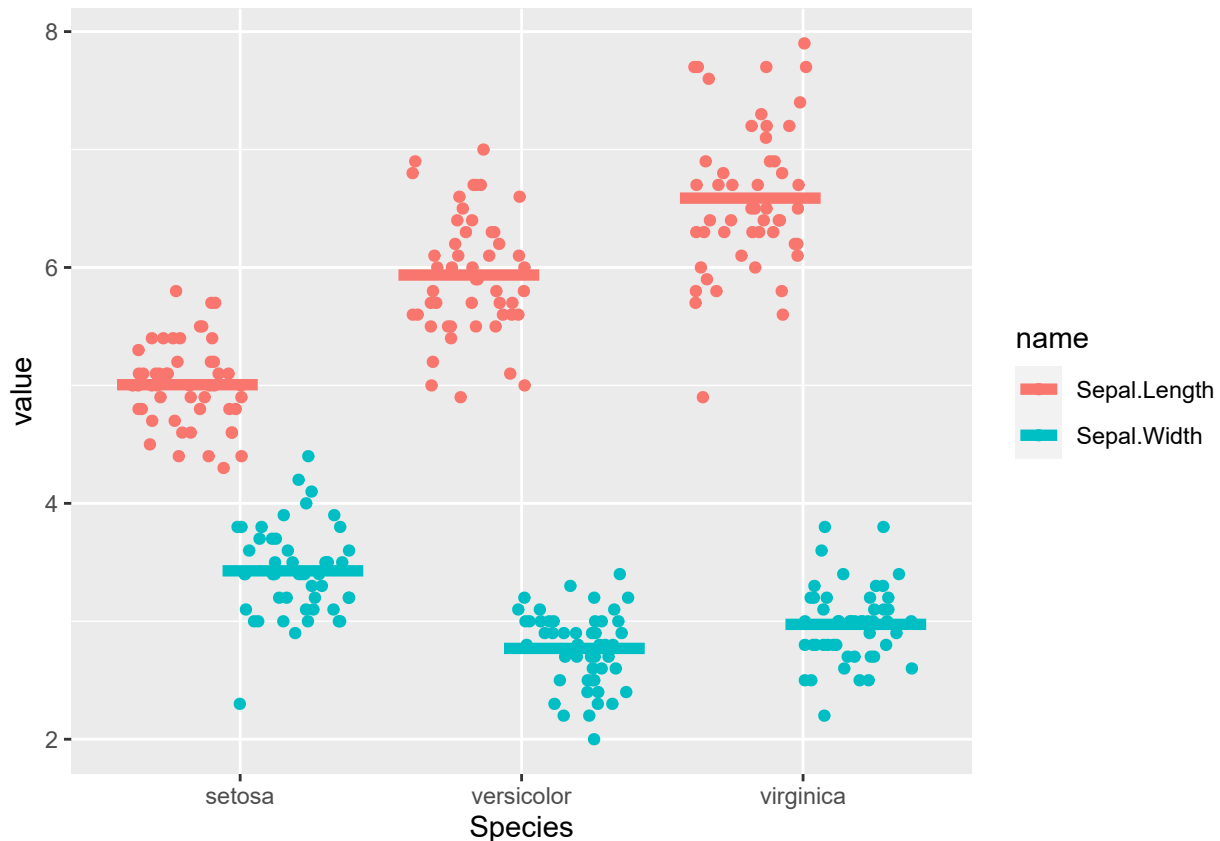
Poglejmo si še en zanimiv graf, kjer bomo prikazali točke in povprečja na istem grafu. Pogledali si bomo porazdelitve dolžin in širin čašnih listov različnih perunik. Najprej si pripravimo `data.frame`.

```
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_hline` 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 = "hline"
  )
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
```

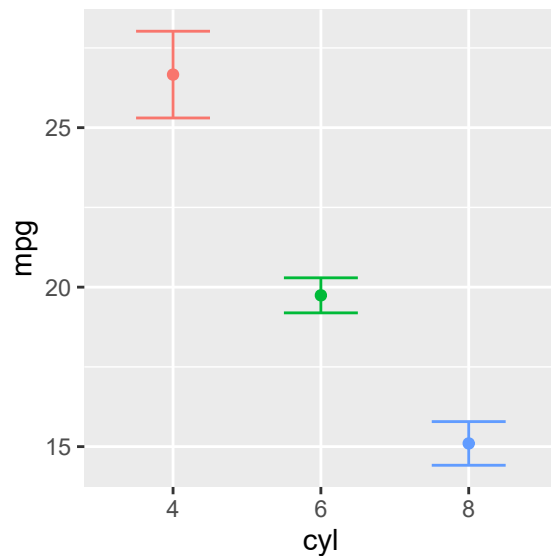
```
mus <- aggregate(mpg ~ cyl, mtcars, FUN = mean)
sds <- aggregate(mpg ~ cyl, mtcars, FUN = function(x) {sd(x) / sqrt(length(x))})
df <- cbind(mus, SE = sds$mpg)
df$cyl <- as.character(df$cyl)

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 narisnemu 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:

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



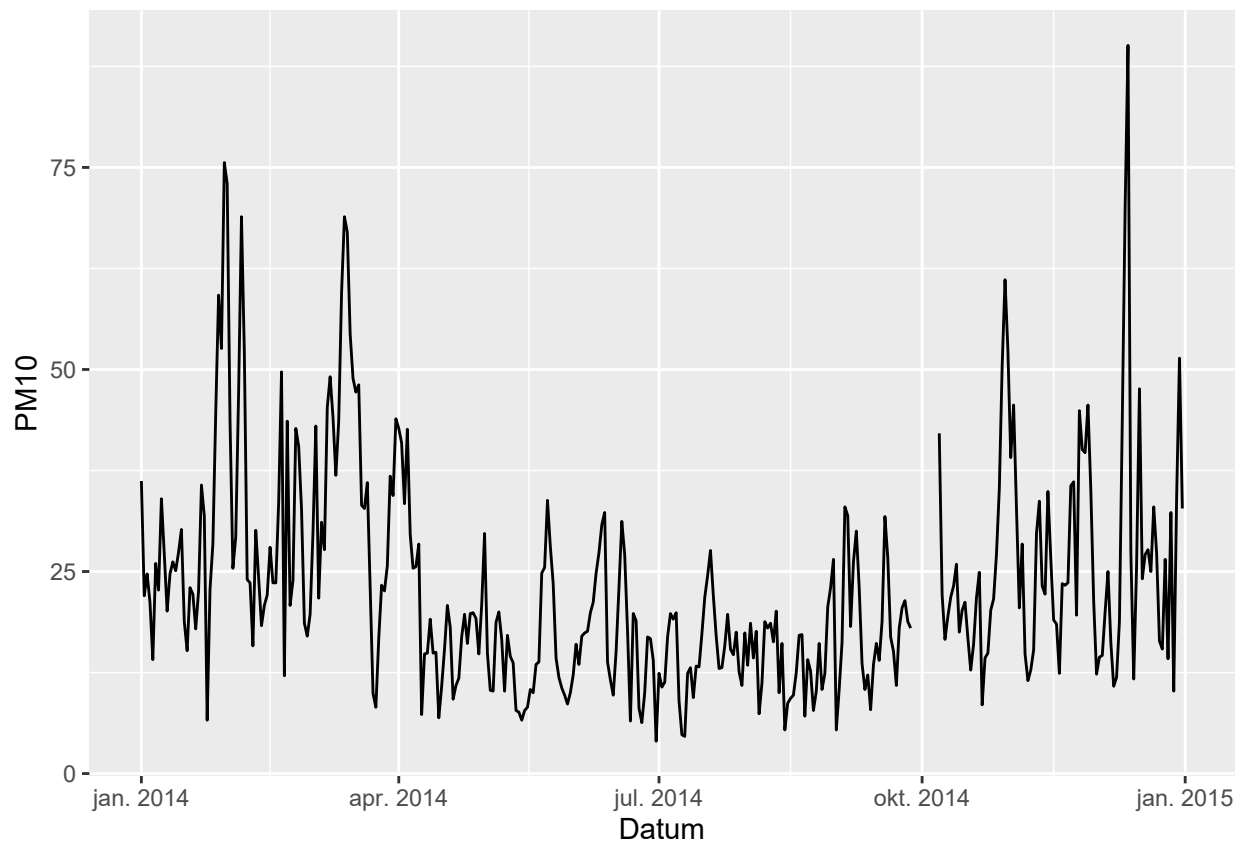
Sedaj pogledjmo, kako lahko pridemo do te rešitve korak po koraku.

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)
```

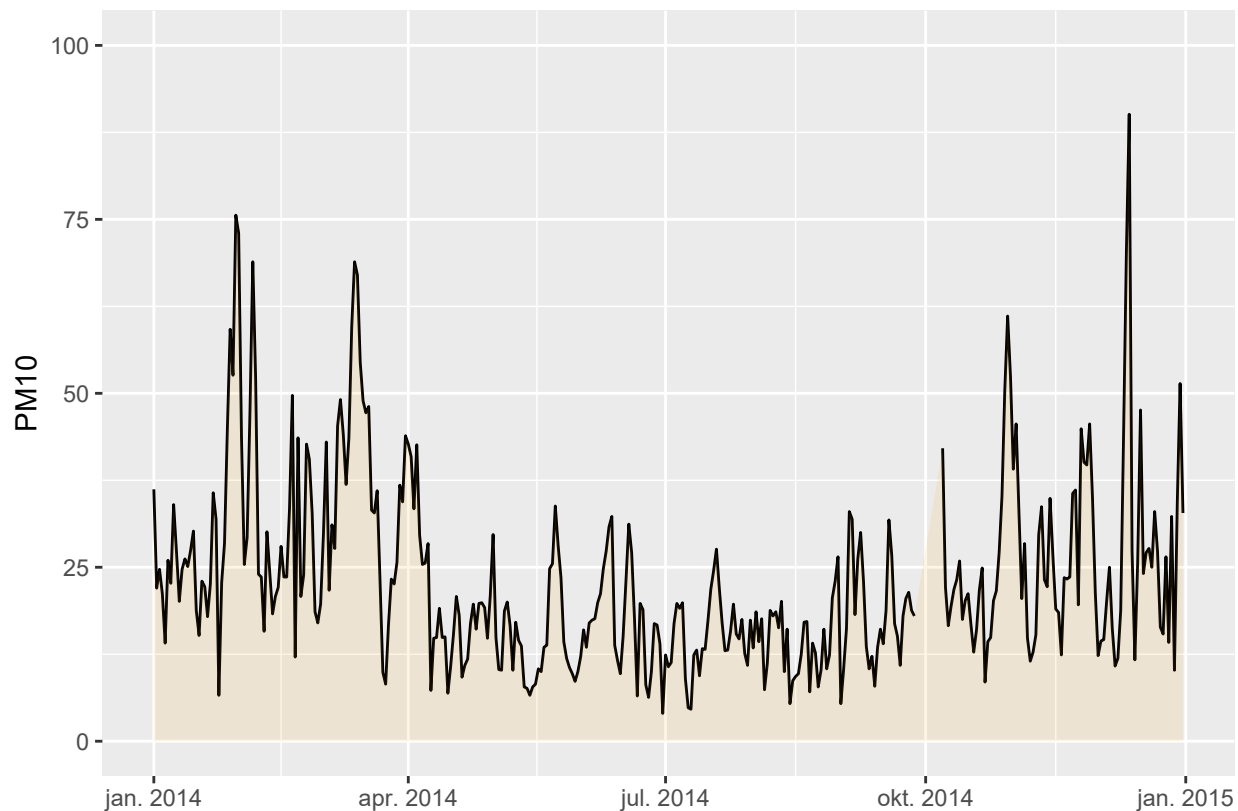
Namesto izrisa treh grafov se bomo za začetek osredotočili za izris vsebnosti delcev PM10 samo za Ljubljano. Izrišimo samo to črto.

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



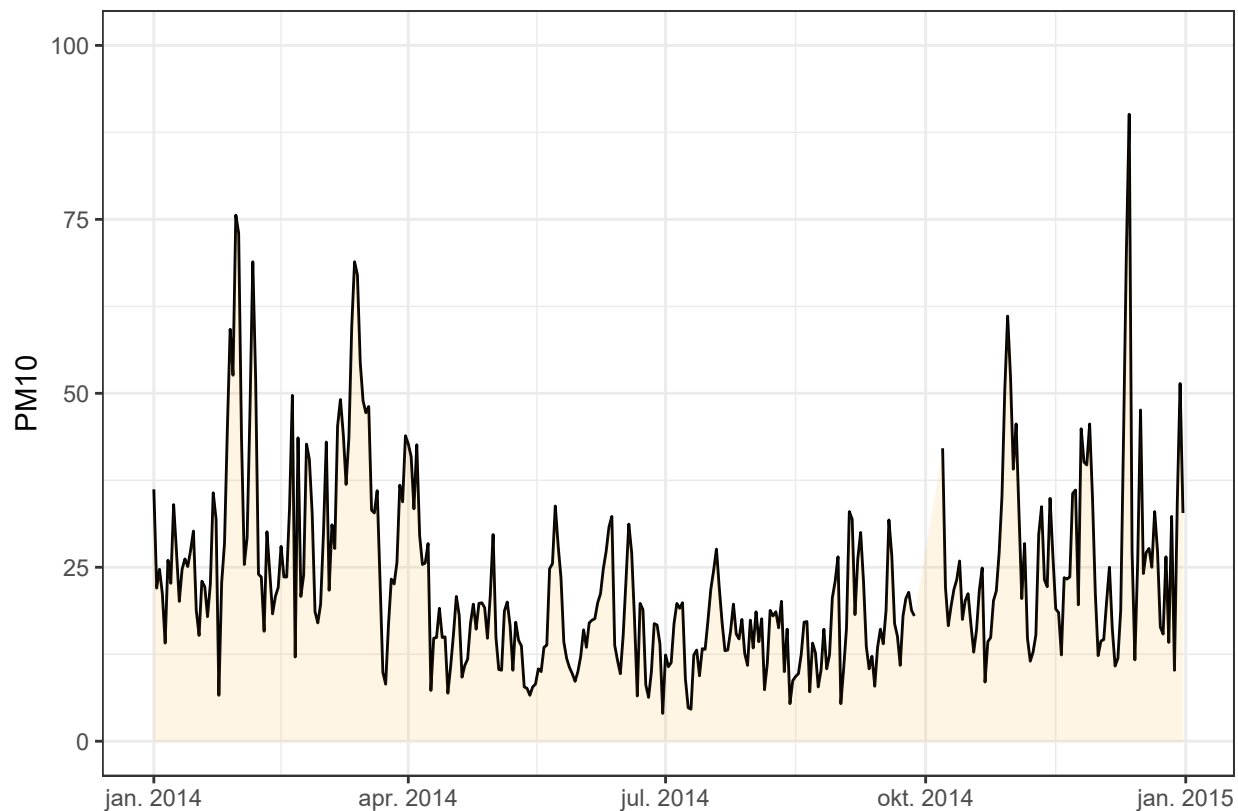
Z ukazom `geom_area` lahko na grafu podamo ploščino. S spodnjim ukazom jo pobarvamo oranžno in naredimo skoraj transparentno. Grafu še omejimo y os med 0 in 100 ter izbrišemo ime osi x.

```
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("")
```



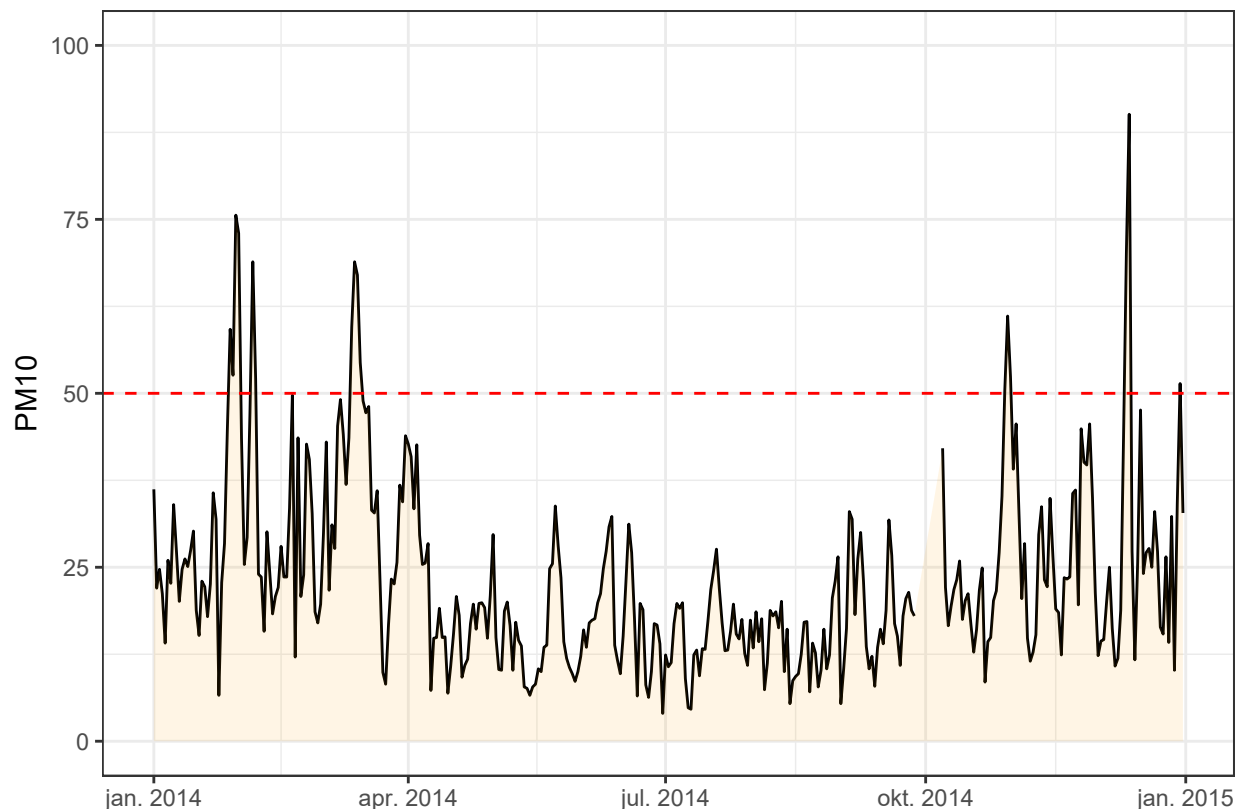
Paket **ggplot2** ima že prednastavljene teme za izris grafov. Ena izmed teh je **theme_bw()**, ki jo bomo uporabili. Z ukazom **?theme_bw()** lahko v zavihku **help** vidite tudi ostale prednastavljene teme. Z ukazom **theme()** pa jih lahko tudi sami definirate.

```
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.

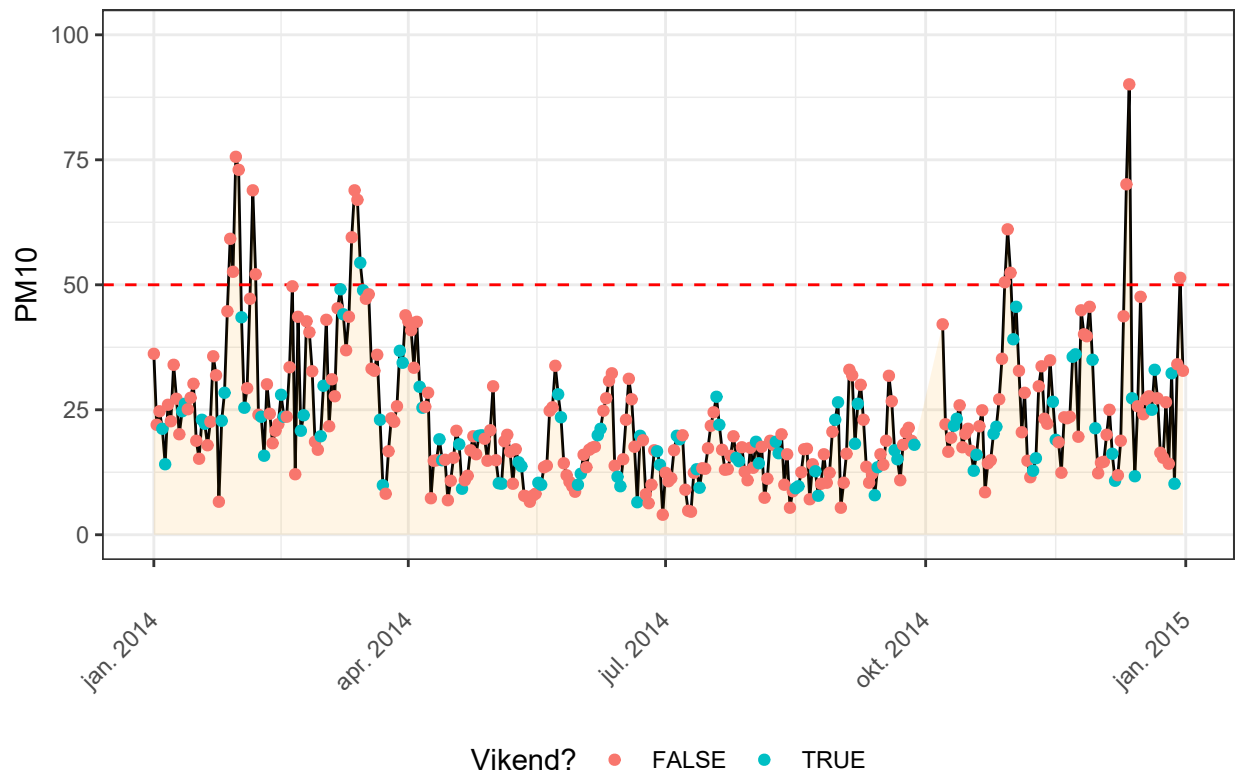
```
ggplot(df[df$kraj == "Ljubljana",], aes(x = Datum, y = PM10)) +
  geom_line(colour = "black") +
  geom_area(alpha = 0.1, fill = "orange") + ylim(0, 100) +
  xlab("") +
  theme_bw() +
  geom_hline(yintercept = 50, lty = "dashed", colour = "red")
```



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 naslov 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 parametrov `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))
```

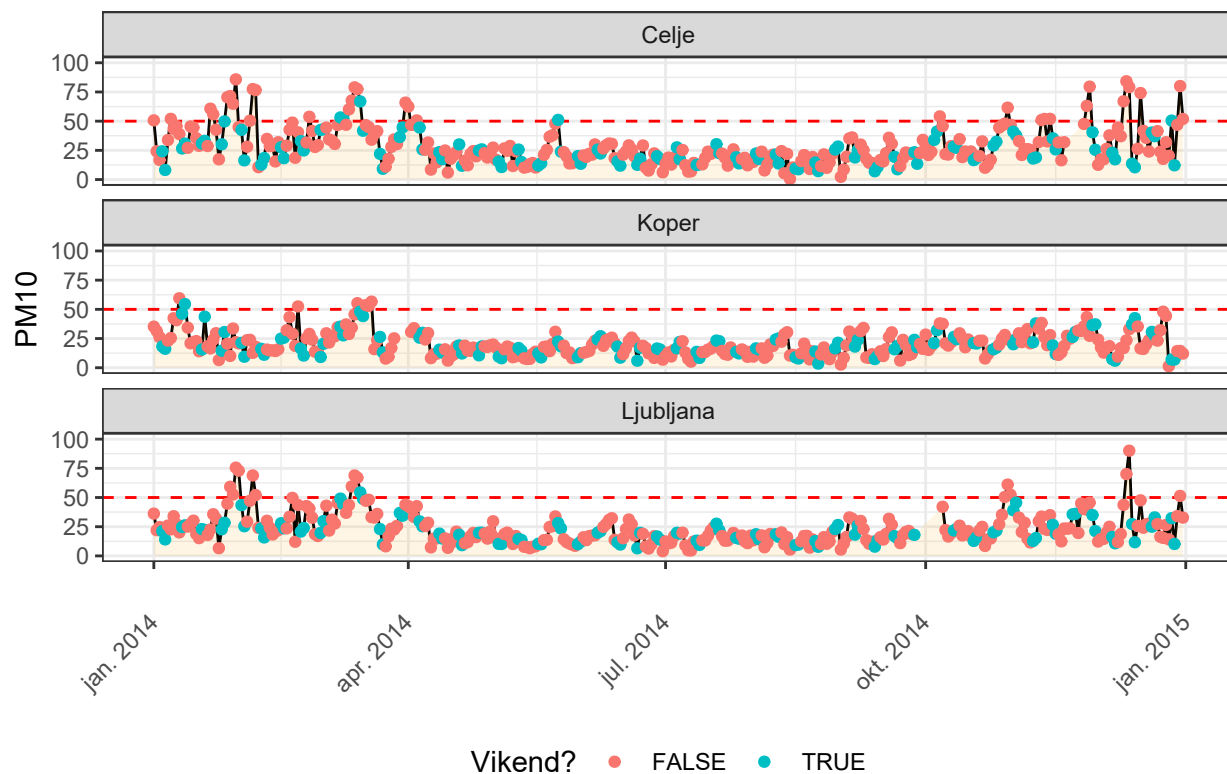
Vrednosti meritev PM10 za tri slovenska mesta.



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` izriše za vsako skupino svoj graf, `ncol = 1` pa pove, da želimo vse skupne izrisati v samo enem stolpcu.

```
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)
```

Vrednosti meritev PM10 za tri slovenska mesta.

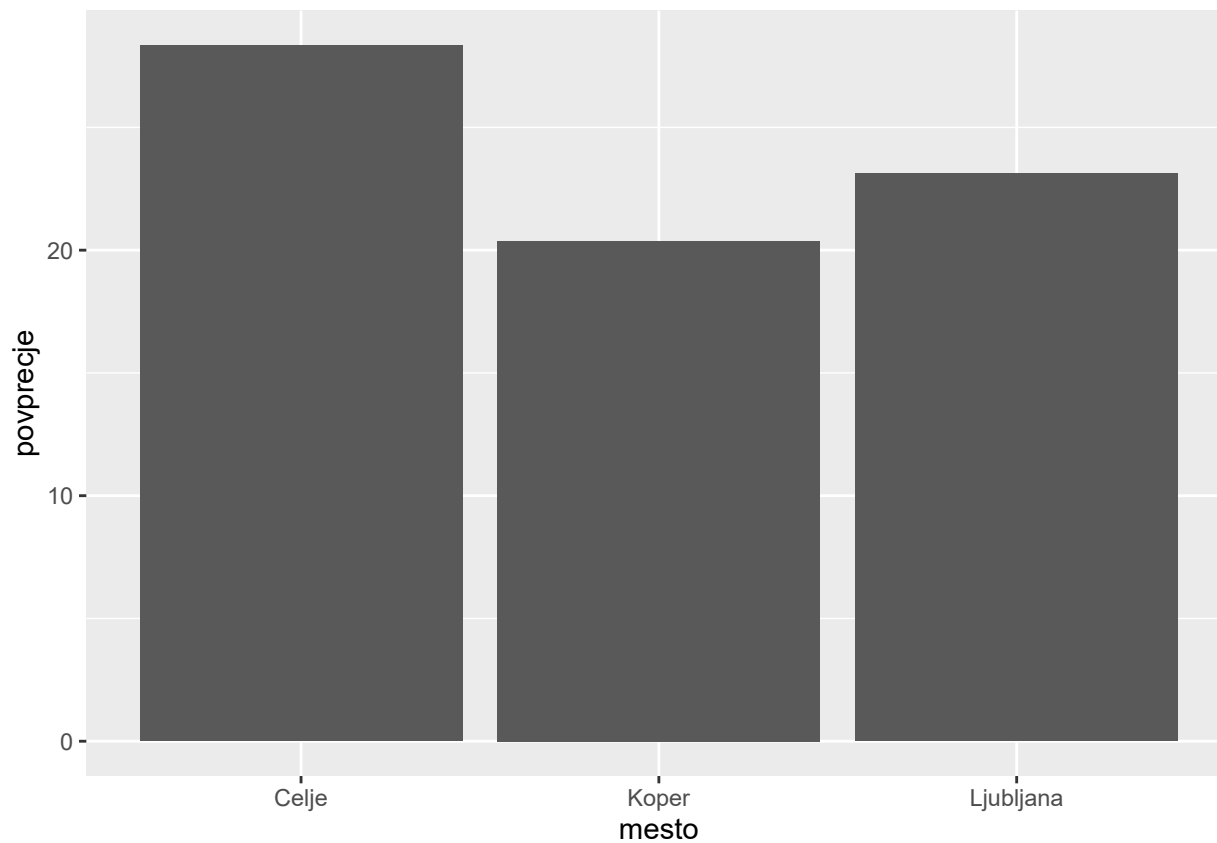


Tako smo prišli do končnega grafa.

Vzorci in slike na grafih

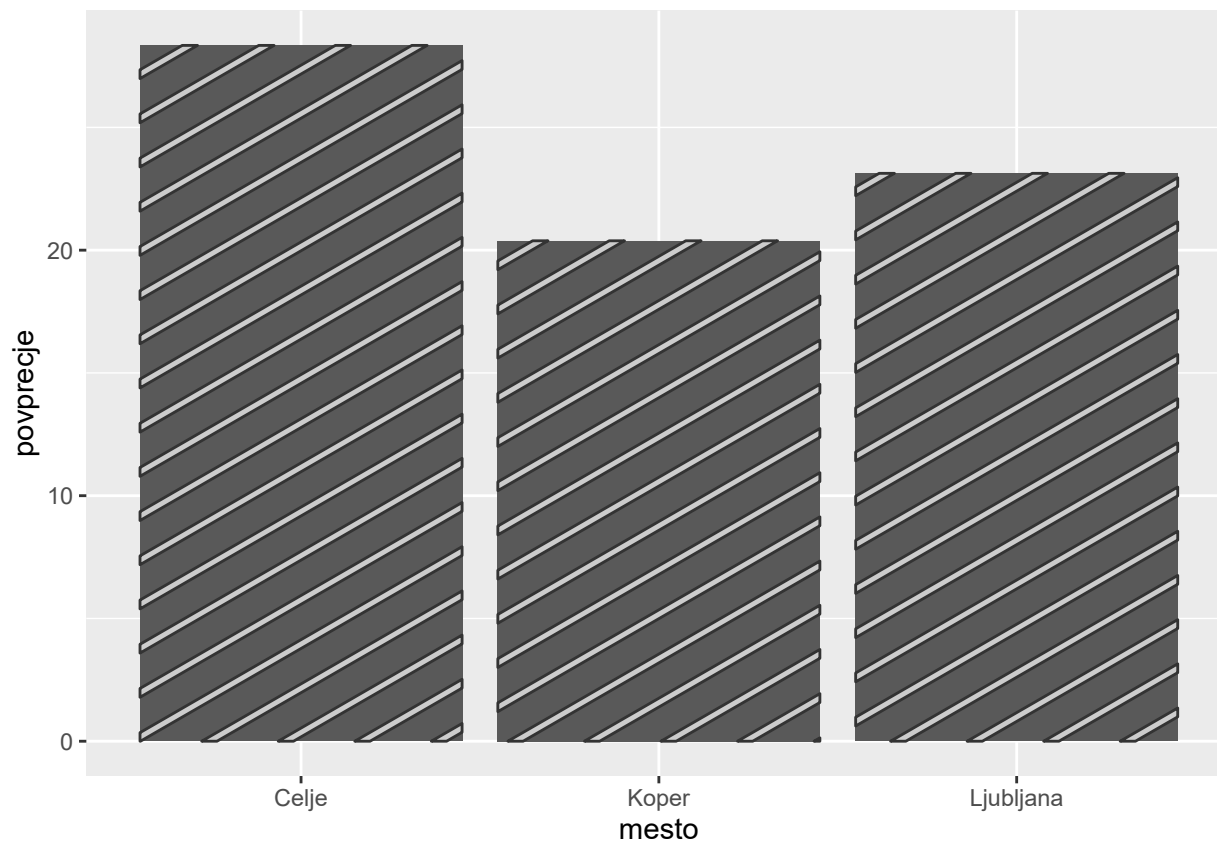
Uporabimo podatke o delcih, da izrišemo stolpični graf, ki prikazuje povprečno onesnaženost glede na mesto.

```
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.

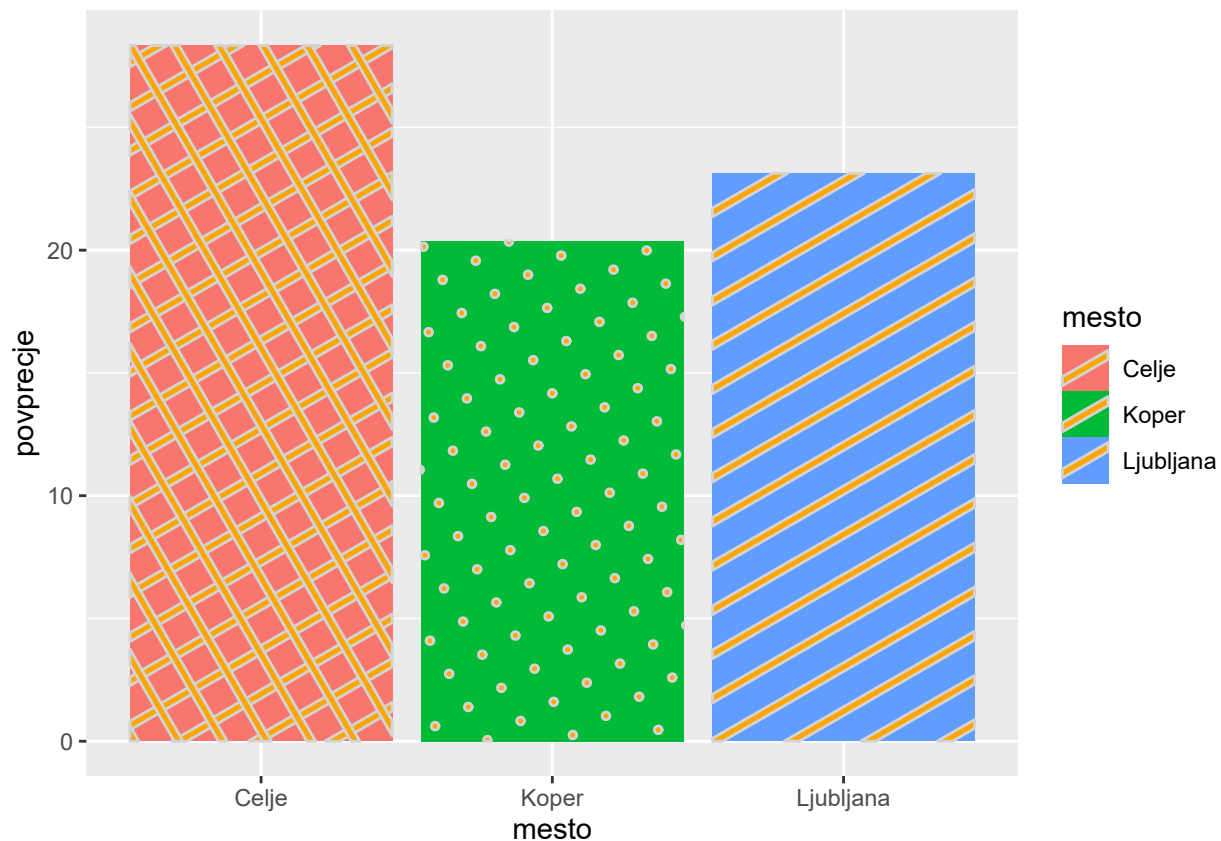
```
library(ggpattern)
ggplot(d, aes(x = mesto, y = povprecje)) +
  geom_bar_pattern(stat = "identity")
```



Opazimo, da avtomatsko izriše črte (stripes).

Dodajmo še barve in nekaj drugih privzetih vzorcev.

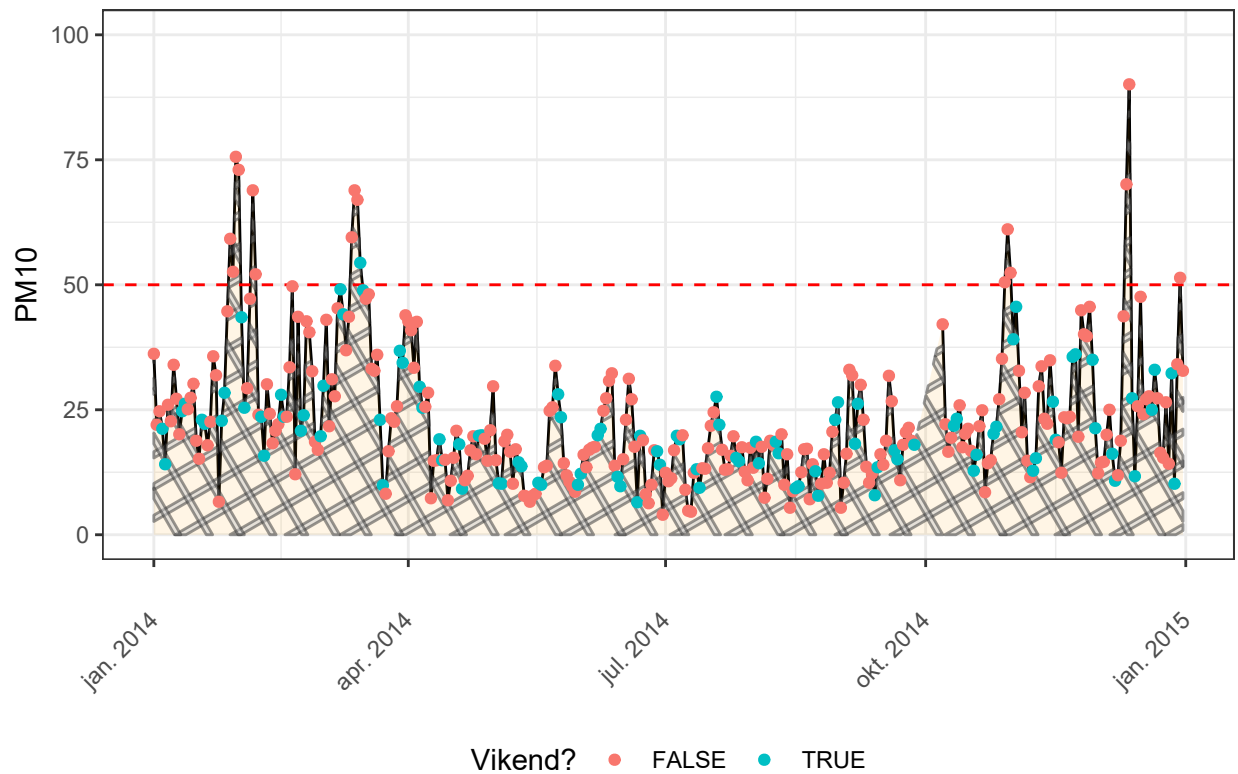
```
ggplot(d, aes(x = mesto, y = povprecje)) +  
  geom_bar_pattern(stat = "identity", aes(fill = mesto, pattern = mesto,  
                                          pattern_fill = "green"),  
                  pattern = c("crosshatch", "circle", "stripe"),  
                  pattern_fill = "orange",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



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

```
ggplot(df[df$kraj == "Ljubljana",], aes(x = Datum, y = PM10)) +
  geom_line(colour = "black") +
  geom_area_pattern(alpha = 0.1, fill = "orange", pattern = "crosshatch", pattern_alpha = 0.5) + ylim(0, 50) +
  xlab("") +
  theme_bw() +
  geom_hline(yintercept = 50, lty = "dashed", colour = "red") +
  geom_point(data = df[df$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))
```

Vrednosti meritev PM10 za tri slovenska mesta.

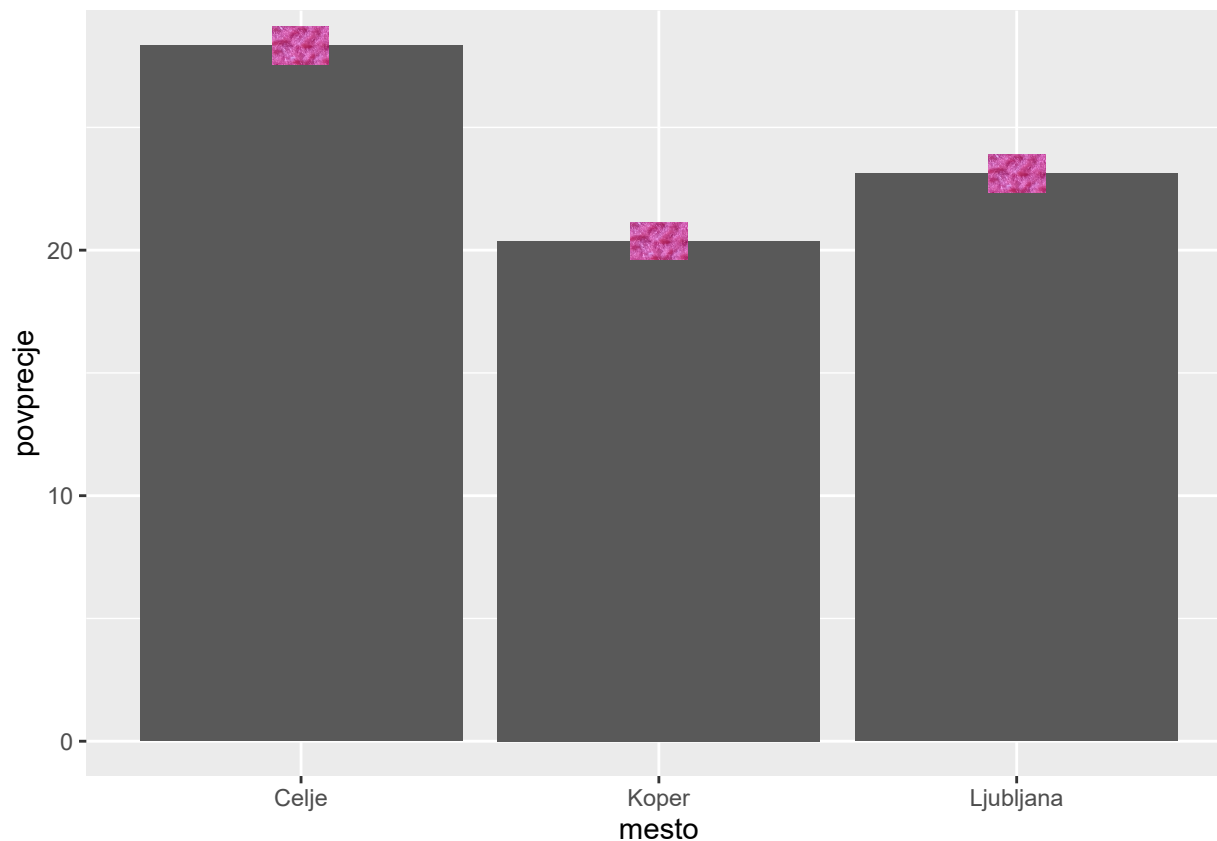


Na grafe lahko s paketom **ggimage** dodamo tudi slike. Paket uporablja drug paket **magick**, ki ga po potrebi predhodno naložite.

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()
```

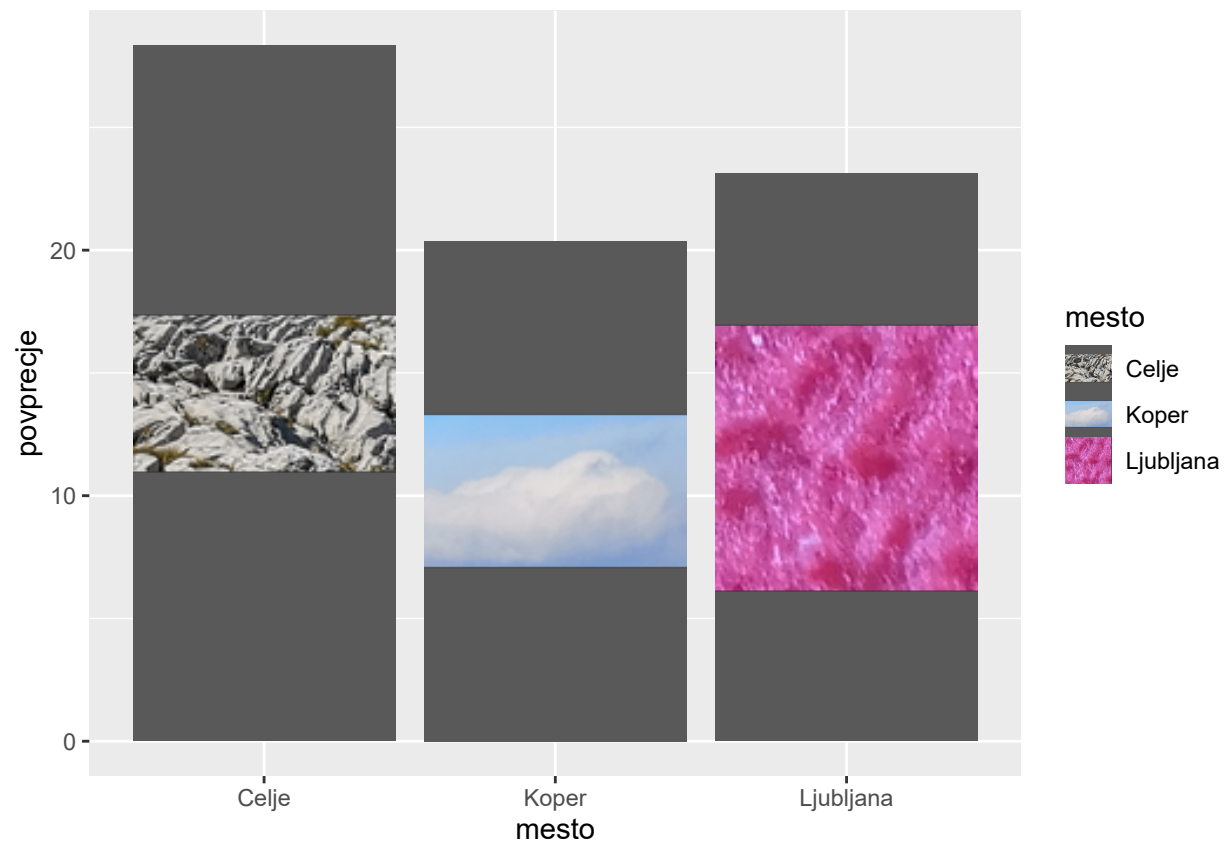



Vidimo, da slikico izriše na vrhu. Podobno lahko tudi namesto točk izrisujemo slike. Poglejmo si, kako bi celoten stolpec zamenjali s poljubnim vzorcem (sliko). Najprej si naredimo vektor s potmi do vzorcev na računalniku ali spletu.

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

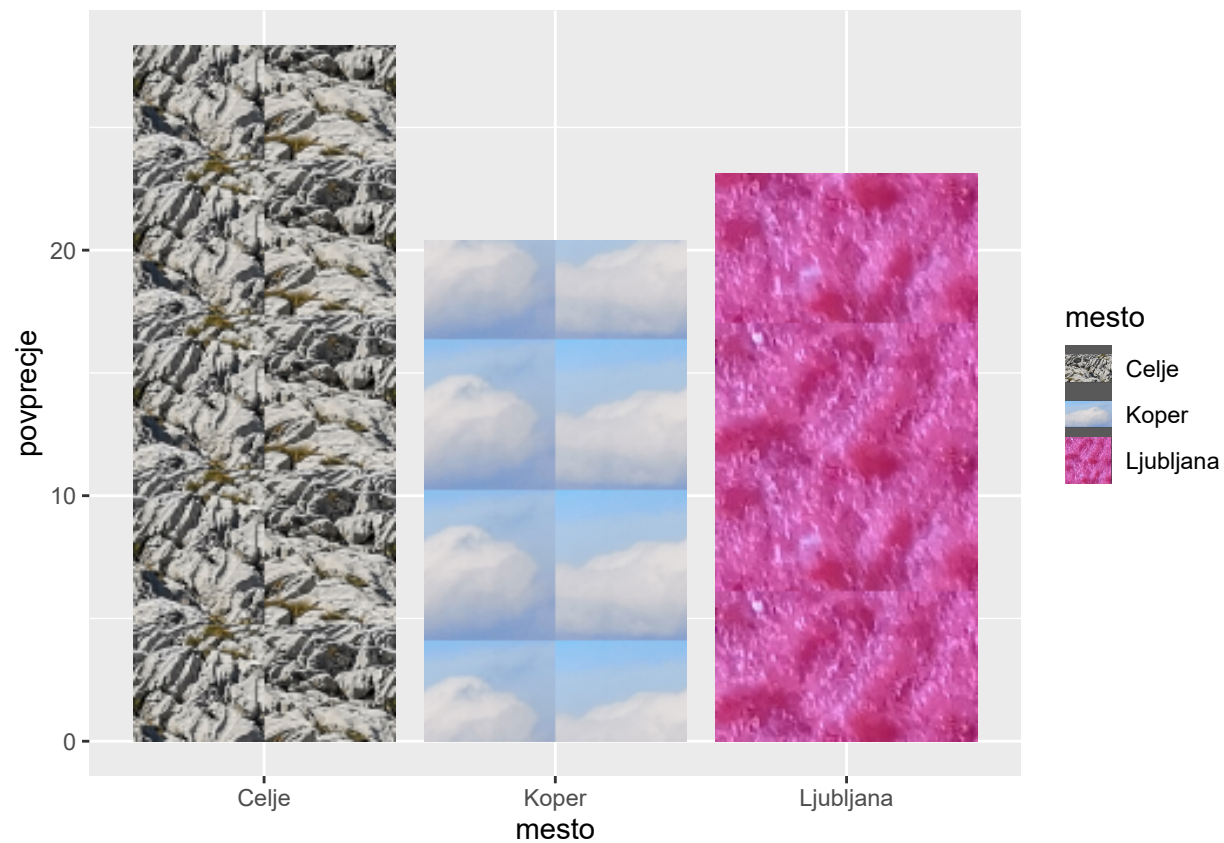
Parameter `pattern_filename` nam omogoča, da izbiramo različne spremenljivke glede na vrednosti stolpca `mesto`. Če uporabimo ta parameter moramo podati poti do slik s funkcijo `scale_pattern_filename_discrete`. Zadnja beseda te funkcije se lahko spreminja, glede na to kateri geom smo predhodno uporabili.

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



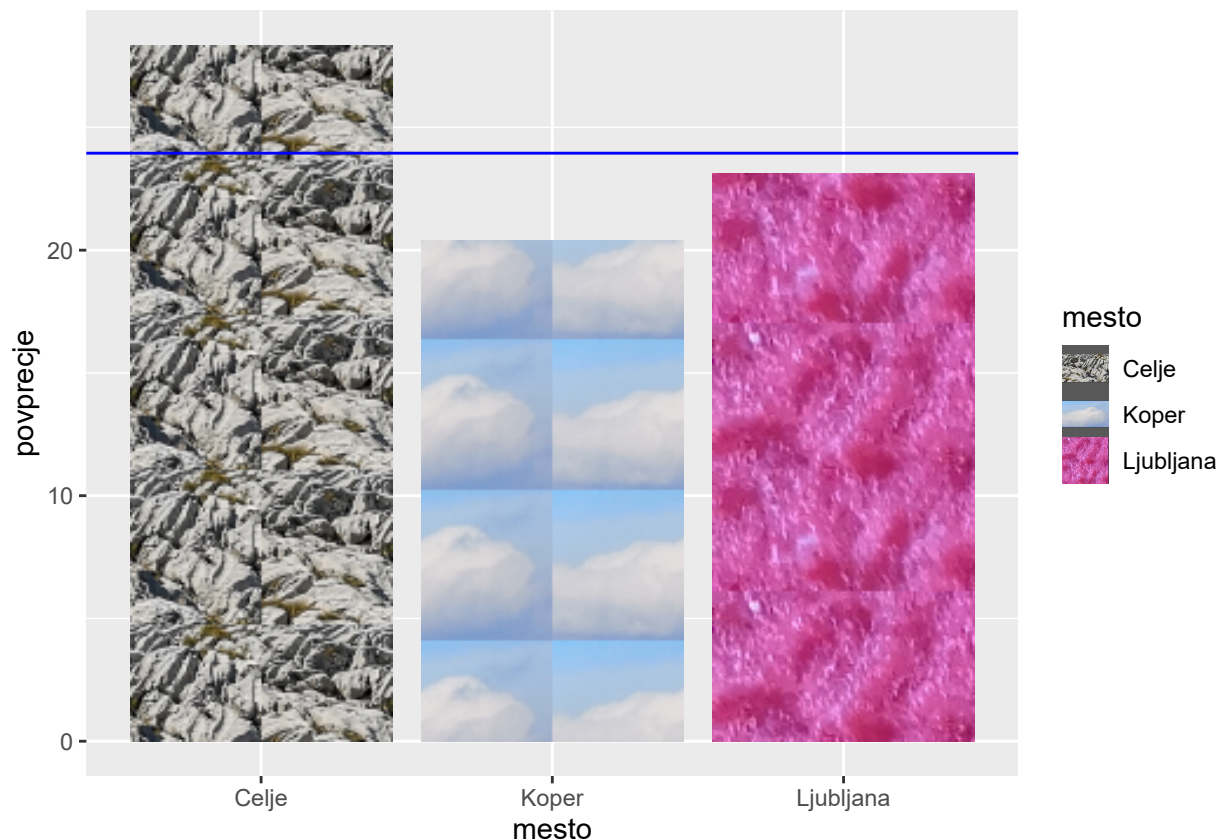
Na sliki vidimo, da je slika centrirana na sredino stolpca. Z dodatnimi parametri jo lahko raztegnemo oziroma razmnožimo cez celotno območje stolpca.

```
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] "                                Z A   L E T O   "
## [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 vrstice
  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], "0prostitutev|Izguba", simplify = T, n = 2)
  tmp2 <- str_match(vrstice_1[, 5], "0prostitutev|Izguba")
  # zdruzimo skupaj
  vrstice_1 <- cbind(vrstice_1[, -5],
                    tmp[, 1], tmp2[, -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 stolpce
  # \\d+ je ujemanje s katerokoli števk
  tmp <- str_split(vrstice_2, "\\d+", simplify = T, 4)[, 2:3]
  # Izvozimo numericne stolpce
  # \\D+ je ujemanje s katerikolnim 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

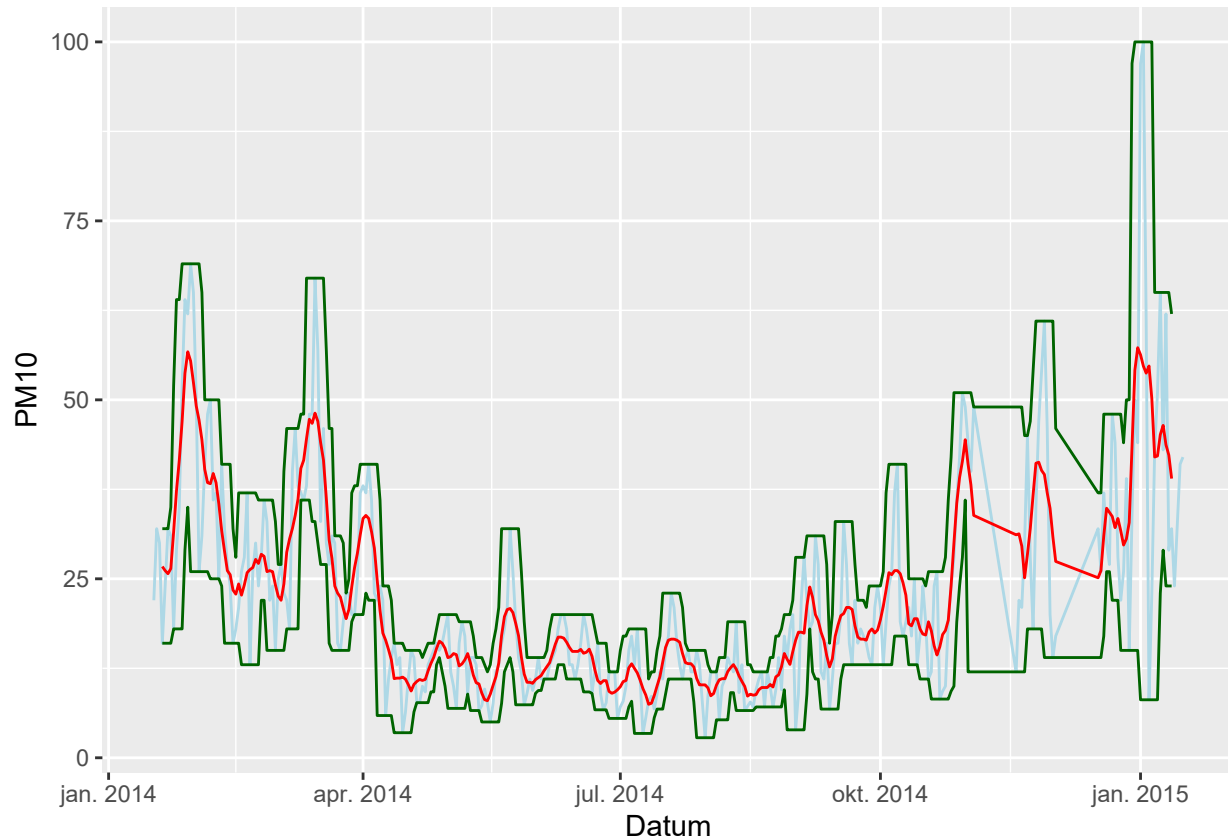
okna <- rollmean(delci$PM10, k = 7, fill = 'NA')
head(okna, n = 10)
```

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

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

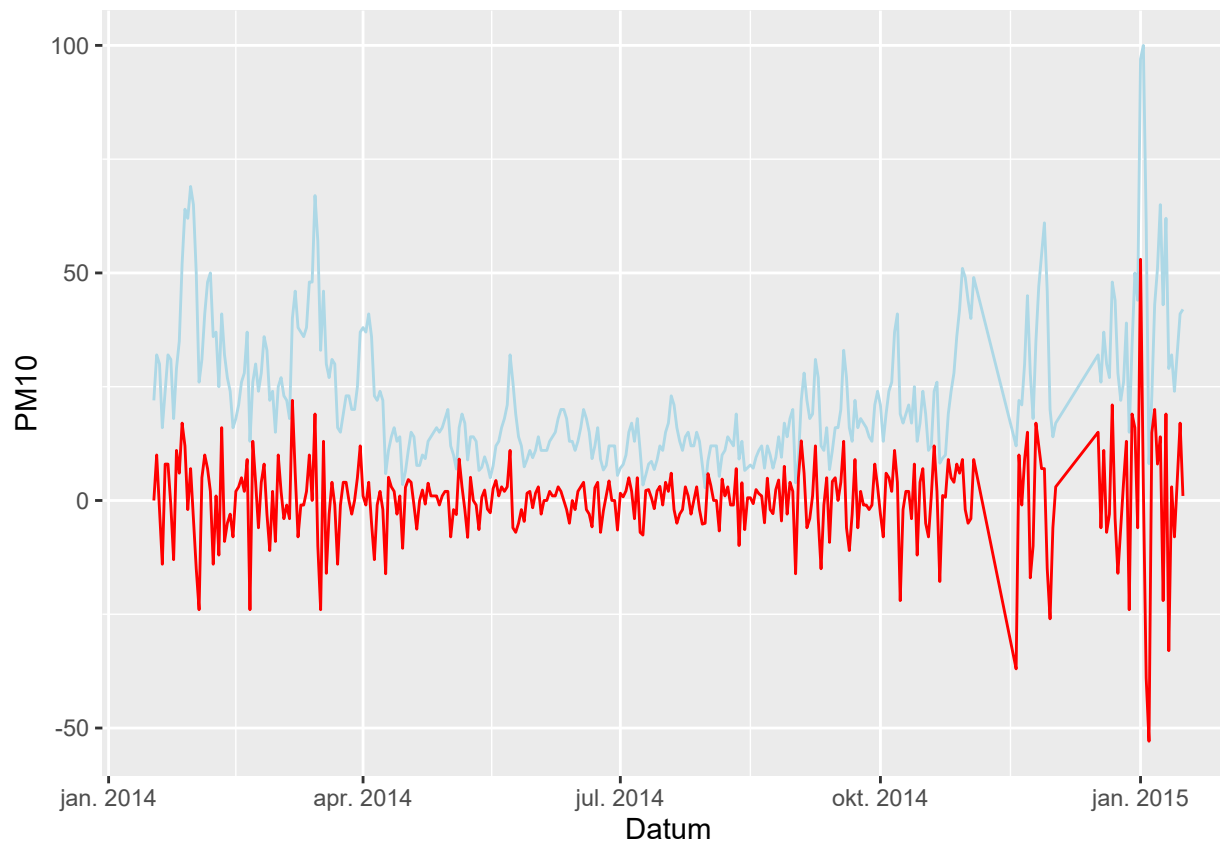
Obstaja še nekaj podobnih funkcij, kot je recimo `rollmax`, ki izračuna maksimalno vrednost okna. V primeru, da želite svoj izračun pa lahko uporabite `rollapply`, ki 'applicira' podano funkcijo na izbrane podatke. Poglejmo si te funkcije na spodnjem grafu.

```
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))))
```



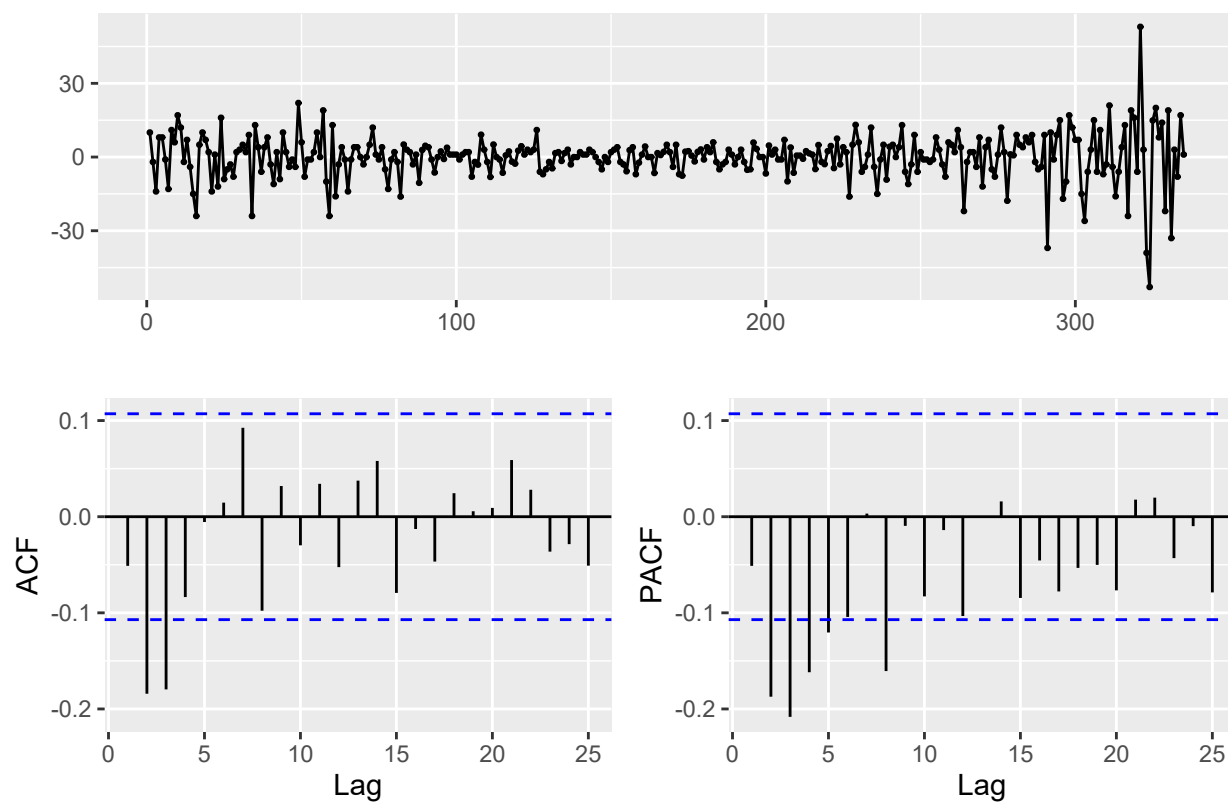
Preskočimo sedaj nekoliko naprej v poglavje strojnega učenja in na časovnih podatkih uporabimo znan model **arima** (autoregressive integrated moving average).

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)
```

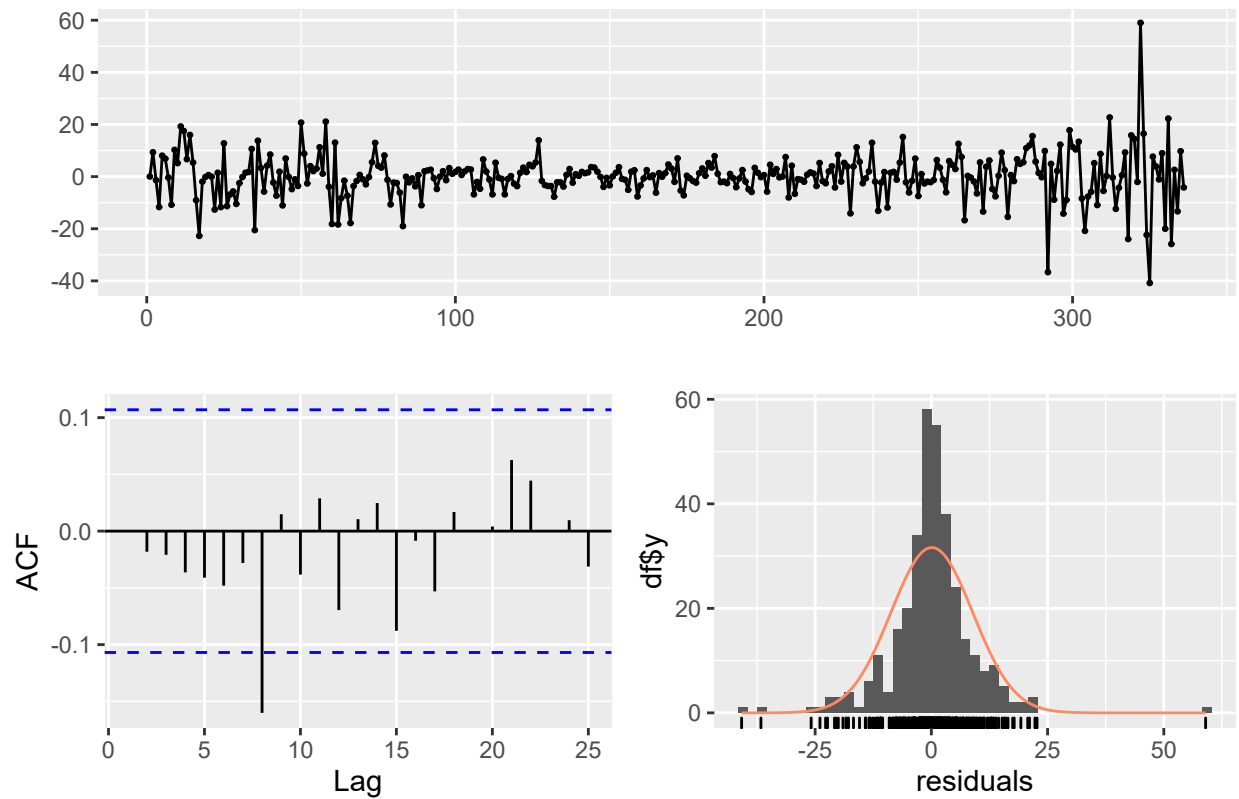
```
## Registered S3 method overwritten by 'quantmod':
##   method      from
##   as.zoo.data.frame zoo
##
## Attaching package: 'forecast'
## The following object is masked from 'package:gghpubr':
##
##   gghistogram
ggttsdisplay(diff(delci$PM10), main = "")
```

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 residuijalov.

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

Residuals from ARIMA(7,1,0)

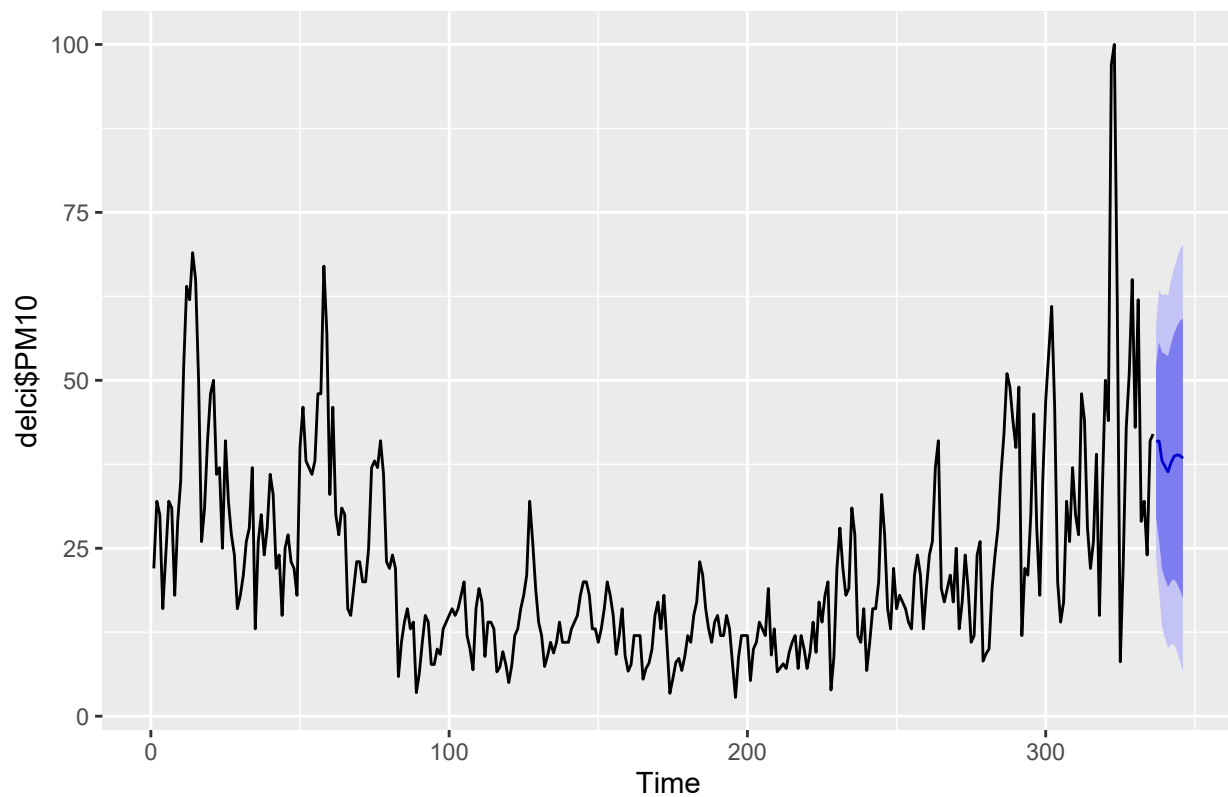


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(7,1,0)
## Q* = 11.822, df = 3, p-value = 0.008017
##
## Model df: 7.   Total lags used: 10
```

Ustvarjen model lahko tudi uporabimo za napovedovanje prihodnjih vrednosti.

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

Forecasts from ARIMA(7,1,0)



Pogljemo si samo še tabularični izpis:

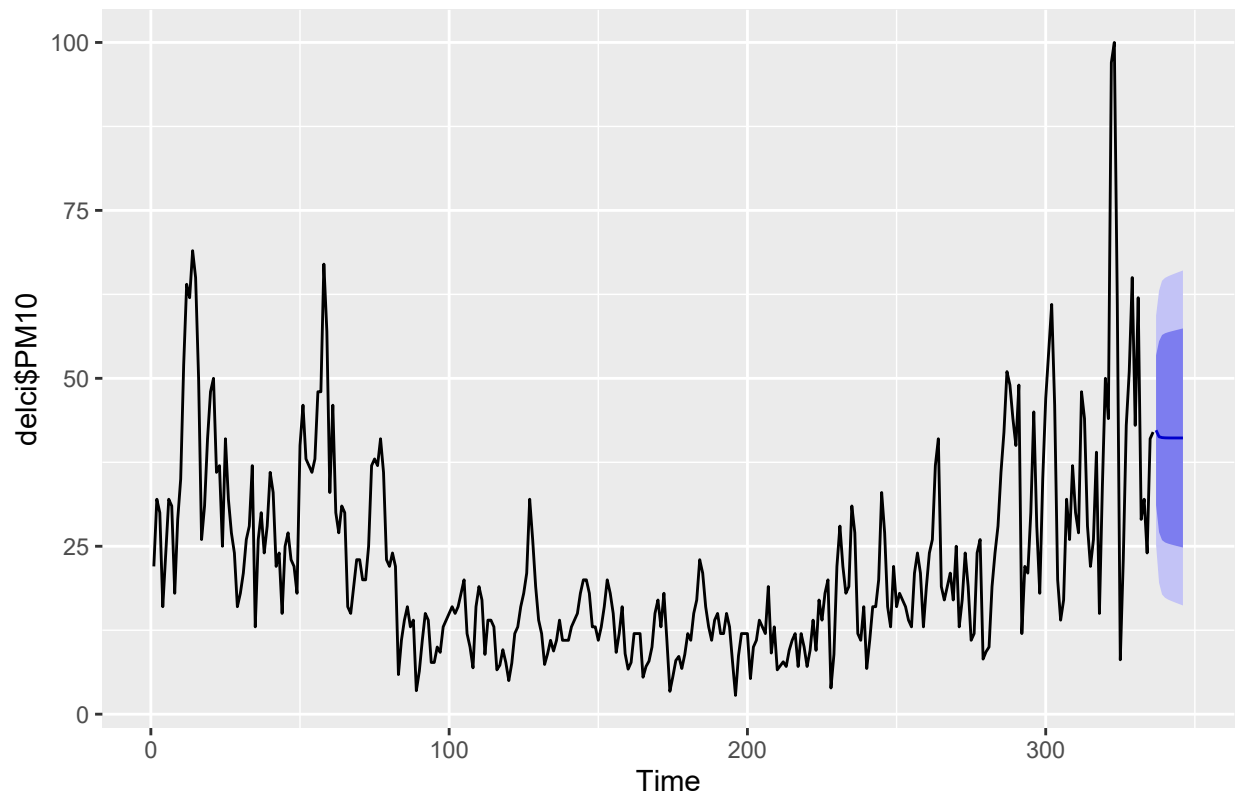
```
forecast(fit)
```

##	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))
```

Forecasts from ARIMA(1,1,3)



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).

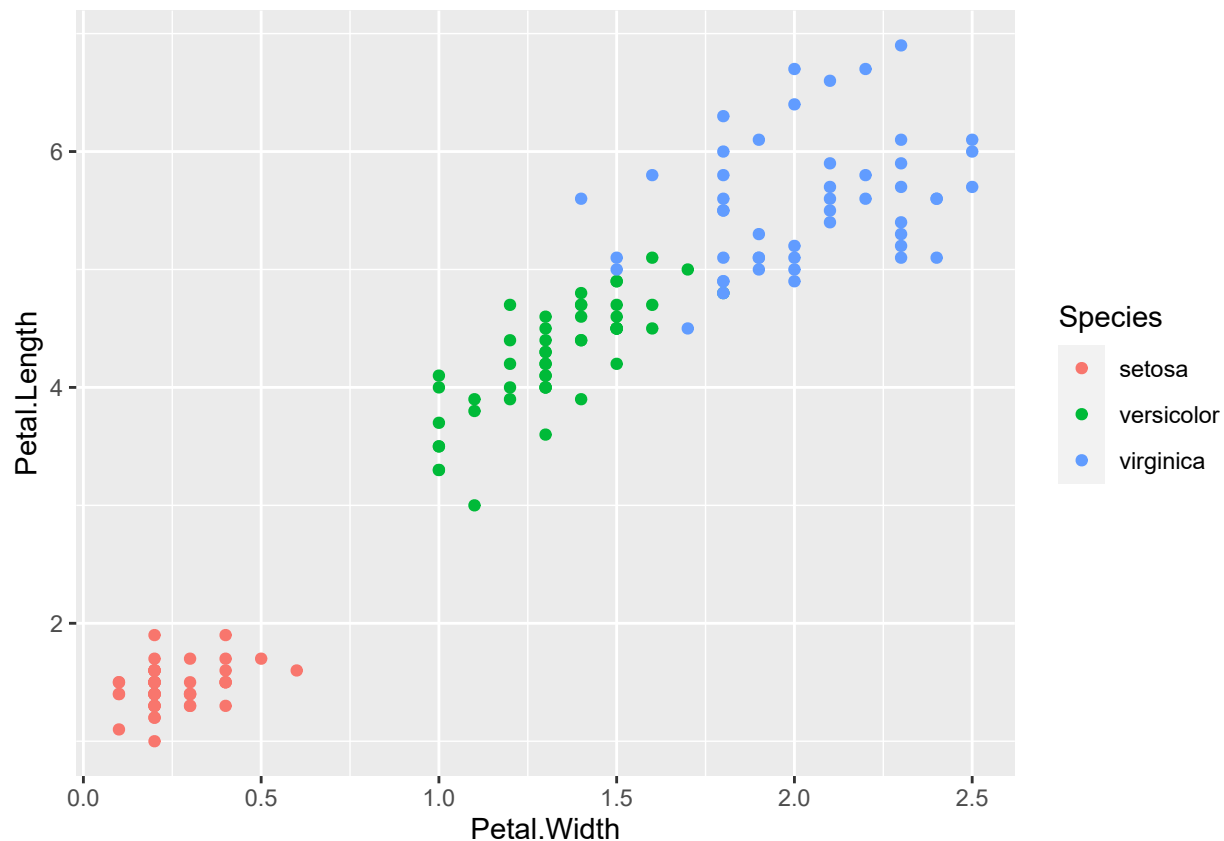
Uporabili bomo podatke `iris`, ki so že priloženi R-ju. Gre za preprost primer, kjer imamo podatke o velikosti čašnih in cvetnih listov preunik, želimo pa napovedati vrsto rože.

```
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
```

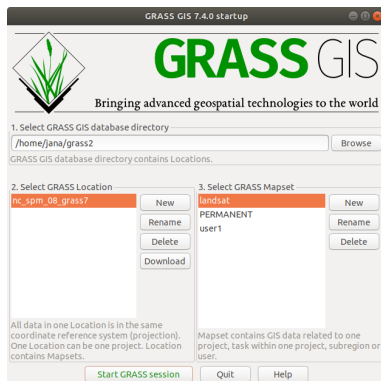
```
##          2  3 1 2  3 1  3  3 2 1  3 1 2 1  3  2 2 1 1 1 1
## napovedi_r 2  3 1 2  3 1  3  3 3 1  3 1 2 1  3  2 2 1 1 1 1
##          138 54 49 102
##          3  2 1  3
## napovedi_r  3  2 1  3
```

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

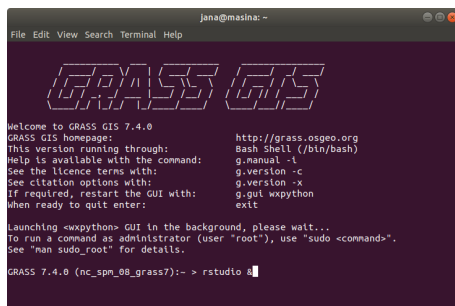
Predavanje 08 – GRASS GIS

GRASS GIS z RGrass7

Zaženemo GrassGis iz konzole:



V konzoli iz GrassGis-a zaženemo RStudio.



Potrebujemo dve R knjižnici, **rgrass7** in **rgdal**.

```
library(rgrass7)
```

```
## Loading required package: XML
```

```
## GRASS GIS interface loaded with GRASS version: GRASS 7.4.0 (2018)
## and location: nc_spm_08_grass7
```

```
library(rgdal)
```

```
## Loading required package: sp
```

```
## Please note that rgdal will be retired by the end of 2023,
## plan transition to sf/stars/terra functions using GDAL and PROJ
## at your earliest convenience.
```

```
##
## rgdal: version: 1.5-29, (SVN revision 1165M)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
## Path to GDAL shared files: /usr/share/gdal/2.2
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
## Path to PROJ shared files: (autodetected)
## Linking to sp version:1.4-6
```

Beremo vzorčni primer iz podatkov pridobljenih na <https://grassbook.org/datasets/datasets-3rd-edition/>.

Lahko si pogledamo metapodatke za svojo lokacijo z:

```
G <- gmeta()
str(G)
```

```
## List of 27
## $ LOCATION_NAME: chr "nc_spm_08_grass7"
## $ GISDBASE      : chr "/home/jana/grass2"
## $ MAPSET        : chr "landsat"
## $ GUI           : chr "wxpython"
## $ PID           : chr "19616"
## $ GUI_PID       : chr "19617"
## $ projection     : chr "99"
## $ zone          : chr "0"
## $ n             : num 228500
## $ s             : num 215000
## $ w             : num 630000
## $ e             : num 645000
## $ t             : num 1
## $ b             : num 0
## $ nsres         : num 10
## $ nsres3        : num 10
## $ ewres         : num 10
## $ ewres3        : num 10
## $ tbres         : num 1
## $ rows          : int 1350
## $ rows3         : int 1350
## $ cols          : int 1500
## $ cols3         : int 1500
## $ depths        : int 1
## $ cells         : chr "2025000"
## $ cells3        : chr "2025000"
## $ proj4          : chr "+proj=lcc +lat_1=36.16666666666666 +lat_2=34.33333333333334 +lat_0=33.75 +lon_0=101.25"
## - attr(*, "class")= chr "gmeta"
```

Prikaže vektorske zemljevide, ki so na voljo:

```
execGRASS("g.list", parameters = list(type = "vector"))
```

```
## P079214
## P079215
```

```

## P079218
## P079219
## boundary_county
## boundary_municp
## bridges
## busroute1
## busroute11
## busroute6
## busroute_a
## busroutesall
## busstopsall
## census_wake2000
## censusblk_swwake
## comm_colleges
## elev_lid792_bepts
## elev_lid792_cont1m
## elev_lid792_randpts
## elev_lidrural_mrpts
## elev_lidrural_mrptsft
## elev_ned10m_cont10m
## firestations
## geodetic_pts
## geodetic_swwake_pts
## geology
## geonames_NC
## geonames_wake
## hospitals
## lakes
## nc_state
## overpasses
## poi_names_wake
## precip_30ynormals
## precip_30ynormals_3d
## railroads
## roadsmajor
## schools_wake
## soils_general
## soils_wake
## streams
## streets_wake
## swwake_10m
## urbanarea
## usgsgages
## zipcodes_wake

```

Prikaži mrežne zemljevide

```
execGRASS("g.list", parameters = list(type = "raster"))
```

```

## aspect
## basin_50K
## boundary_county_500m
## cfactorbare_1m

```

```

## cfactorgrow_1m
## el_D782_6m
## el_D783_6m
## el_D792_6m
## el_D793_6m
## elev_lid792_1m
## elev_ned_30m
## elev_srtm_30m
## elev_state_500m
## elevation
## elevation_shade
## facility
## geology_30m
## lakes
## landclass96
## landcover_1m
## landuse96_28m
## lsat5_1987_10
## lsat5_1987_20
## lsat5_1987_30
## lsat5_1987_40
## lsat5_1987_50
## lsat5_1987_60
## lsat5_1987_70
## lsat7_2000_10
## lsat7_2000_20
## lsat7_2000_30
## lsat7_2000_40
## lsat7_2000_50
## lsat7_2000_61
## lsat7_2000_70
## lsat7_2000_80
## lsat7_2002_10
## lsat7_2002_20
## lsat7_2002_30
## lsat7_2002_40
## lsat7_2002_50
## lsat7_2002_61
## lsat7_2002_62
## lsat7_2002_70
## lsat7_2002_80
## ncmask_500m
## ortho_2001_t792_1m
## roadsmajor
## slope
## soilsID
## soils_Kfactor
## streams_derived
## towns
## urban
## zipcodes
## zipcodes_dbl

```

Preberemo dva GRASS mrežna zemljevidi ("geology_30m" in "elevation" iz vzorčne podatkovne zbirke

North Carolina) v R kot "ncdata".

Lahko pogledamo kaj smo prebrali.

```
str(ncdata)
```

```
## Formal class 'SpatialGridDataFrame' [package "sp"] with 4 slots
##   ..@ data      : 'data.frame':  2025000 obs. of  2 variables:
##   .. ..$ geology_30m: Factor w/ 13 levels "CZfg_217","CZlg_262",...: 5 5 5 4 4 4 4 4 4 4 ...
##   .. ..$ elevation  : num [1:2025000] 142 141 141 142 143 ...
##   ..@ grid       : Formal class 'GridTopology' [package "sp"] with 3 slots
##   .. .. ..@ cellcentre.offset: num [1:2] 630005 215005
##   .. .. ..@ cellsize         : num [1:2] 10 10
##   .. .. ..@ cells.dim        : int [1:2] 1500 1350
##   ..@ bbox       : num [1:2, 1:2] 630000 215000 645000 228500
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "min" "max"
##   ..@ proj4string: Formal class 'CRS' [package "sp"] with 1 slot
##   .. .. ..@ projargs: chr "+proj=lcc +lat_1=36.16666666666666 +lat_2=34.33333333333334 +lat_0=33.75
```

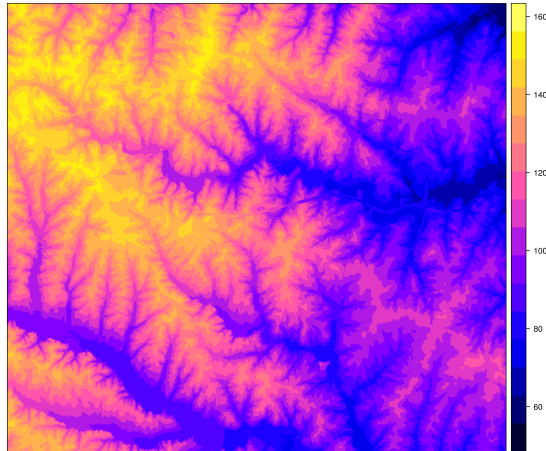
Pogledamo lahko strukturo podatkov:

```
str(ncdata@data)
```

```
## 'data.frame':  2025000 obs. of  2 variables:
## $ geology_30m: Factor w/ 13 levels "CZfg_217","CZlg_262",...: 5 5 5 4 4 4 4 4 4 4 ...
## $ elevation  : num  142 141 141 142 143 ...
```

```
summary(ncdata)
```

```
## Object of class SpatialGridDataFrame
## Coordinates:
##      min      max
## [1,] 630000 645000
## [2,] 215000 228500
## Is projected: TRUE
## proj4string :
## [+proj=lcc +lat_1=36.16666666666666 +lat_2=34.33333333333334
## +lat_0=33.75 +lon_0=-79 +x_0=609601.22 +y_0=0 +no_defs +a=6378137
## +rf=298.257222101 +towgs84=0.000,0.000,0.000 +to_meter=1]
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## 1              630005         10      1500
## 2              215005         10      1350
## Data attributes:
##   geology_30m      elevation
## CZfg_217:725562   Min.    : 55.58
## CZlg_262:198684   1st Qu.: 94.79
## CZbg_405:253710   Median :108.88
## CZlg_262:198684   Mean    :110.38
## CZam_862: 61722   3rd Qu.:126.79
## CZbg_910: 44964   Max.    :156.33
## (Other) : 50985
```



Tako izrišemo mrežne zemljevide

Pregledmo vektorske podatke:

```
library(sf)
```

```
## Linking to GEOS 3.6.2, GDAL 2.2.3, PROJ 4.9.3; sf_use_s2() is TRUE
```

```
use_sf()
execGRASS("v.info", map="hospitals", layer="1")
```

```
## +-----+
## | Name:          hospitals |
## | Mapset:        PERMANENT |
## | Location:      nc_spm_08_grass7 |
## | Database:      /home/jana/grass2 |
## | Title:         North Carolina hospitals (points map) |
## | Map scale:     1:1 |
## | Name of creator: helena |
## | Organization:  NC OneMap |
## | Source date:   Fri Feb 9 23:53:39 2007 |
## | Timestamp (first layer): none |
## |-----+
## | Map format:    native |
## |-----+
## |   Type of map: vector (level: 2) |
## | |
## |   Number of points:      160           Number of centroids: 0 |
## |   Number of lines:       0             Number of boundaries: 0 |
## |   Number of areas:       0             Number of islands: 0 |
## | |
## |   Map is 3D:             No |
## |   Number of dblinks:     1 |
## | |
## |   Projection: Lambert Conformal Conic |
## | |
## |           N: 308097.93740056   S: 20235.56440056 |
## |           E: 914347.8748615   W: 156998.1718615 |
## | |
```

```
## | Digitization threshold: 0 |
## | Comment: |
## | |
## +-----+

```

```
vInfo("hospitals")
```

```
##      nodes      points      lines boundaries centroids      areas      islands
##          0         160          0          0          0          0          0
## primitives      map3d
##          160          0

```

```
myschools <- readVECT("hospitals")
```

```
## Warning: Package rgrass7 transitioning to package rgrass for GRASS 8.
## 'readVECT' is deprecated. Use 'read_VECT' instead.

## Exporting 160 features...
## v.out.ogr complete. 160 features (Point type) written to <hospitals> (GPKG
## format).
## Reading layer 'hospitals' from data source
##   '/home/jana/grass2/nc_spm_08_grass7/landsat/.tmp/masina/640.0.gpkg'
##   using driver 'GPKG'
## Simple feature collection with 160 features and 16 fields
## Geometry type: POINT
## Dimension:      XY
## Bounding box:   xmin: 156998.2 ymin: 20235.56 xmax: 914347.9 ymax: 308097.9
## CRS:            3358

```

```
print(summary(myschools))
```

```
##      cat      OBJECTID      AREA      PERIMETER      HLS_
## Min.   : 1.00   Min.   : 1.00   Min.   :0      Min.   :0      Min.   : 1.00
## 1st Qu.: 40.75  1st Qu.: 40.75  1st Qu.:0      1st Qu.:0      1st Qu.: 40.75
## Median : 80.50  Median : 80.50  Median :0      Median :0      Median : 80.50
## Mean   : 80.50  Mean   : 80.50  Mean   :0      Mean   :0      Mean   : 80.50
## 3rd Qu.:120.25  3rd Qu.:120.25  3rd Qu.:0      3rd Qu.:0      3rd Qu.:120.25
## Max.   :160.00  Max.   :160.00  Max.   :0      Max.   :0      Max.   :160.00
##      HLS_ID      NAME      ADDRESS      CITY
## Min.   : 1.00   Length:160   Length:160   Length:160
## 1st Qu.: 40.75   Class :character   Class :character   Class :character
## Median : 80.50   Mode  :character   Mode  :character   Mode  :character
## Mean   : 80.50
## 3rd Qu.:120.25
## Max.   :160.00
##      ZIP      COUNTY      PHONE      CANCER
## Length:160   Length:160   Length:160   Length:160
## Class :character   Class :character   Class :character   Class :character
## Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##

```

```
##
## POLYGONID SCALE ANGLE geom
## Min. :0 Min. :1 Min. :0.0000 POINT :160
## 1st Qu.:0 1st Qu.:1 1st Qu.:1.0000 epsg:3358 : 0
## Median :0 Median :1 Median :1.0000 +proj=lcc ...: 0
## Mean :0 Mean :1 Mean :0.8313
## 3rd Qu.:0 3rd Qu.:1 3rd Qu.:1.0000
## Max. :0 Max. :1 Max. :1.0000
```

Povzemanje podatkov: Naredimo tabelo kolikokrat se posamezna vrednost pojavi.

```
table(ncdata$geology_30m)
```

```
## < table of extent 0 >
```

Primerjamo z GRASS izpisom:

```
execGRASS("r.stats", flags=c("c", "l"), parameters=list(input="geology_30m"), ignore.stderr=TRUE)
```

```
## 217 CZfg 725562
## 262 CZlg 198684
## 270 CZig 689373
## 405 CZbg 253710
## 583 CZve 21609
## 720 CZam 4824
## 766 CZg 7074
## 862 CZam 61722
## 910 CZbg 44964
## 921 Km 12528
## 945 CZbg 9
## 946 CZam 4068
## 948 CZam 873
```

Lahko narišemo “škatle z brki” različnih geoloških tipov po višinah

```
ncdata <- read_RAST(c("geology_30m", "elevation"), cat=c(TRUE, FALSE))
```

```
## Warning in .read_rast_non_plugin_ng(vname = vname, cat = cat, NODATA = NODATA, :
## non-unique category labels; category number appended
```

```
da <- data.frame(elevation = ncdata$elevation, geology_30m = ncdata$geology_30m)
library(ggplot2)
ggplot(da, aes(x = geology_30m, y = elevation)) + geom_boxplot()
```