Lekérdezés, összefűzés, transzformálás

Térinformatika R-ben

2023.11.07.

Section 1

Lekérdezés és logikai kifejezések

Lekérdezés

Az `sf` objektum `data.frame` is.
Lekérdezés és módosítás hasonló:

- `$`: oszloplekérés oszlopnév alapján (eredmény: vektor)
- `[i, j, drop]`: résztáblázat lekérése (eredmény: `sf`)

A `drop` paramétert kevésbé ismert, pedig fontos!

Lekérdezés

```
library(sf)
load("varosok.RData")
```

```
colnames(varosok)
```

```
[1] "nev"        "lakossag"   "terulet"    "hatarhossz"
[5] "geometry"
```

```
class(varosok$lakossag)
```

```
[1] "integer"
```

Lekérdezés

A $ után oszlopnevet írhatunk idézőjellel vagy anélkül.
Eredményként vektort ad vissza.

```
varosok$lakossag
```

```
 [1] 1861383   72473  211038 1861383 1861383   55859
 [7]   81924  106350   77634   77634   68700  168276
[13]  162502   67971   62853   55313  107752   61657
[19]   52109  118799   58112  129415   56179 1861383
[25]  184129
```

Lekérdezés

Ha az oszlopnév szóközt vagy egyéb, nem szokványos karaktert tartalmaz, rakjuk idézőjelek közé!

```
colnames(varosok)[1] <- "a varos neve"
```

```
varosok$a varos neve
```

```
Error: <text>:1:11: unexpected symbol
1: varosok$a varos
               ^
```

```
varosok$"a varos neve"
```

```
 [1] "Budapest"       "Tatabánya"      "Debrecen"
 [4] "Budapest"       "Budapest"       "Érd"
 [7] "Szombathely"    "Székesfehérvár" "Szolnok"
[10] "Szolnok"        "Kaposvár"       "Szeged"
```

...

Lekérdezés

```
colnames(varosok)[1] <- "nev"
```

Inkább visszaállítjuk az oszlopnevet...

Lekérdezés

Lekérdezés a `[i, j, drop]` operátorral

- `i`: sorok száma, neve vagy a szükséges sorok logikai maszkja
- `j`: oszlopok száma, neve vagy a szükséges oszlopok logikai maszkja
- `drop`: logikai kapcsoló. Egyszerűsítsük a struktúrát, ha lehetséges? (később)

Eltérések a `data.frame`-es esethez képest

- visszaadott érték `sf` lesz
- vagyis a geometriát tartalmazó oszlop megőrződik
- tehát azt nem kell az oszlopok között feltüntetni
- `drop` alapértelmezetten `FALSE`

Lekérdezés

```
class(varosok[, "lakossag"])
```

```
[1] "sf"         "data.frame"
```

```
colnames(varosok[, "lakossag"])
```

```
[1] "lakossag" "geometry"
```

```
fontos_oszlopok <- c("nev", "terulet", "hatarhossz")
length(fontos_oszlopok)
```

```
[1] 3
```

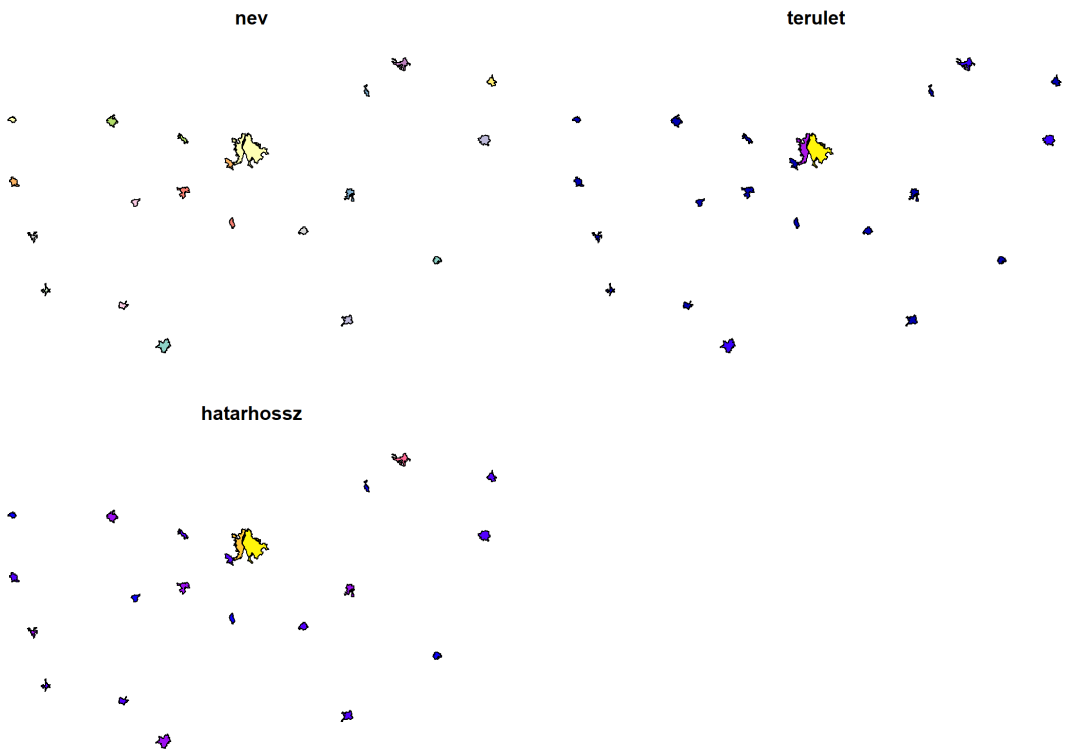
```
ncol(varosok[, fontos_oszlopok])
```

```
[1] 4
```

Lekérdezés

Bár a lekérendő oszlopok között nem soroltuk fel a geometriaoszlopot, az mégis megmaradt.
Ha nem maradna meg a geometriaoszlop, nem tudnánk megjeleníteni...

```
plot(varosok[, fontos_oszlopok])
```



Lekérdezés

Sorok és oszlopok egyszerre leválogathatóak.

```
rownames(varosok)
```

```
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
[11] "11" "12" "13" "14" "15" "16" "17" "18" "19" "20"
[21] "21" "22" "23" "24" "25"
```

```
dim(varosok[c("4", "3"), c("terulet", "hatarhossz")])
```

```
[1] 2 3
```

Megint eggyel több oszlopot kaptunk, mint ahányat kértünk.

Lekérdezés

A szűkétésre többféle lehetőségünk van (ismétlés):

- számvektorral (hanyadik sorok/oszlopok)
- negatív számvektorral (értsd: ezek kivételével mindegyik)
- szövegvektorral (sor- és oszlopnevek)
- logikai vektorral (kell-e az adott sor/oszlop?)
- logikai kifejezéssel (ez végülis logikai vektor lesz)
- üresen hagyjuk (összes sor/oszlop szükséges)

Logikai kifejezéssel jellemzően sorokra szoktunk szűkíteni, szövegvektorral pedig oszlopokra.

Lekérdezés

```
load("orszagok_osszes.RData")
str(orszagok_osszes)
```

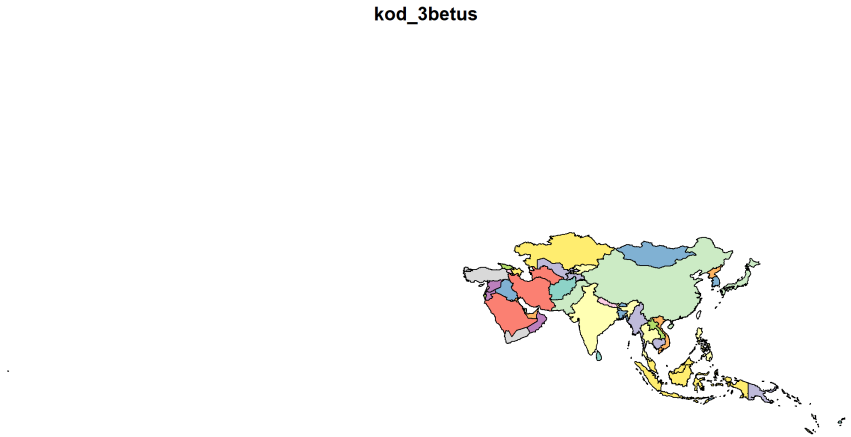
```
Classes 'sf' and 'data.frame':  211 obs. of  7 variables:
    $ nev : Factor w/ 211 levels "Afghanistan",..: 1 2
     3 4 5 6 7 8 9 10 ...
    $ kod_3betus: Factor w/ 204 levels
     "AFG","AGO","ALB",..: 1 3 52 4 2 5 9 7 8 10 ...
    $ kod_2betus: Factor w/ 203 levels
     "AD","AE","AF",..: 3 5 51 1 7 10 4 8 6 11 ...
    $ terulet : Factor w/ 190 levels
     "0","1000","100000",..: 151 81 69 1 25 1 121 80
     87 164 ...
    $ regio : Factor w/ 10 levels
     "Antarctica","Asia",..: 2 5 8 5 10 1 4 6 2 3 ...
    $ nepesseg : Factor w/ 204 levels
     "0","10086387",..: 69 92 97 172 39 1 188 109 89
     56 ...
    $ geometry :sfc_MULTIPOLYGON of length 211; first
     list element: List of 1
```

Lekérdezés

```
levels(orszagok_osszes$regio)
```

```
 [1] "Antarctica"          "Asia"
 [3] "Australia"           "Caribbean"
 [5] "Europe"              "Latin America"
 [7] "North America"       "NorthAfrica"
 [9] "Pacific"             "Sub Saharan Africa"
```

```
plot(orszagok_osszes[orszagok_osszes$regio == "Asia",
 "kod_3betus"])
```



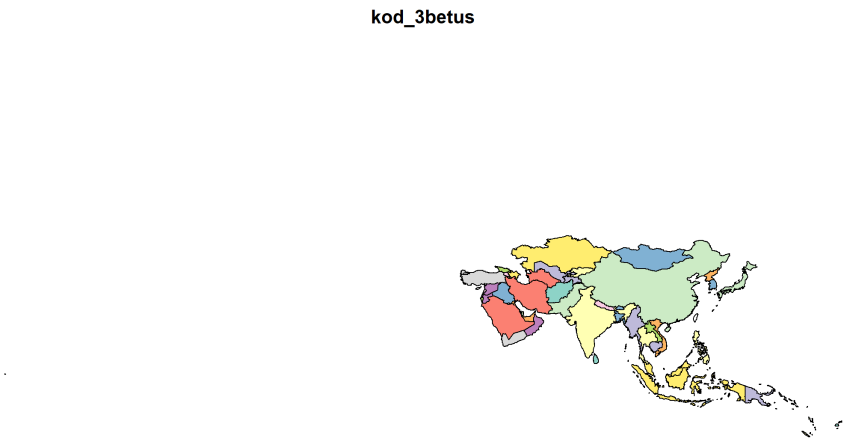
Lekérdezés

A logikai kifejezést előzetesen is kiértékelhetjük, majd szűkíthetünk logikai maszkkal.

```
logikai_maszk <- orszagok_osszes$regio == "Asia"
str(logikai_maszk)
```

```
 logi [1:211] TRUE FALSE FALSE FALSE FALSE FALSE ...
```

```
plot(orszagok_osszes[logikai_maszk, 2])
```



Lekérdezés

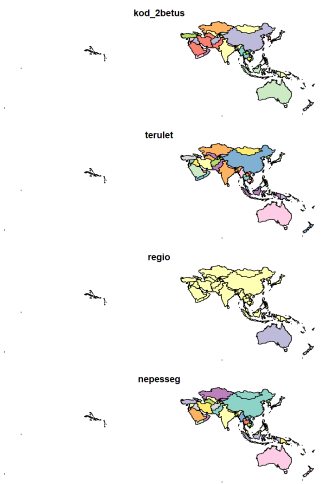
Emlékeztető: az %in% operátor minden bal oldali elemre logikai értékként megmondja, hogy része-e a jobb oldali vektornak mint halmaznak.

```
logikai_maszk <- orszagok_osszes$regio %in% c("Australia",
 "Caribbean", "Asia")
str(logikai_maszk)
```

```
 logi [1:211] TRUE FALSE FALSE FALSE FALSE FALSE ...
```

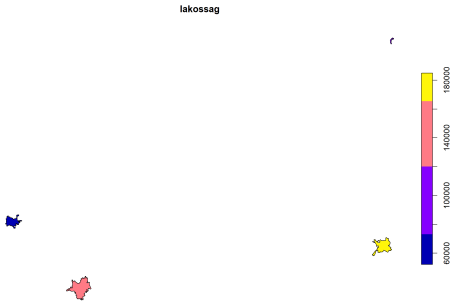
Lekérdezés

```
plot(orszagok_osszes[logikai_maszk, -c(1:2)])
```



1. feladat (órai)

- Mi a városok “nev” oszlopának osztálya?
- Mik a szintjei?
- Hozd létre Eger és Miskolc városok logikai maszkját varosok_maszkja néven.
- Jelenítsd meg ezen városokat a 3. és 4. oszlop szerint színezve.
- Jelenítsd meg a 10–13 sorszámú városokat a “lakossag” oszlop szerint színezve.
- Végül jelenítsd meg azon városokat a területük (“terulet” oszlop) alapján színezve, amelyek sorszáma nem esik az 1-4 tartományba, valamint nem is 6.



1. feladat (órai) – megoldás

```
class(varosok$nev)
```

```
[1] "factor"
```

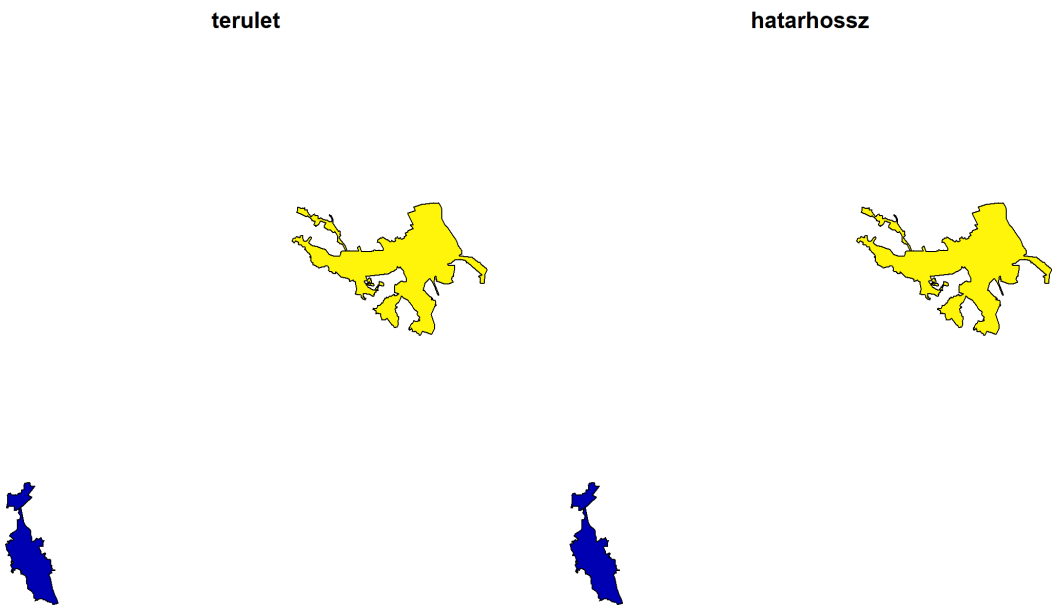
```
levels(varosok$nev)
```

```
 [1] "Békéscsaba"      "Budapest"        "Debrecen"
 [4] "Dunaújváros"     "Eger"            "Érd"
 [7] "Gyor"            "Kaposvár"        "Kecskemét"
[10] "Miskolc"         "Nagykanizsa"     "Nyíregyháza"
[13] "Pécs"            "Sopron"          "Szeged"
[16] "Székesfehérvár" "Szolnok"         "Szombathely"
[19] "Tatabánya"       "Veszprém"        "Zalaegerszeg"
```

```
varosok_maszkja <- varosok$nev %in% c("Eger", "Miskolc")
```

1. feladat (órai) – megoldás

```
plot(varosok[varosok_maszkja, 3:4])
```

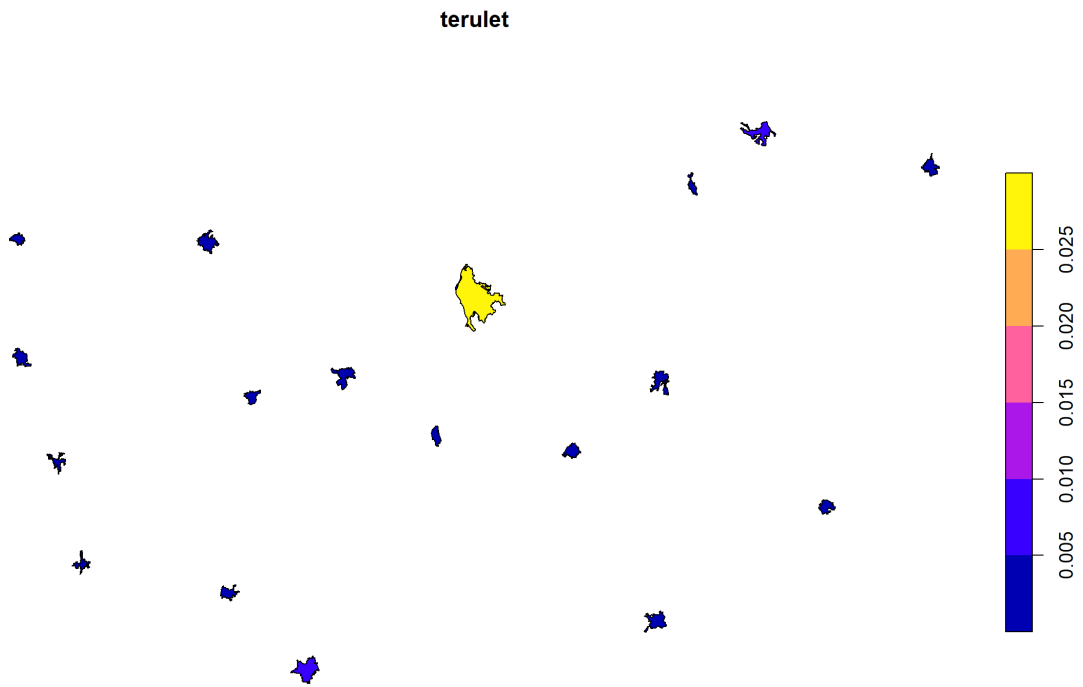


1. feladat (órai) – megoldás

```
plot(varosok[10:13, "lakossag"])
```

1. feladat (órai) – megoldás

```
plot(varosok[-c(1:4, 6), "terulet"])
```



Lekérdezés

A `drop` paraméterrel szabályozhatjuk, hogy ha egydimenziós a lekérdezés eredménye, akkor elvesszen-e a táblázatos struktúra.

- `data.frame`-ek esetén alapértelmezetten `TRUE`
- emiatt kiszámíthatatlan az `[i, j]` típusú lekérdezés eredményének típusa!
- megoldás: `drop` paraméter megadása, vagy a `tibble` csomag használata (link)
- `sf` esetén szerencsére alapértelmezetten `FALSE`

`drop = TRUE` esetén az eredmény

- vektor (egy oszlop) vagy
- lista (több oszlop, akár egy sor).

Listából vektort az `unlist()` függvénnyel készíthetünk.

Lekérdezés

```
load("nuts_regiok.RData")
str(nuts_regiok)
```

```
Classes 'sf' and 'data.frame':  1951 obs. of  5 variables:
    $ regio_kod : Factor w/ 1951 levels
     "AT","AT1","AT11",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ regio_szint: int  0 1 2 3 3 3 2 3 3 3 ...
 $ terulet    : num  10.1465 2.9406 0.5315 0.0809 0.2529 ...
 $ hatarhossz : num  20.79 9.53 4.8 1.09 2.23 ...
    $ geometry :sfc_MULTIPOLYGON of length 1951; first
     list element: List of 1
  ..$ :List of 1
  .. ..$ : num [1:72, 1:2] 16.9 16.9 16.9 17 17.1 ...
```

Lekérdezés

```
class(nuts_regiok[1, c("terulet", "hatarhossz")])
```

```
[1] "sf"         "data.frame"
```

```
class(nuts_regiok[1, c("terulet", "hatarhossz"), drop =
 FALSE])
```

```
[1] "sf"         "data.frame"
```

```
class(nuts_regiok[1, c("terulet", "hatarhossz"), drop =
 TRUE])
```

```
[1] "list"
```

Lekérdezés

```
str(nuts_regiok[1, c("terulet", "hatarhossz"), drop = TRUE])
```

```
List of 2
 $ terulet   : num 10.1
 $ hatarhossz: num 20.8
```

Használjuk az `unlist()` függvényt, hogy vektort kapjunk!

```
unlist(nuts_regiok[1, c("terulet", "hatarhossz"), drop =
 TRUE])
```

```
   terulet hatarhossz
  10.14648   20.78541
```

Lekérdezés

A `drop = TRUE` egy oszlop esetén vektort ad eredményül:

```
str(nuts_regiok[, "terulet", drop = TRUE])
```

```
 num [1:1951] 10.1465 2.9406 0.5315 0.0809 0.2529 ...
```

Logikai kifejezések

Sorokra (elemekre) lekgyakrabban logikai kifejezéssel szűkítünk.
A logikai kifejezések lényegileg logikai vektorok, amikkel logikai műveleteket végezhetünk.
Leggyakrabban használt műveletek:

- `<`, `<=`, `>`, `>=`: szám, szám $\rightarrow$ logikai
- `==`, `!=`, `%in%`: bármi, bármi $\rightarrow$ logikai
- `is.na()`: bármi $\rightarrow$ logikai
- `!`: logikai $\rightarrow$ logikai
- `&`, `|`, `xor()`: logikai, logikai $\rightarrow$ logikai

Logikai kifejezések

```
logikai_vektor <- c(TRUE, FALSE)
logikai_vektor
```

```
[1]  TRUE FALSE
```

```
logikai_vektor == TRUE
```

```
[1]  TRUE FALSE
```

```
!logikai_vektor
```

```
[1] FALSE  TRUE
```

```
logikai_vektor != TRUE
```

```
[1] FALSE  TRUE
```

Logikai kifejezések

Igazságtábla: `FALSE` (0) vagy `TRUE` (1) értékek kombinálásakor mit ad eredményként a logikai művelet?

AND Truth Table

Inputs		Output
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

Inputs		Output
A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

XOR Truth Table

Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Logikai kifejezések

```
logikai_vektor1 <- c(TRUE, TRUE, FALSE, FALSE)
logikai_vektor2 <- c(TRUE, FALSE, FALSE, TRUE)
logikai_vektor1 & logikai_vektor2
```

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

```
logikai_vektor1 | logikai_vektor2
```

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

```
xor(logikai_vektor1, logikai_vektor2)
```

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

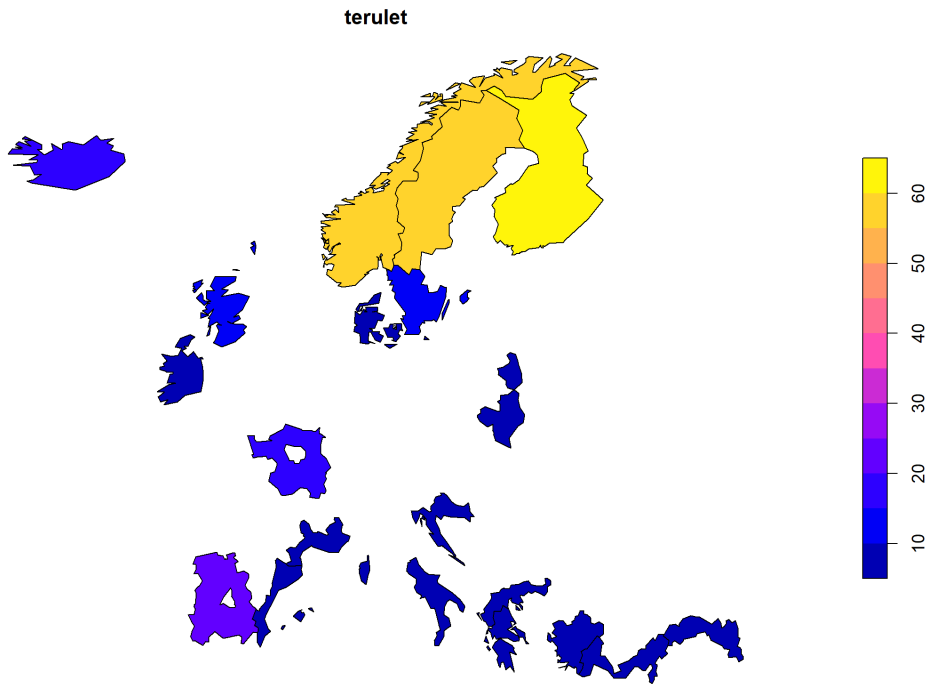
```
!logikai_vektor1 & (logikai_vektor2 | logikai_vektor1)
```

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

Logikai kifejezések

Nem szükséges előbb tulajdonságonként létrehozni a logikai maszkokat, és azokat logikai művelettel összekötni.
De ha úgy átláthatóbbnak érezzük, nyugodtan!

```
plot(nuts_regiok[nuts_regiok$regio_szint == 1 &
 nuts_regiok$hatarhossz > 20, "terulet"])
```



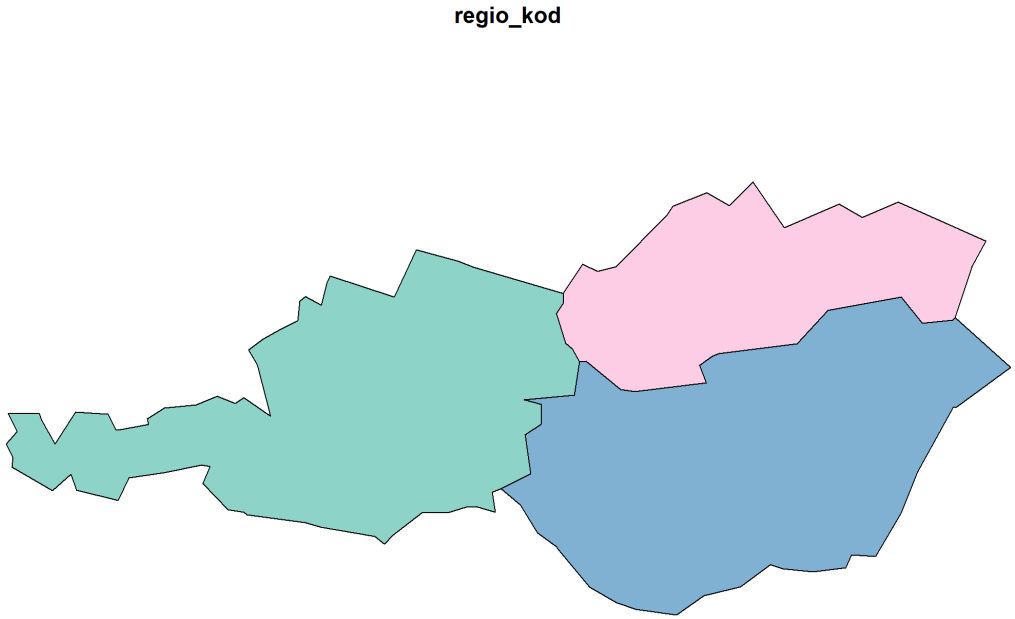
Logikai kifejezések

A műveletek precedencia-sorrendjében az & és a | hátul kullog, ezért nem muszáj zárójelezni.

```
plot(nuts_regiok[nuts_regiok$regio_szint == 2 &
 nuts_regiok$terulet < 2, "hatarhossz"])
```

Logikai kifejezések

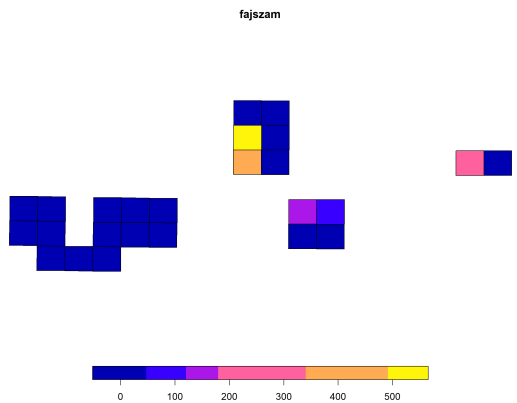
```
plot(nuts_regiok[nuts_regiok$regio_kod %in% c("AT", "HU",
 "SK", "RO") & !(nuts_regiok$terulet > 15), "regio_kod"])
```



2. feladat (órai)

- Töltsd be a “kef_halo.RData” fajlt.
- Jelenítsd meg
  - a felmért fajok számát (“fajszam”)
  - azon négyzetekben, ahol a felmérő (“felmero”) a következő botanikusok valamelyike volt: “Somodi I.”, “Bölöni J.” vagy “Isépy I.”.
- Ezután jelenítsd meg a felmért fajok számát (“fajszam”) azon négyzetekben, ahol
  - vagy ismeretlen a felmérő,
  - vagy nagyon alacsony (100-nál kevesebb) a fajok száma.

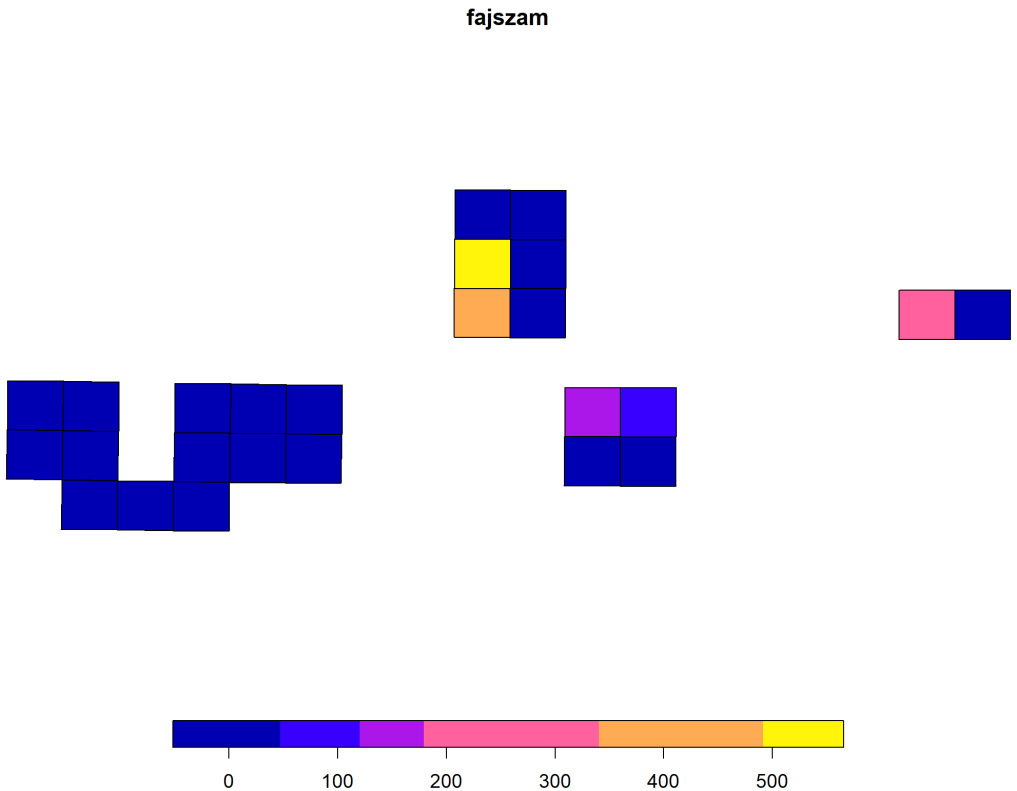
(Logikai vagy, nem pedig kizáró vagy.)



2. feladat (órai) – megoldás

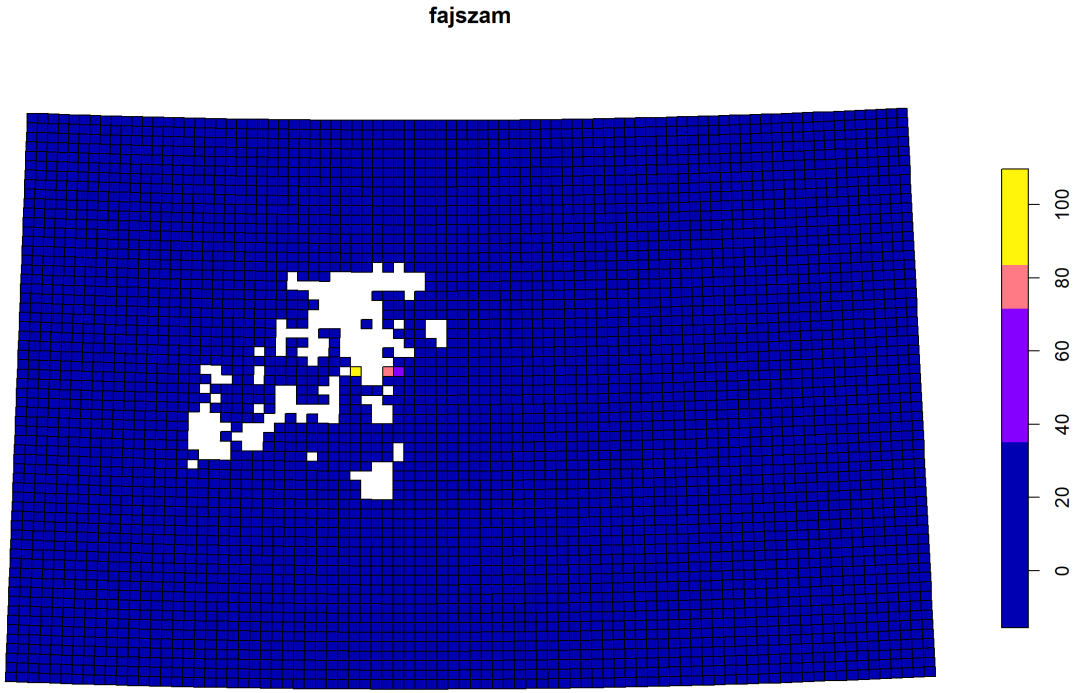
```
load("kef_halo.RData")
```

```
plot(kef_halo[kef_halo$felmero %in% c("Somodi I.", "Bölöni
 J.", "Isépy I."), "fajszam"])
```



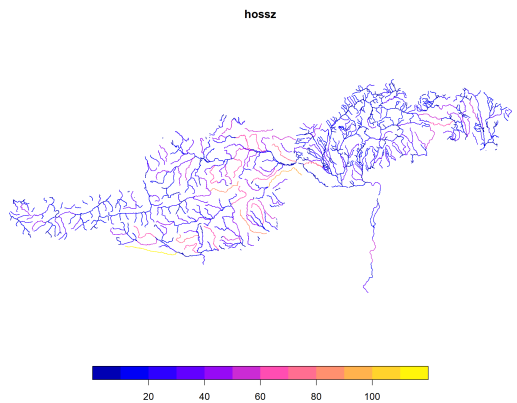
2. feladat (órai) – megoldás

```
plot(kef_halo[is.na(kef_halo$felmero) | kef_halo$fajszam <
 100, "fajszam"])
```



3. feladat (házi)

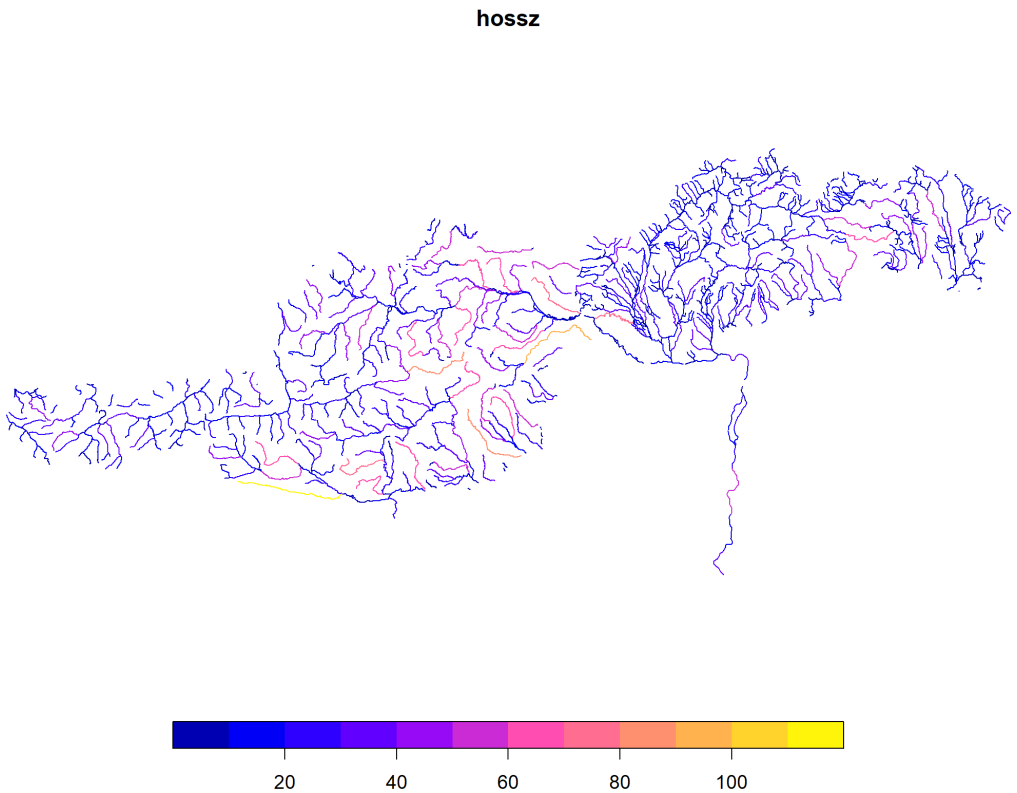
- Töltsd be a “folyok.RData” fájlt.
- Először jelenítsd meg a hosszuk szerint színezve (“hossz” oszlop) azon folyószakaszokat, amelyekre teljesül, hogy
  - Ausztriában (“AT”) vagy Szlovákiában (“SK”) találhatóak (“orszag” oszlop), vagy
  - a Duna 5 km-nél hosszabb szakaszai (“nev” és “hossz” oszlopok).
- Ezután jelenítsd meg kétszeres vonalvastagsággal
  - a kivezető ország (“orszag_ki” oszlop) szerint színezve
  - azon svájci (“orszag” oszlopban “CH” áll) folyószakaszokat,
  - amelyek hossza kisebb 20 km-nél.



3. feladat (házi) – megoldás

```
load("folyok.RData")
```

```
plot(folyok[folyok$orszag %in% c("AT", "SK") | (folyok$nev
 == "Duna" & folyok$hossz > 5), "hossz"])
```



3. feladat (házi) – megoldás

```
plot(folyok[folyok$orszag == "CH" & folyok$hossz < 20,
 "orszag_ki"], lwd = 2)
```

Section 2

Módosítás

Módosítás

A módosítás ugyanúgy történik, mint a `data.frame`-eknél.
(Néhány apró kivételtől eltekintve.)

- `$<-`: oszlopot cserél
- `[i, j]<-`: cellaértékeket cserél

Módosítás

`n` hosszúságú oszlop cserélhető az alábbi hosszúságú vektorokkal:

- `n` (lecserél, akár más típusúra)
- 1 (végig kitölt)
- `n` osztója (ismétel)

Ha az oszlop nem létezik, létrehozza.

Módosítás

Véletlenszámok generálása

`runif(n, min = 0, max = 1)`

- **r**andom numbers from **unif**orm distribution
- egyenletes eloszlásból vesz számokat
- `n`: az előállítandó véletlenszámok darabszáma
- `min`, `max`: a számok ezen tartományon belül lehetnek

Módosítás

```
head(st_drop_geometry(varosok))
```

```
        nev lakossag      terulet hatarhossz
1  Budapest  1861383 0.0134331226 1.47253200
2 Tatabánya    72473 0.0025361967 0.55176661
3  Debrecen   211038 0.0059036668 0.53653104
4  Budapest  1861383 0.0001479875 0.05727080
5  Budapest  1861383 0.0001268366 0.05530916
6       Érd    55859 0.0036080615 0.44043862
```

Módosítás

```
lakossagszam_2020 <- runif(n = nrow(varosok), min =
 200000, max = 2000000)
varosok$lakossag <- lakossagszam_2020
```

```
head(st_drop_geometry(varosok))
```

```
        nev  lakossag      terulet hatarhossz
1  Budapest 1497627.0 0.0134331226 1.47253200
2 Tatabánya 1776391.7 0.0025361967 0.55176661
3  Debrecen 1569768.2 0.0059036668 0.53653104
4  Budapest 1795024.2 0.0001479875 0.05727080
5  Budapest 1021665.7 0.0001268366 0.05530916
6       Érd  499469.2 0.0036080615 0.44043862
```

Módosítás

```
varosok$terulet <- 400
```

```
head(st_drop_geometry(varosok))
```

```
        nev  lakossag terulet hatarhossz
1  Budapest 1497627.0     400 1.47253200
2 Tatabánya 1776391.7     400 0.55176661
3  Debrecen 1569768.2     400 0.53653104
4  Budapest 1795024.2     400 0.05727080
5  Budapest 1021665.7     400 0.05530916
6       Érd  499469.2     400 0.44043862
```

Módosítás

```
varosok$terulet <- c(300, 500, 200, 400, 600)
```

```
head(st_drop_geometry(varosok))
```

```
        nev  lakossag terulet hatarhossz
1  Budapest 1497627.0     300 1.47253200
2 Tatabánya 1776391.7     500 0.55176661
3  Debrecen 1569768.2     200 0.53653104
4  Budapest 1795024.2     400 0.05727080
5  Budapest 1021665.7     600 0.05530916
6       Érd  499469.2     300 0.44043862
```

```
varosok$terulet <- c(300, 500)
```

```
Error in `[[<-.data.frame`(`*tmp*`, i, value = c(300,
 500)): replacement has 2 rows, data has 25
```

2 nem osztója 25-nek...

Módosítás

```
class(varosok$nev)
```

```
[1] "factor"
```

```
varosok$nev <- as.character(varosok$nev)
class(varosok$nev)
```

```
[1] "character"
```

Módosítás

```
eves_koltsegvetes <- runif(n = nrow(varosok), min = 1000,
 max = 30000)
varosok$koltsegvetes <- eves_koltsegvetes
```

```
head(st_drop_geometry(varosok))
```

```
        nev  lakossag terulet hatarhossz koltsegvetes
1  Budapest 1497627.0     300 1.47253200    21906.213
2 Tatabánya 1776391.7     500 0.55176661    26397.423
3  Debrecen 1569768.2     200 0.53653104    23068.488
4  Budapest 1795024.2     400 0.05727080    26697.612
5  Budapest 1021665.7     600 0.05530916    14237.948
6       Érd  499469.2     300 0.44043862     5824.782
```

Módosítás

Egy vagy több cellaérték módosítása:

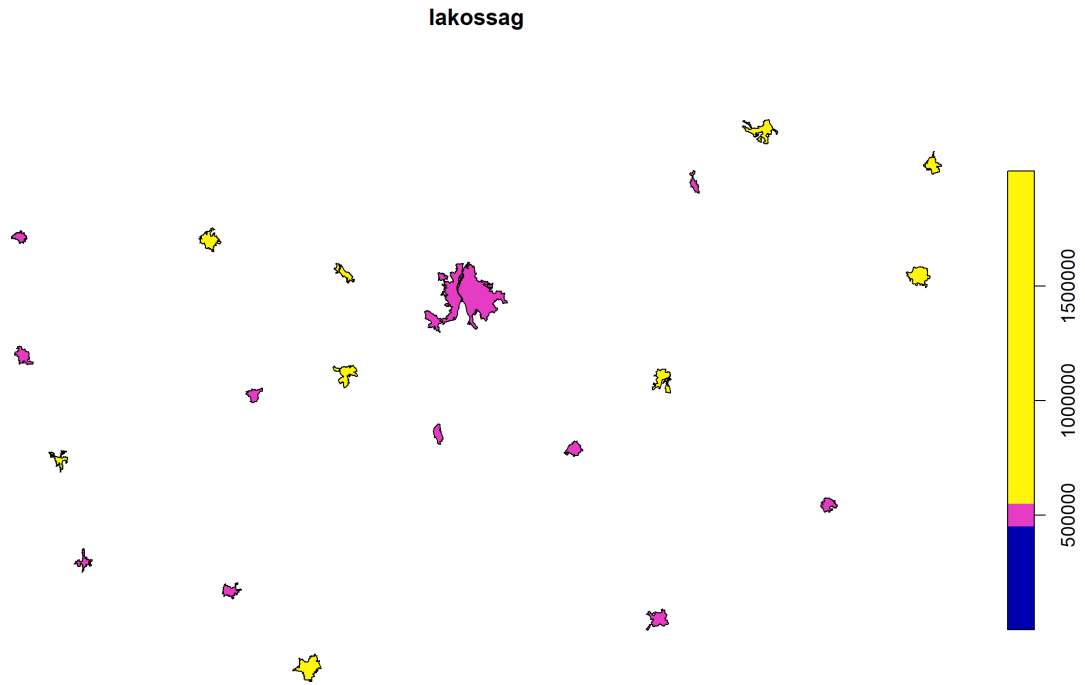
```
varosok[3:4, "terulet"] <- 700
```

```
head(st_drop_geometry(varosok))
```

```
        nev  lakossag terulet hatarhossz koltsegvetes
1  Budapest 1497627.0     300 1.47253200    21906.213
2 Tatabánya 1776391.7     500 0.55176661    26397.423
3  Debrecen 1569768.2     700 0.53653104    23068.488
4  Budapest 1795024.2     700 0.05727080    26697.612
5  Budapest 1021665.7     600 0.05530916    14237.948
6       Érd  499469.2     300 0.44043862     5824.782
```

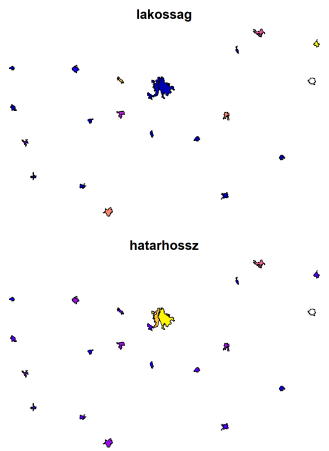
Módosítás

```
varosok[varosok$nev == "Budapest" | varosok$hatarhossz <
 0.5, "lakossag"] <- 500000
plot(varosok[, "lakossag"], breaks = c(0, 450000, 550000,
 2000000))
```



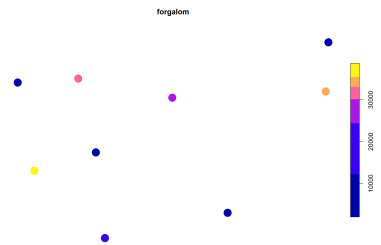
Módosítás

```
varosok[varosok$nev == "Debrecen", c("lakossag",
 "hatarhossz")] <- NA
plot(varosok[, c("lakossag", "hatarhossz")])
```



4. feladat (órai)

- Olvasd be a “repterek.RData” fájlt.
- Hozz létre egy “forgalom” nevű új oszlopot, amelyben éppen annyi, 2 ezer és 40 ezer közötti – egyenletes eloszlásból származó – véletlen szám szerepel, ahány repterünk van.
- Módosítsd a forgalom értékét 5000-re ott, ahol az azonosító “UNK” (unknown), vagy a reptér neve (“nev” oszlop) Ferihegy.
- Ellenőrzésképpen jelenítsd meg a reptereket a forgalom szerint színezve, háromszoros méretű, kitöltött kör (16-os pontjel) szimbólummal, az eloszlásfüggvény alapján vágva a színskálát.
- Bónuszfeladat: e második (5000-re történő) forgalommódosítást végezd el úgy, hogy a $ operátort használod.

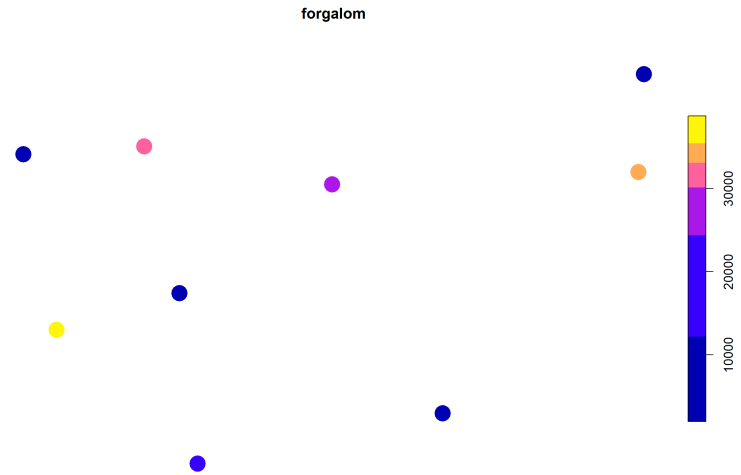


4. feladat (órai) – megoldás

```
load("repterek.RData")
repterek$forgalom <- runif(n = nrow(repterek), min = 2000,
 max = 40000)
repterek[repterek$azonosito == "UNK" | repterek$nev ==
 "Feriegy", "forgalom"] <- 5000
```

4. feladat (órai) – megoldás

```
plot(repterek[, "forgalom"], breaks = "quantile", pch =
 16, cex = 3)
```



Bónuszfeladat megoldása:

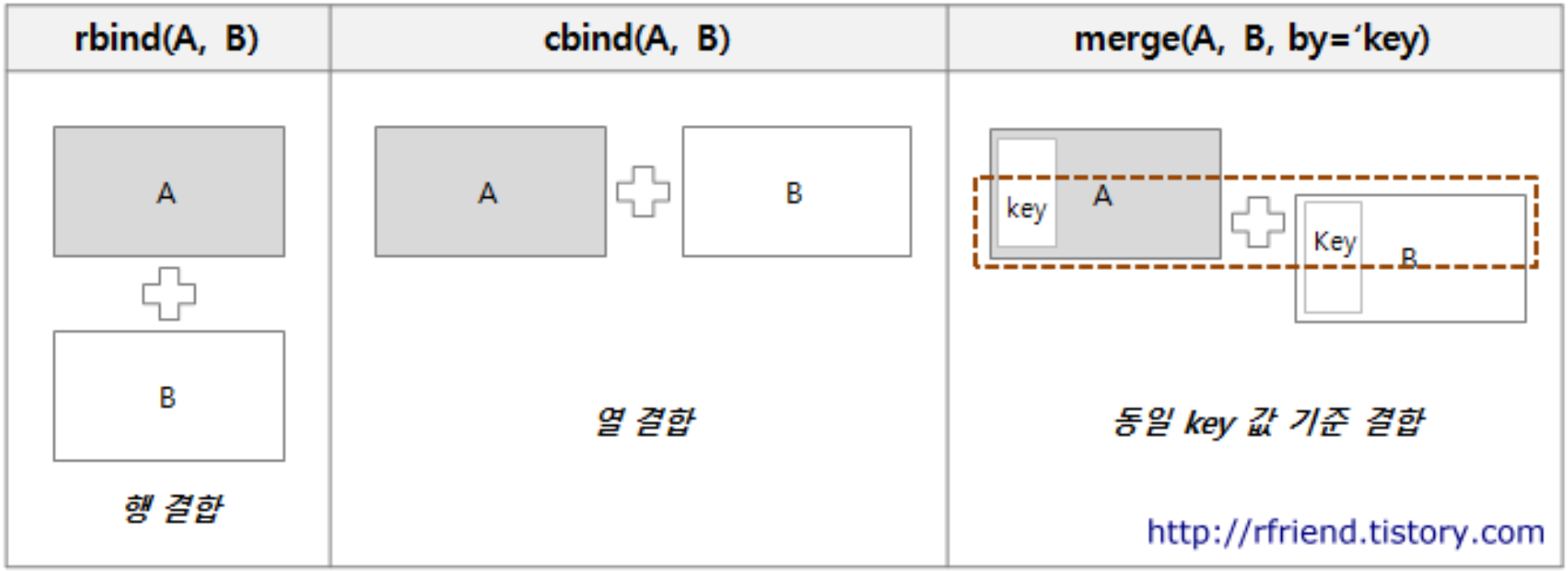
```
repterek$forgalom[repterek$azonosito == "UNK" |
 repterek$nev == "Feriegy"] <- 5000
```

Section 3

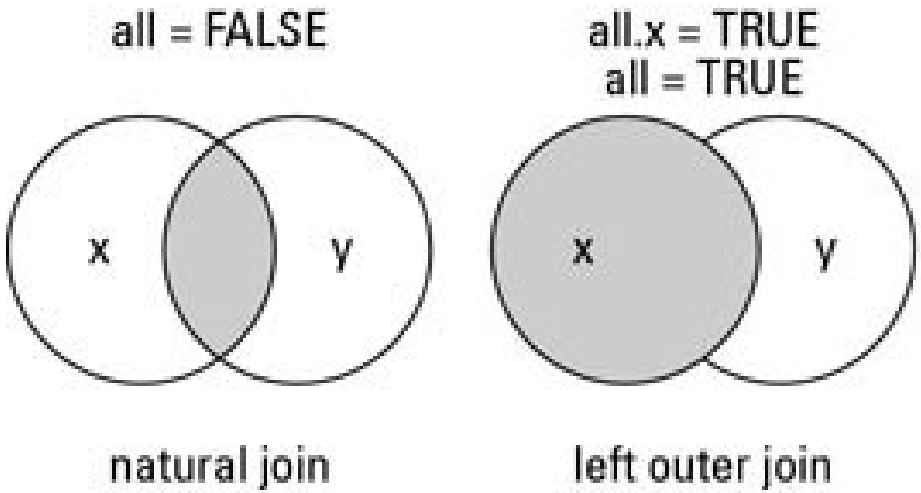
Összefűzés

Összefűzés

A `merge()` függvénnyel két `data.frame`-et, vagy egy `sf`-et és egy `data.frame`-et lehet összefűzni megadott kulcsoszlop(ok) alapján.



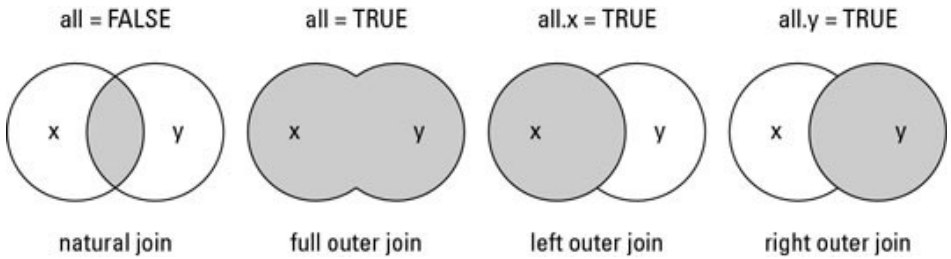
Összefűzés



`merge(x, y, by, all = FALSE)`

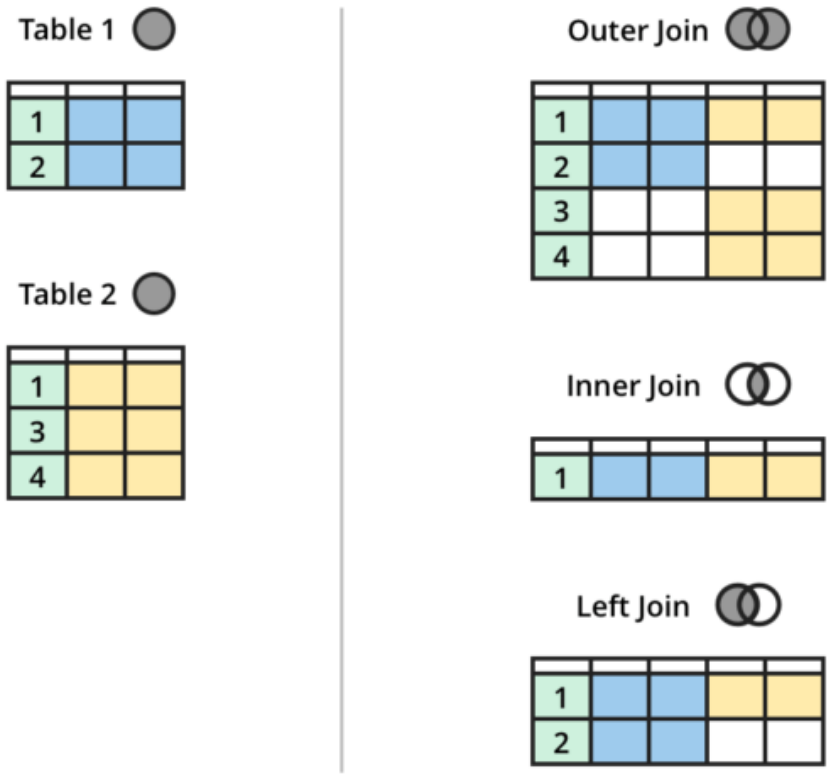
- `x`: a Simple Features
- `y`: a `data.frame`
- `by`: a kulcsoszlop(ok) neve
- vagy `by.x`, `by.y`: a kulcsoszlop(ok) neve, ha eltér a két adatsorban
- `all` (vagy `all.x`): a Simple Features sorait megtartsa ott is, ahol a `data.frame`-ben nem talált hozzáfűzhető adatokat?

Összefűzés



Két `data.frame` összefűzése esetén minden kombináció elképzelhető (ezért van külön `all.x` és `all.y` paraméter).
Két `sf` nem fűzhető össze, mert az eredmény csak az egyiknek a geometriáját örökölheti.
Egy `sf` és egy - akár `sf`-ből képzett - `data.frame` viszont összefűzhető.
Értelemszerűen ekkor az `all.y` paraméter nem használható.

Összefűzés



Az `all = FALSE` esetet inner/natural joinnak nevezzük, az `all = TRUE` esetet pedig left joinnak.

Összefűzés

```
load("kozeptajak.RData")
```

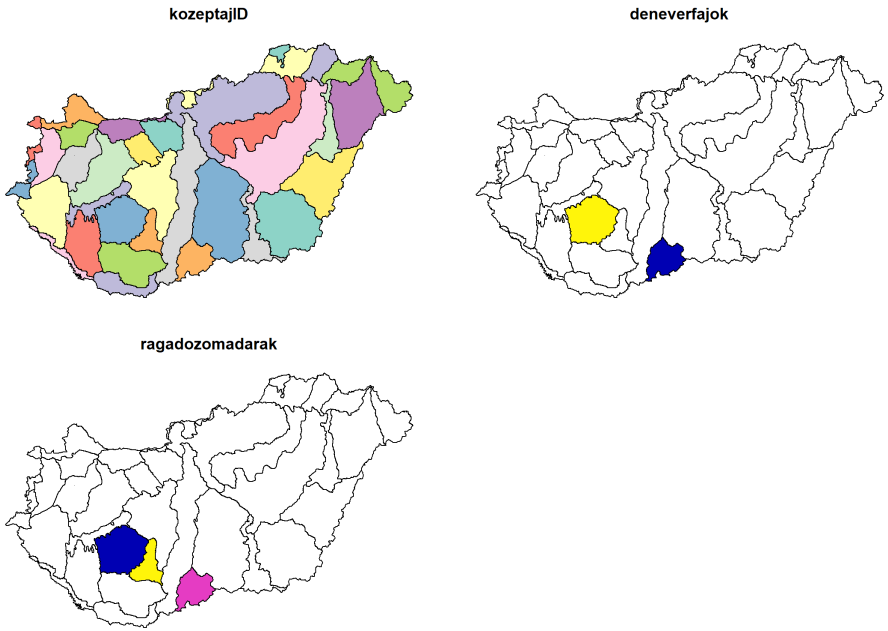
```
allatszamlalasi_adatok <- data.frame(kozeptajID =
 c("C.3.3.6", "C.3.1.4", "C.3.1.5"), deneverfajok = c(4,
 6, NA), ragadozomadarak = c(5, 3, 6), stringsAsFactors =
 FALSE)
```

```
allatszamlalasi_adatok
```

```
  kozeptajID deneverfajok ragadozomadarak
1    C.3.3.6             4                5
2    C.3.1.4             6                3
3    C.3.1.5            NA                6
```

Összefűzés

```
plot(merge(x = kozeptajak, y = allatszamlalasi_adatok, by
 = "kozeptajID", all = TRUE))
```

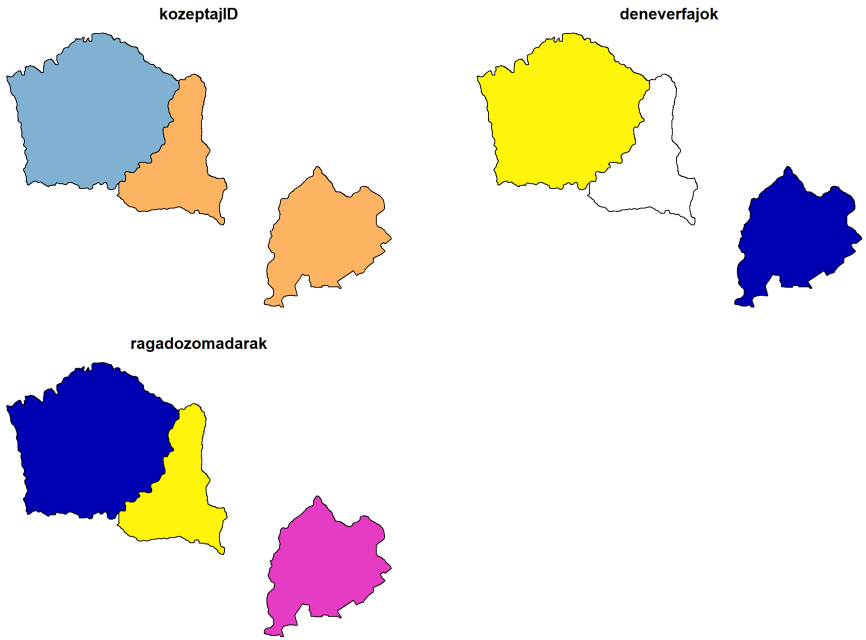


Most már három tulajdonságoszlopa van.
Az `all = TRUE` esetben az összes eredeti sor (poligon) megmarad, ahol szükséges, `NA` kerül az új oszlopba.

Összefűzés

Az `all = FALSE` esetben szűkül(het) az adatsor.

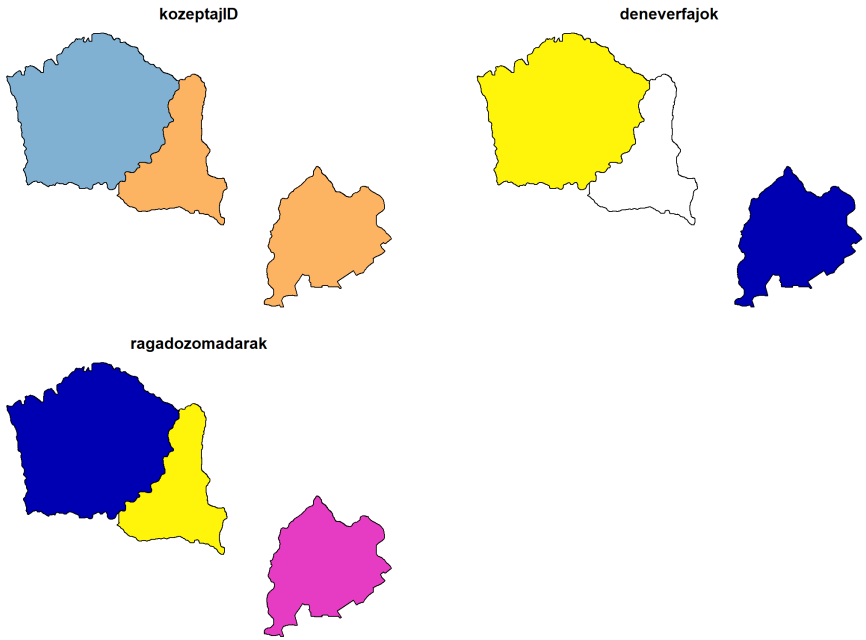
```
plot(merge(x = kozeptajak, y = allatszamlalasi_adatok, by
 = "kozeptajID", all = FALSE))
```



Összefűzés

Az `all` alapértelmezetten `FALSE`, vagyis csak azt tartja meg, ami mindkét adatsorban szerepelt.

```
plot(merge(x = kozeptajak, y = allatszamlalasi_adatok, by
 = "kozeptajID"))
```



Összefűzés

```
load("orszag_eu.RData")
```

```
colnames(orszag_eu)
```

```
[1] "orszagkod"  "terulet"    "hatarhossz" "geometry"
```

```
nrow(orszag_eu)
```

```
[1] 35
```

```
fovarosok <- data.frame(ketbetus_kod = c("HU", "AT", "SK",
 "HR", "RU", "RO"), fovaros = c("Budapest", "Becs",
 "Pozsony", "Zagrab", "Moszkva", "Bukarest"),
 stringsAsFactors = FALSE)
```

Összefűzés

```
fovarosok
```

```
  ketbetus_kod  fovaros
1           HU Budapest
2           AT     Becs
3           SK  Pozsony
4           HR   Zagrab
5           RU  Moszkva
6           RO Bukarest
```

```
nrow(fovarosok)
```

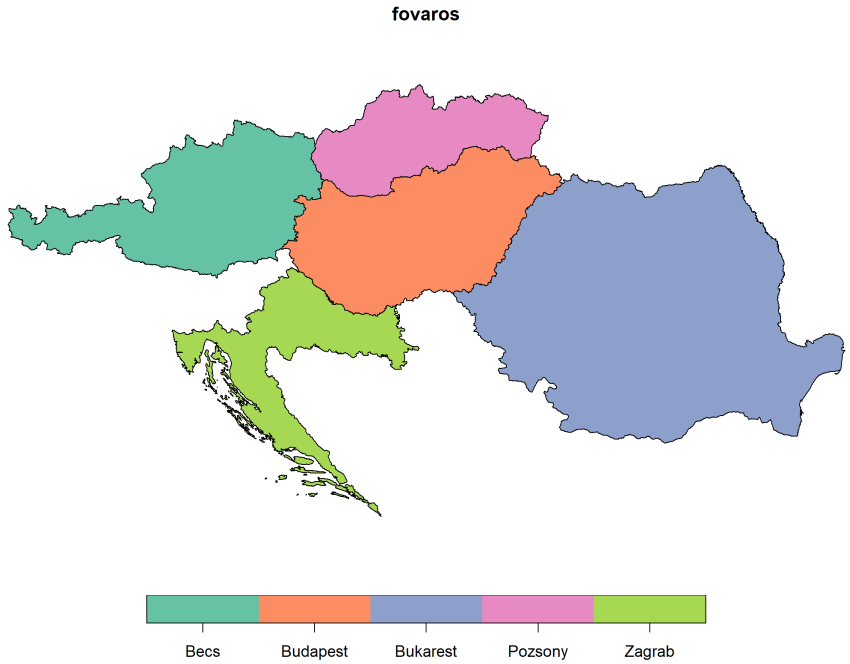
```
[1] 6
```

```
orszag_eu_fovarossal <- merge(x = orszag_eu, y =
 fovarosok, by.x = "orszagkod", by.y = "ketbetus_kod", all
 = FALSE)
nrow(orszag_eu_fovarossal)
```

```
[1] 5
```

Összefűzés

```
plot(orszag_eu_fovarossal[, "fovaros"])
```



5. feladat (órai)

- Olvasd be a következő két fájlt: “eghajlat.RData”, “viztavolsag_es_domborzat.RData”.
- Írd ki a képernyőre az egyiknek, majd a másiknak az oszlopneveit.
- Hozz létre egy “uj_oszlopok” nevű szövegvektort, amely a viztavolsag_es_domborzat nevű `data.frame` azon oszlopneveit tartalmazza, amelyek nem szerepelnek az eghajlat nevű Simple Features oszlopnevei között.
- Fűzd az éghajlati adatokhoz a `data.frame` adatait a “meta_id” kulcsoszlopot használva úgy, hogy minden olyan pont megmaradjon, amihez éghajlati adat rendelkezésre állt.
- Jelenítsd meg a pontokat a három új oszlop szerint színezve (3 térkép, egy lépésben).

5. feladat (órai) – megoldás

```
load("eghajlat.RData")
load("viztavolsag_es_domborzat.RData")
```

```
colnames(eghajlat)
```

```
 [1] "meta_id"  "P_DJF"    "P_MAM"    "P_JJA"
 [5] "P_SON"    "T_DJF"    "T_MAM"    "T_JJA"
 [9] "T_SON"    "geometry"
```

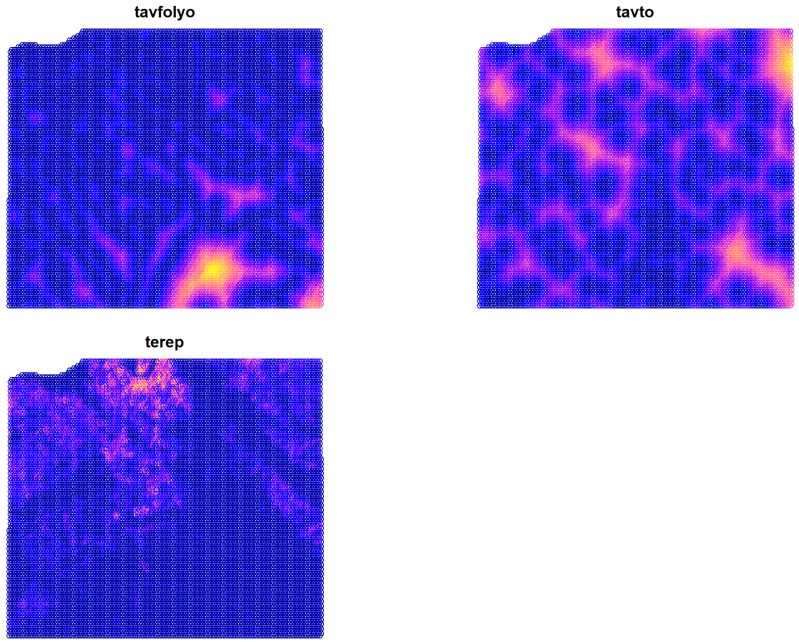
```
colnames(viztavolsag_es_domborzat)
```

```
[1] "meta_id"  "tavfolyo" "tavto"    "terep"
```

```
uj_oszlopok <- colnames(viztavolsag_es_domborzat)
 [!(colnames(viztavolsag_es_domborzat) %in%
 colnames(eghajlat))]
eghajlat_bovitett <- merge(x = eghajlat, y =
 viztavolsag_es_domborzat, by = "meta_id", all = TRUE)
```

5. feladat (órai) – megoldás

```
plot(eghajlat_bovitett[, uj_oszlopok])
```



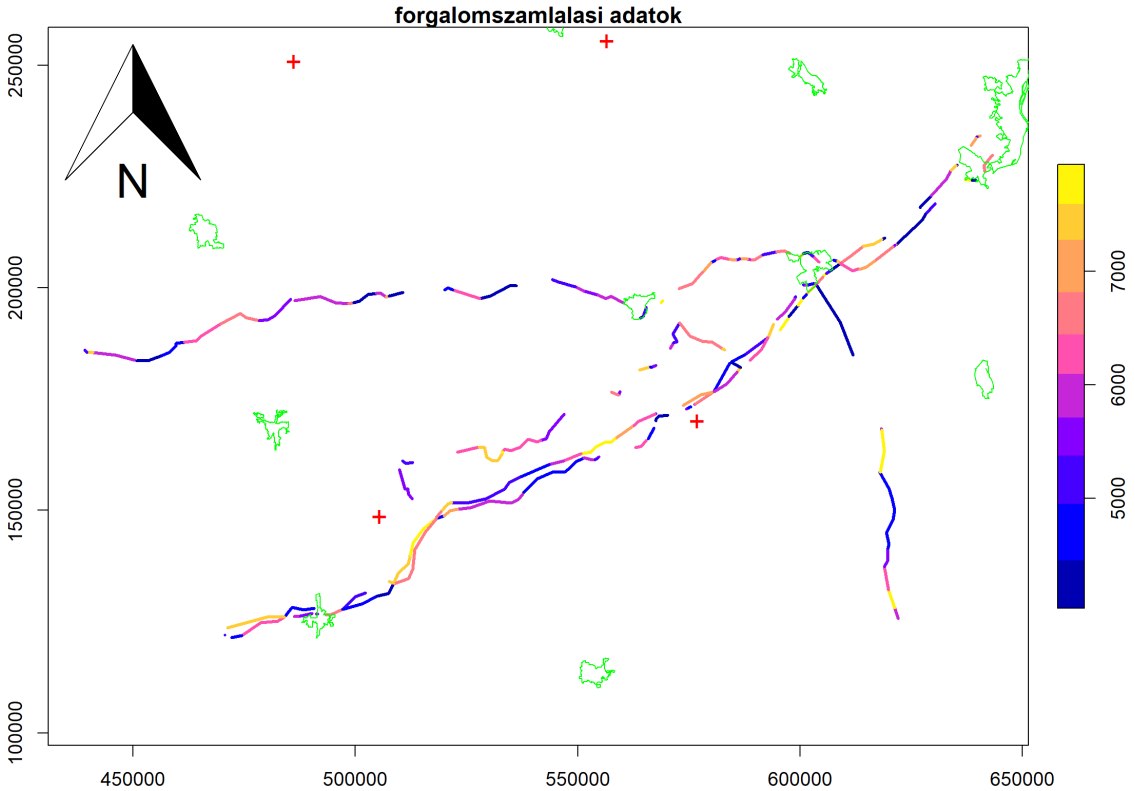
6. (összefoglaló) feladat (házi)

- Olvasd be a `prettymapr` csomagot és az “utak.RData” fájlt.
- Alakítsd át a “nev” oszlopot szöveg (`character`) típusúvá.
- Adj hozzá egy új oszlopot “forgalom” néven, amely 3000 és 8000 közötti, megfelelő számú véletlen értékeket tartalmaz.
- Ábrázold az “M7”, “7”, “8”, “63” és “71” nevű utak közül azok forgalmát 10 osztatú Jenks-féle skálán, amelyek forgalma 4000-nél nagyobb.
  - A jelmagyarázat jobbra kerüljön, a címfelirat legyen “forgalomszamlalasi adatok”, az utak vastagságát az alapértelmezett háromszorosára növeld.
  - Lásd el koordináta-feliratokkal is az ábrát.

...

6. (összefoglaló) feladat (házi)

- Add hozzá a reptereket (geometriát) másfeles méretű, piros pluszjellel.
- Add hozzá a városok körvonalát zölddel.
- Helyezz el az ábrán egy kétszeres méretű északjelet a bal felső sarokba.

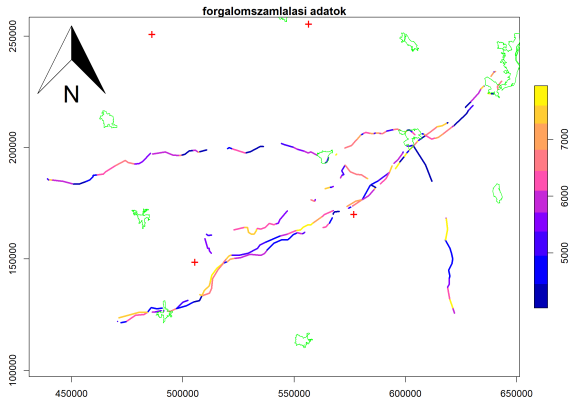


6. (összefoglaló) feladat (házi) – megoldás

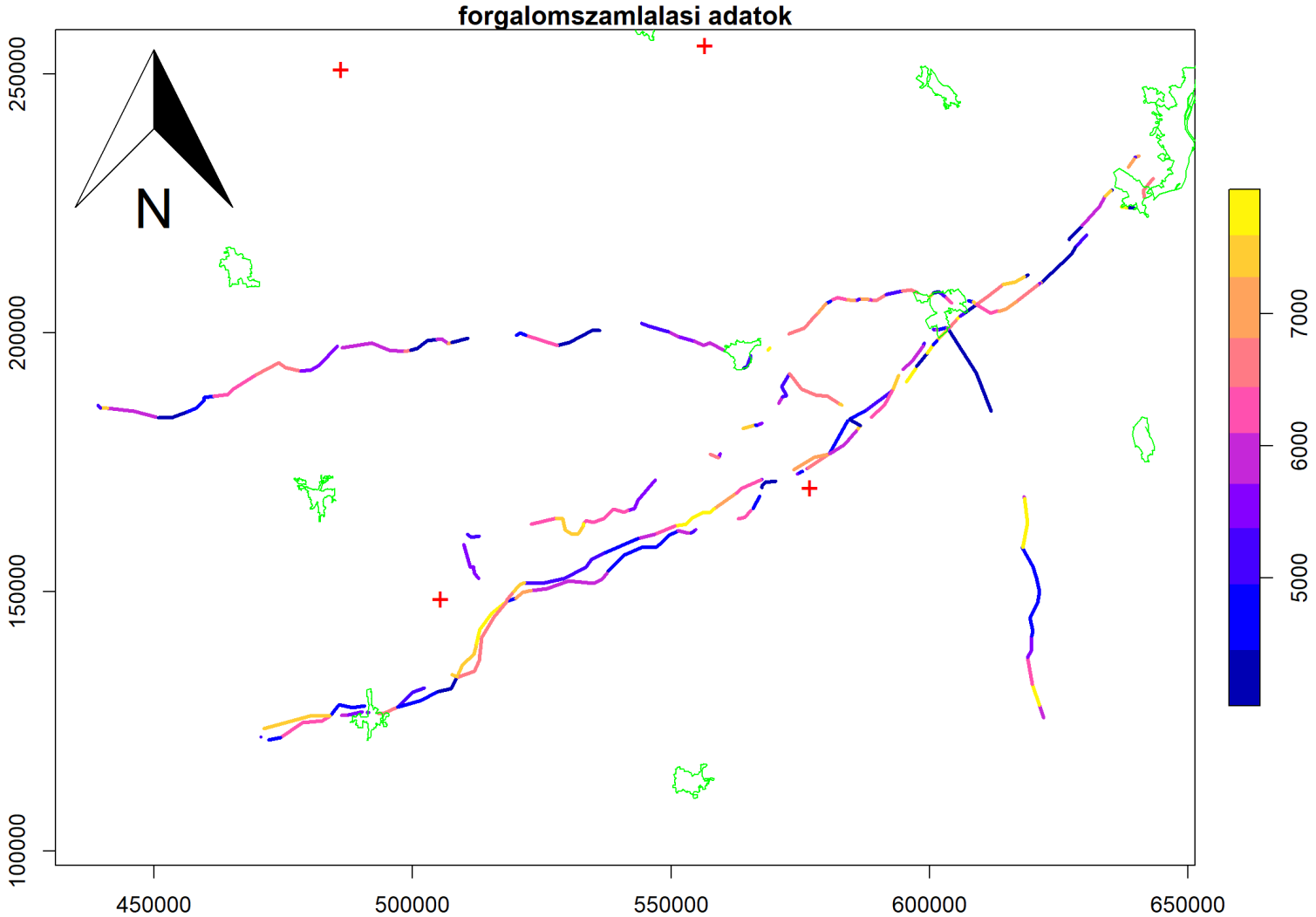
```
library(prettymapr)
load("utak.RData")
utak$nev <- as.character(utak$nev)
utak$forgalom <- runif(n = nrow(utak), min = 3000, max =
 8000)
```

6. (összefoglaló) feladat (házi) – megoldás

```
plot(utak[utak$nev %in% c("M7", "7", "8", "63", "71") &
 utak$forgalom > 4000, "forgalom"], main =
 "forgalomszamlalasi adatok", axes = TRUE, lwd = 3,
 nbreaks = 10, breaks = "jenks", key.pos = 4, reset =
 FALSE)
plot(st_geometry(repterek), cex = 1.5, pch = "+", col =
 "red", add = TRUE)
plot(st_geometry(varosok), border = "green", add = TRUE)
addnortharrow(pos = "topleft", scale = 2)
```



6. (összefoglaló) feladat (házi) – megoldás



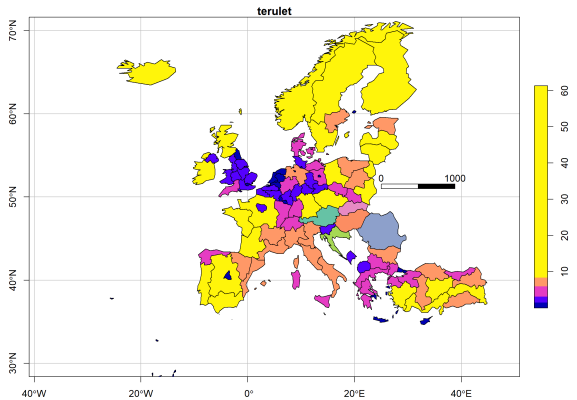
7. (összefoglaló) feladat (házi)

- Olvasd be a raster csomagot.
- A “fovarosok” nevű `data.frame`-et fűzd hozzá a “nuts_regiok” nevű Simple Featureshöz.
  - Vigyázz, a kulcsoszlop neve különbözik a két adattáblában!
  - Az összes régió maradjon meg az összefűzés után.
- Azon régiók fővárosa, amelyek szintje (“regio_szint” oszlop) nem 0 (vagyis nem országok, hanem pl. megyék), legyen “nincsen”.
- Vigyázz, megbújik az adatsorban egy régió, aminek nem ismert a szintje, ezt megfelelő módon kezelned kell, amúgy hibát kapsz...

...

7. (összefoglaló) feladat (házi)

- Ábrázold az első szintű NUTS-régiókat a terület szerint színezve úgy, hogy az 5 színkategória mindegyikébe azonos számú régió essen.
  - Az ábrázolt terület a 20°Ny és 30°K hosszúságok, valamint 30°É és 70°É szélességek közé essen.
  - A tengelyfeliratok legyenek “hosszusag” és “szelesseg”, továbbá koordináta-feliratokat és rácshálót is rakj az ábrára.
  - Ugyanezen térképen ábrázold a főváros szerint színezve azon 0. szintű régiókat, amelyek fővárosa nem ismeretlen.
- Helyezz a térképre egy 1000 egység (km) hosszú léptékrudat a 25°K és 51°É pontba.



7. (összefoglaló) feladat (házi) – megoldás

```
library(raster)
```

```
nuts_regiok <- merge(x = nuts_regiok, y = fovarosok, by.x
 = "regio_kod", by.y = "ketbetus_kod", all = TRUE)
```

```
nuts_regiok[nuts_regiok$regio_szint != 0, "fovaros"] <-
 "nincsen"
```

```
Error in `[<-.data.frame`(`*tmp*`, nuts_regiok$regio_szint
 != 0, "fovaros", : missing values are not allowed in
 subscripted assignments of data frames
```

Upsz, a `nuts_regiok_fovarossal$regio_szint != 0` logikai kifejezés `NA`-t tartalmazott, mivel a regio_szint oszlopban volt `NA` érték.

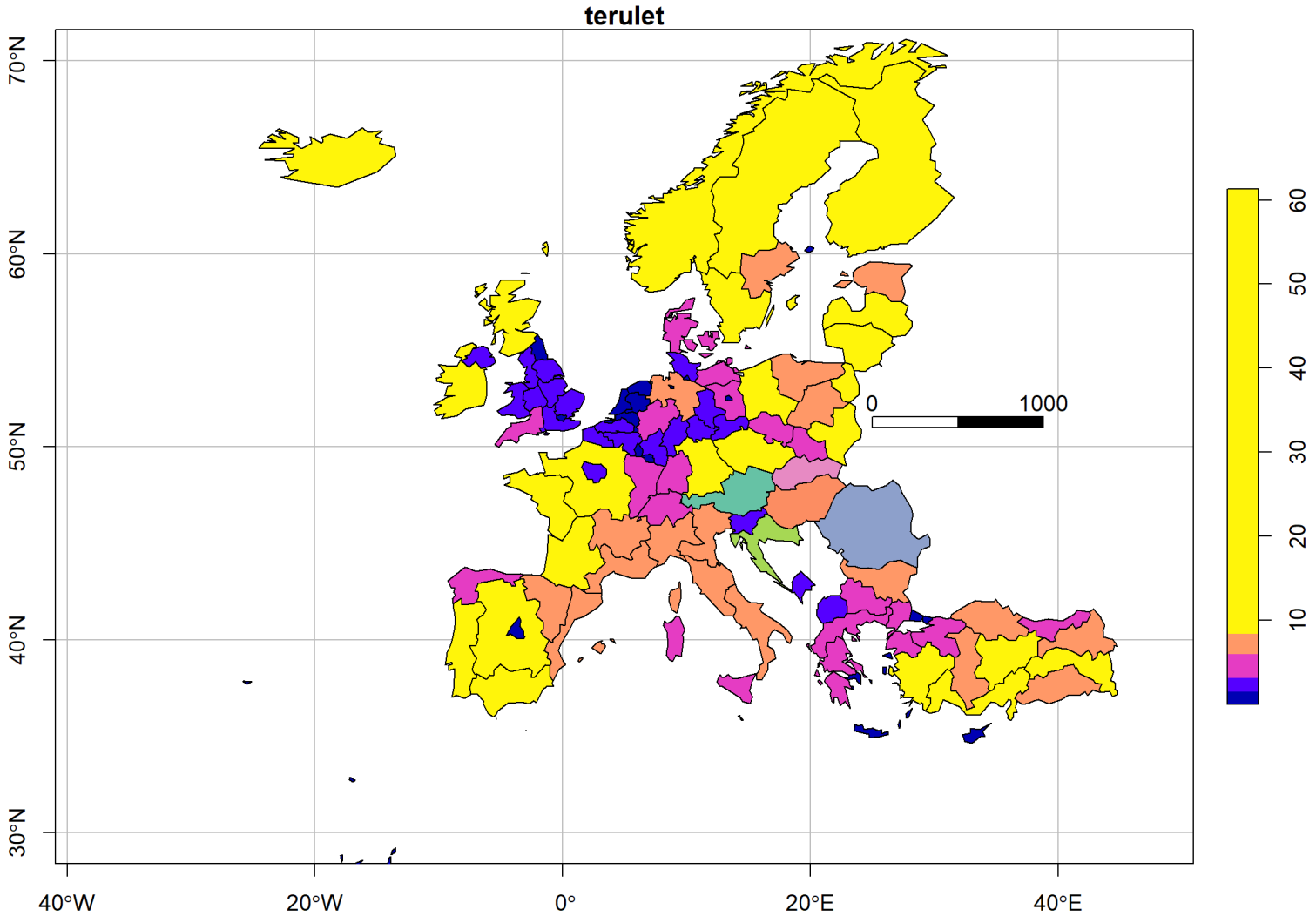
7. (összefoglaló) feladat (házi) – megoldás

Ezt ki kell küszöbölni:

```
nuts_regiok_fovarossal[nuts_regiok_fovarossal$regio_szint
 != 0 & !is.na(nuts_regiok_fovarossal$regio_szint),
 "fovaros"] <- "nincsen"
```

```
plot(nuts_regiok_fovarossal[nuts_regiok_fovarossal$regio_szint
 == 1, "terulet"], nbreaks = 5, breaks = "quantile", xlim
 = c(-20, 30), ylim = c(30, 70), xlab = "hosszusag", ylab
 = "szelesseg", axes = TRUE, graticule = TRUE, reset =
 FALSE)
plot(nuts_regiok_fovarossal[!is.na(nuts_regiok_fovarossal$fovaros)
 & nuts_regiok_fovarossal$regio_szint == 0, "fovaros"],
 add = TRUE)
scalebar(d = 1000, xy = c(25, 51), type = "bar")
```

7. (összefoglaló) feladat (házi) – megoldás

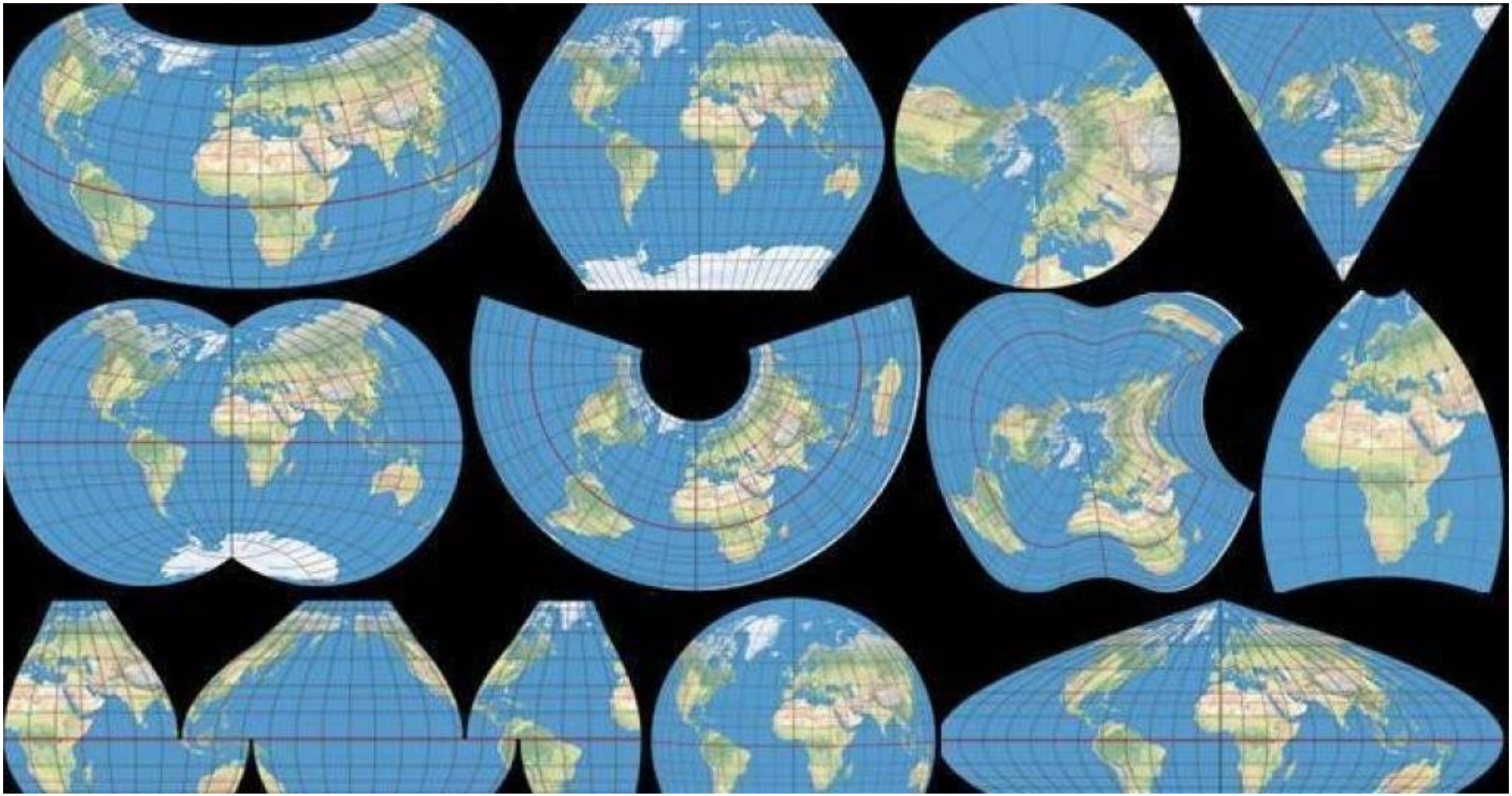


Section 4

Vetületek, transzformációk

Vetületek

Egy kis elméleti rész jön, amin gyorsan végigszaladunk...



Vetületek

A térinformaikai adatoknak (beleértve a rasztereket is) vannak

- koordinátái, és
- koordináta-rendszere.

Koordináta:

- minden rasztercellának, pontnak, vonaltöréspontnak, poligonsarokpontnak van koordinátája,
- 2 (vagy több).

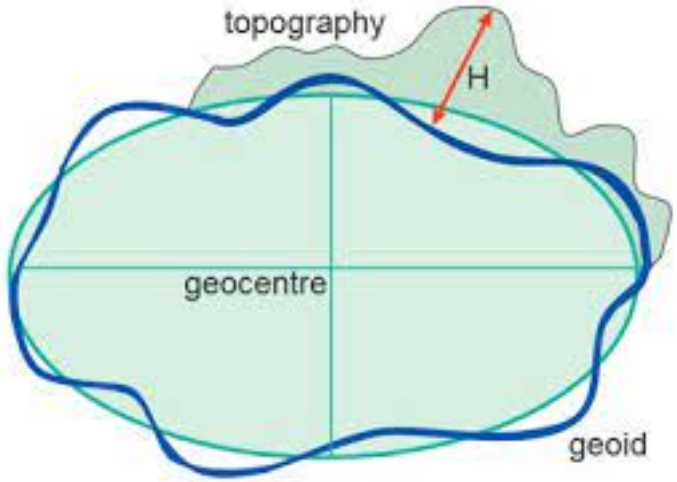
Koordináta-rendszer:

- az, amelyben a koordináták értelmezhetőek
- több típusa létezik, többféle névvel illetik
- én pongyolán lehet, hogy vetületnek fogom mondani azt is, ami valójában nem az (pl. WGS-84)

Vetületek

A Föld felszínén lévő pont helyének egyértelmű meghatározása több, egymásra épülő, közelítő lépésből áll:

- felszín $\rightarrow$ geoidmodell (gravitáció)
- geoid $\rightarrow$ ellipszoid (dátum)
- dátum $\rightarrow$ sík (térképi vetület)



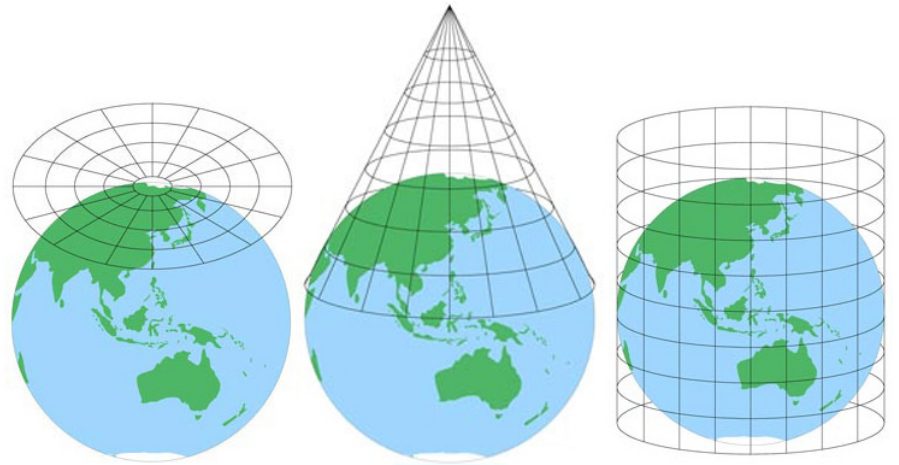
Vetületek

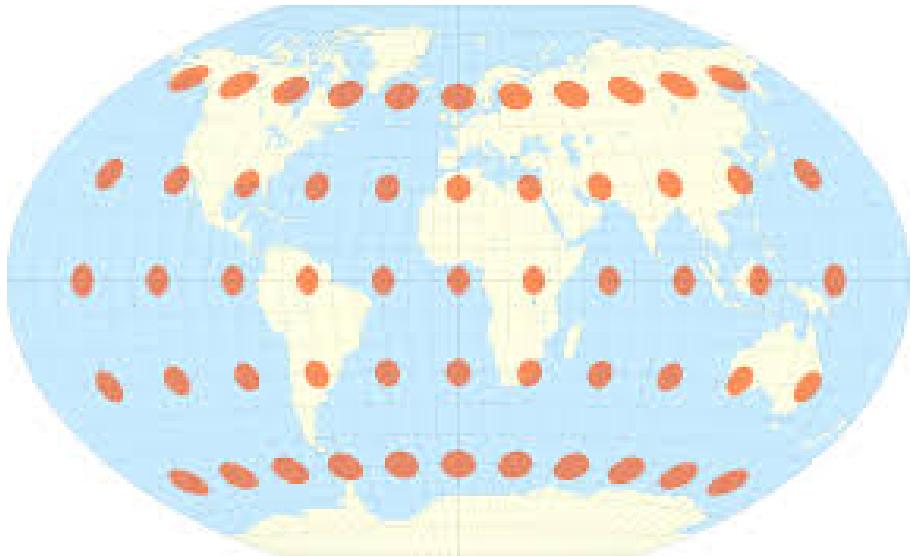
Ha van dátum, akkor már

- ábrázolható,
- van koordináta-rendszere (geographic coordinate system, spatial reference system, **c**oordinate **r**eference **s**ystem, CRS).

Vetület

- projection, projected coordinate system
- számos típusa van: pl. sík/henger/kúp
- szögtartó/távolságtartó/ területtartó (Tissot)





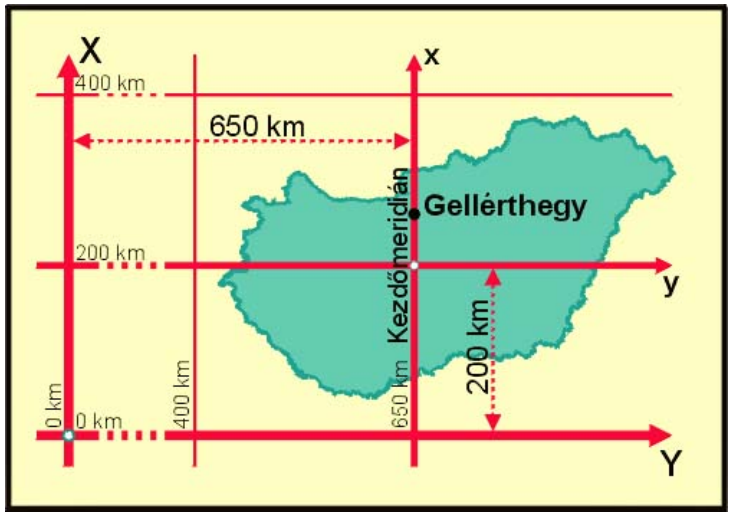
Vetületek

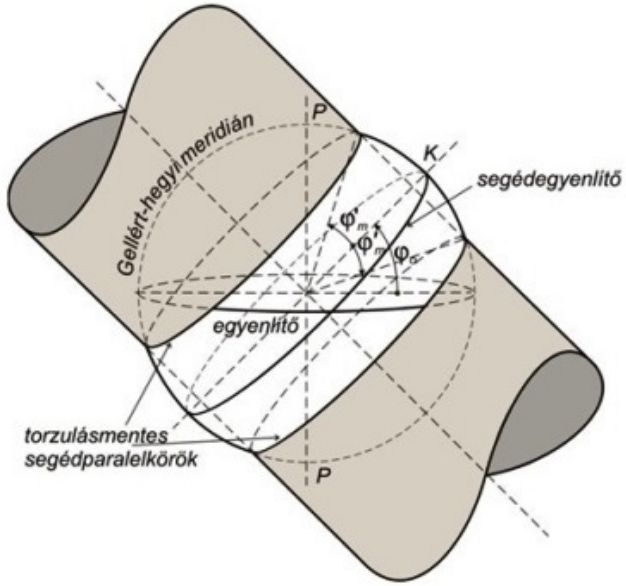
WGS-84

- dátum és koordinátarendszer egyben
- **W**orld **G**eodetic **S**ystem
- GPS ezt használja
- szélesség (É) és hosszúság (K)

EOV

- igazi vetület
- **E**gységes **O**rszágos **V**etület, Hungarian Datum 1972 (HD72)
- ferdetengelyű, szögtartó, süllyesztett hengervetület
- X koordináta az északi (<400e), Y koordináta a keleti (>400e)
- mértékegysége: méter





Vetületek

A koordináta-rendszerek kezelése R-ben a PROJ nevű geodéziai/térinformatikai könyvtár alapján történik.

- multiplatform
- proj.org/usage/projections.html
- minden koordináta-rendszernek pontos szöveges leírása van
- a formátuma régebben PROJ.4 string, ma WKT (mindkettő bonyolult)

De minden koordináta-rendszernek van egy azonosítója (sorszáma) is

- ezt sokkal egyszerűbb használni/megjegyezni
- EOV: 23700, WGS-84: 4326, LAEA Europe: 3035, Pseudo-Mercator/Web Mercator: 3857
- **E**uropean **P**etroleum **S**urvey **G**roup (EPSG)
- spatialreference.org/ref/epsg/

Vetületek

Fontos: az elmúlt években nagyon sokat változott a vetületek kezelése R-ben.
A mára elavult PROJ.4 támogatása megszűnt, a korábban mentett `sf`-változók beolvasásakor figyelmeztetést kaphatunk.
Az `sp` és a `raster` szintén dobhat figyelmeztetéseket.
Azeket általában minden gond nélkül ignorálhatjuk, kivéve, ha nagy pontosságú vetületi transzformációkat szeretnénk végezni.
Részletek: itt.

Vetületek kiolvasása és módosítása

```
load("magyarorszag.RData")
```

```
magyarorszag
```

```
Geometry set for 1 feature
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box: xmin: 16.11385 ymin: 45.73705 xmax: 22.89627
 ymax: 48.58523
Geodetic CRS:  WGS 84

MULTIPOLYGON (((21.44006 48.58523, 21.45073 48....
```

Vetületek kiolvasása és módosítása

sf/sfc vetületének kiolvasása: st_crs(x)

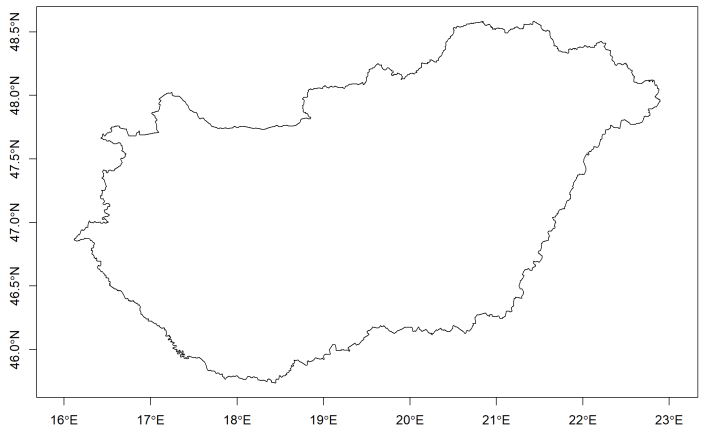
```
st_crs(magyarorszag)
```

```
Coordinate Reference System:
  User input: EPSG:4326
  wkt:
GEOGCRS["WGS 84",
    ENSEMBLE["World Geodetic System 1984 ensemble",
        MEMBER["World Geodetic System 1984 (Transit)"],
        MEMBER["World Geodetic System 1984 (G730)"],
        MEMBER["World Geodetic System 1984 (G873)"],
        MEMBER["World Geodetic System 1984 (G1150)"],
        MEMBER["World Geodetic System 1984 (G1674)"],
        MEMBER["World Geodetic System 1984 (G1762)"],
        MEMBER["World Geodetic System 1984 (G2139)"],
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]],
        ENSEMBLEACCURACY[2.0]],
    PRIMEM["Greenwich",0,
```

Vetületek kiolvasása és módosítása

Próbáljuk ábrázolni egy térképen Magyarországot és a városait!

```
plot(magyarorszag, axes = TRUE)
plot(st_geometry(varosok), add = TRUE)
```



Hol vannak a városok?

Vetületek kiolvasása és módosítása

Nem sikerült, mert a városok vetülete más, ezért a koordinátái teljesen más tartományban vannak.
(Ha a tartományban lenne átfedés, akkor látnánk a városokat, de teljesen rossz helyen.)

```
st_crs(magyarorszag)$epsg
```

```
[1] 4326
```

```
st_crs(varosok)$epsg
```

```
[1] 23700
```

A $epsg segítségével lehetőségünk van a részletes leírás helyett csak az EPSG-kódot lekérni.

Vetületek kiolvasása és módosítása

Első – logikusnak tűnő – megoldási kísérlet: módosítsuk a vetületet!
`sf`/`sfc` vetületének módosítása kétféle módon történhet:

- `st_crs(x) <- value`
- `st_set_crs(x, value)`
- előbbi helyben változtat, utóbbi visszaad eredményként egy módosított vetületű objektumot (csőszintaxishoz megfelelő)

Figyelem! Ezeket a függvényeket csak akkor használjuk, ha valamiért hiányzott (`NA`) a vetületi információ, vagy téves volt!
(Vagyis ha a koordináták és a koordináta-rendszer nem volt összhangban.)

Vetületek kiolvasása és módosítása

Szerencsére figyelmeztet is, hogy valószínűleg tévedésből módosítottuk a vetületet:

```
st_crs(varosok) <- 4326
```

```
Warning: st_crs<- : replacing crs does not reproject
data; use st_transform for that
```

Gyorsan csináljuk vissza (most a másik függvényt használva, pusztán a változatosság kedvéért)!

```
varosok <- st_set_crs(x = varosok, value = 23700)
```

Vetületek kiolvasása és módosítása

```
eghajlat <- read.table(file =
 "eghajlat_fix_szelesseg.csv", sep = "", dec = ".", header
 = TRUE)
```

```
eghajlat <- st_as_sf(x = eghajlat, coords = c("eov_y",
 "eov_x"))
st_crs(eghajlat)
```

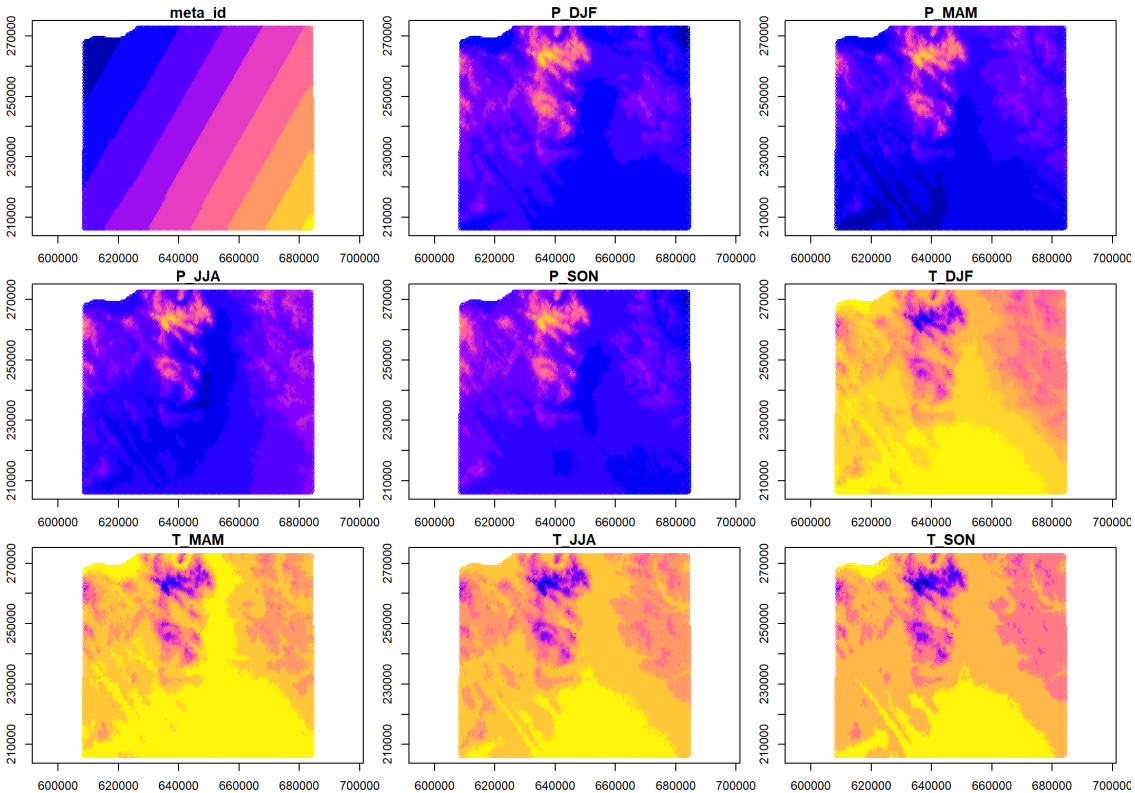
```
Coordinate Reference System: NA
```

```
st_crs(eghajlat) <- 23700
```

Ha nem ismert (vagy téves) a vetület, akkor hasznos az `st_crs()`.

Vetületek kiolvasása és módosítása

```
plot(eghajlat, axes = TRUE)
```



Átvetítés

Átvetítés (transzformáció)

- ha a koordináták és a vetület eddig is összhangban voltak
- de más vetületbe szeretnénk átvetíteni
- vagyis egyszerre kell módosítanunk a vetületet és átszámolnunk a koordinátákat

`st_transform(x, crs)`

- `x`: átvetítendő Simple Features
- `crs`: célvetület EPSG-száma (vagy PROJ.4-leírása)

Átvetítés

Visszatérünk Magyarország és városainak problémájához...
Második - ezúttal már tényleg sikeres - megoldási kísérlet: vetítsük át!

```
st_coordinates(varosok)[1:4, ]
```

```
             X        Y L1 L2
[1,] 643341.9 225654.8  1  1
[2,] 643313.8 225670.5  1  1
[3,] 643529.5 225985.4  1  1
[4,] 643200.3 226705.4  1  1
```

```
varosok_wgs <- st_transform(x = varosok, crs = 4326)
```

Átvetítés

```
st_coordinates(varosok_wgs)[1:4, ]
```

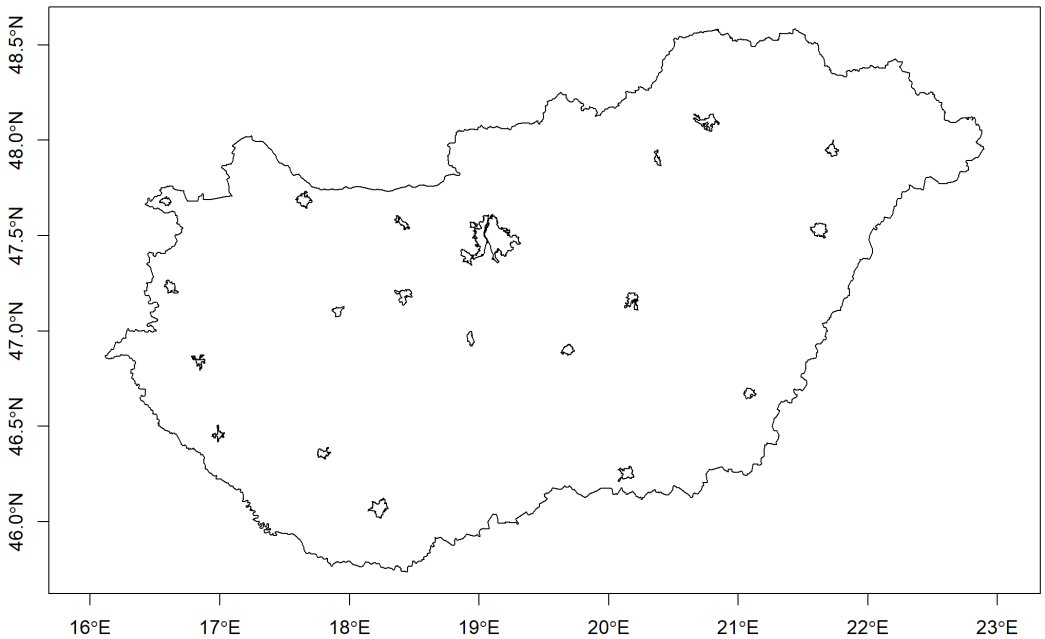
```
             X        Y L1 L2
[1,] 18.95928 47.37486  1  1
[2,] 18.95891 47.37500  1  1
[3,] 18.96176 47.37784  1  1
[4,] 18.95739 47.38431  1  1
```

```
st_crs(varosok_wgs)$epsg
```

```
[1] 4326
```

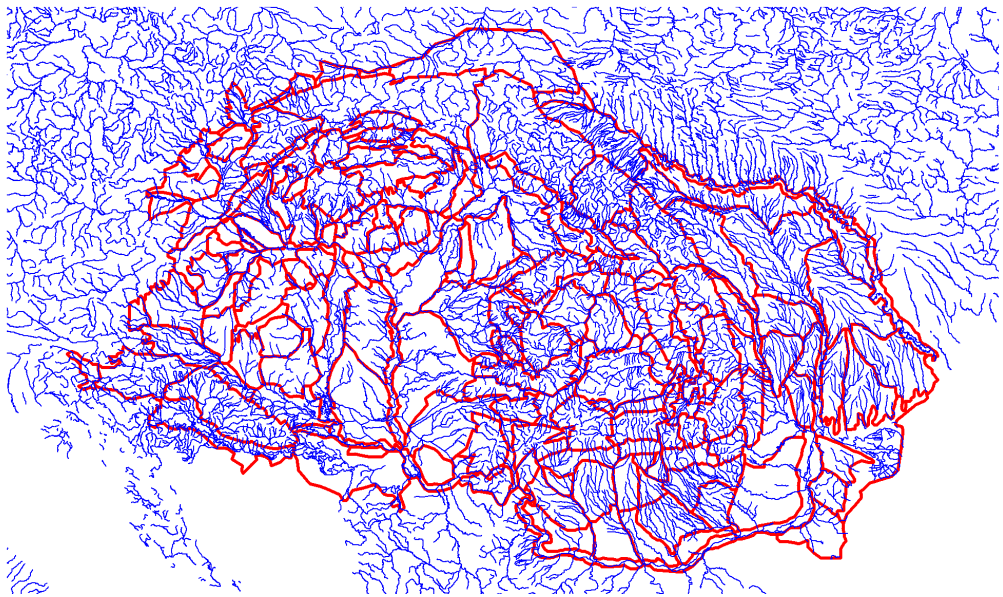
Átvetítés

```
plot(magyarorszag, axes = TRUE)
plot(st_geometry(varosok_wgs), add = TRUE)
```



8. feladat (órai)

- Olvasd be a “tajbeosztas_geometria.RData” fájlt.
- Mi a beolvasott `sfc`-objektum vetülete? És mi a folyók vetülete?
- Vetítsd át a tájbeosztás poligonjait WGS-84-be (EPSG: 4326),
- majd jelenítsd meg piros, háromszoros vastagságú körvonallal.
- Rakd rá a térképre a folyók vonalait (geometriáját) kék színnel.



8. feladat (órai) – megoldás

```
load("tajbeosztas_geometria.RData")
```

```
st_crs(tajbeosztas_geometria)$epsg
```

```
[1] 4326
```

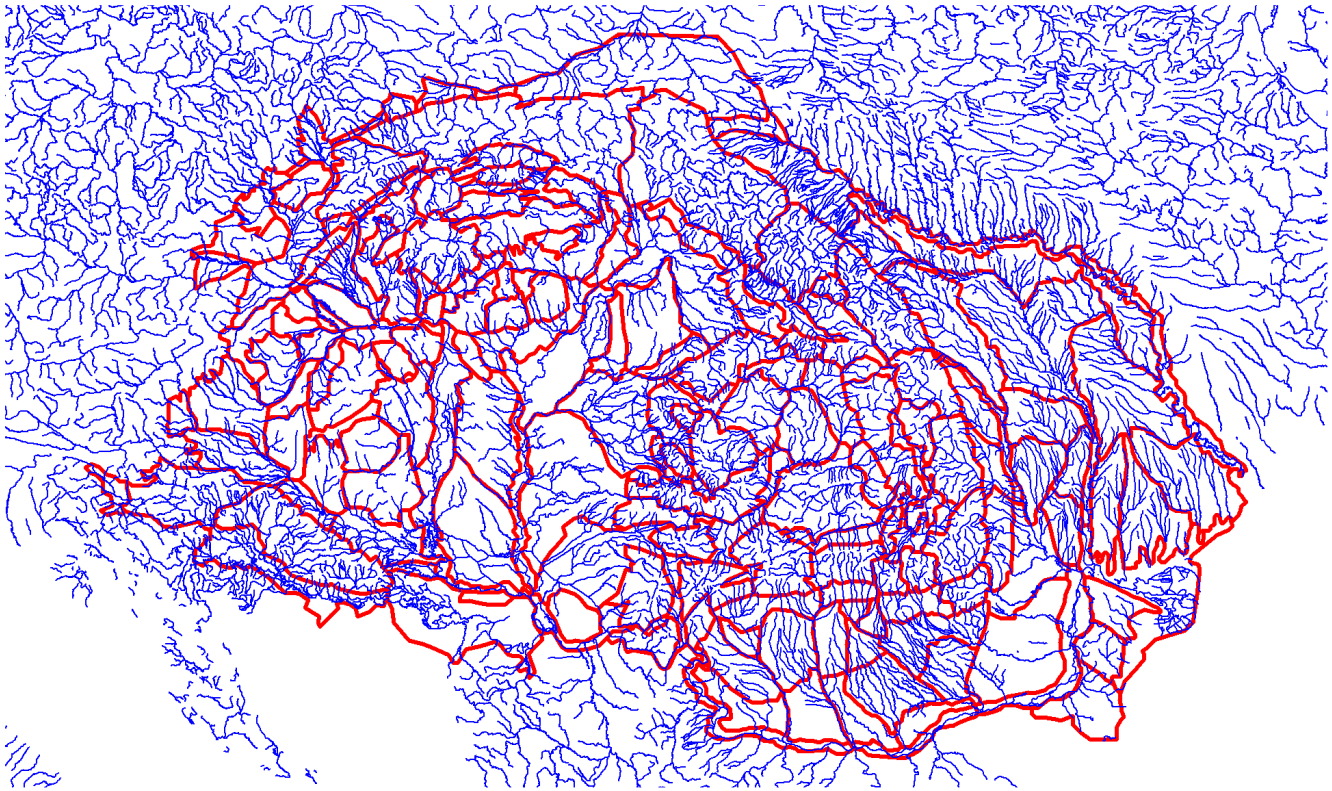
```
st_crs(folyok)$epsg
```

```
[1] 4326
```

```
tajbeosztas_geometria <- st_transform(x =
 tajbeosztas_geometria, crs = 4326)
```

8. feladat (órai) – megoldás

```
plot(tajbeosztas_geometria, border = "red", lwd = 3)
plot(st_geometry(folyok), col = "blue", add = TRUE)
```



9. feladat (házi)

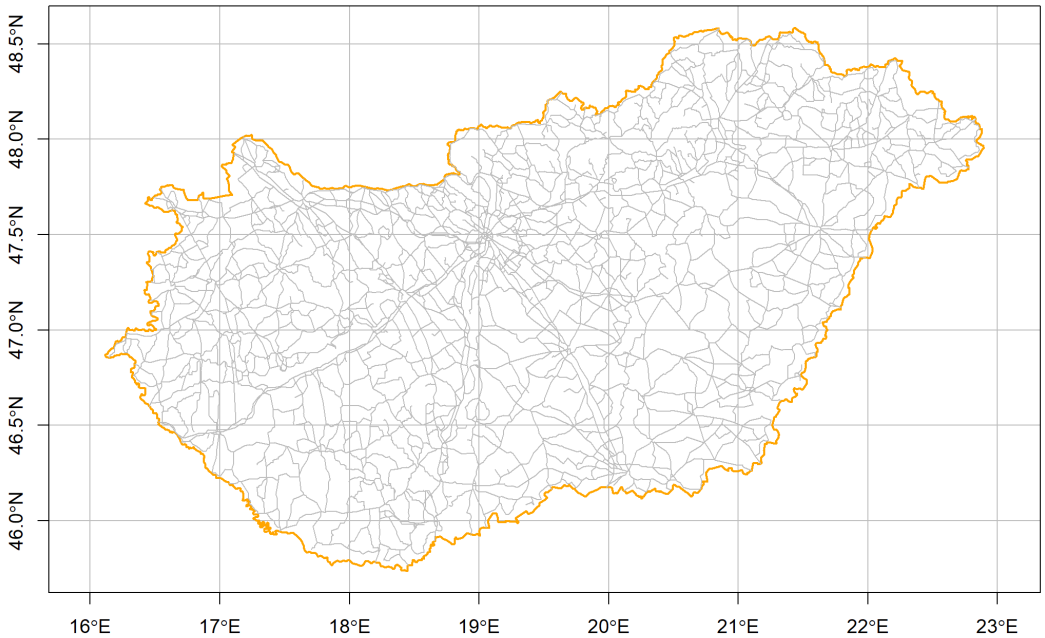
- Hozz létre az “utak” Simple Featuresből egy “utak_wgs” nevű másolatot, amit - a nevének megfelelően - WGS-84 koordináta-rendszerbe vetítesz (EPSG: 4326) EOV-ból.
- Hasonló logikával készíts egy “magyarorszag_eov” nevű, EOV-s (EPSG: 23700) másolatot is a WGS-84-es Magyarországról (“magyarorszag” nevű objektum).
- Készíts két, hasonló térképet:
  - az elsőn ábrázold a WGS-84-ben lévő (eredeti) Magyarországot narancssárga, dupla vastagságú körvonallal, koordináta-feliratokkal és rácshálóval, rajta a WGS-84-be vetített utak geometriájával, szürke színnel.
  - a második térkép ugyanígy épüljön fel, de az EOV vetületű geometriákat használd hozzá.

9. feladat (házi) – megoldás

```
utak_wgs <- st_transform(x = utak, crs = 4326)
magyarorszag_eov <- st_transform(x = magyarorszag, crs =
 23700)
```

9. feladat (házi) – megoldás

```
plot(magyarorszag, border = "orange", lwd = 2, graticule =
  TRUE, axes = TRUE)
plot(st_geometry(utak_wgs), col = "gray", add = TRUE)
```



9. feladat (házi) – megoldás

```
plot(magyarorszag_eov, border = "orange", lwd = 2,
  graticule = TRUE, axes = TRUE)
plot(st_geometry(utak), col = "gray", add = TRUE)
```