Predavanje 08 – Odgovori na vprašanja - Junij 23

#Delo z mapami in datotekami

Za brskanje po mapah v R-ju obstajajo tri funkcije in sicer `dir()`, `list.files()` in `list.dirs()`. Funkcija `list.dirs()` je najpreporstejša in izpisuje le mape, ostali dve pa sta enaki in omogočata bolj usmerjeno iskanje po mapah in datotekah.

Funkcija `list.dirs()` brez parametrov izpiše vse mape in podmape v trenuti delovni mapi.

```
list.dirs()
```

```
##  [1] "."                              "./data_raw"
##  [3] "./data_raw/podatki"             "./data_raw/podatki/analiza_1"
##  [5] "./data_raw/podatki/analiza_2"   "./data_raw/podatki/analiza_2/a"
##  [7] "./data_raw/podatki/analiza_2/b" "./data_raw/podatki/analiza_2/c"
##  [9] "./data_raw/podatki/analiza_3"   "./data_raw/ternarni_diagrami"
## [11] "./img"
```

Izbrano mapo lahko relativno premaknemo z dodajnjem poti v prvem parametru. V mapi *data_raw/podatki/* imamo pripravljeno strukturo map na kateri bomo prikazali primer delovanja teh funkcij.

```
list.dirs("./data_raw/podatki/")
```

```
## [1] "./data_raw/podatki/"             "./data_raw/podatki//analiza_1"
## [3] "./data_raw/podatki//analiza_2"   "./data_raw/podatki//analiza_2/a"
## [5] "./data_raw/podatki//analiza_2/b" "./data_raw/podatki//analiza_2/c"
## [7] "./data_raw/podatki//analiza_3"
```

Funkcija `list.files()` privzeto izpiše le imena map in datotek v trenutnem direktoriju.

```
list.files("./data_raw/podatki")
```

```
## [1] "analiza_1" "analiza_2" "analiza_3"
```

Če želimo prebrati le datoteke brez map nastavimo dodatne parametre, ki jih ponuja funkcija `list.files()` oziroma lahko uporabimo funkcijo `dir()`, ki naredi enako ima pa le krajše ime. Za ostale parametre si oglejte dokumentacijo z `?dir()`.

```
datoteke <- dir("./data_raw/podatki", include.dirs = FALSE, recursive = TRUE)
datoteke
```

```
##  [1] "analiza_1/3-1-2012_S2A_S2B.txt"    "analiza_1/3-2-2012_S2B.txt"
##  [3] "analiza_1/3-3-2012_S2A.txt"        "analiza_1/4-1-2012_S2A_S2B.txt"
##  [5] "analiza_1/5-5-2012_S2A_S2B.txt"    "analiza_2/a/3-1-2012_S2A.txt"
##  [7] "analiza_2/a/3-2-2012_S2A.txt"      "analiza_2/a/3-3-2012_S2A.txt"
##  [9] "analiza_2/a/4-1-2012_S2A.txt"      "analiza_2/a/5-5-2012_S2A.txt"
## [11] "analiza_2/b/13-1-2012_S2B.txt"     "analiza_2/b/15-5-2012_S2B.txt"
## [13] "analiza_2/b/23-3-2012_S2B.txt"     "analiza_2/b/3-2-2012_S2B.txt"
## [15] "analiza_2/b/4-1-2012_S2B.txt"      "analiza_2/c/_S22_13-1-2012_.txt"
## [17] "analiza_2/c/_S2A_S2B_3-1-2012.txt" "analiza_2/c/_S2A_S2B_4-1-2012.txt"
## [19] "analiza_2/c/_S2B_3-2-2012_S2A.txt" "analiza_2/c/3-3-2012_S2A_S2B.txt"
## [21] "analiza_2/c/5-5-2012_S2A_S2B.txt"  "analiza_3/14-1-2012_S2A_S2B.csv"
## [23] "analiza_3/3-1-2012_S2A_S2B.csv"    "analiza_3/3-2-2012_S2B.csv"
```

```
## [25] "analiza_3/3-3-2012_S2A.csv"        "analiza_3/5-5-2012_S2A_S2B.csv"
```

Kot vidimo so v vektorju `datoteke` vse relative poti vključno z mapami. Z uporabo **for** zanke se lahko preprosto sprehodimo čez vse datoteke in jih vsako posebej obdelamo. Okvirno tako:

```
for(d in datoteke){
  #obdeljaj datoteko
}
```

Če imamo informacije skrite v samih imenih datotek oziroma mapah potrebujemo ukaze za delo z nizi, da iz njih izvlečemeo koristne informacije. Zelo na hitro si oglejmo par uporabnih ukazov. Zelo priročen paket za delo z nizi je **stringr**, ki je del paketov **tidyverse**. Naložimo ta paket, da bomo imeli vse funkcije na voljo.

Ena bolj uporabnih funkcij za delo z nizi je strsplit, ki nam omogoča, da niz razbijemo na več manjših podnizov, glede na specifične znake.

```
pokosih <- strsplit("mapa1/mapa2/datoteka.txt", split = "/")
pokosih
```

```
## [[1]]
## [1] "mapa1"        "mapa2"        "datoteka.txt"
```

Funkcija nam vrne vrednosti v seznamu, zato je velikokrat priporočeno, da uporabimo še funkcijo `unlist()`, ki iz seznama ustvari vektor, če je le-to mogoče. Če v takem vektorju indeksiramo zadnji element in delimo pot do datoteke po znaku `/` dobimo ravno ime datoteke.

```
pokosih <- unlist(pokosih)
pokosih[length(pokosih)]
```

```
## [1] "datoteka.txt"
```

Če imamo v imenih datotek na določenem mestu (recimo prvih nekaj znakov) pomemben podatek, ga lahko iz niza izluščimo s funkcijo `substr()`. Primer:

```
podniz <- substr("moj neznan niz znakov", 5, 9)
podniz
```

```
## [1] "nezna"
```

Nekoliko zahtevnejša funkcija je `grep()`, ki nam pove ali niz vsebuje nek znak ali vzorec. Vrne nam indeks niza v katerem se nahaja iskan niz. V našem primeru je to znak 'a'. Funkcija `grep()` je podobna funkcija, le da vrača logični vektor.

```
grep("a", c("ah", "oh", "eh", "aha", "pa res"))
```

```
## [1] 1 4 5
```

```
grepl("a", c("ah", "oh", "eh", "aha", "pa res"))
```

```
## [1]  TRUE FALSE FALSE  TRUE  TRUE
```

Če želimo vedeti na katerih mestih znotraj niza se nahaja nek znak lahko uporabimo funkcijo `gregexpr()`.

```
gregexpr("a", "banana")
```

```
## [[1]]
## [1] 2 4 6
## attr(,"match.length")
## [1] 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
```

```
## [1] TRUE
```

Funkciji `sub()` in `gsub()` (substitution) pa se uporabljajo za menjavo iskanih znakov in vzorcev. Najprej podamo iskani vzorec, nato niz s katerim iskan vzorec zamenjamo in nato naš željen vhodni niz ali nize.

```
gsub("a", "", c("ah", "oh", "eh", "aha", "pa res"))
```

```
## [1] "h"     "oh"    "eh"    "h"     "p res"
```

```
gsub("[aeiou]", "", c("ah", "oh", "eh", "aha", "pa res"))
```

```
## [1] "h"    "h"    "h"    "h"    "p rs"
```

```
gsub("[^aeiou]", "", c("ah", "oh", "eh", "aha", "pa res"))
```

```
## [1] "a"  "o"  "e"  "aa" "ae"
```

Prvi parameter za opis vzorca vpišemo v obliko regularnega izraza. Razlaga te teme sicer presega obseg tega predavanja, če pa želite izvedeti več oziroma preizkušati regularne izraze je na voljo stran regex101.

Z regularnimi izrazi lahko zamenjamo oziroma poiščemo številke.

```
gsub("[0-9]+", "$", "crk36in5tev1lk3")
```

```
## [1] "crk$in$tev$lk$"
```

```
gsub("[^0-9]+", "", "crk3in5tev1lk3")
```

```
## [1] "3513"
```

Opis datumov, kot jih imamo v struturi datotek je že nekoliko kompleksnejše in v tej fazi ni mišljeno, da bi že razumeli. Spodnji primer išče vzorec "številke-številke-številke", kjerkoli v nizu in jih zamejna z skupino med oklepaji, kar pomeni, da ga izlušči.

```
datum <- gsub(".*([0-9]+-[0-9]+-[0-9]+).*","\\1","4-1-2012_S2B.txt")
datum
```

```
## [1] "4-1-2012"
```

Dobljeni datum je tipa `character()`, kar pomeni, da se primerjave z drugimi datumi delajo leksikografsko, kar pa ni pravilno. Zato je dobro, da nize pretovorim v tip `Date` s funkcijo `as.Date()`.

```
as.Date(datum, format = "%d-%m-%Y")
```

```
## [1] "2012-01-04"
```

```
class(as.Date(datum, format = "%d-%m-%Y"))
```

```
## [1] "Date"
```

Sedaj lahko delamo primerjave z datumi.

```
as.Date(datum, format = "%d-%m-%Y") > as.Date("2011-1-1")
```

```
## [1] TRUE
```

```
as.Date(datum, format = "%d-%m-%Y") > as.Date("2012-1-1")
```

```
## [1] TRUE
```

Naredimo sedaj primer, kjer bomo prebrali le datoteke, ki v svojem imenu vsebujejo datum nekje v Januarju leta 2012 in v imenu datoteke ne vsebujejo vzorca *S2B* in niso tipa *.csv*. Iz teh datotek bomo zgradili **data.frame**, ki bo vseboval imena datotek, izluščene datume in logično vrednost, ki nam pove ali ime datoteke vsebuje vzorec *S2A*.

```
izbrane <- data.frame(file=character(), date=Date(), S2A=logical())
for(d in datoteke){
  print("--------------------------")
  print(d)
  #odstranimo mape
  d <- unlist(strsplit(d, split = "/"))
  d <- d[length(d)]
  print(d)
  #odstranimo .csv in datoteke z S2B
  if(grepl("\\.csv", d) | grepl("S2B", d)){
    print("Removing .csv or S2B")
    next
  }
  #preverimo ustrezne datume
  datum <- gsub(".*([0-9]+-[0-9]+-[0-9]+).*","\\1",d)
  datum <- as.Date(datum, format = "%d-%m-%Y")
  if(datum < as.Date("2012-1-31") & datum > as.Date("2012-1-1")){
    #če datum ustrza, shranimo v tabelo
    S2A <- grepl("S2A", d)
    izbrane <- rbind(izbrane, data.frame(d, datum, S2A))
  }
}
```

```
## [1] "--------------------------"
## [1] "analiza_1/3-1-2012_S2A_S2B.txt"
## [1] "3-1-2012_S2A_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_1/3-2-2012_S2B.txt"
## [1] "3-2-2012_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_1/3-3-2012_S2A.txt"
## [1] "3-3-2012_S2A.txt"
## [1] "--------------------------"
## [1] "analiza_1/4-1-2012_S2A_S2B.txt"
## [1] "4-1-2012_S2A_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_1/5-5-2012_S2A_S2B.txt"
## [1] "5-5-2012_S2A_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/a/3-1-2012_S2A.txt"
## [1] "3-1-2012_S2A.txt"
## [1] "--------------------------"
## [1] "analiza_2/a/3-2-2012_S2A.txt"
## [1] "3-2-2012_S2A.txt"
## [1] "--------------------------"
## [1] "analiza_2/a/3-3-2012_S2A.txt"
## [1] "3-3-2012_S2A.txt"
## [1] "--------------------------"
## [1] "analiza_2/a/4-1-2012_S2A.txt"
## [1] "4-1-2012_S2A.txt"
```

```
## [1] "--------------------------"
## [1] "analiza_2/a/5-5-2012_S2A.txt"
## [1] "5-5-2012_S2A.txt"
## [1] "--------------------------"
## [1] "analiza_2/b/13-1-2012_S2B.txt"
## [1] "13-1-2012_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/b/15-5-2012_S2B.txt"
## [1] "15-5-2012_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/b/23-3-2012_S2B.txt"
## [1] "23-3-2012_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/b/3-2-2012_S2B.txt"
## [1] "3-2-2012_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/b/4-1-2012_S2B.txt"
## [1] "4-1-2012_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/c/_S22_13-1-2012_.txt"
## [1] "_S22_13-1-2012_.txt"
## [1] "--------------------------"
## [1] "analiza_2/c/_S2A_S2B_3-1-2012.txt"
## [1] "_S2A_S2B_3-1-2012.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/c/_S2A_S2B_4-1-2012.txt"
## [1] "_S2A_S2B_4-1-2012.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/c/_S2B_3-2-2012_S2A.txt"
## [1] "_S2B_3-2-2012_S2A.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/c/3-3-2012_S2A_S2B.txt"
## [1] "3-3-2012_S2A_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_2/c/5-5-2012_S2A_S2B.txt"
## [1] "5-5-2012_S2A_S2B.txt"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_3/14-1-2012_S2A_S2B.csv"
## [1] "14-1-2012_S2A_S2B.csv"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_3/3-1-2012_S2A_S2B.csv"
## [1] "3-1-2012_S2A_S2B.csv"
## [1] "Removing .csv or S2B"
```

```
## [1] "--------------------------"
## [1] "analiza_3/3-2-2012_S2B.csv"
## [1] "3-2-2012_S2B.csv"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_3/3-3-2012_S2A.csv"
## [1] "3-3-2012_S2A.csv"
## [1] "Removing .csv or S2B"
## [1] "--------------------------"
## [1] "analiza_3/5-5-2012_S2A_S2B.csv"
## [1] "5-5-2012_S2A_S2B.csv"
## [1] "Removing .csv or S2B"
```

```
izbrane
```

```
##                     d      datum   S2A
## 1    3-1-2012_S2A.txt 2012-01-03  TRUE
## 2    4-1-2012_S2A.txt 2012-01-04  TRUE
## 3 _S22_13-1-2012_.txt 2012-01-03 FALSE
```

#Krajša analiza pajkov in izris intevalov zaupanja, korelacije in podobno

V analizmo smo dobili manjši vzorec o pajkih, ki so v datoteki *Nephilia_data.xlsx*. Naložimo te podatke:

```
library(openxlsx)
data <- read.xlsx(
  xlsxFile = "./data_raw/Nephila_data.xlsx",
  sheet = 1)

head(data)
```

```
##        Specimen Leg.length Adult.female      Locality Canopy Males Kleptos
## 1 MK20230201-25       9362           no Telok Blangah   open     0       0
## 2 MK20230201-32       9499           no Telok Blangah   open     0       2
## 3 MK20230201-42       9915           no Telok Blangah closed     0       2
## 4 MK20230201-28      10020           no Telok Blangah closed     0       0
## 5 MK20230201-07      10252           no Telok Blangah   open     0       2
## 6 MK20230201-34      10979           no Telok Blangah   open     0       2
##   Nearest.web Above.ground Web.diameter
## 1           2          160           24
## 2          10          168           39
## 3          10          154           39
## 4          10          133           41
## 5           8           95           30
## 6          10          123           47
```

Samo za lažji pregled naredimo še `summary()`, da vidimo tipe stolpcev in razpone vrednosti.

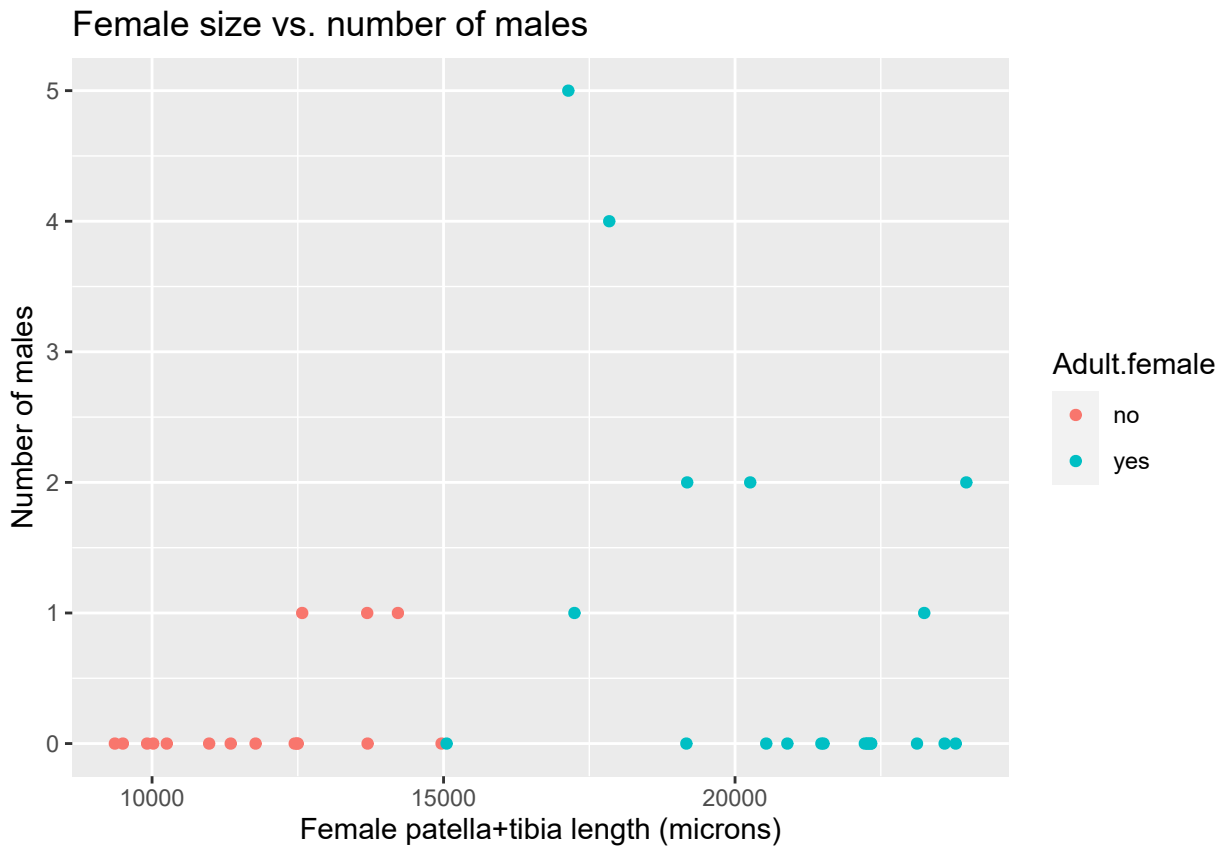
```
summary(data)
```

```
##    Specimen           Leg.length    Adult.female         Locality
##  Length:37          Min.   : 9362   Length:37          Length:37
##  Class :character   1st Qu.:12486   Class :character   Class :character
##  Mode  :character   Median :17245   Mode  :character   Mode  :character
##                     Mean   :17005
##                     3rd Qu.:22226
##                     Max.   :23967
##     Canopy              Males           Kleptos       Nearest.web
```

```
##  Length:37          Min.   :0.0000   Min.   :0.000   Min.   : 1.000
##  Class :character   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.: 5.000
##  Mode  :character   Median :0.0000   Median :0.000   Median : 8.000
##                     Mean   :0.5405   Mean   :1.189   Mean   : 6.973
##                     3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:10.000
##                     Max.   :5.0000   Max.   :6.000   Max.   :10.000
##   Above.ground    Web.diameter
##  Min.   : 90.0   Min.   : 22.00
##  1st Qu.:137.0   1st Qu.: 39.00
##  Median :176.0   Median : 47.00
##  Mean   :179.1   Mean   : 52.68
##  3rd Qu.:210.0   3rd Qu.: 66.00
##  Max.   :320.0   Max.   :111.00
```

Na podatkih je bil že narejen razsevni diagram, ki prikazuje število moških v odvisnosti od velikosti ženske.

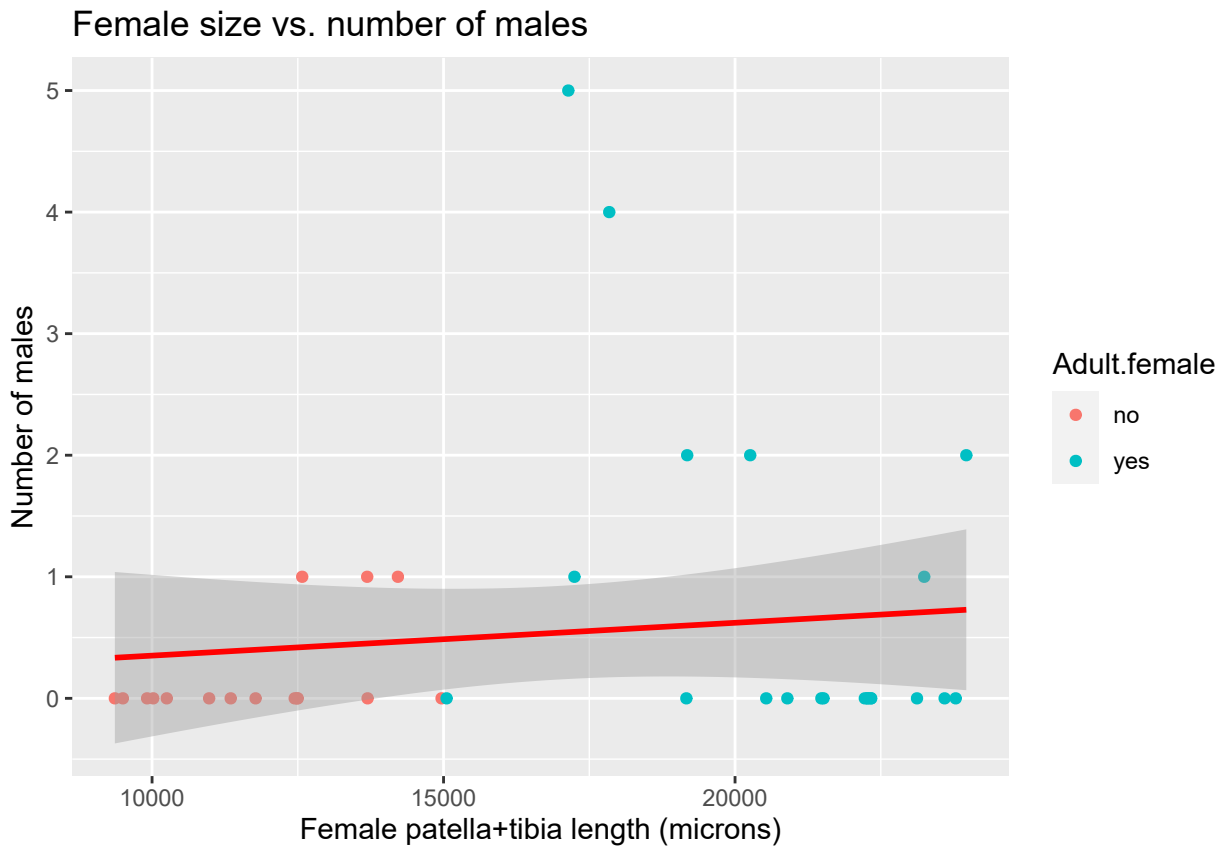
```
library(ggplot2)

#Female size vs. number of males

g1 <- ggplot(data, aes(x = Leg.length, y = Males, colour = Adult.female)) + geom_point() + xlab("Female
g1
```



Če želimo zaznati vzorce v točkovnih podatkih lahko naredimo linearni model in ga izrišemo na graf. Paket **ggplot2** nam s funkcijo `stat_smoot` to omogoča avtomatsko. Doda celo intervale zaupanja.

```
g1 + stat_smooth(method = "lm", col = "red")
```

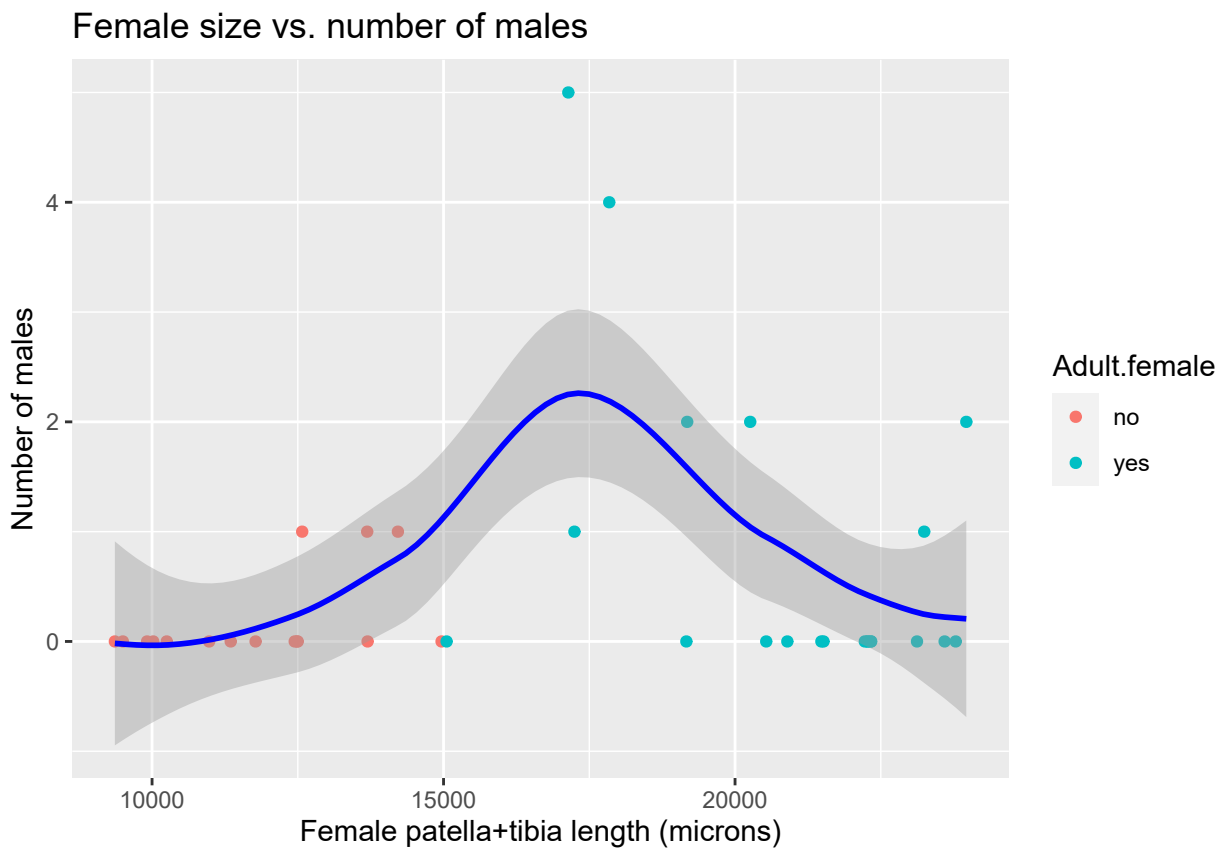
```
## `geom_smooth()` using formula = 'y ~ x'
```



Uporabimo lahko tudi druge metode, kot so lokalni modeli.

```
g1 + stat_smooth(method = 'loess', col = "blue")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

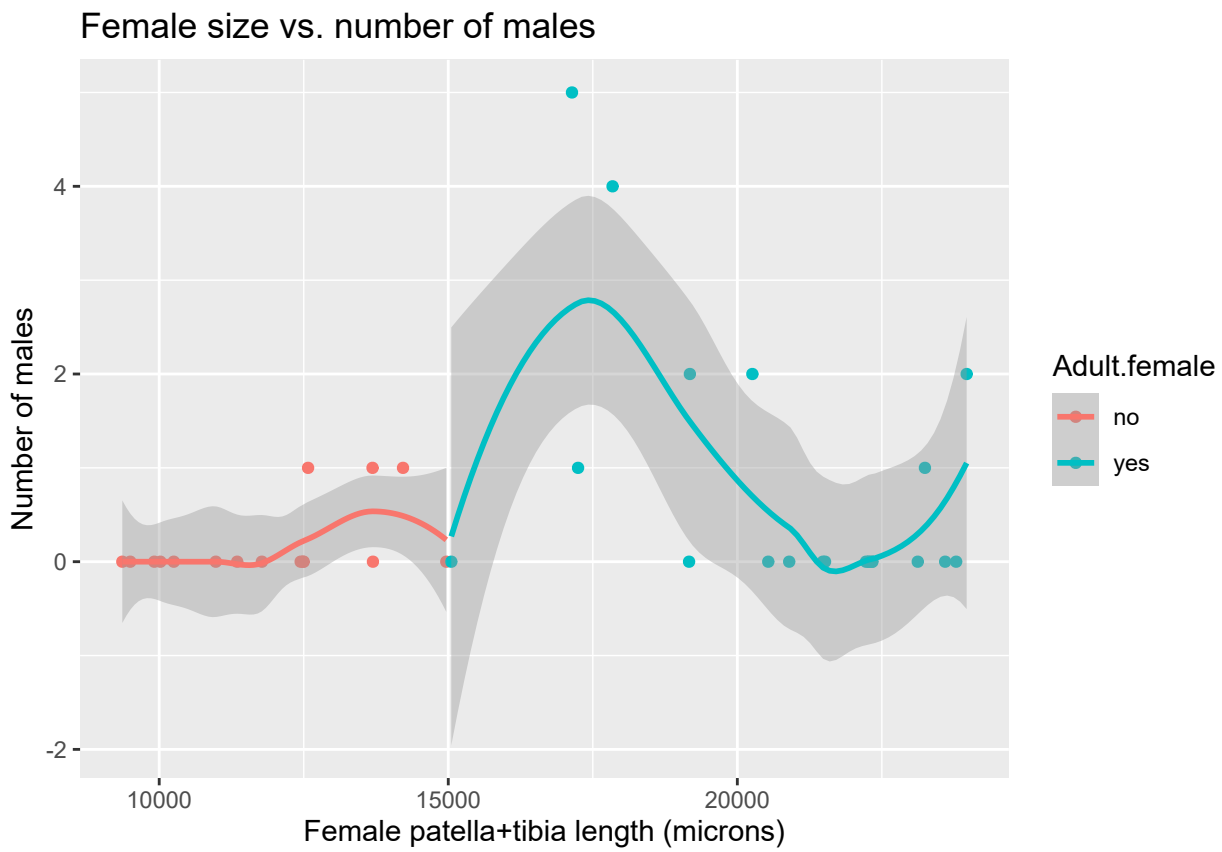


Pri tem primeru vidimo, da je največ samcev na mrežah srednje velikih samic.

Izrise pa lahko naredimo tudi za vsako skupino posebej.

```
g1 + stat_smooth(method = "loess")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Metodi `lm` in `loess` sta del paketa **stats**, ki je del osnovnega R-ja. Če želimo tem modelom podati kakšne druge parametre lahko to naredimo preko argumenta `method.args`, kot je prikazano spodaj.

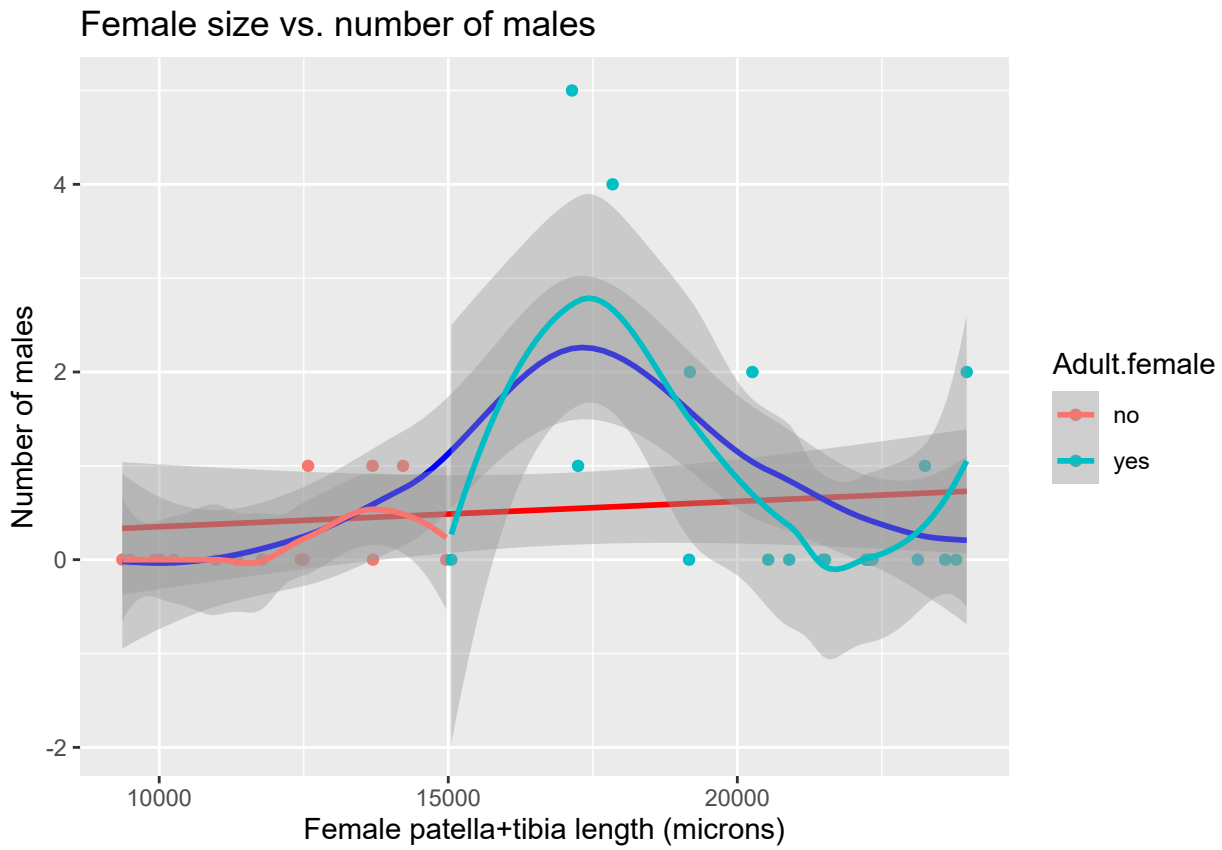
```
g1 + stat_smooth(method = "loess", method.args = list(degree = 1))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

Paket **ggplot2** nam omogoča, da na enem grafu prikažemo poljubno mnogo informacij. Seveda moramo v tem primeru paziti, da graf ne postane nepregleden, kot je v naslednjem primeru.

```
g1 + stat_smooth(method = "lm", col = "red") +
  stat_smooth(method = 'loess', col = "blue") +
  stat_smooth(method = "loess")
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



Uporabno povezavo, ki prikazuje tudi uporabo mešanih modelov lahko dobite tukaj.

Seveda pa lahko koristne informacije dobimo tudi računsko. Recimo izračunajmo korelacijmo med dolžino pajkov in število samcev.

```
#korelacija
cor(data$Leg.length, data$Males)
```

```
## [1] 0.1190133
```

Oziroma lahko korelacijo izračunamo med vsemi številskimi atributi naenkrat.

```
corr <- cor(data[,c("Leg.length", "Males", "Kleptos", "Nearest.web",
                    "Above.ground", "Web.diameter")])
corr
```

```
##                Leg.length       Males     Kleptos Nearest.web Above.ground
## Leg.length     1.00000000  0.11901335  0.11265690 -0.02020371   0.47645068
## Males          0.11901335  1.00000000  0.21614028 -0.36058813   0.24861497
## Kleptos        0.11265690  0.21614028  1.00000000  0.15990687  -0.08513639
## Nearest.web   -0.02020371 -0.36058813  0.15990687  1.00000000  -0.32573424
## Above.ground   0.47645068  0.24861497 -0.08513639 -0.32573424   1.00000000
## Web.diameter   0.62082269 -0.07789458  0.08468174  0.08928130   0.19407864
##               Web.diameter
## Leg.length      0.62082269
## Males          -0.07789458
## Kleptos         0.08468174
## Nearest.web     0.08928130
## Above.ground    0.19407864
```

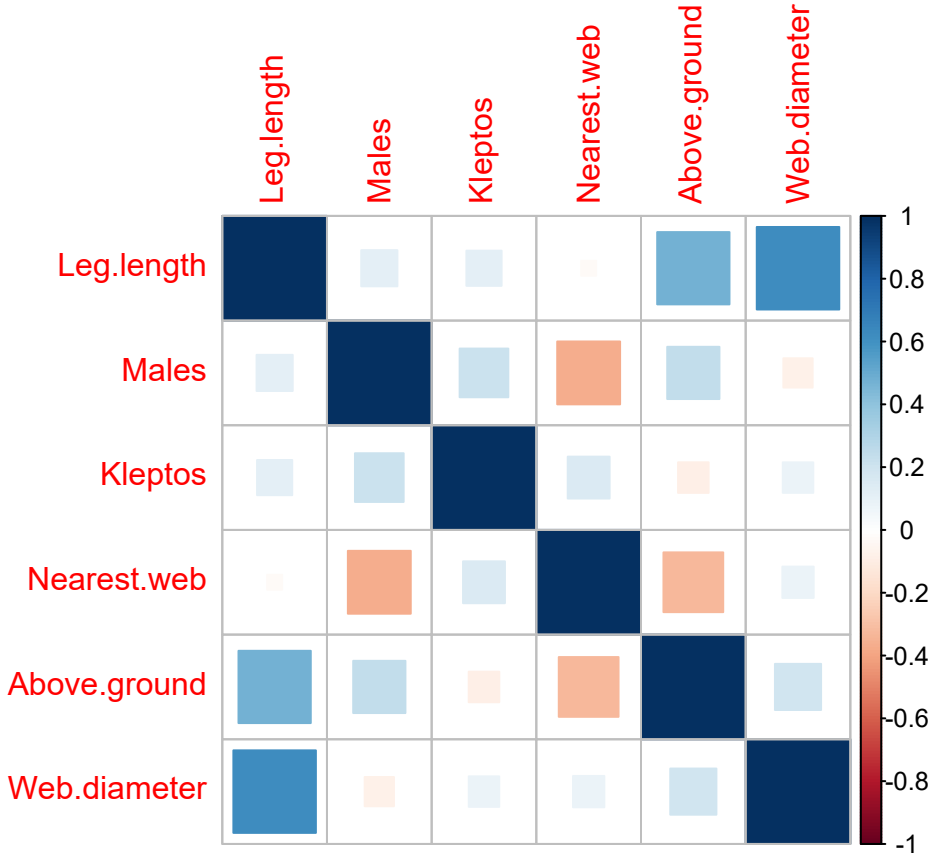
```
## Web.diameter   1.00000000
```

Te korelacije lahko z uporabo paketa **corrplot** tudi grafično prikažemo.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
corrplot(corr, method="square")
```



Modele, ki smo jih prikazali na na grafih lahko tudi poženemo in njihov rezultat interpetiramo glede na dobljen izpis. Zgradimo linearni model.

```
model <- lm(Males ~ Kleptos + Nearest.web + Leg.length, data)
model
```

```
##
## Call:
## lm(formula = Males ~ Kleptos + Nearest.web + Leg.length, data = data)
##
## Coefficients:
## (Intercept)      Kleptos  Nearest.web   Leg.length
##   9.452e-01    1.947e-01   -1.357e-01    1.822e-05
```

Zgrajen model nam prikazuje koeficient za vsak atribut. Pozorni bodite na znak~, ki se lahko bere kot *v odvisnosti od* oziroma kot enačaj.

Upoben je tudi ukaz `summary()` nad modelom, ki nam poda tudi ocene p-vrednosti.

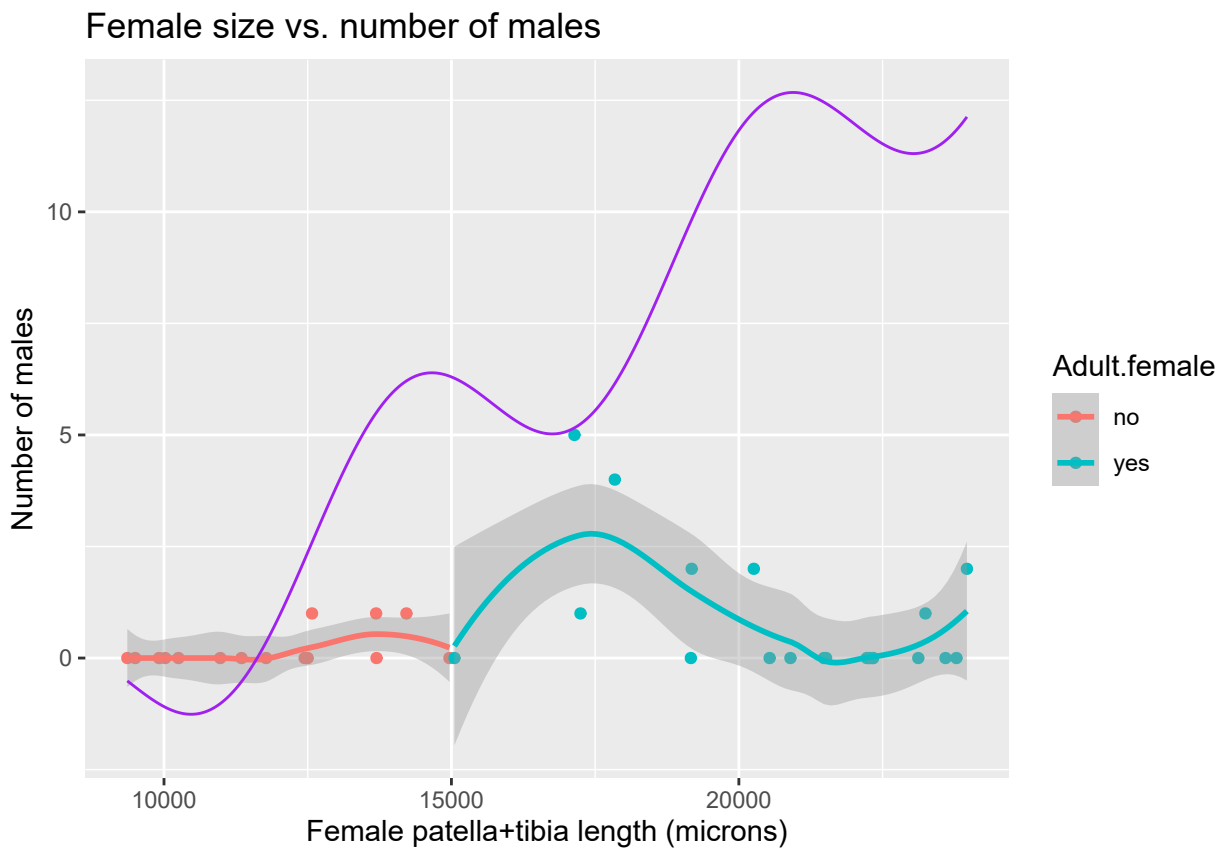
```
summary(model)
```

```
##
## Call:
## lm(formula = Males ~ Kleptos + Nearest.web + Leg.length, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4375 -0.5218 -0.1779  0.2291  3.4296
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.452e-01  7.254e-01   1.303   0.2016
## Kleptos      1.947e-01  1.130e-01   1.723   0.0942 .
## Nearest.web -1.357e-01  5.278e-02  -2.571   0.0149 *
## Leg.length   1.822e-05  3.529e-05   0.516   0.6090
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.061 on 33 degrees of freedom
## Multiple R-squared:  0.2133, Adjusted R-squared:  0.1418
## F-statistic: 2.983 on 3 and 33 DF,  p-value: 0.04532
```

Za konec analize si poglejmo, še kako dodamo poljubno funkcijo na graf. To lahko storimo s funkcijo `stat_function()`. V spodnjem primeru `fun = function(x){2*sin(x/1000)+x/1000-10}` implicitno definira funkcijo, ki jo želimo izrisati, lahko pa jo definiramo posbej, kot smo to delali na sedmem predavanju.

```
#poljubne funkcije
g1 + stat_smooth(method = "loess") +
  stat_function(fun = function(x){2*sin(x/1000)+x/1000-10},
                n = 1000, col = "purple")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



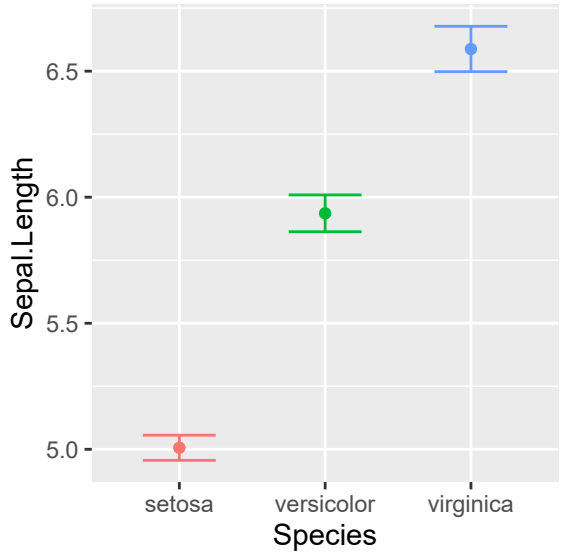
ggplot2 – errorbar

Drugi zelo nazorem prikaz podatkov je z **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.

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

head(df)
```

```
##      Species Sepal.Length         SE
## 1     setosa        5.006 0.04984957
## 2 versicolor        5.936 0.07299762
## 3  virginica        6.588 0.08992695
```

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



ggplot- heatmap

Heatmap lahko naredimo s pomočjo paketa **ggplot2**. To je način, da z barvami prikažemo matriko vrednosti, oziroma tabelo. V spodnem primeru imamo vrednosti delcev PM10 po mesecih za več merilnih mest. Najprej si poglejmo pdatke.

```
library(tidyr)
pod <- read.table('./data_raw/PM10_heat.csv', sep=',', header = T)
pod
```

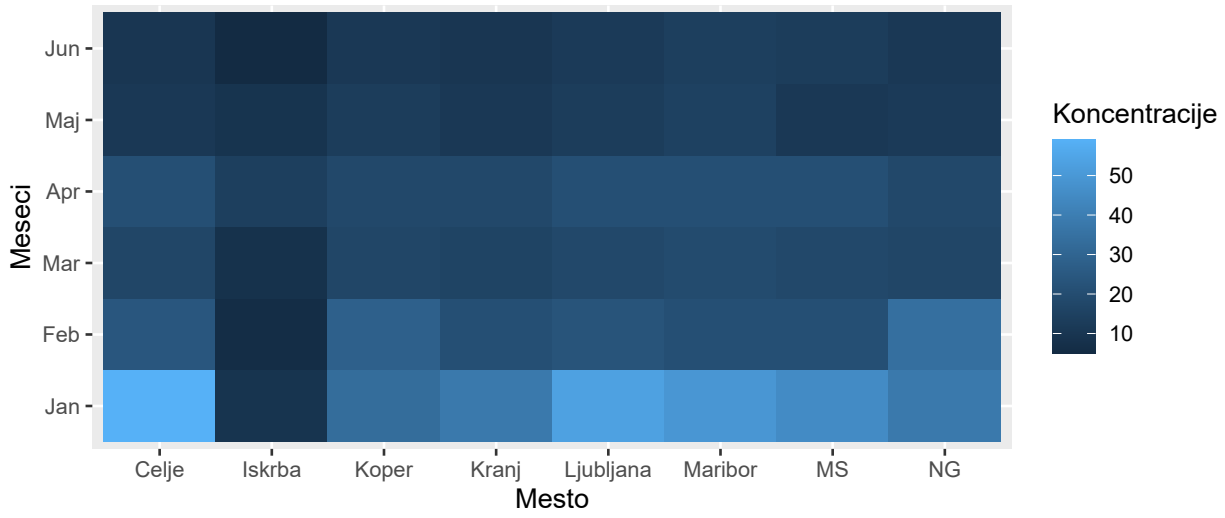
```
##       Mesto Jan Feb Mar Apr Maj Jun
## 1     Celje  59  24  17  21  11  10
## 2    Iskrba   9   6   8  14   9   5
## 3     Koper  33  28  17  18  13  11
## 4     Kranj  38  21  16  18  11  10
## 5 Ljubljana  53  23  18  21  13  12
## 6   Maribor  49  21  19  21  15  14
## 7        MS  45  21  18  21  11  13
## 8        NG  38  34  17  18  12  11
```

Podatke pretvorimo v dolgo obliko. Tukaj smo bili pozorni na atribut `levels`, ki bo določal vrsti red izpisa podatkov.

```
pod_l <- pivot_longer(pod, cols = 2:7, names_to = "Meseci", values_to = "Koncentracije")
pod_l$Meseci <- factor(pod_l$Meseci, levels = c("Jan", "Feb", "Mar", "Apr", "Maj", "Jun"))
```

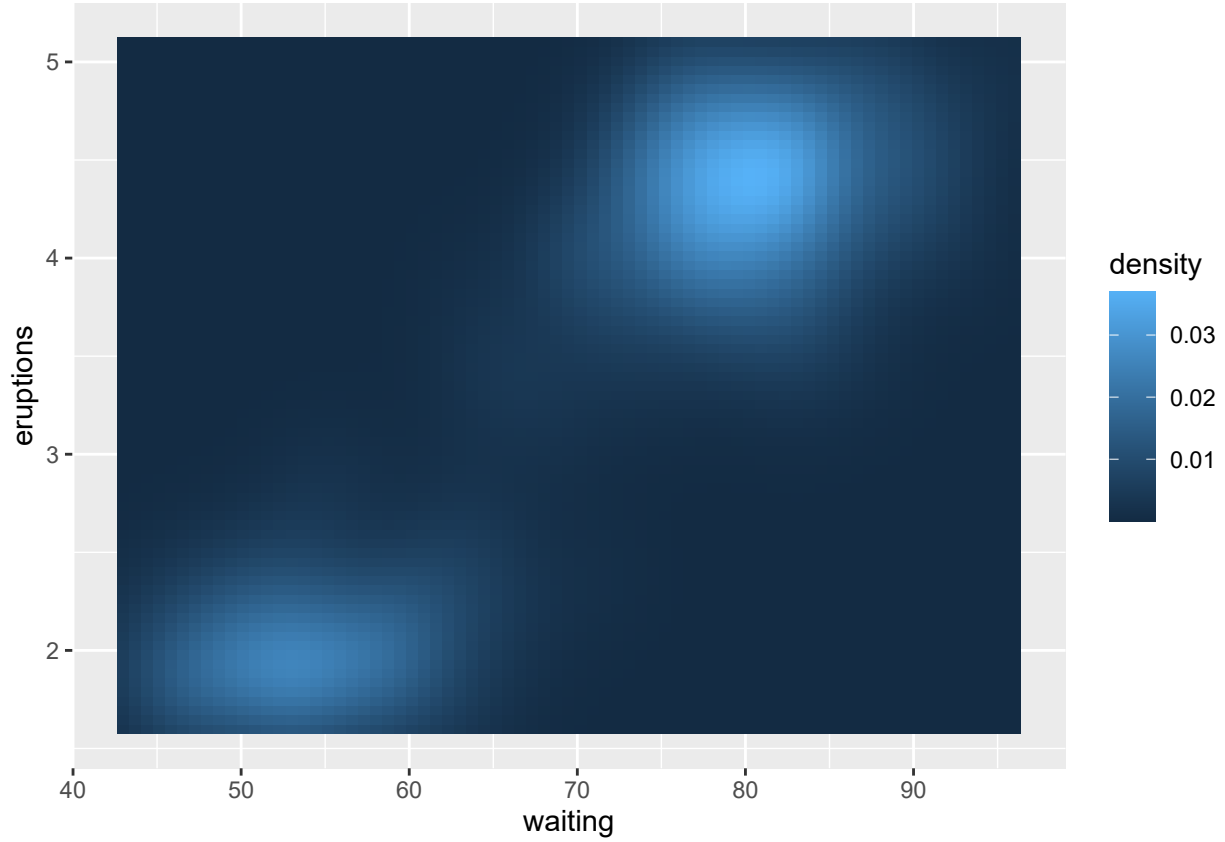
Z funkcijo `geom_tile()` lahko izrišemo heatmap.

```
ggplot(pod_l, aes(Mesto, Meseci, fill= Koncentracije)) +
    geom_tile()
```



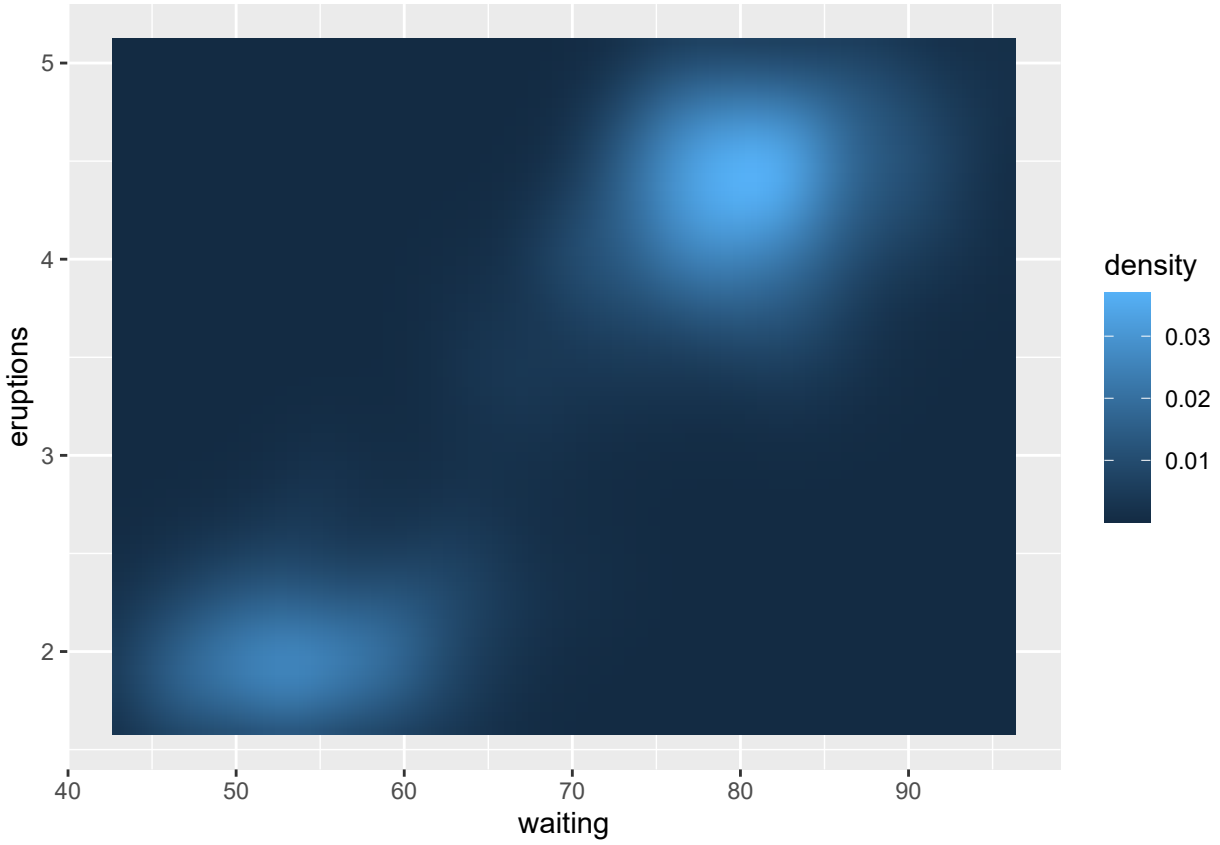
Poglejmo si še izris na nekoliko večji tabeli podatkov o lokacijah izbruhov Old Faithful gejzirja. Večja intenziteta pomeni večjo frekvenco izbruhov na teh lokacijah.

```
ggplot(faithfuld, aes(waiting, eruptions)) +
  geom_tile(aes(fill = density))
```



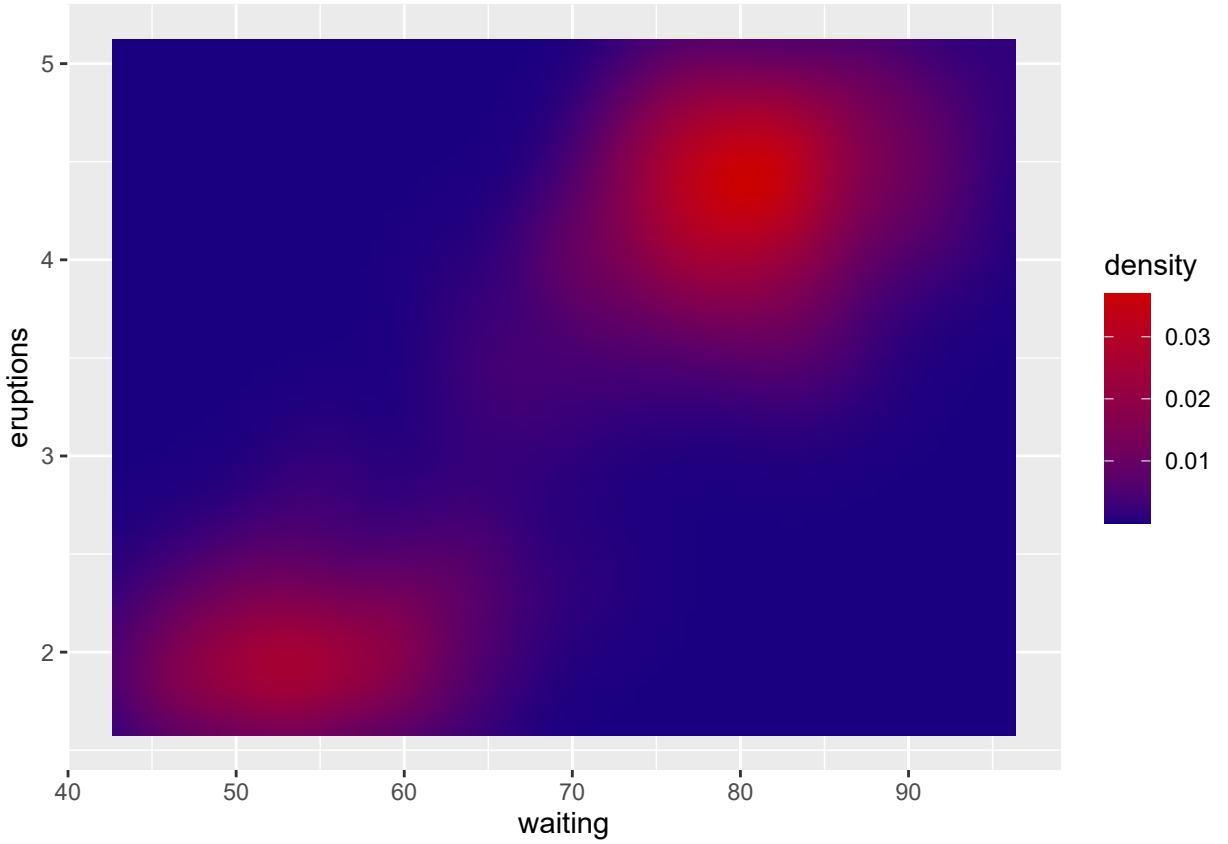
Če bomo takšno sliko vnesli v pdf datoteko zna biti izpis pdf-a počasen. V teh primerih je bolje uporabiti `geom_raster`.

```
ggplot(faithfuld, aes(waiting, eruptions)) +
  geom_raster(aes(fill = density), interpolate = TRUE)
```



Paket **ggplot** sam določa barve izrisa, lahko pa jih tudi sami določimo. To je redkejša operacija, vendar vam lahko včasih pride prav.

```
ggplot(faithfuld, aes(waiting, eruptions)) +
  geom_raster(aes(fill = density), interpolate = TRUE) +
  scale_fill_gradient(low = rgb(0.1, 0.0, 0.5),
                      high = rgb(0.8, 0, 0))
```



Na tej točni, naj samo omenimo, da obstaja še veliko drugih možnosti za izrise. Na tej strani lahko dobite nekaj osnovnih in naprednih idej.

#Izris genomskih podatkov

Vprašanje se je nanašalo, kako bi v R-ju izrisali graf podoben spodnjemu.

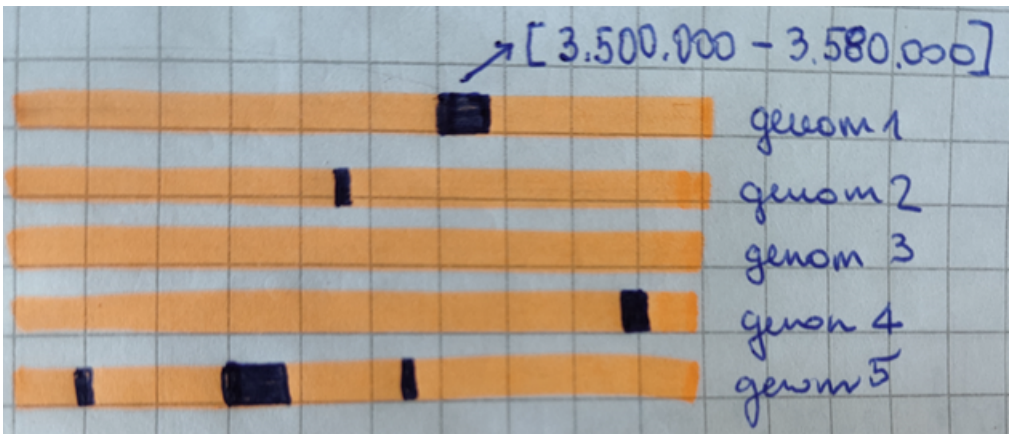
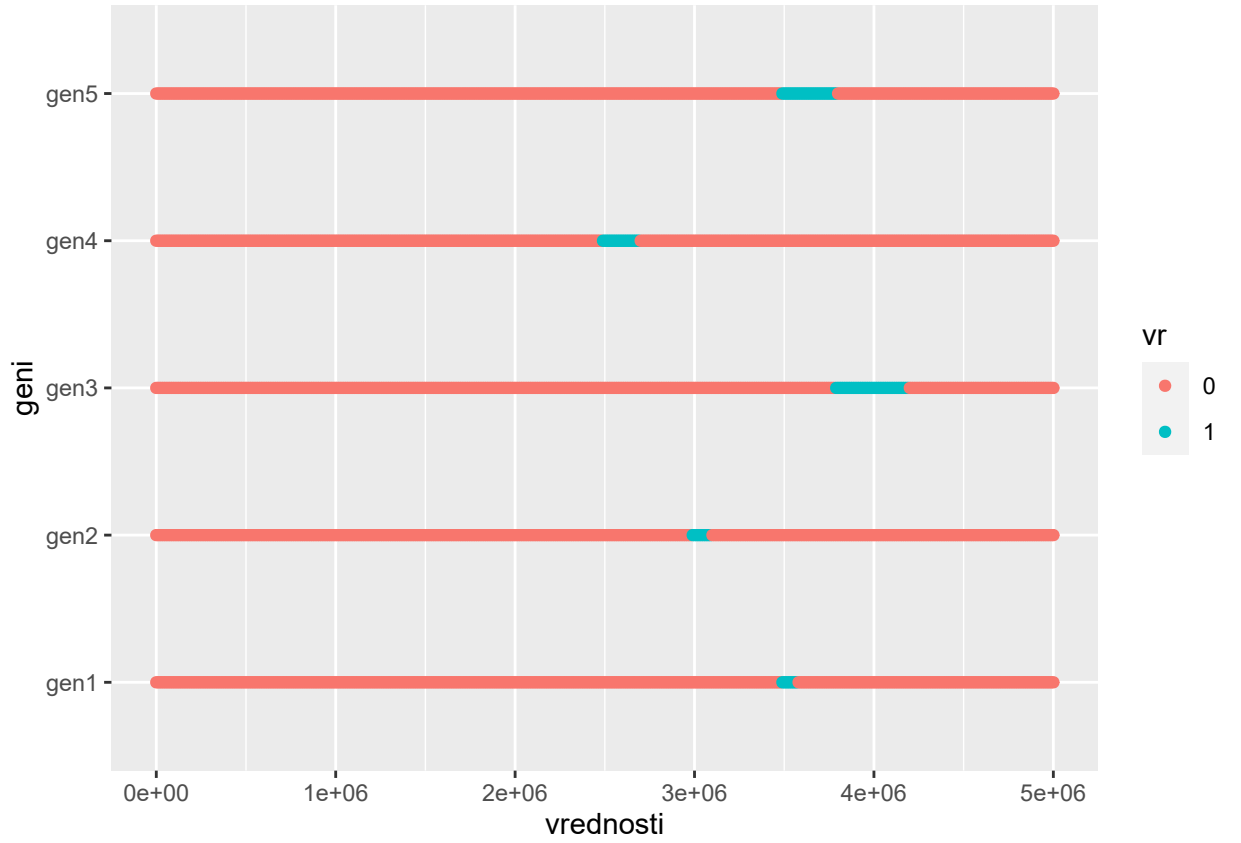


Figure 1: Primer izrisa genomov

Na prvi pogled preprost graf se izkaže za nekoliko zahtevnejšega.

Naredimo najprej primer, kjer bomo izrisali ta graf kar s pomočjo točk. Na graf nanizamo točke eno ob drugi in vsaki določimo barvo 0 ali 1. V tem primeru smo nekatere odseke točk obarvali na roke.

```
df <- data.frame(vrednosti = seq(0, 5000000, by=10000),
                 gen1 = rep(0,501),
                 gen2 = rep(0, 501),
                 gen3=rep(0, 501),
                 gen4 = rep(0, 501),
                 gen5= rep(0,501))
df$gen1[350:358] <- 1
df$gen2[300:310] <- 1
df$gen3[380:420] <- 1
df$gen4[250:270] <- 1
df$gen5[350:380] <- 1

df_l <- pivot_longer(df, cols=2:ncol(df),
                     names_to = "geni", values_to = "vr")
df_l$vr <- factor(df_l$vr)
ggplot(df_l, aes(x = geni, y = vrednosti, color = vr)) +
  geom_point() + coord_flip()
```



Zgornji primer ima težave, če v graf preveč zoomiramo. Če pa tega ne potrebujemo bo ok. Drugi primer prikazuje, kako bi lahko izrisali stolpce. Najprej pripravimo podatke.

```
blocks <- data.frame(
  gens = c(rep("genom1", 3), rep("genom2", 3), rep("genom3", 1),
           rep("genom4", 3), rep("genom5", 7)),
  type = c("off", "on", "off", #genome1
           "off", "on", "off", #genome 2...
           "off",
```

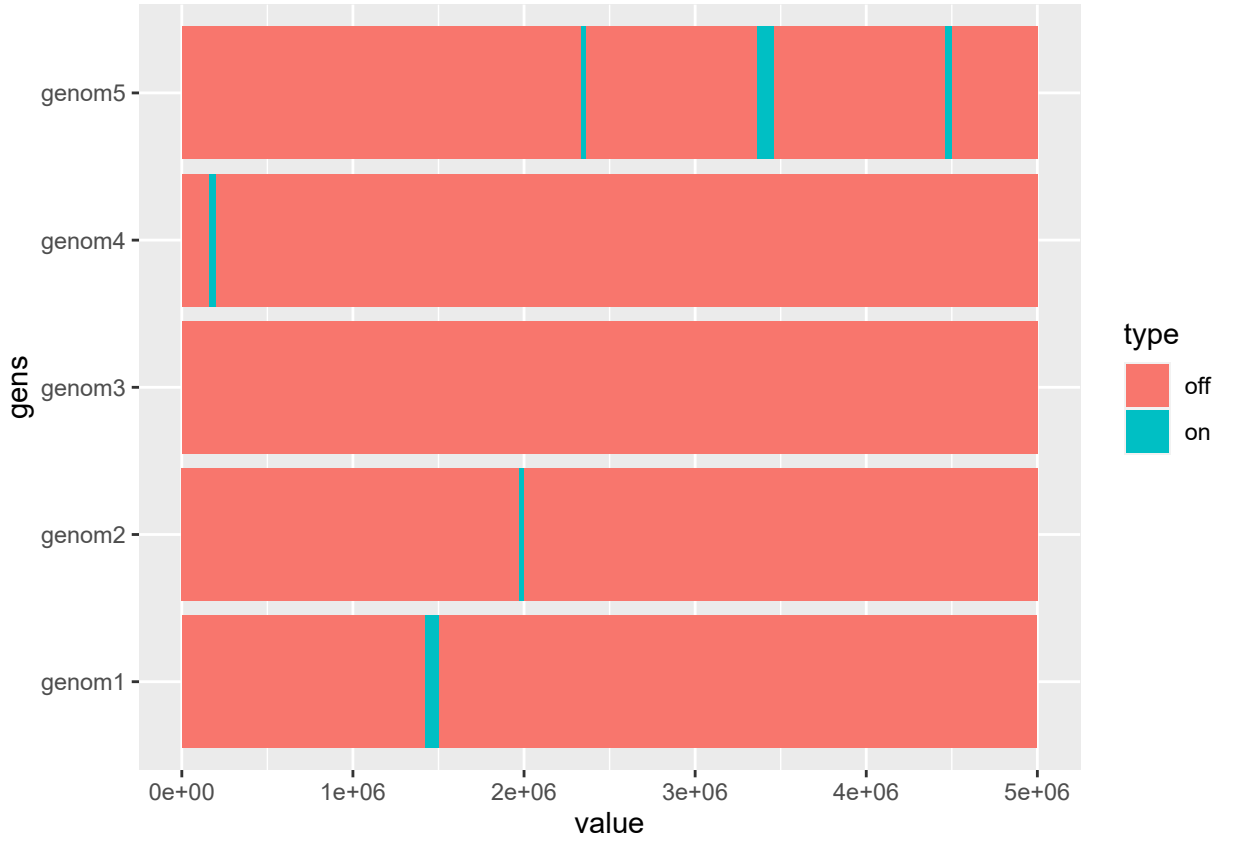
```
            "off", "on", "off",
            "off", "on", "off", "on", "off", "on", "off"
            ),
  group = c(1:3, 1:3, 1, 1:3, 1:7),
  value = c(3500000, 80000, 5000000-3500000-80000,
             3000000, 30000, 5000000-3000000-30000,
             5000000,
             4800000, 45000, 5000000-4800000-45000,
             500000, 40000, 1000000, 100000, 1000000, 30000, 5000000-2670000)
)

head(blocks)
```

```
##     gens type group   value
## 1 genom1  off     1 3500000
## 2 genom1   on     2   80000
## 3 genom1  off     3 1420000
## 4 genom2  off     1 3000000
## 5 genom2   on     2   30000
## 6 genom2  off     3 1970000
```

Tukaj je gens le ime genoma, ki ga izrisujemo, type (on/off) uporabljamo le za barvo (off - rdeča, on - modra). Group predstavlja odsek stolpca, *value* pa širino tega stolpca. Seveda bi lahko namesto širine uporabili tudi drugačno predstavitev, kot je naprimer interval *value_od* in *value_do*.

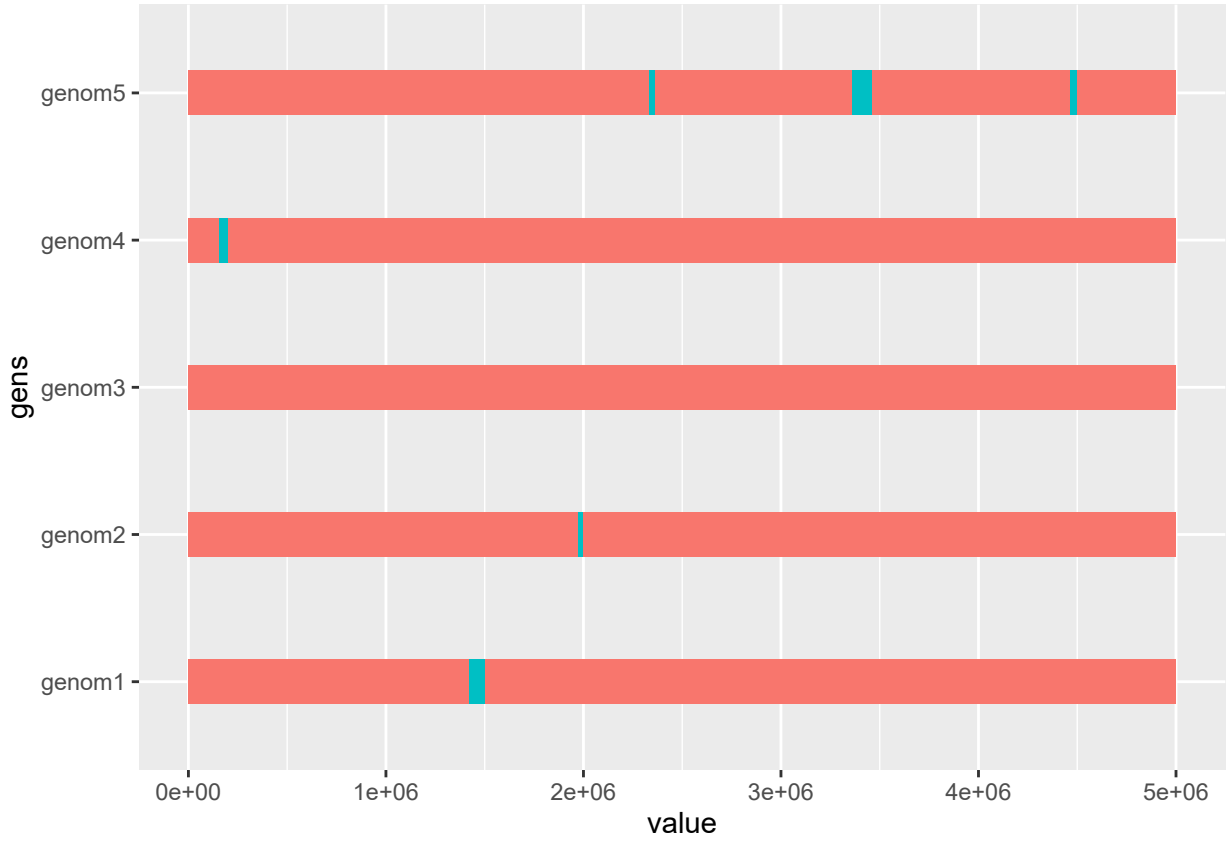
Graf lahko izrišemo z `geom_col()` oziroma `geom_bar()`. V kombinaciji z `coord_flip()` pa ga prestavimo v ležeči položaj.

```
ggplot(blocks, aes(gens, value, group = group)) +
  geom_col(aes(fill = type)) +
  coord_flip()
```



Še par preprostih izboljšav.

```
ggplot(blocks, aes(gens, value, group = group)) +
  geom_col(aes(fill = type), width = 0.3) +
  coord_flip() +
  theme(legend.position = "none")
```



Za delo z genomskimi podatki je uporaben tudi BiocManager oziroma BioConductor, ki ima nekaj tisoč paketov za obdelavo teh podatkov.

Instalacija le tega je zahtevna vendar uporabna, če delate s takšnimi podatki.

Spodnji primer vam lahko pomaga pri instalaciji skupaj z dvema paketoma za vizualizacijo in z primerim podatkov.

```
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

BiocManager::install("Gviz")
BiocManager::install("GenomicRanges")
```

Spodnji primer je pobran s strani STHDA in prikazuje par možnih izrisov genomskih podatkov.

Naložimo podatke.

```
library(Gviz)
```

```
## Loading required package: S4Vectors
## Loading required package: stats4
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:lubridate':
##
```

```
##     intersect, setdiff, union

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:lubridate':
##
##     second, second<-

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:tidyr':
##
##     expand

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:lubridate':
##
##     %within%

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:purrr':
##
##     reduce

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb
```

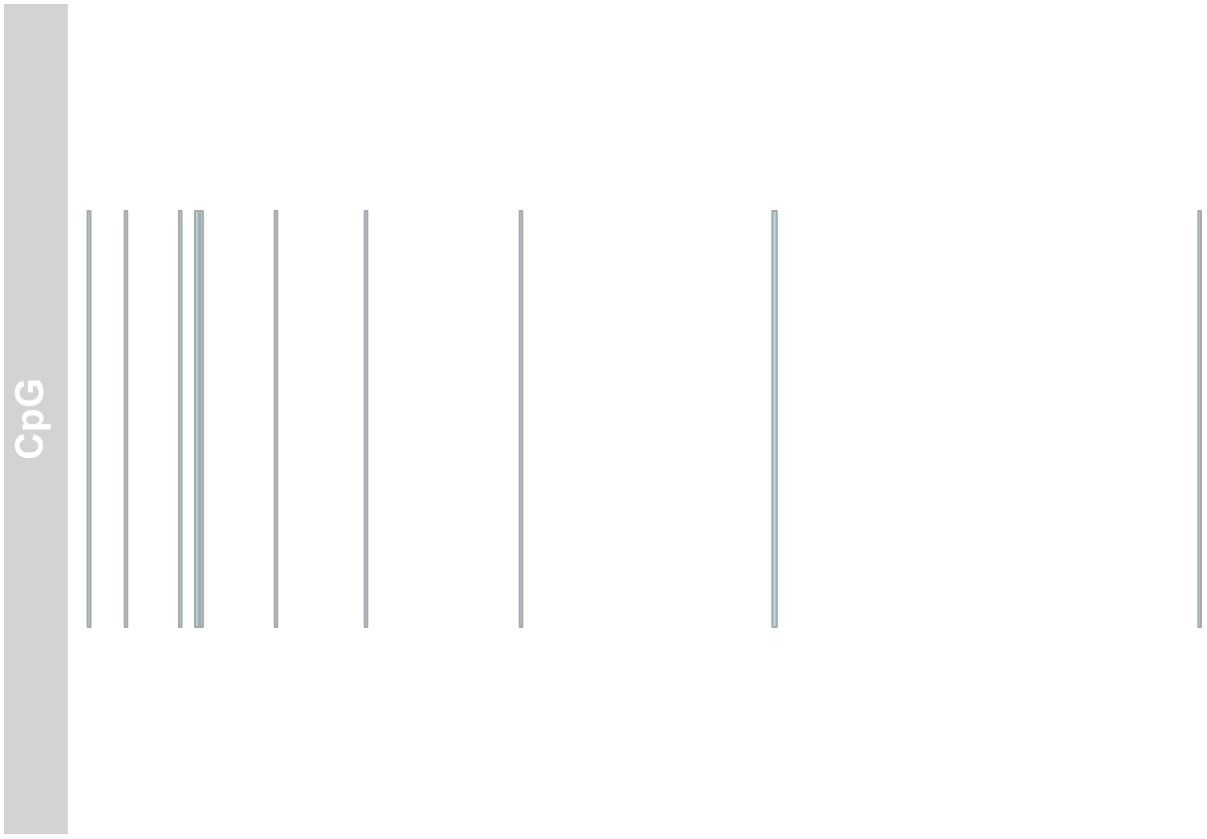
```
## Loading required package: grid
```

```
library(GenomicRanges)
  #Load data : class = GRanges
data(cpgIslands)
cpgIslands
```

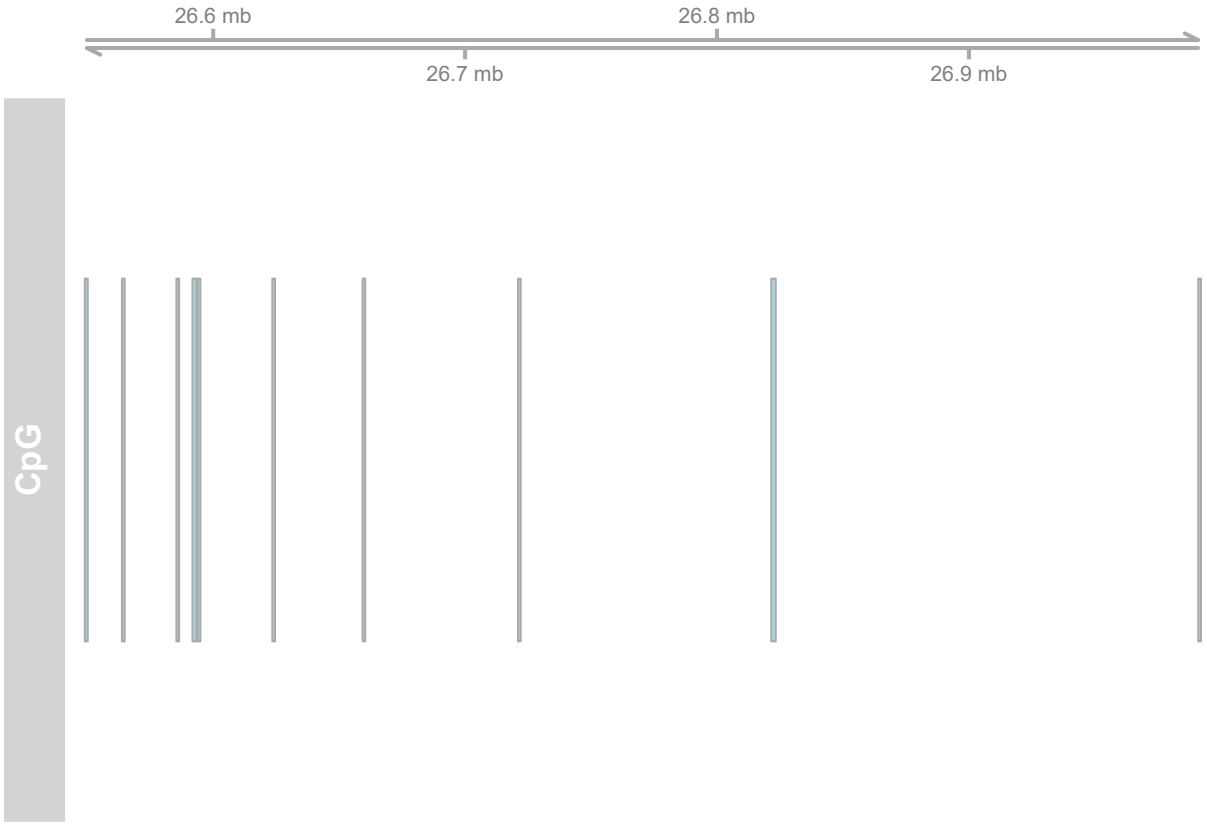
```
## GRanges object with 10 ranges and 0 metadata columns:
##        seqnames            ranges strand
##           <Rle>         <IRanges>  <Rle>
##    [1]     chr7 26549019-26550183      *
##    [2]     chr7 26564119-26564500      *
##    [3]     chr7 26585667-26586158      *
##    [4]     chr7 26591772-26593309      *
##    [5]     chr7 26594192-26594570      *
##    [6]     chr7 26623835-26624150      *
##    [7]     chr7 26659284-26660352      *
##    [8]     chr7 26721294-26721717      *
##    [9]     chr7 26821518-26823297      *
##   [10]     chr7 26991322-26991841      *
##   -------
##   seqinfo: 1 sequence from hg19 genome; no seqlengths
```

Sledijo izrisi. Kot uporabniku R-ja v večinoma namene strojnega učenja so mi spodnji izrisi dokaj nerazumljivi, tako da razumevanje prepušam vam.

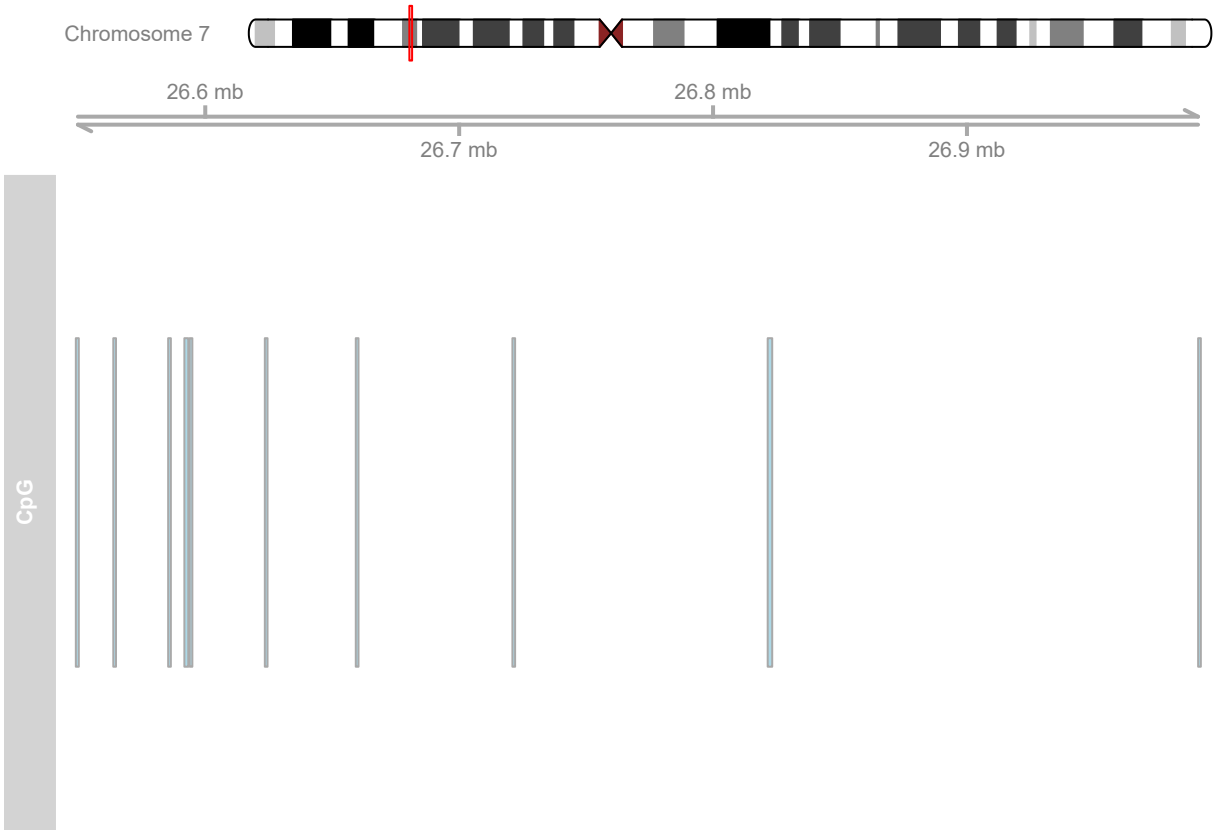
```
atrack <- AnnotationTrack(cpgIslands, name = "CpG")
plotTracks(atrack)
```



```
## genomic coordinates
gtrack <- GenomeAxisTrack()
plotTracks(list(gtrack, atrack))
```



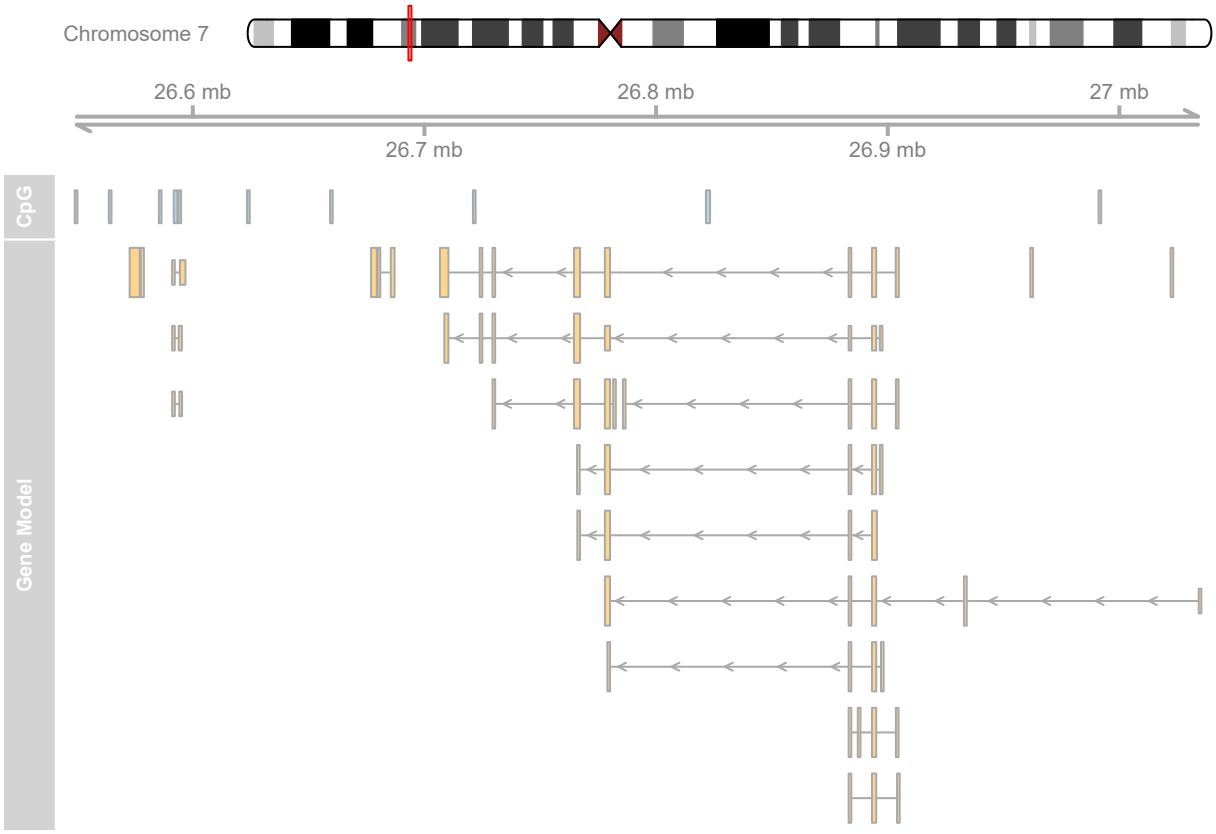

```
#genome : "hg19"
gen<-genome(cpgIslands)
#Chromosme name : "chr7"
chr <- as.character(unique(seqnames(cpgIslands)))
#Ideogram track
itrack <- IdeogramTrack(genome = gen, chromosome = chr)
plotTracks(list(itrack, gtrack, atrack))
```




```
#Load data
data(geneModels)
head(geneModels)
```

```
##   chromosome    start      end width strand feature            gene
## 1       chr7 26591441 26591829   389      + lincRNA ENSG00000233760
## 2       chr7 26591458 26591829   372      + lincRNA ENSG00000233760
## 3       chr7 26591515 26591829   315      + lincRNA ENSG00000233760
## 4       chr7 26594428 26594538   111      + lincRNA ENSG00000233760
## 5       chr7 26594428 26596819  2392      + lincRNA ENSG00000233760
## 6       chr7 26594641 26594733    93      + lincRNA ENSG00000233760
##              exon      transcript     symbol
## 1 ENSE00001693369 ENST00000420912 AC004947.2
## 2 ENSE00001596777 ENST00000457000 AC004947.2
## 3 ENSE00001601658 ENST00000430426 AC004947.2
## 4 ENSE00001792454 ENST00000457000 AC004947.2
## 5 ENSE00001618328 ENST00000420912 AC004947.2
## 6 ENSE00001716169 ENST00000457000 AC004947.2
```

```
#Plot
grtrack <- GeneRegionTrack(geneModels, genome = gen,
                           chromosome = chr, name = "Gene Model")
plotTracks(list(itrack, gtrack, atrack, grtrack))
```



Branje rasterskih podatkov

Za branje rasterskih podatkov potrebujemo različne pakete. Uporabili bomo paket *terra* (https://cran.r-project.org/web/packages/terra/index.html), ki nam omogoča obdelavo prostorskih podatkov v vektorski ali rasterski obliki. Paket podpira procesiranje tudi zelo velikih podatkov, ker je večina fukcij programiranih tako, da omogočajo paralelno procesiranje (na več jedrih). Za kratek uvod v paket *terra* si lahko pogledate (https://rspatial.org/index.html).

Pokažimo prepost primer na podatkih tipa GeoTiff. S funkcijo `describe` dobimo podatke o prebrani GeoTiff datoteki.

```
library(terra)
```

```
## terra 1.7.37
```

```
##
## Attaching package: 'terra'
```

```
## The following objects are masked from 'package:Gviz':
##
##     values, values<-, width
```

```
## The following object is masked from 'package:grid':
##
##     depth
```

```
## The following objects are masked from 'package:GenomicRanges':
##
```

```
##     distance, gaps, nearest, shift, trim, values, values<-, width
## The following objects are masked from 'package:IRanges':
##
##     distance, gaps, nearest, shift, trim, width
## The following objects are masked from 'package:S4Vectors':
##
##     values, values<-, width
## The following object is masked from 'package:BiocGenerics':
##
##     width
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(ggplot2)
library(dplyr)

describe('./img/cea.tif')
```

```
##  [1] "Driver: GTiff/GeoTIFF"
##  [2] "Files: ./img/cea.tif"
##  [3] "Size is 514, 515"
##  [4] "Coordinate System is:"
##  [5] "PROJCRS[\"unnamed\","
##  [6] "    BASEGEOGCRS[\"NAD27\","
##  [7] "        DATUM[\"North American Datum 1927\","
##  [8] "            ELLIPSOID[\"Clarke 1866\",6378206.4,294.978698213898,"
##  [9] "                LENGTHUNIT[\"metre\",1]]],"
## [10] "        PRIMEM[\"Greenwich\",0,"
## [11] "            ANGLEUNIT[\"degree\",0.0174532925199433]],"
## [12] "        ID[\"EPSG\",4267]],"
## [13] "    CONVERSION[\"Lambert Cylindrical Equal Area\","
## [14] "        METHOD[\"Lambert Cylindrical Equal Area\","
## [15] "            ID[\"EPSG\",9835]],"
## [16] "        PARAMETER[\"Latitude of 1st standard parallel\",33.75,"
## [17] "            ANGLEUNIT[\"degree\",0.0174532925199433],"
## [18] "            ID[\"EPSG\",8823]],"
## [19] "        PARAMETER[\"Longitude of natural origin\",-117.333333333333,"
## [20] "            ANGLEUNIT[\"degree\",0.0174532925199433],"
## [21] "            ID[\"EPSG\",8802]],"
## [22] "        PARAMETER[\"False easting\",0,"
## [23] "            LENGTHUNIT[\"metre\",1],"
## [24] "            ID[\"EPSG\",8806]],"
## [25] "        PARAMETER[\"False northing\",0,"
## [26] "            LENGTHUNIT[\"metre\",1],"
## [27] "            ID[\"EPSG\",8807]]],"
## [28] "    CS[Cartesian,2],"
## [29] "        AXIS[\"easting\",east,"
## [30] "            ORDER[1],"
## [31] "            LENGTHUNIT[\"metre\",1,"
## [32] "                ID[\"EPSG\",9001]]],"
## [33] "        AXIS[\"northing\",north,"
## [34] "            ORDER[2],"
```

```
## [35] "            LENGTHUNIT[\"metre\",1,"
## [36] "                ID[\"EPSG\",9001]]]]"
## [37] "Data axis to CRS axis mapping: 1,2"
## [38] "Origin = (-28493.166784412522247,4255884.543802191503346)"
## [39] "Pixel Size = (60.022136983193739,-60.022136983193739)"
## [40] "Metadata:"
## [41] "  AREA_OR_POINT=Area"
## [42] "Image Structure Metadata:"
## [43] "  INTERLEAVE=BAND"
## [44] "Corner Coordinates:"
## [45] "Upper Left  (  -28493.167, 4255884.544) (117d38'27.05\"W, 33d56'37.74\"N)"
## [46] "Lower Left  (  -28493.167, 4224973.143) (117d38'27.05\"W, 33d39'53.81\"N)"
## [47] "Upper Right (    2358.212, 4255884.544) (117d18'28.38\"W, 33d56'37.74\"N)"
## [48] "Lower Right (    2358.212, 4224973.143) (117d18'28.38\"W, 33d39'53.81\"N)"
## [49] "Center      (  -13067.478, 4240428.844) (117d28'27.71\"W, 33d48'15.38\"N)"
## [50] "Band 1 Block=514x15 Type=Byte, ColorInterp=Gray"
```

Preberemo datoteko v R in s funkcijo `summary()` dobimo osnovne informacije o podatkih na sliki (statistika intenzitete pikslov).

```
Slika <- terra::rast("./img/cea.tif")
summary(Slika)
```

```
##       cea
##  Min.   :  0.0
##  1st Qu.: 58.0
##  Median : 99.0
##  Mean   :102.9
##  3rd Qu.:140.0
##  Max.   :255.0
```

Sliko lahko pretvorimo v **data.frame** in pogledamo njene osnovne lastnosti:

```
slika_df <- as.data.frame(Slika, xy = TRUE)
head(slika_df)
```

```
##           x       y cea
## 1 -28463.16 4255855   0
## 2 -28403.13 4255855   0
## 3 -28343.11 4255855   0
## 4 -28283.09 4255855   0
## 5 -28223.07 4255855   0
## 6 -28163.05 4255855   0
```

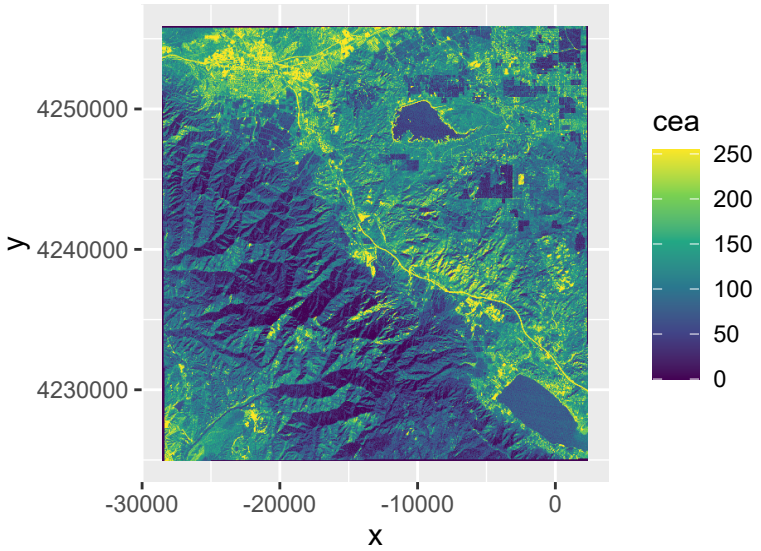
```
summary(slika_df)
```

```
##        x                y                cea
##  Min.   :-28463   Min.   :4225003   Min.   :  0.0
##  1st Qu.:-20780   1st Qu.:4232686   1st Qu.: 58.0
##  Median :-13067   Median :4240429   Median : 99.0
##  Mean   :-13067   Mean   :4240429   Mean   :103.1
##  3rd Qu.: -5355   3rd Qu.:4248172   3rd Qu.:140.0
##  Max.   :  2328   Max.   :4255855   Max.   :255.0
```

V **data.frame** so zabeleženi podatki o lokacijah in intenzitetah posameznih pikslov.

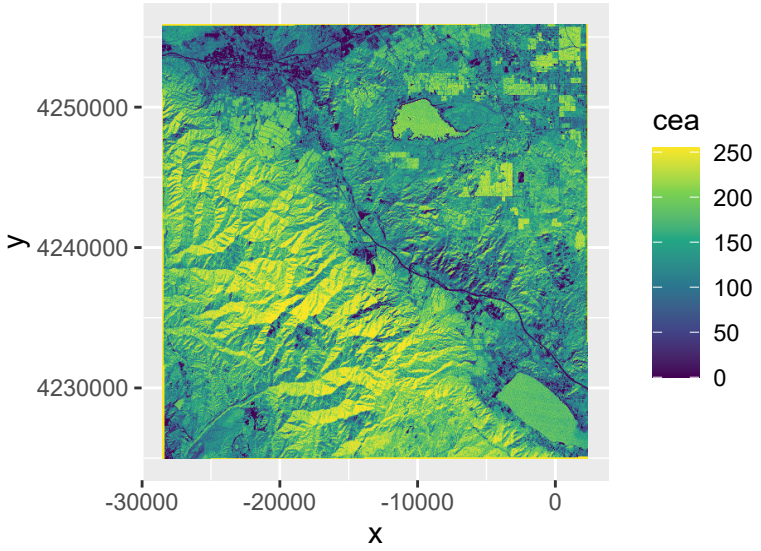
S pomočjo paketa *ggplot* jo lahko narišemo:

```
library(ggplot2)
ggplot() +
    geom_raster(data = slika_df , aes(x = x, y = y, fill = cea)) +
    scale_fill_viridis_c() + # barve v katerih prikažemo
    coord_quickmap() #omogoča projekcijo dela zemlje na ravnino z določeno projekcijo
```



Lahko naredimo inverz slike (zamenjamo barve pikslov):

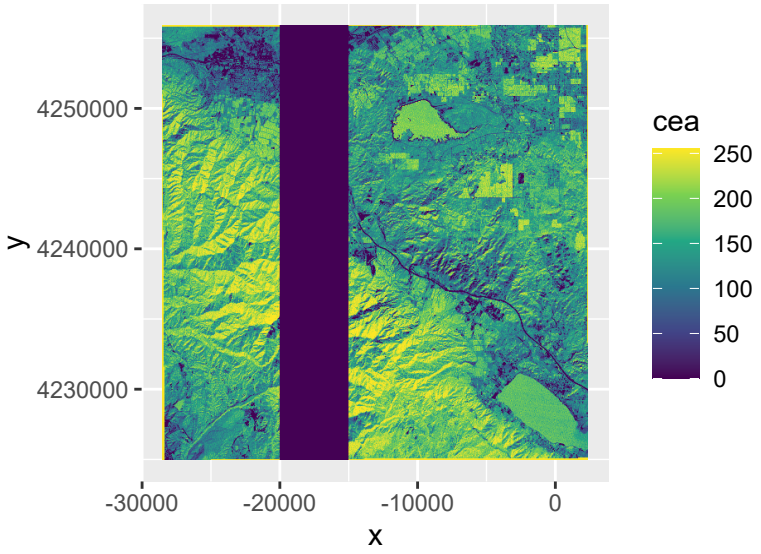
```
slika_df$cea <- 255 - slika_df$cea
ggplot() +
  geom_raster(data = slika_df , aes(x = x, y = y, fill = cea)) +
  scale_fill_viridis_c() +
  coord_quickmap()
```



Oziroma "odstranimo" del slike:

```
slika_df[slika_df$x > -20000 & slika_df$x < -15000,3] <- 0
```

```
ggplot() +
  geom_raster(data = slika_df , aes(x = x, y = y, fill = cea)) +
  scale_fill_viridis_c() +
  coord_quickmap()
```



Sliko lahko shranimo s paketom *ggplot*, kot sliko:

```
ggsave("./img/inverz.png")
```

```
## Saving 6.5 x 4.5 in image
```

Oziroma **data.frame** shranimo nazaj kot GeoTiff:

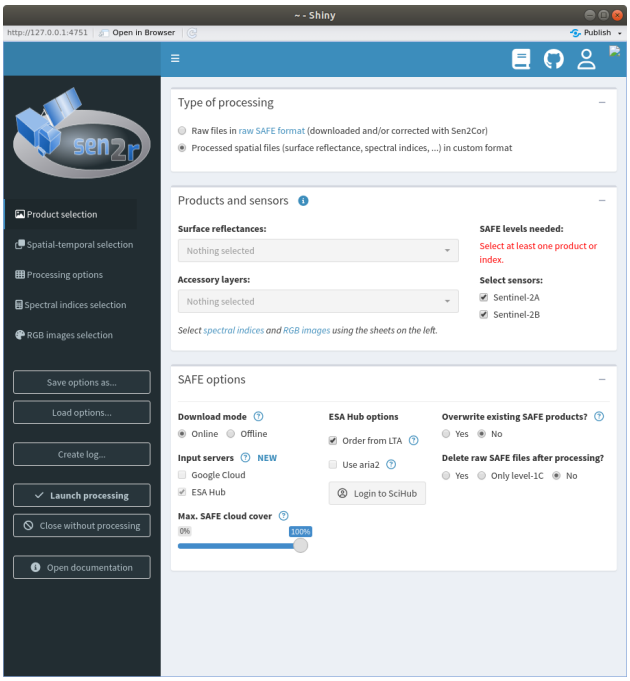
```
writeRaster(terra::rast(slika_df), "./img/inverz.tif",
            wopt= list(gdal=c("COMPRESS=NONE"), datatype='INT1U'),
            overwrite = TRUE)
```

Satelitski posnetki s paketom *sen2r*

Za delo s podatki sateliv Santinel-2 je na voljo paket *sen2r*, ki omogoča shranjevanje podatkov iz spletne strani na disk in obdelavo v R-ju. Več informacij najdete na (https://sen2r.ranghetti.info/).

```
library(sen2r)
library(sf)
# potrebujete shape datoteko, da definirate območje
border = read_sf("./Ljubljana_border.shp")
sen2r()
```

Z ukazom `sen2r()` se nam odpre uporabniški vmesnik, ki nam pomoga izbrati podatke satelitov Sentinel.



S pomočjo funkcije `s2_list()` filtriramo satelitske posnetke, ki smo jih shranili. Seznam posnetkov shranimo v **data frame**. Funkcija `s2_list()` omogoča veliko število vhodnih parametrov, med njimi tudi parameter *spatial_extent*, ki predstavlja obravnavano območje. To moramo podatki v obliki objekta tipa *sf*, *sfc* ali *sfg*, ki so objekti, ki predstavljajo prostorske podatke. To so ponavadi **shape** datoteke, ki predstavljajo konturo območja. Ker tako podakti niso bili na boljo, ne moremo pokazati izpisa ob zagonu kode.

```
images_list = s2_list(
   spatial_extent = border,
   time_interval = as.Date(c("2015-05-01", "2020-08-30")),
   max_cloud = 1
)

images_list = as.data.frame(images_list)
```