Megjelenítés

Térinformatika R-ben

2023.11.07.

Section 1

Geometriák megjelenítése

Grafikai paraméterek

A `plot()` függvény meghívható `sf` (később...) és `sfc`-objektumokra is.
Utóbbi esetben ugyanúgy paraméterezhető, mint a hagyományos `plot()`.
Fontosabb paraméterek:

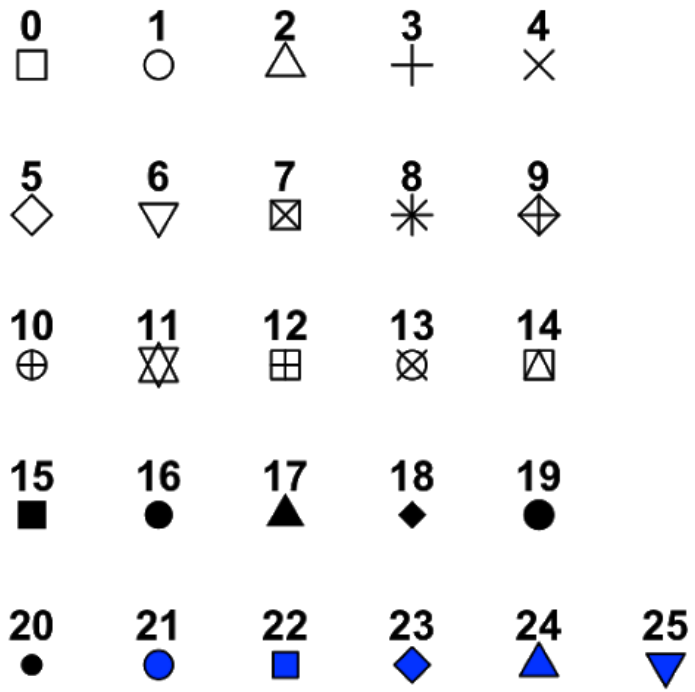
- `add`: a megnyitott képvászonra rajzoljon-e (alapértelmezett: újat nyit)
- `col`: **col**or, pont/vonal színe, poligon kitöltőszíne
- `border`: poligon határvonalszíne
- `lwd`: **l**ine **wid**th, vonal és poligon-határvonal relatív vastagsága
- `pch`: **p**oint **ch**aracter, pontjel típusa (szám vagy betű)
- `cex`: **c**haracter **ex**pansion, pontjel relatív mérete

Grafikai paraméterek

Alapértelmezett színek:

- pont/vonal: `col = "black"`
- poligon: `col = "transparent", border = "black"`

Pontjelek (`pch`) megadhatóak szövegként (pl. `"+"`) vagy számkóddal:



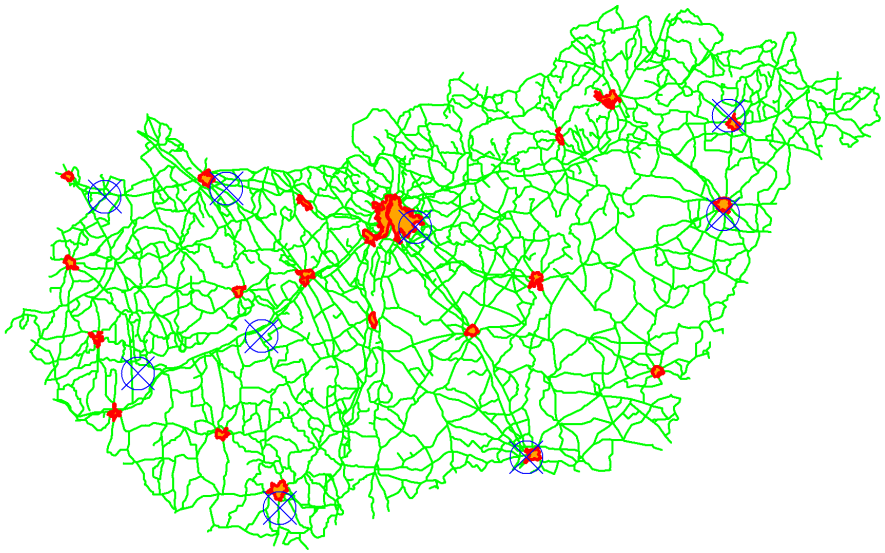
Grafikai paraméterek

```
library(sf)
load("utak_geometria.RData")
load("varosok_geometria.RData")
load("repterek.RData")
```

```
repterek_geometria <- st_geometry(repterek)
plot(utak_geometria)
plot(varosok_geometria, add = TRUE)
plot(repterek_geometria, add = TRUE)
```

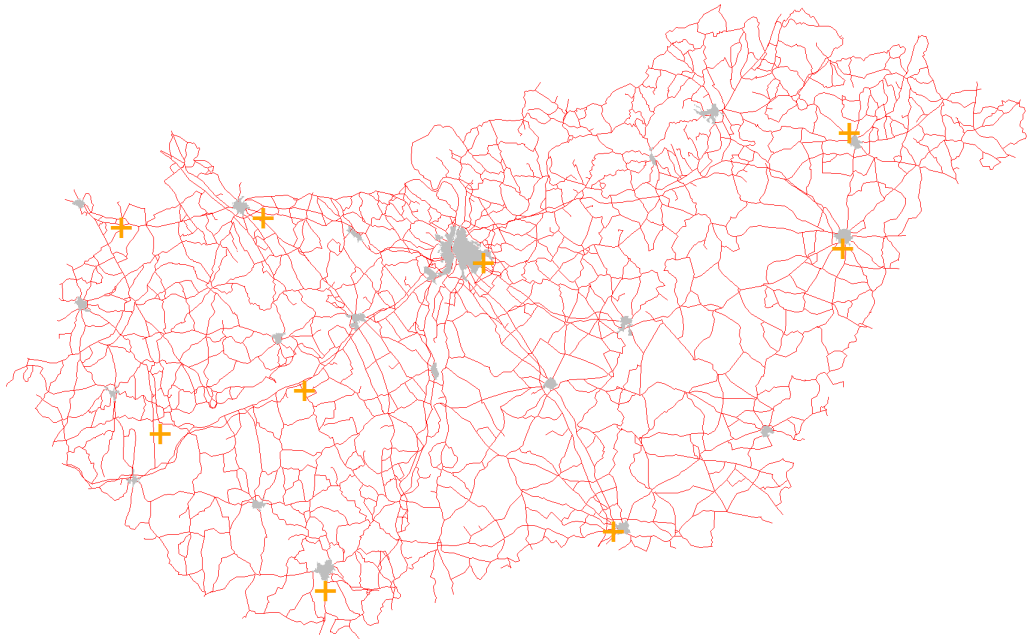
Grafikai paraméterek

```
plot(utak_geometria, col = "green", lwd = 2)
plot(varosok_geometria, col = "orange", border = "red",
 lwd = 3, add = TRUE)
plot(repterek_geometria, col = "blue", pch = 13, cex = 4,
 add = TRUE)
```



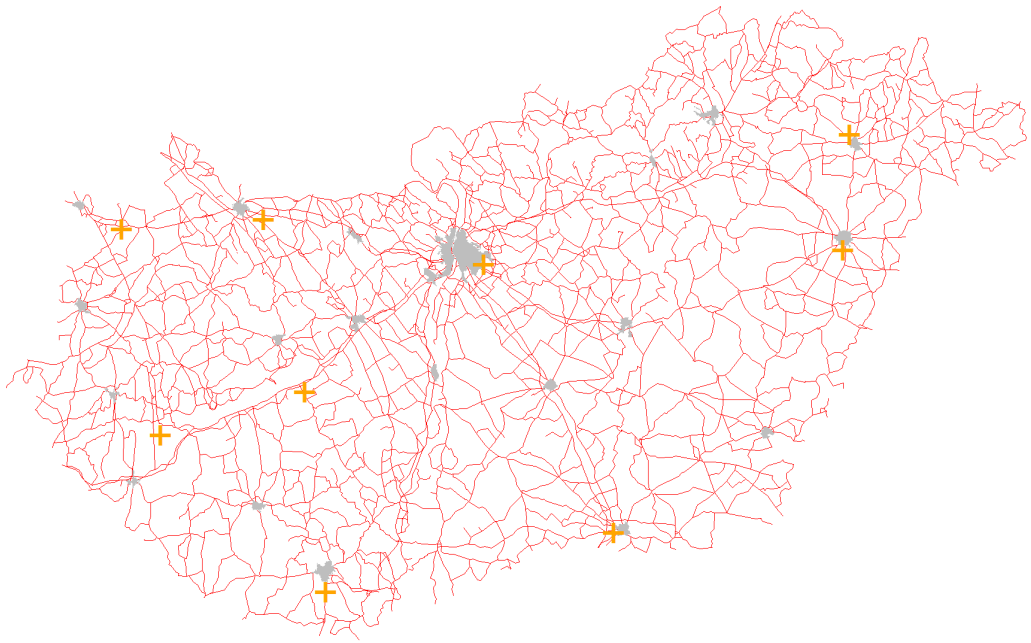
1. feladat (órai)

- Jelenítsd meg az utakat piros, vékony (0,5-ös) vonallal.
- Add hozzá a városokat szürke kitöltéssel és körvonal nélkül (vagyis átlátszó körvonallal),
- illetve a reptereket narancssárga, kétszeres méretű pluszjellel.



1. feladat (órai) – megoldás

```
plot(utak_geometria, col = "red", lwd = 0.5)
plot(varosok_geometria, col = "gray", border =
 "transparent", add = TRUE)
plot(repterek_geometria, col = "orange", pch = "+", cex =
 2, add = TRUE)
```



Ábrázolás sorrendje

Fontos az ábrázolás sorrendje!

- az új kitakarja a régit
- az első kép határozza meg az ábrázolási területet
- utóbbi persze az `xlim`/`ylim`-mel felülírható (később...)

Ábrázolás sorrendje

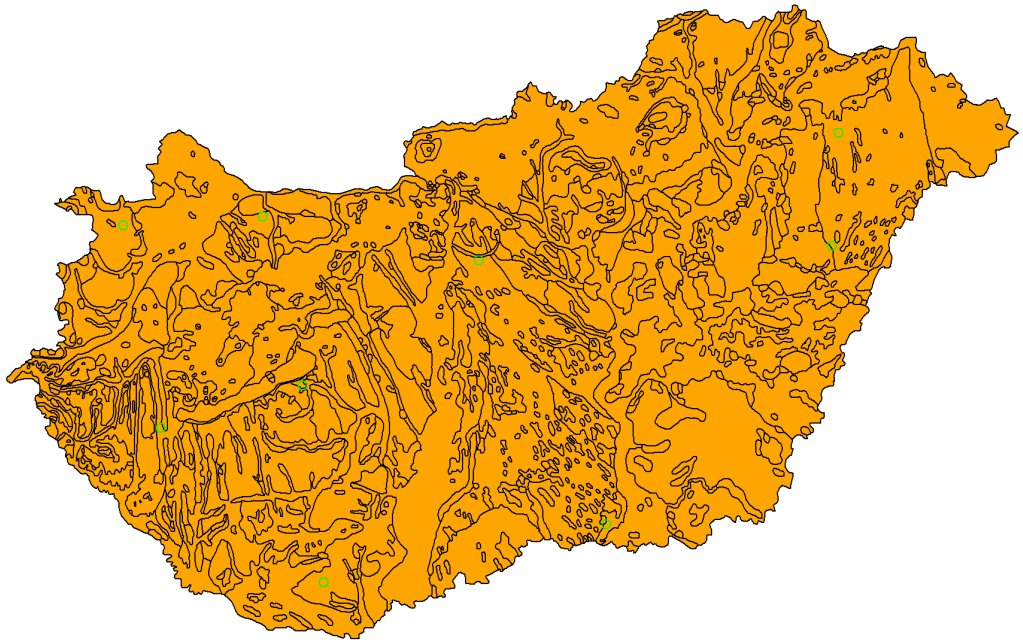
```
load("zolyomi_geometria.RData")
```

```
plot(repterek_geometria, col = "green")
plot(zolyomi_geometria, col = "orange", add = TRUE)
```

Hol vannak a repterek?
Hová tűnt Északkelet-Magyarország?

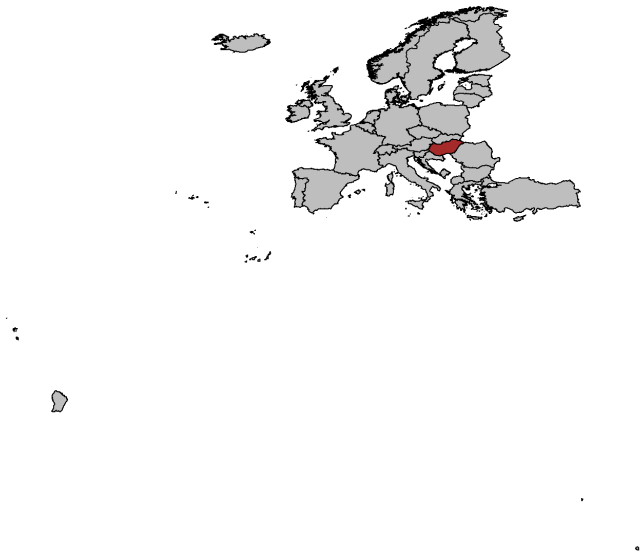
Ábrázolás sorrendje

```
plot(zolyomi_geometria, col = "orange")
plot(repterek_geometria, col = "green", add = TRUE)
```



2. feladat (házi)

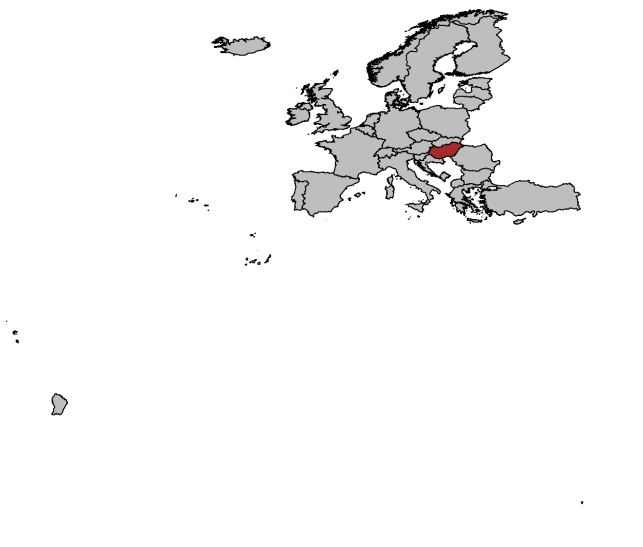
- Töltsd be az orszag_eu_geometria.RData és magyarorszag.RData fájlokat.
- Jelenítsd meg alul az EU országait szürke kitöltőszínnel,
- rajta pedig Magyarországot barna kitöltőszínnel.



2. feladat (házi) – megoldás

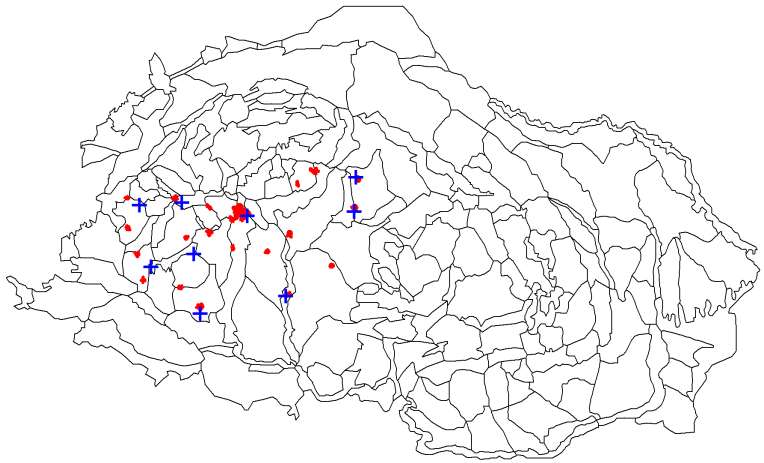
```
load("orszag_eu_geometria.RData")
load("magyarorszag.RData")
```

```
plot(orszag_eu_geometria, col = "gray")
plot(magyarorszag, col = "brown", add = TRUE)
```



3. feladat (órai)

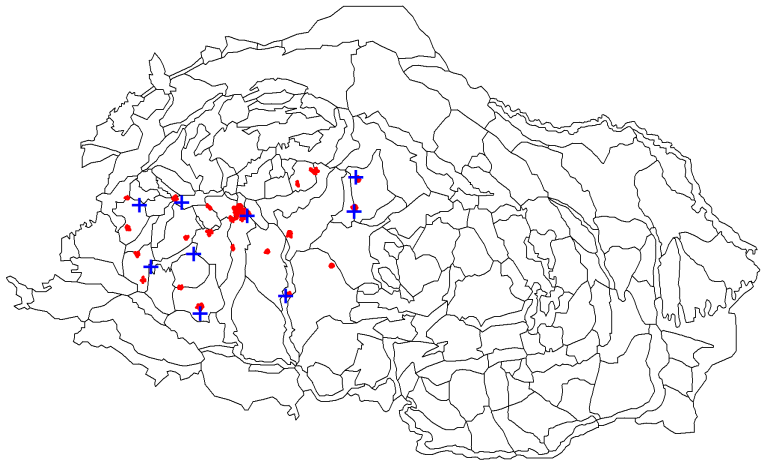
- Töltsd be az tajbeosztas_geometria.RData és varosok_geometria.RData fájlokat.
- Jelenítsd meg
  - alul a tájbeosztás poligonjait átlátszó kitöltőszínnel,
  - fölötte a városokat piros kitöltő- és körvonalszínnel, háromszoros vastagságú körvonallal,
  - majd rajtuk a reptereket kék, kétszeres méretű pluszjellel.



3. feladat (órai) – megoldás

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

```
plot(tajbeosztas_geometria, col = "transparent")
plot(varosok_geometria, col = "red", border = "red", lwd =
 3, add = TRUE)
plot(repterek_geometria, col = "blue", pch = "+", cex = 2,
 add = TRUE)
```



Feliratok és ábrázolt terület

Tengelyfeliratok, tengelyek és cím hozzáadása a szokásos módon:

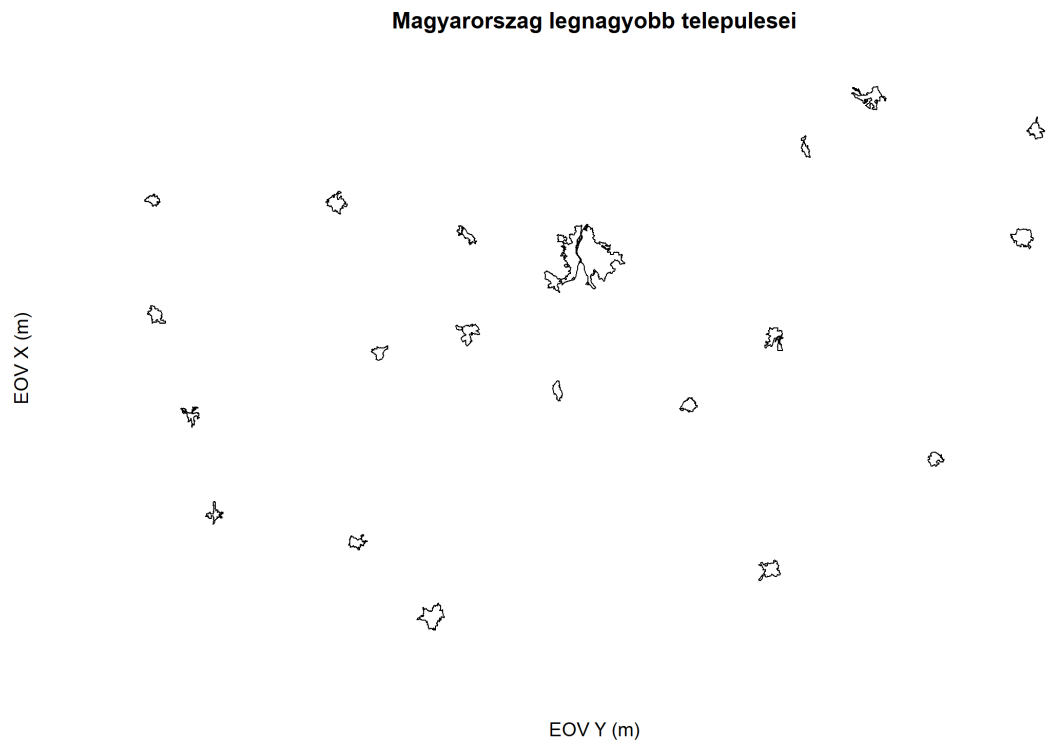
- `xlab`: **x lab**el, vízszintes tengely címe
- `ylab`: **y lab**el, függőleges tengely címe
- `main`: ábra címe
- `axes`: ábrázolja-e a tengelyt (+ tüskéket és koordináta-feliratokat)? Alapértelmezetten `FALSE`.

A képvászon méretét mindig az elsőként megjelenített geometria kiterjedése határozza meg (kis ráhagyással).
Ez akár csökkenthető, akár növelhető:

- `xlim`: **x lim**it, az ábra vízszintes kiterjedése (kételemű vektor)
- `ylim`: **y lim**it, az ábra függőleges kiterjedése (kételemű vektor)

Feliratok és ábrázolt terület

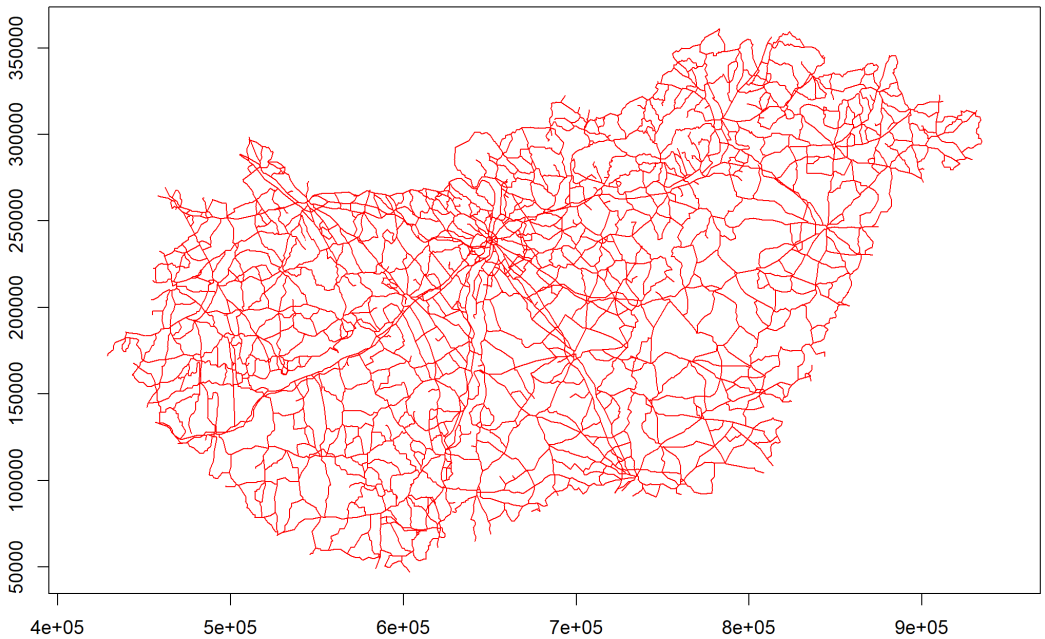
```
plot(varosok_geometria, main = "Magyarorszag legnagyobb
 telepulesei", xlab = "EOV Y (m)", ylab = "EOV X (m)")
```



Persze a tengelyfeliratoknak tengelyek híján nincs sok veleje...

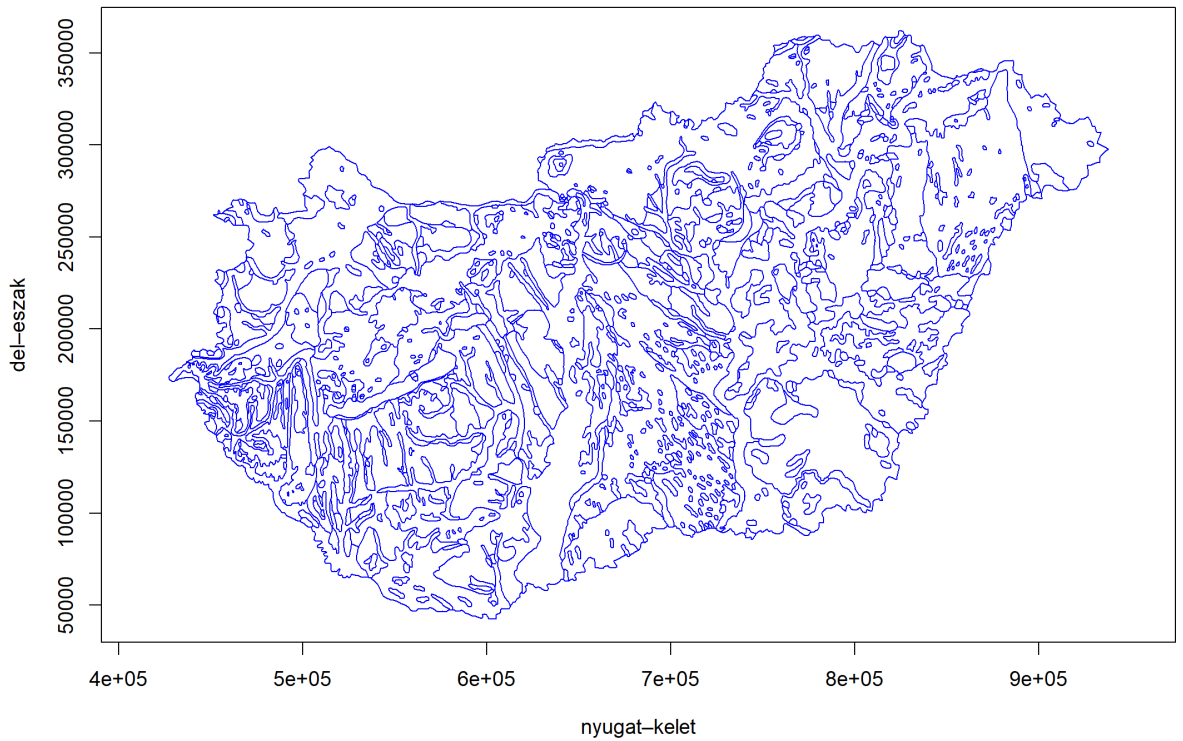
Feliratok és ábrázolt terület

```
plot(utak_geometria, col = "red", axes = TRUE)
```



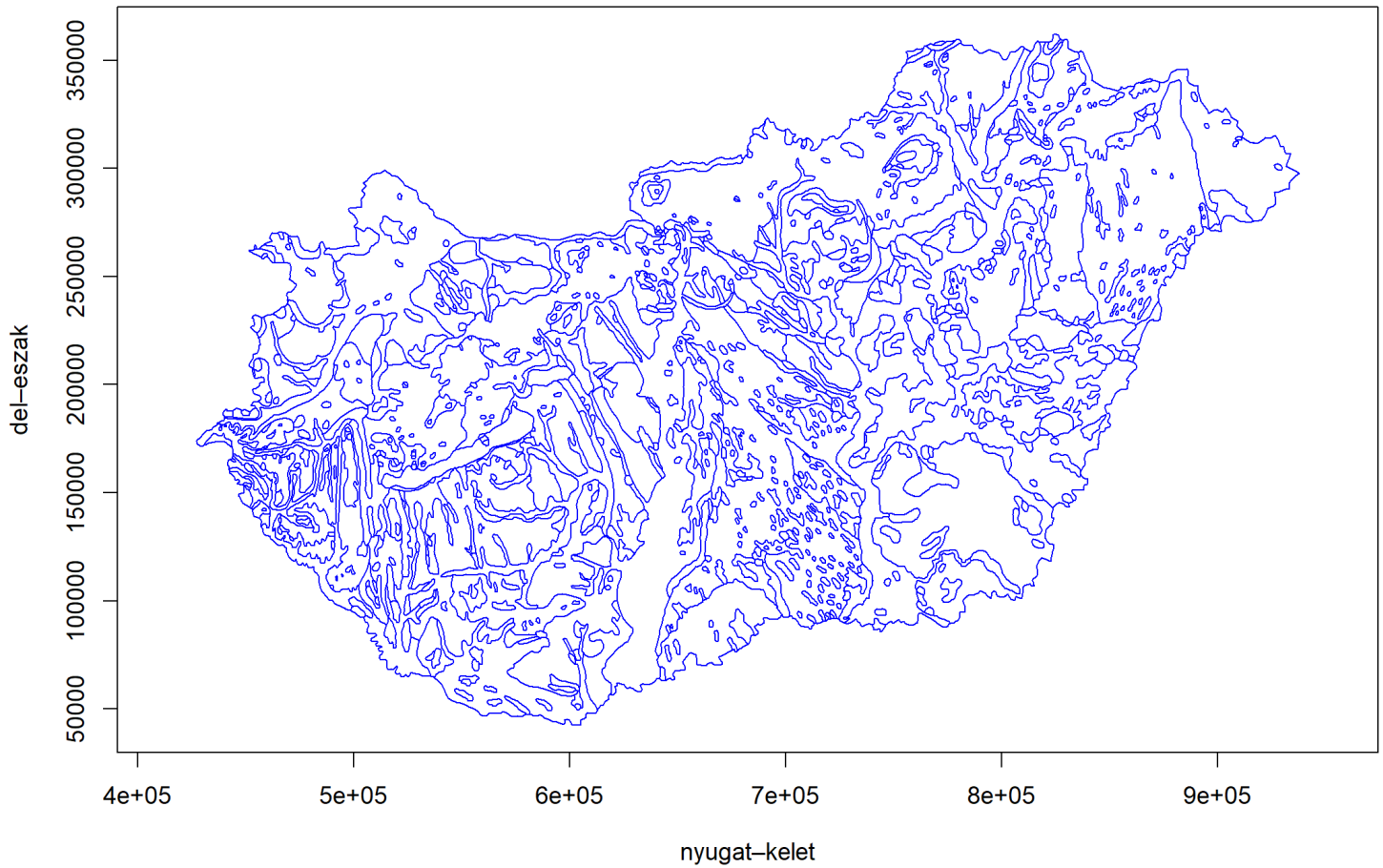
4. feladat (házi)

- Ábrázold a Zólyomi Bálint-féle vegetációzónák határait kék színnel.
- Az ábrát keretezze tengely tüskékkel és koordináta-feliratokkal.
- A vízszintes tengely felirata legyen “nyugat–kelet”, a függőlegesé pedig “del–eszak”.



4. feladat (házi) – megoldás

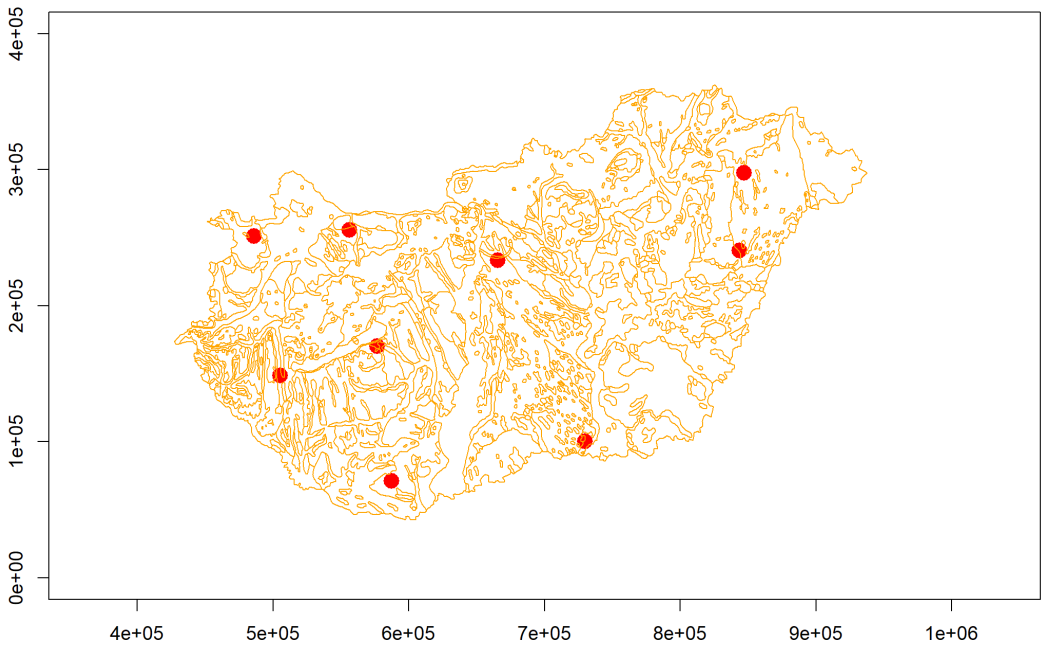
```
plot(zolyomi_geometria, axes = TRUE, border = "blue", xlab
 = "nyugat-kelet", ylab = "del-eszak")
```



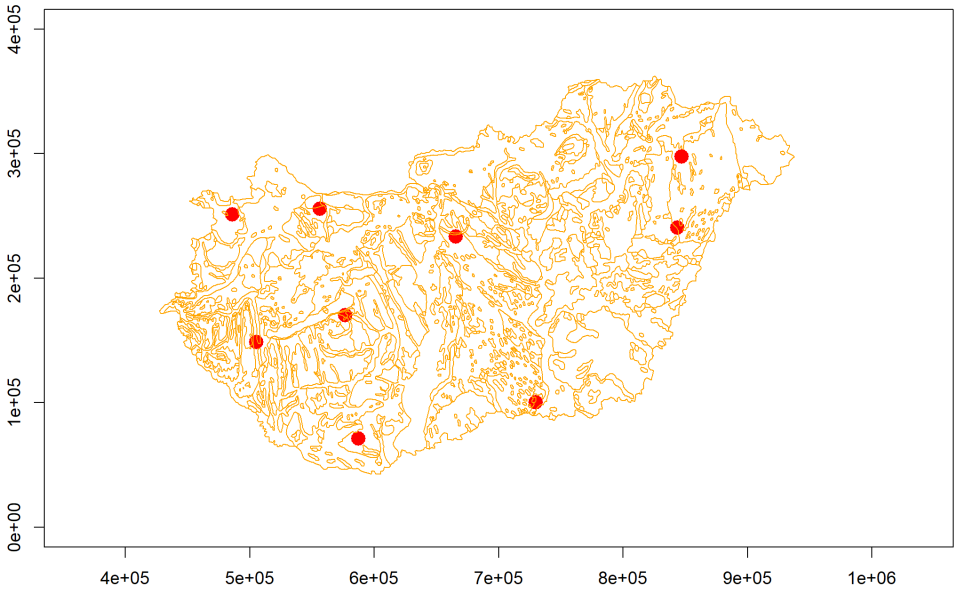
Feliratok és ábrázolt terület

Előbb leolvastuk a tengelyeken az ábrázolandó tartományt.
Most már könnyen megjelenítünk bármit ezen tartományok között!

```
plot(repterek_geometria, pch = 16, col = "red", cex = 2,
 xlim = c(400000, 1000000), ylim = c(0, 400000), axes =
 TRUE)
plot(zolyomi_geometria, border = "orange", add = TRUE)
```



Feliratok és ábrázolt terület



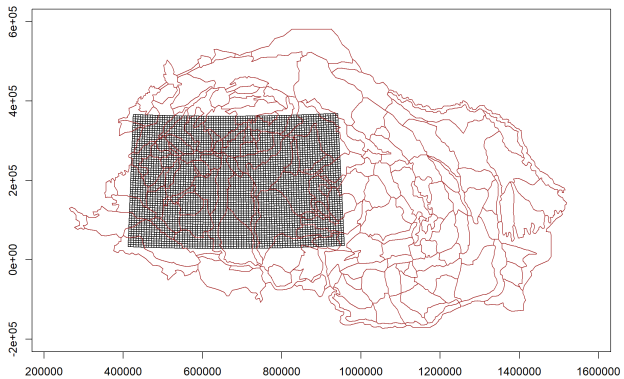
Mi értelme meghatározni az ábrázolt területet?

- szűkítés céljából
- vagy mert nem mindig a legnagyobb kiterjedésű az elsőként ábrázolt (alsó) réteg
- egymásralapolás miatt fontos a sorrend
- ha tulajdonság szerint színezünk (később...), azt általában elsőként

5. feladat (órai)

- Olvasd be a tajbeosztas_geometria.RData és kef_halo_geometria.RData fájlokat,
- majd jelenítsd meg a kettőt az alábbiak szerint:
  - alul legyen a KEF-háló, tengellyel (tüskék és koordináták),
  - kerüljön rá barna körvonalszínnel a tájbeosztás geometriája,
  - ne lógjon le az ábráról semmi a tájbeosztás poligonjaiból.

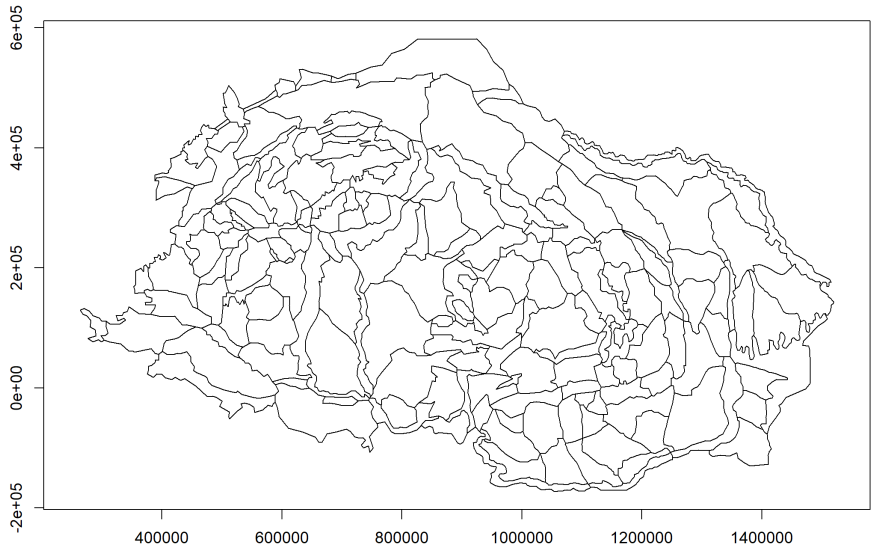
Nyugodtan kísérletezz előtte a tájbeosztás ábrázolásával, hogy megtudd, milyen vízszintes és függőleges tartományban érdemes létrehozni a képvásznat.



5. feladat (órai) – megoldás

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

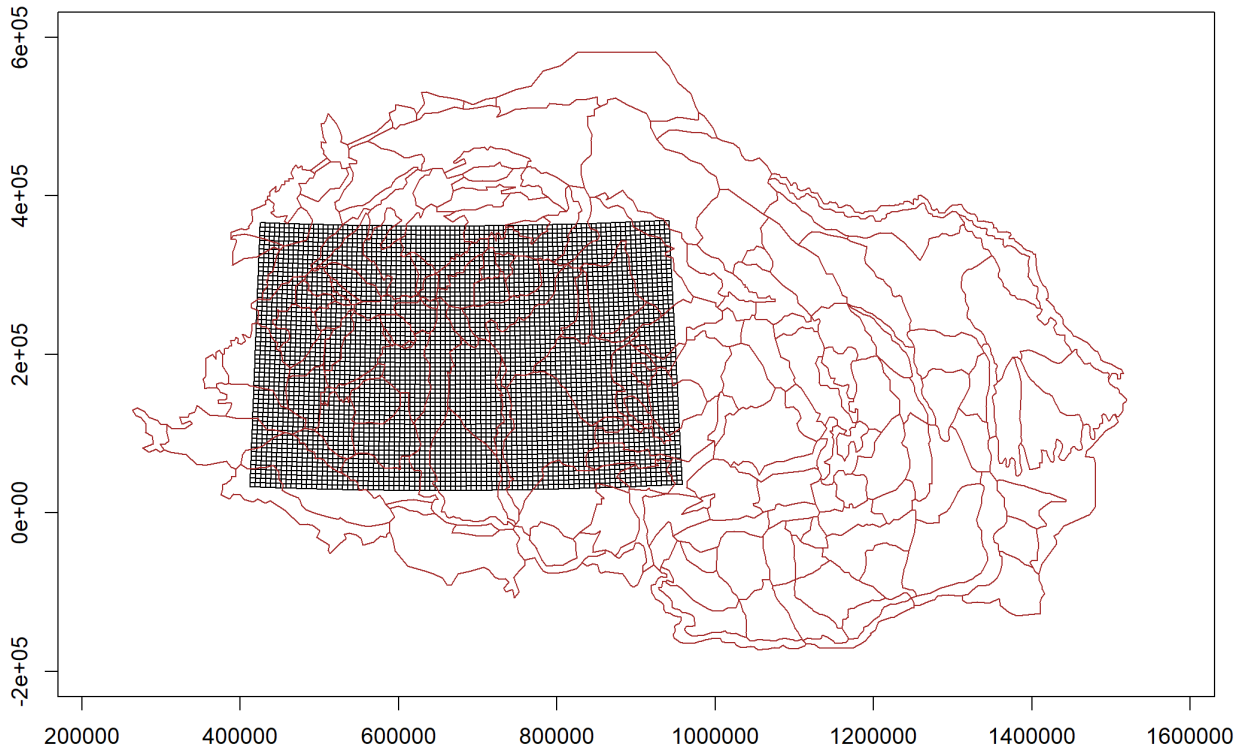
```
plot(tajbeosztas_geometria, axes = TRUE)
```



Leolvassuk az ábrázolandó tartományt.

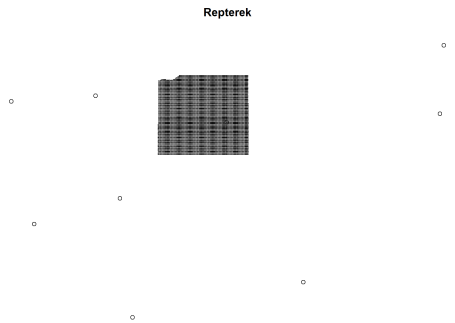
5. feladat – megoldás

```
plot(kef_halo_geometria, axes = TRUE, xlim = c(300000,
 1500000), ylim = c(-200000, 600000))
plot(tajbeosztas_geometria, border = "brown", add = TRUE)
```



6. (összefoglaló) feladat (házi)

- Olvasd be a repterek_pontosvesszo.csv és eghajlat_tab_tvesszo.csv fájlokat, és mindkettőből készíts Simple Featurest.
- Az éghajlat geometriáját írd egy új, eghajlat_geometria nevű változóba.
- Ezt és a reptereket mentsd el a repter_es_eghajlat.RData fájlba, a munkakönyvtár hazi_feladat nevű alkönyvtára alá. (Hozd létre, ezt a könyvtárat, ha még nem létezik.)
- Jelenítsd meg a repterek helyét egy “Repterek” című ábrán,
- és add hozzá pontokként ábrázolva az éghajlati adatsor geometriáját.

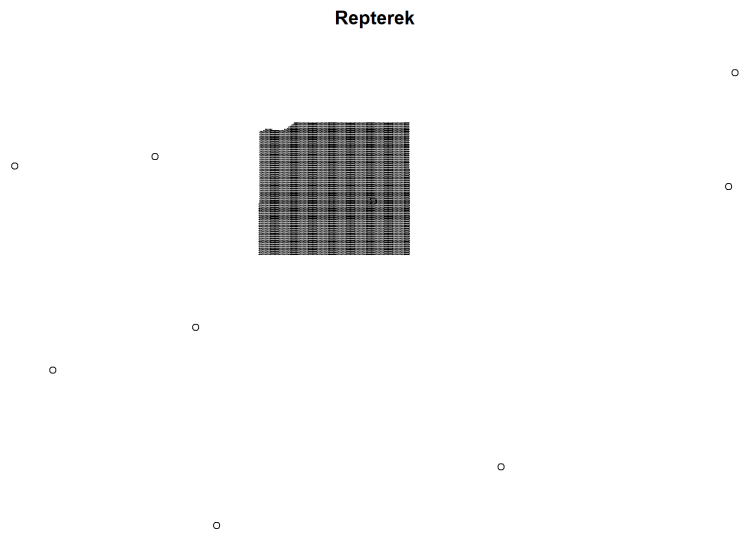


6. (összefoglaló) feladat (házi) – megoldás

```
repterek <- read.table(file = "repterek_pontosvesszo.csv",
 sep = ";", dec = ".", header = TRUE)
repterek <- st_as_sf(x = repterek, crs = 23700, coords =
 c("eov_y", "eov_x"))
eghajlat <- read.table(file = "eghajlat_tab_tvesszo.csv",
 sep = "\t", dec = ",", header = TRUE)
eghajlat <- st_as_sf(x = eghajlat, crs = 23700, coords =
 c("eov_y", "eov_x"))
eghajlat_geometria <- st_geometry(eghajlat)
dir.create("hazi_feladat", showWarnings = FALSE, recursive
 = TRUE)
save(list = c("repterek", "eghajlat_geometria"), file =
 "hazi_feladat/repter_es_eghajlat.RData")
```

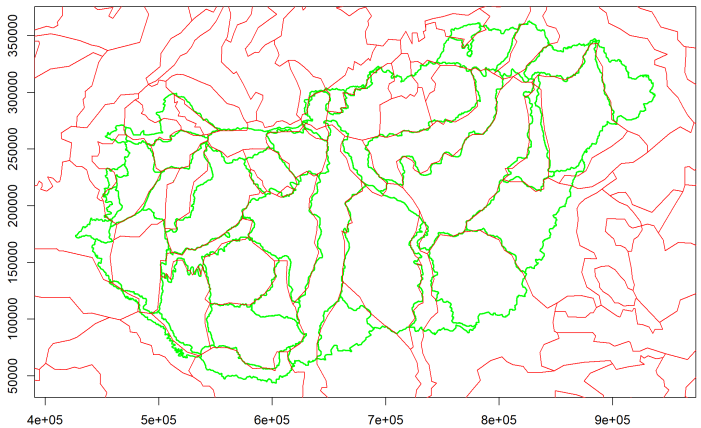
6. (összefoglaló) feladat (házi) – megoldás

```
plot(st_geometry(repterek), main = "Repterek")
plot(eghajlat_geometria, pch = ".", add = TRUE)
```



7. (összefoglaló) feladat (házi)

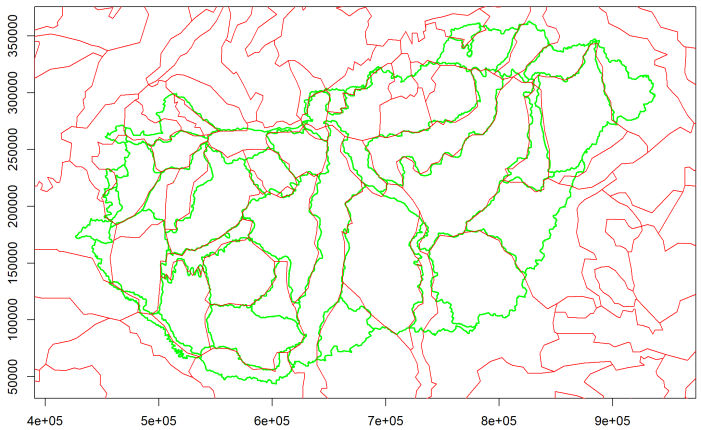
- Olvasd be a kozeptajak_geometria.shp fájlt,
- majd mentsd el az előző feladatban létrehozott hazi_feladat mappába kozeptajak_geometria.gpkg néven, geopackage-ként. Ha már létezik a fájl, írd felül.
- Jelenítsd meg kétszeres vastagságú zöld körvonallal a középtájak határait,
- majd ezen piros körvonallal ábrázold a tájbeosztást.
- Az ábrát keretezze tengely tüskékkel és koordináta-feliratokkal.



7. (összefoglaló) feladat (házi) – megoldás

```
kozeptajak_geometria <- st_read(dsn =
 "kozeptajak_geometria.shp", quiet = TRUE)
st_write(obj = kozeptajak_geometria, dsn =
 "hazi_feladat/kozeptajak_geometria.gpkg", quiet = TRUE,
 delete_dsn = TRUE)
```

```
plot(kozeptajak_geometria, border = "green", lwd = 2, axes
 = TRUE)
plot(tajbeosztas_geometria, border = "red", add = TRUE)
```



Section 2

Tulajdonságok megjelenítése

Tulajdonságok vs. geometriák megjelenítése

A `plot(sf)` és `plot(sfc)` funkciója, és ezért a paraméterezése is különbözik.

- `sf`: tulajdonságok szerint színez, `sfc`: geometriát jelenít meg
- `sf`: több tulajdonság esetén több térképet készít, `sfc`: csak egy geometria van
- `sf`: egy tulajdonság esetén színskálás jelmagyarázatot készít, `sfc`: nincs mit a jelmagyarázatra rakni
- `sf`: megjelenítés után alapértelmezés szerint lezárja a képvászont, `sfc`: a képvászon nyitva marad

Tulajdonságok megjelenítése

Sok paraméter, a legfontosabbak:

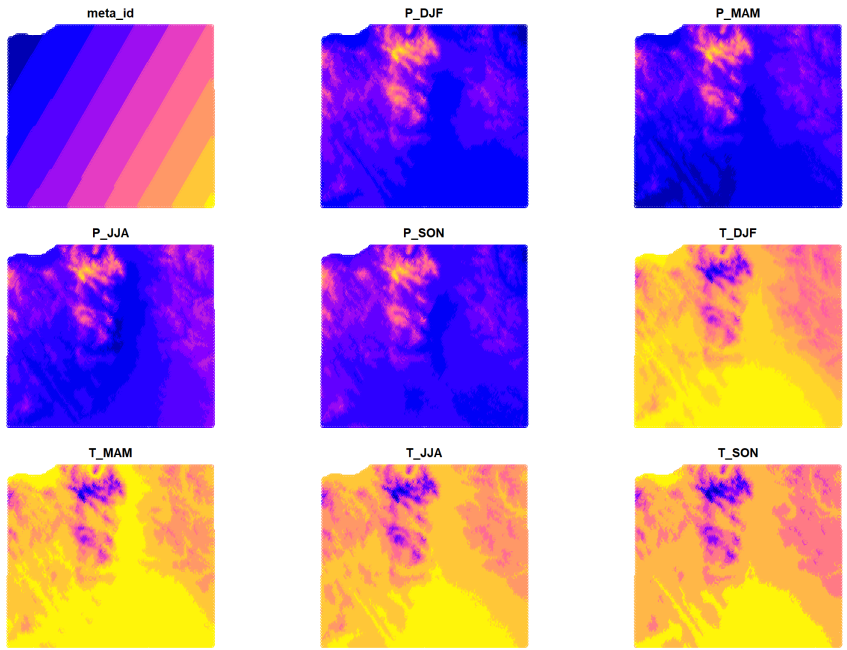
- `max.plot`: ábrázolandó tulajdonságok (oszlopok) maximális száma (alapértelmezetten 9)
- `key.pos`: **key pos**ition, jelmagyarázat helye
- `pal`: **pal**ette, színskála vagy a színskálakészítő függvény neve
- `nbreaks`: **n**umber of **breaks**, a színskálát hány részre ossza?
- `breaks`: a színskála osztópontjai vagy az osztópontkészítő függvény neve
- `reset`: lezárja-e a képvászont (alapértelmezetten `TRUE`)

Továbbá a geometria ábrázolásánál megismert paramterek (pl. `cex`, `main`) is működnek.

Ábrázolt térképek száma

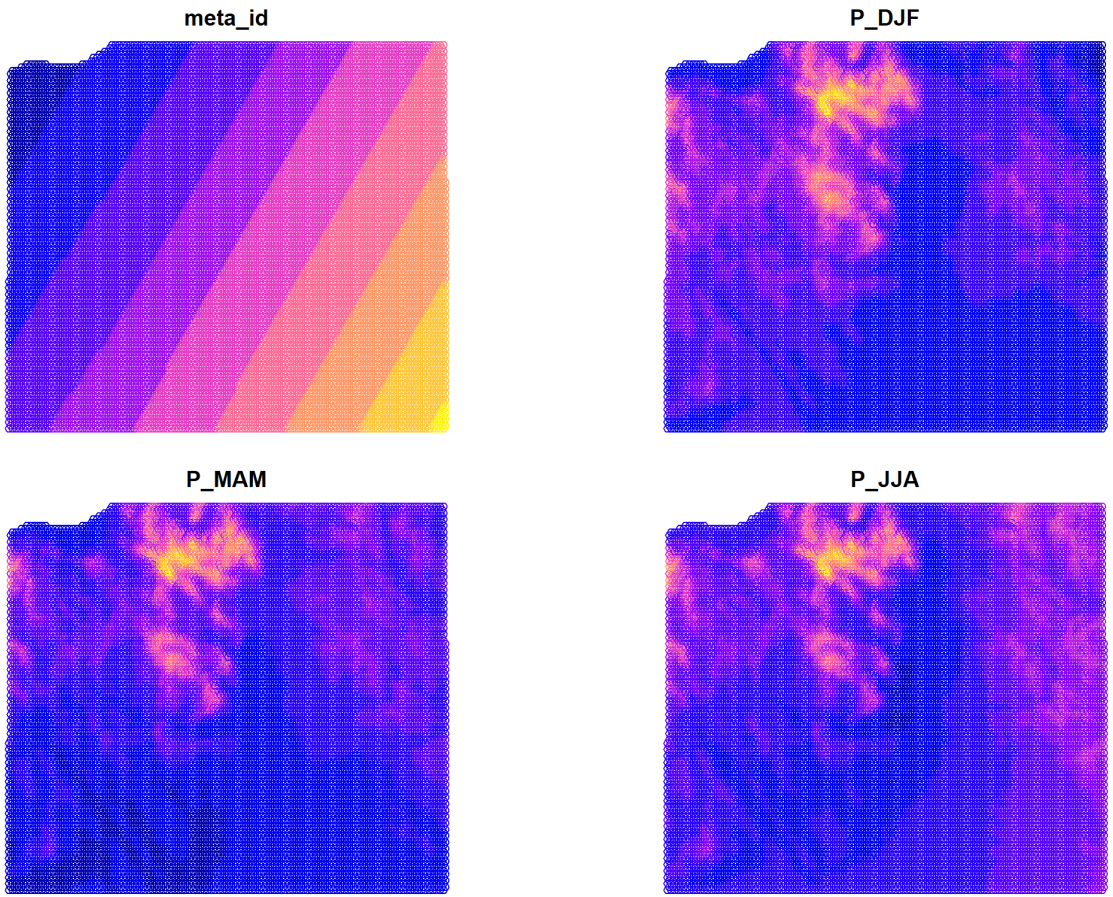
```
load("eghajlat.RData")
```

```
plot(eghajlat)
```



Ábrázolt térképek száma

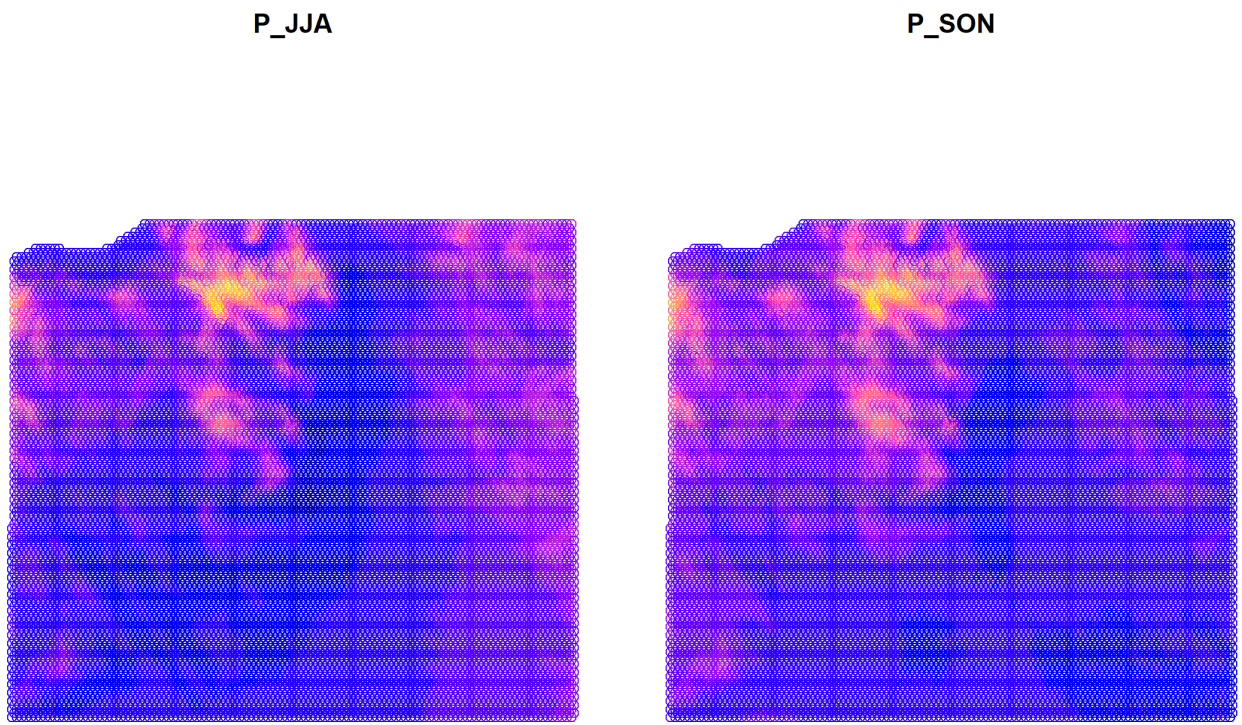
```
plot(eghajlat, max.plot = 4)
```



Ábrázolt térképek száma

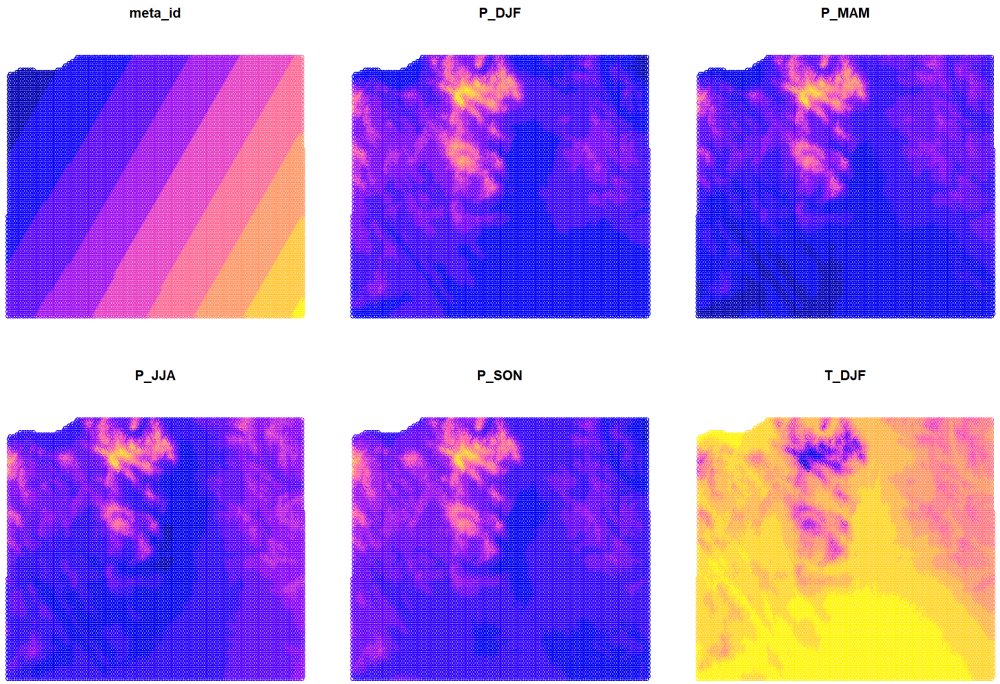
A `max.plot`-tal az ábrázolt tulajdonságoknak nem a tényleges, hanem a maximális számát módosíthatjuk.
A tényleges számot az határozza meg, hogy hány oszlopa van a Simple Featuresnek.

```
plot(eghajlat[, c("P_JJA", "P_SON")], max.plot = 4)
```



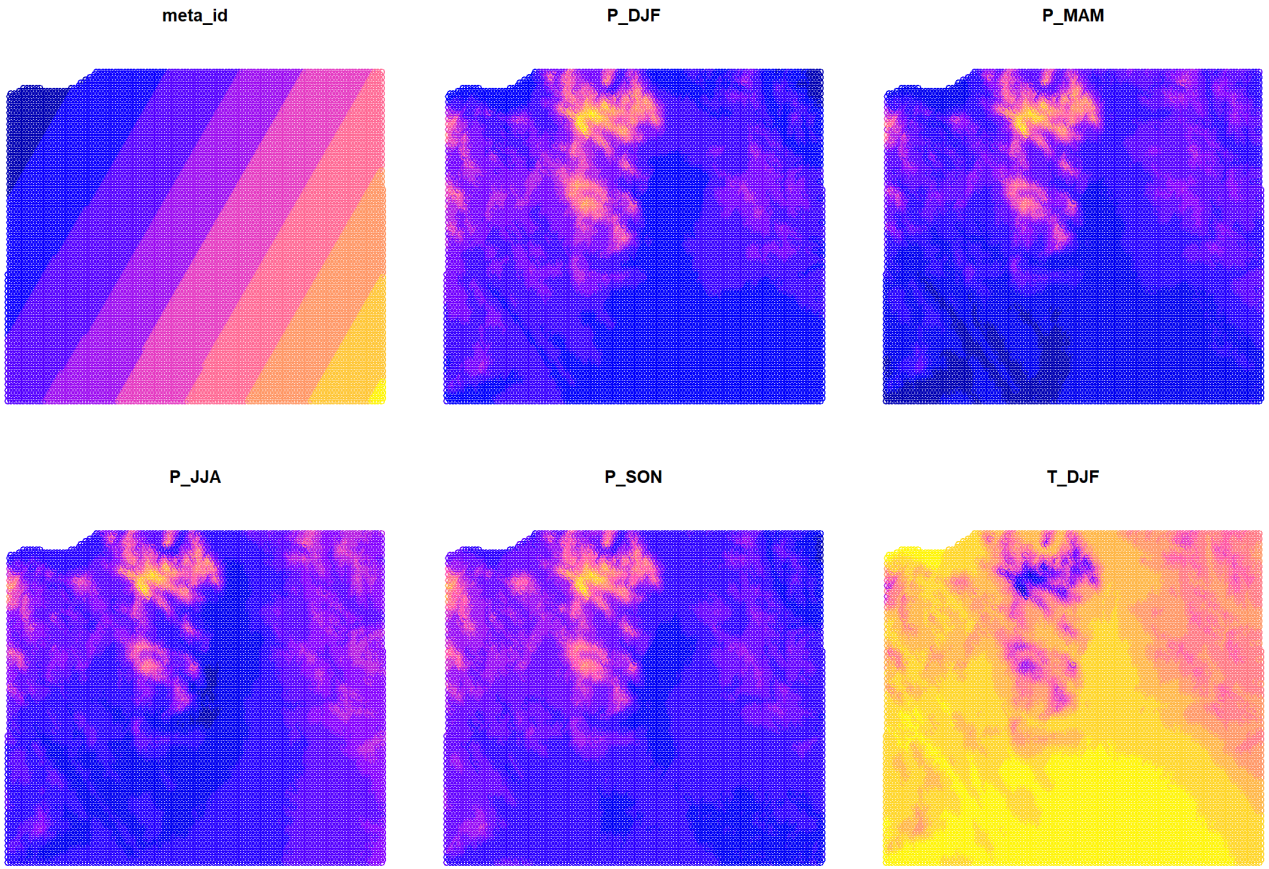
8. feladat (házi)

- Ábrázold az éghajlati jellemzőket, de úgy, hogy legfeljebb 6 térkép készüljön.
- Ezután ábrázold a tavaszi (T_MAM) és nyári (T_JJA) középhőmérsékletet egymás mellett.



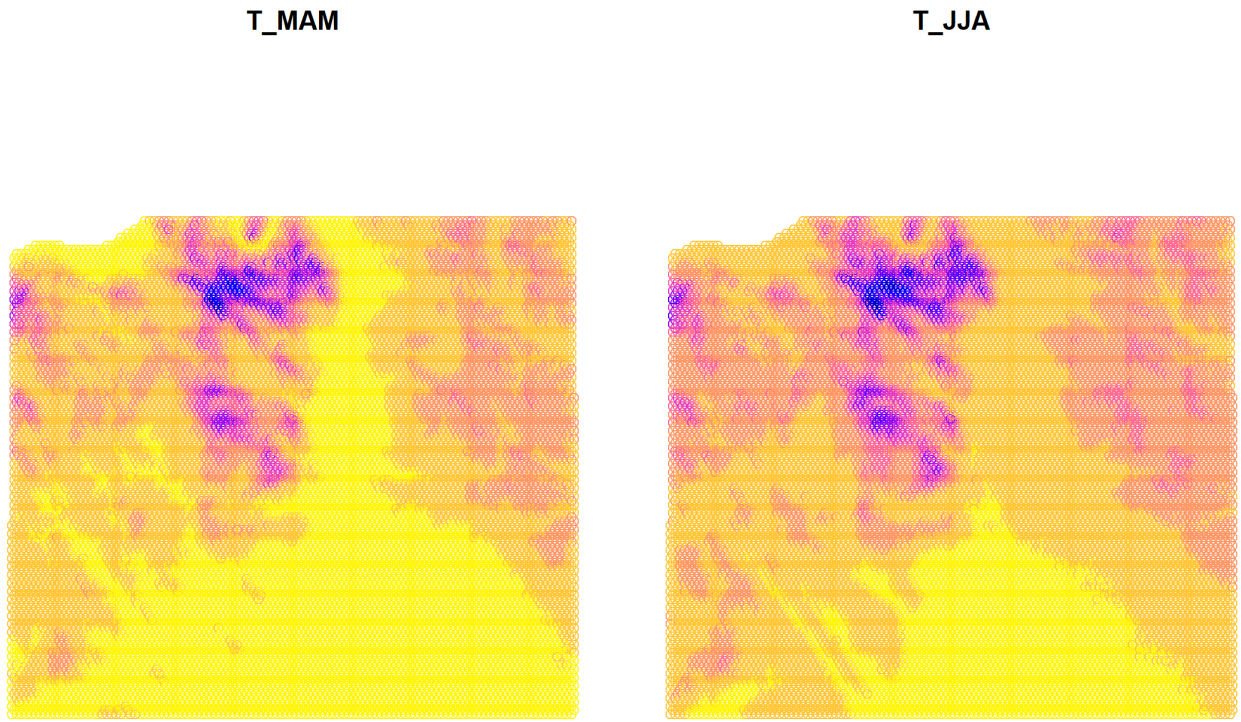
8. feladat (házi) – megoldás

```
plot(eghajlat, max.plot = 6)
```



8. feladat (házi) – megoldás

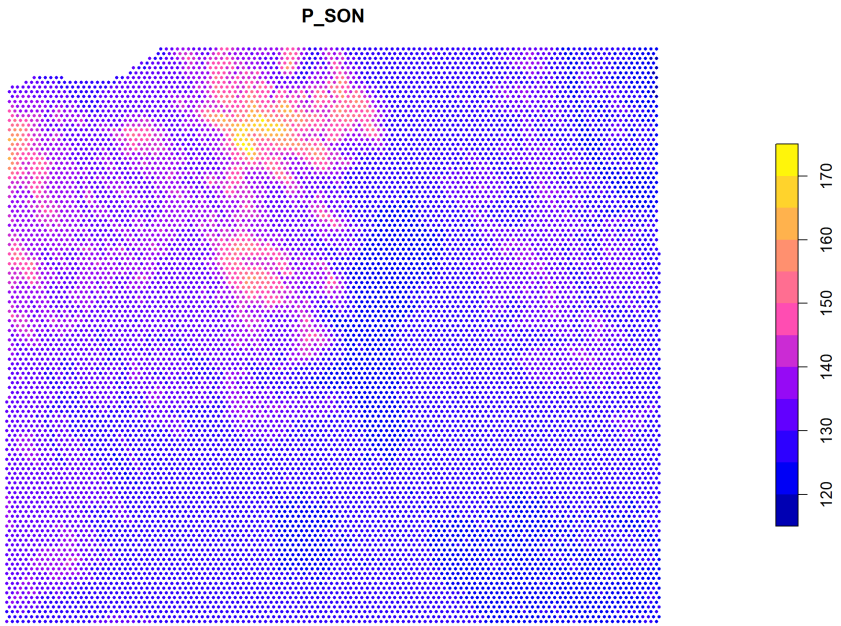
```
plot(eghajlat[, c("T_MAM", "T_JJA")])
```



Színskála és jelmagyarázat

Ha *egy* tulajdonságot ábrázolunk, színskálás jelmagyarázatot kapunk. Poligonok esetén a kitöltőszín változik a tulajdonsággal, a körvonal fekete.

```
plot(eghajlat[, "P_SON"], pch = 16, cex = 0.5)
```



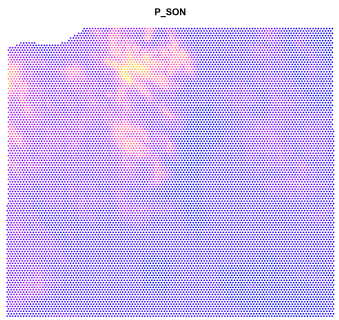
Természetesen kérhetünk jelmagyarázatot több térkép esetén is, valamint egy térkép esetén elrejthetjük azt.

Színskála és jelmagyarázat

A `key.pos` paraméter lehetséges értékei:

- 1: alul
- 2: bal
- 3: felül
- 4: jobb
- –1: automatikus (a képarány és a képvászon méretének függvénye)
- `NULL`: sehol

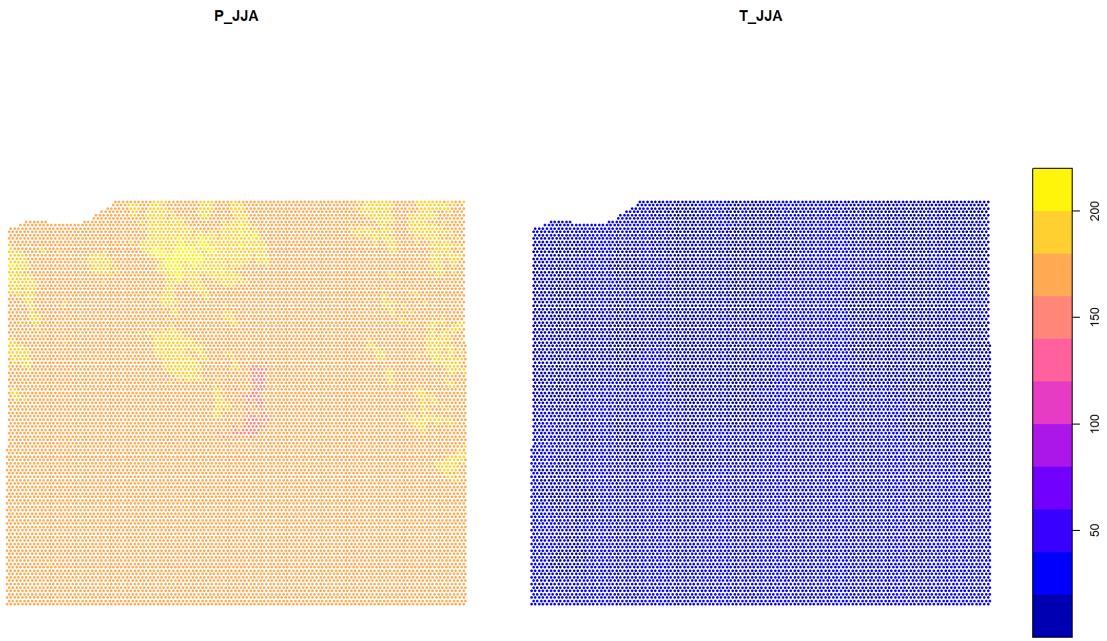
```
plot(eghajlat[, "P_SON"], pch = 16, cex = 0.5, key.pos =
 NULL)
```



Színskála és jelmagyarázat

Ha több tulajdonságra kérünk jelmagyarázatot, közös színskálát használ.

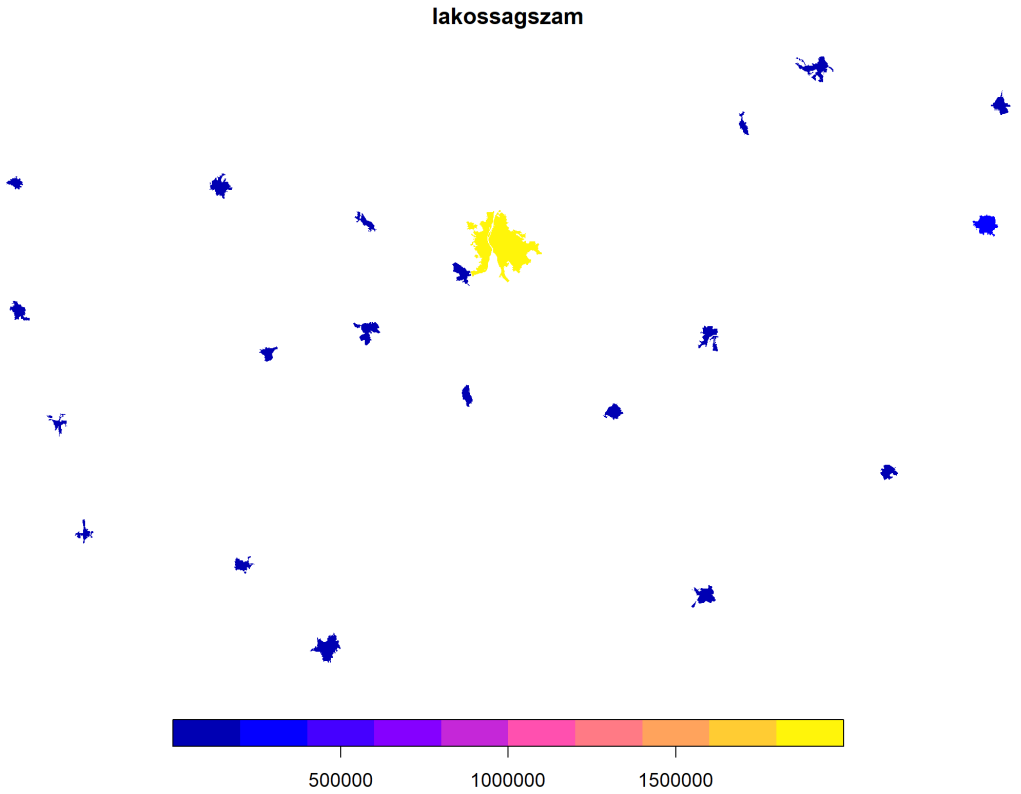
```
plot(eghajlat[, c("P_JJA", "T_JJA")], pch = 16, cex = 0.5,
 key.pos = 4)
```



Ennek nincs mindig értelme (°C és mm)...

9. feladat (órai)

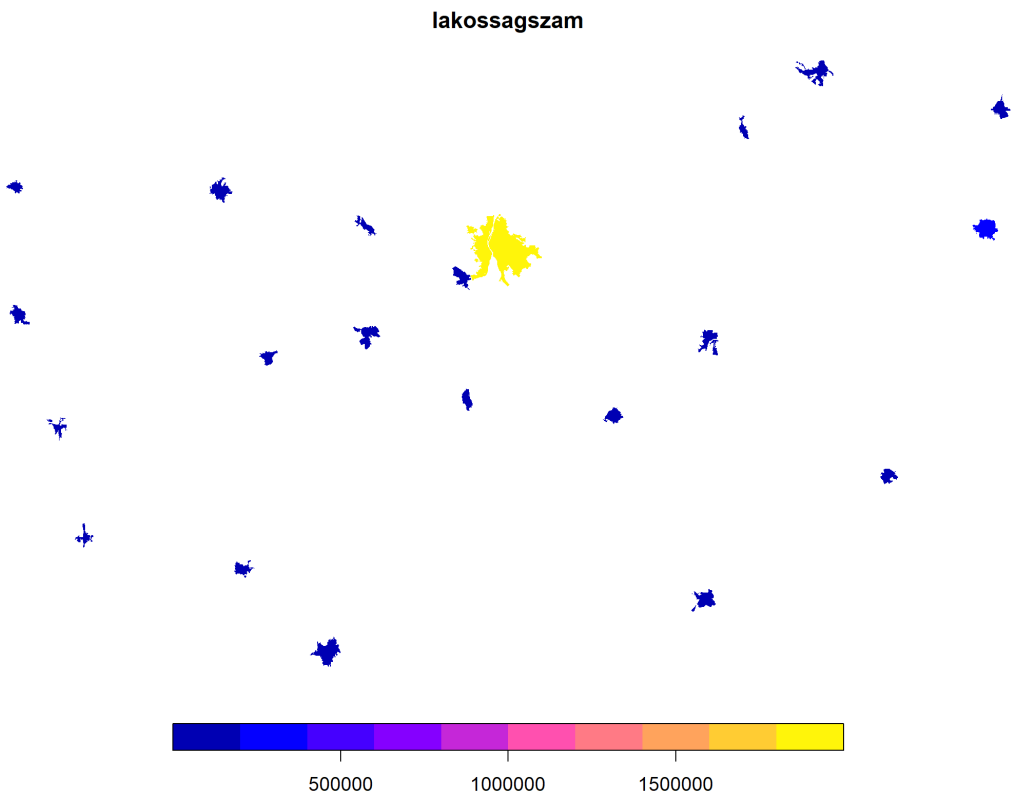
- Olvasd be a varosok.RData fájlt,
- majd ábrázold körvonal nélkül (átlátszó körvonallal) a lakossag nevű oszlopot “lakossagszam” címmel, a jelmagyarázatot alulra helyezve.



9. feladat (órai) – megoldás

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

```
plot(varosok[, "lakossag"], border = "transparent", main =
 "lakossagszam", key.pos = 1)
```

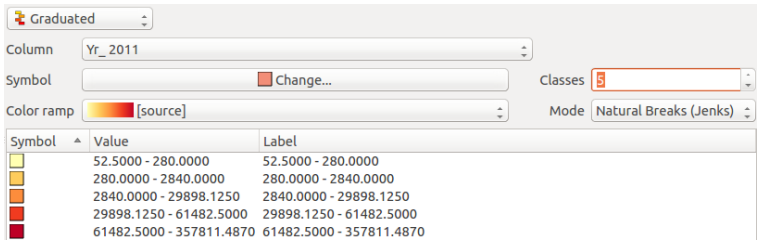


Színskála és jelmagyarázat

Az `nbreaks` és a `breaks` paraméterekkel testreszabható a színskála.
A skála felosztására több módszer használható:

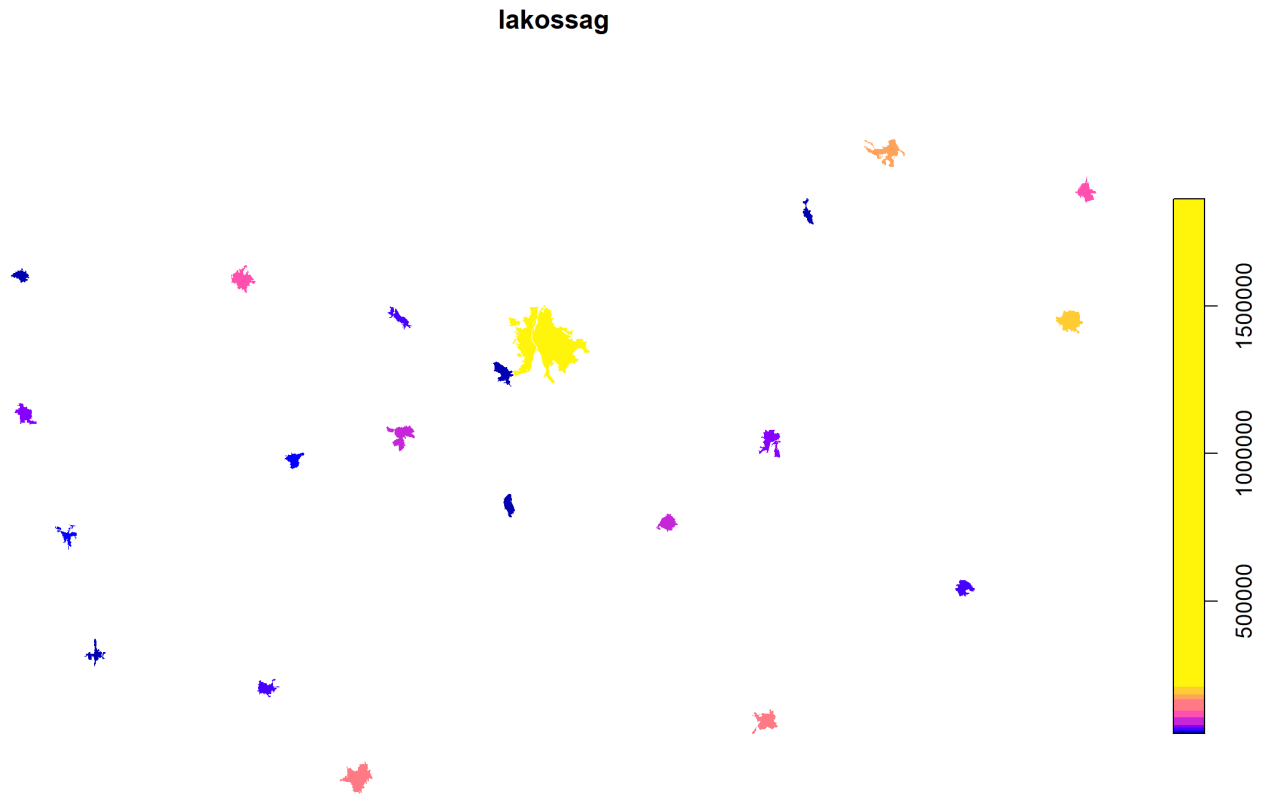
- `equal`: egyenlő felosztás
- `pretty`: szép, kerek számoknál vág (nem garantált, hogy pont `nbreaks` számú kategóriába oszt)
- `quantile`: az eloszlásfüggvény szerinti felosztás (minden kategóriába ugyanannyi alakzat kerül)
- `jenks`: Jenks-féle természetes felosztás (ArcMapben és QGISben is van hasonló)

Vagy mi magunk megadhatjuk a vágópontokat.



Színskála és jelmagyarázat

```
plot(varosok[, "lakossag"], border = "transparent", breaks
 = "jenks")
```

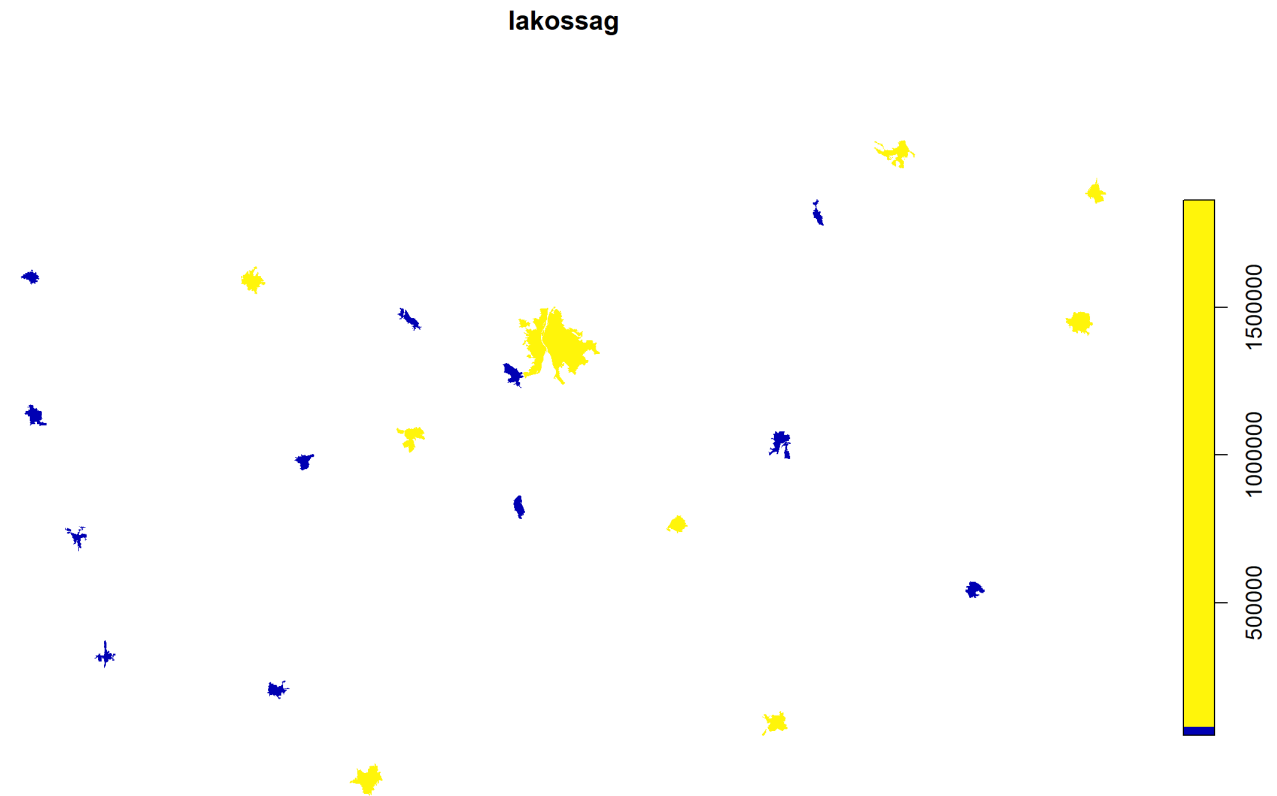


Színskála és jelmagyarázat

```
plot(varosok[, "lakossag"], border = "transparent", breaks
 = "jenks", nbreaks = 3)
```

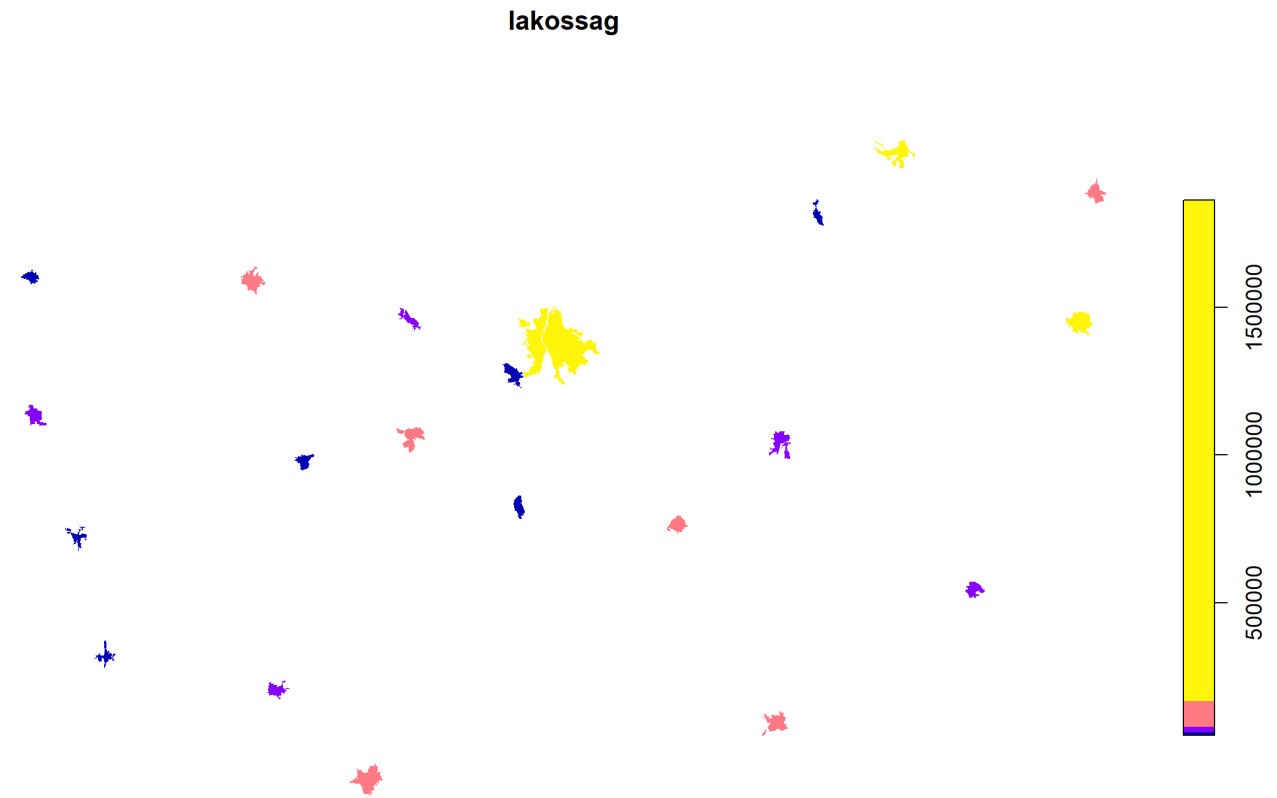
Színskála és jelmagyarázat

```
plot(varosok[, "lakossag"], border = "transparent", breaks
 = "quantile", nbreaks = 2)
```



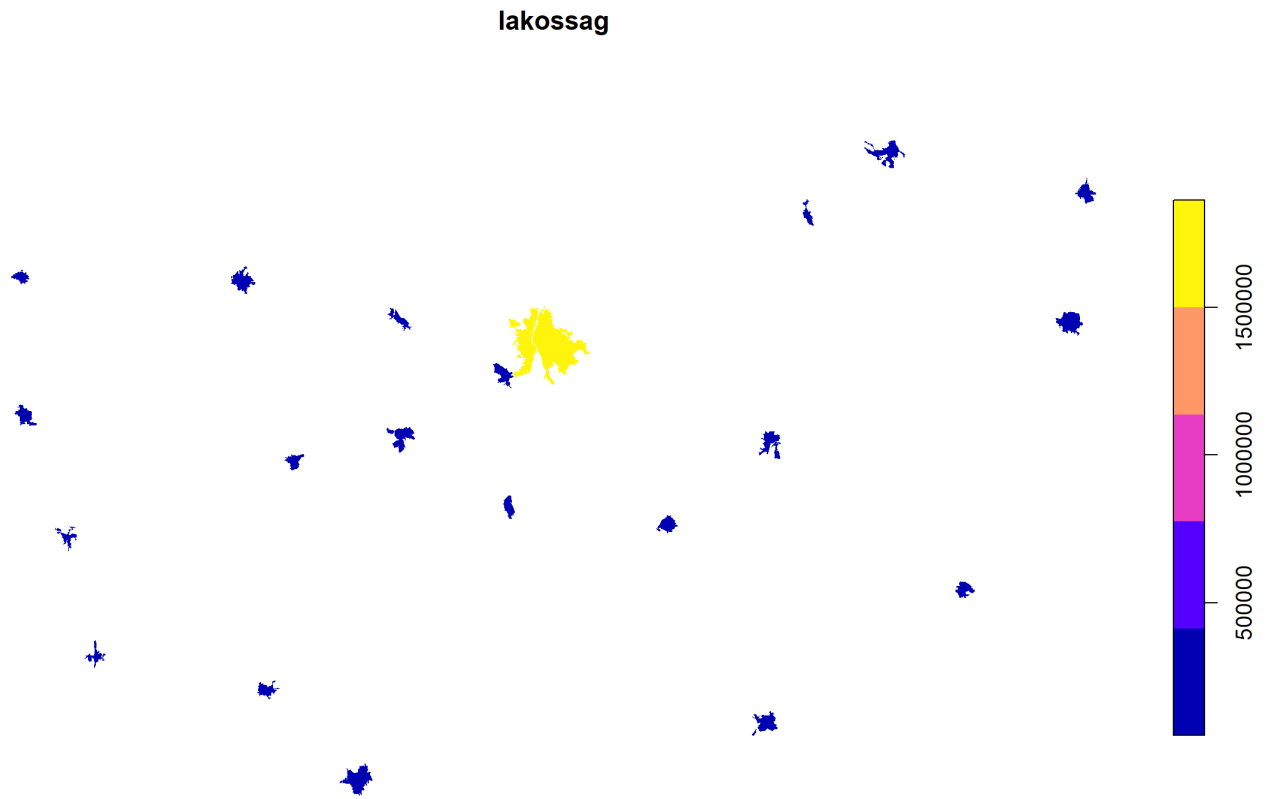
Színskála és jelmagyarázat

```
plot(varosok[, "lakossag"], border = "transparent", breaks
 = "quantile", nbreaks = 4)
```



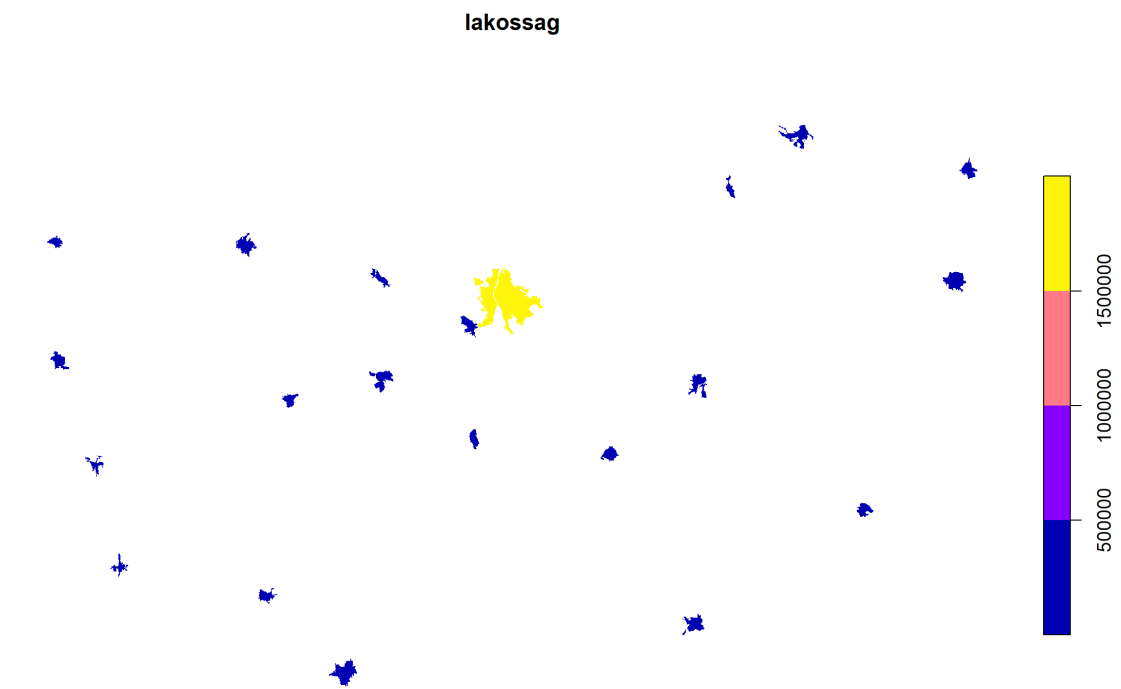
Színskála és jelmagyarázat

```
plot(varosok[, "lakossag"], border = "transparent", breaks
 = "equal", nbreaks = 5)
```



Színskála és jelmagyarázat

```
plot(varosok[, "lakossag"], border = "transparent", breaks
 = "pretty", nbreaks = 5)
```



Az `equal` képes volt öt kategóriára osztani, de a `pretty` nem talált megfelelően kerek számokat, ezért négy kategóriára osztotta a skálát.

Színskála és jelmagyarázat

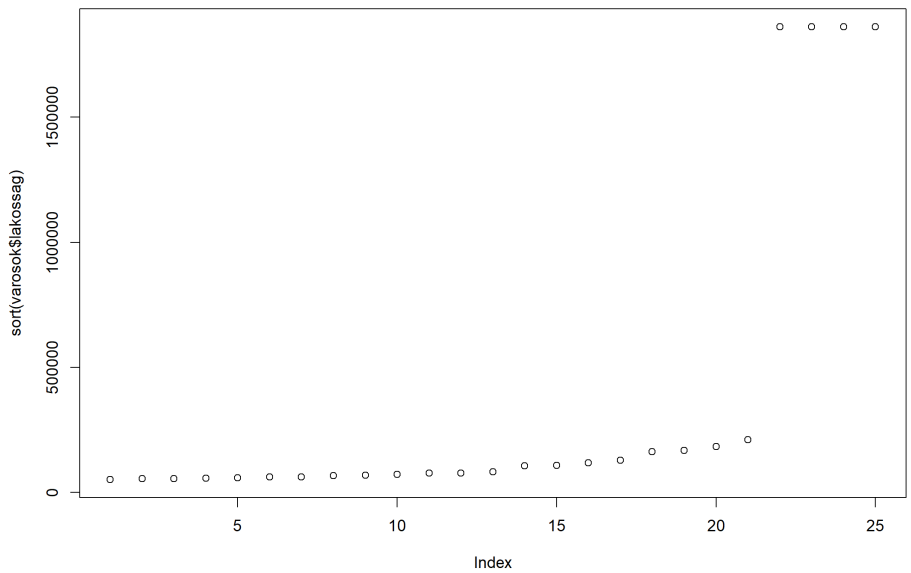
Mi is feloszthatjuk a skálát.

- egy számvektorral
- nem csak a töréspontokat, hanem a skála két szélső értékét is meg kell adni
- ami a szélső értékeken kívül esik, az nem kerül rá a térképre (mintha `NA` lenne)
- ilyenkor az `nbreaks` paraméternek nincs hatása

Színskála és jelmagyarázat

Nézzük meg, mi a lakosságszámok eloszlása!
(A `plot()` függvény számvektorokra is alkalmazható, és persze még sok minden másra is.)

```
plot(sort(varosok$lakossag))
```



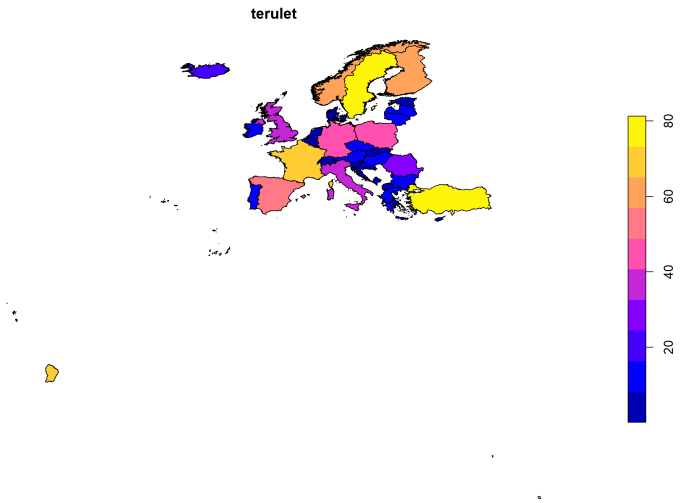
Eléggé vízfejű az ország, Budapest négy poligonja kiugrik, a többi mind 500 ezer alatt.

Színskála és jelmagyarázat

```
plot(varosok[, "lakossag"], border = "transparent", breaks
 = c(0, 100000, 200000, 500000, 2000000))
```

10. feladat (órai)

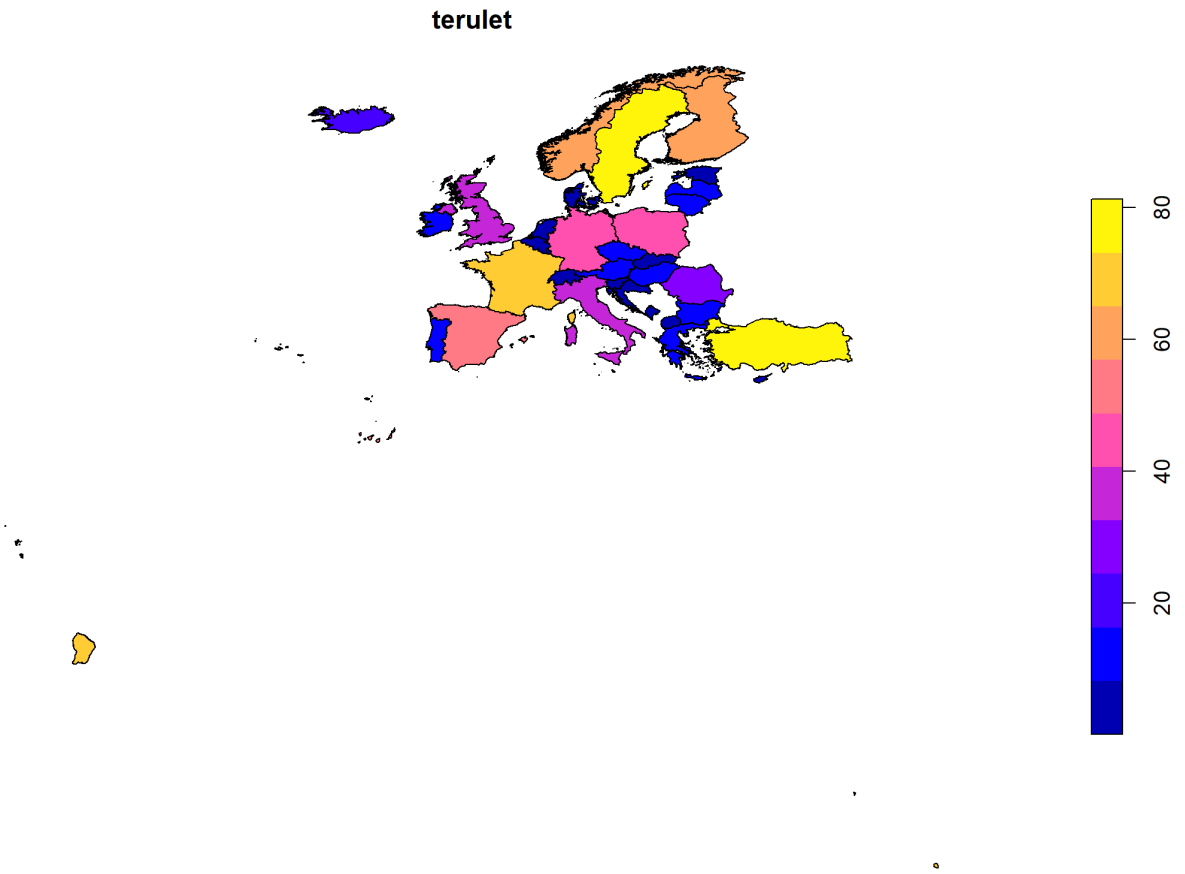
- Olvasd be az Európai Unió országait tartalmazó fájlt (orszag_eu.RData),
- és jelenítsd meg azt a terület szerint színezve, 10, egyenlő részre osztva az ábrázolandó tartományt.
- Ezután készíts újabb két térképet:
  - egyszer az országok 20–20%-a kerüljön egy csoportba,
  - másszor itt legyenek a vágópontok: 10, 20 és 40. Ne felejtsd el az ábrázolandó tartomány széleit is megadni.



10. feladat (órai) – megoldás

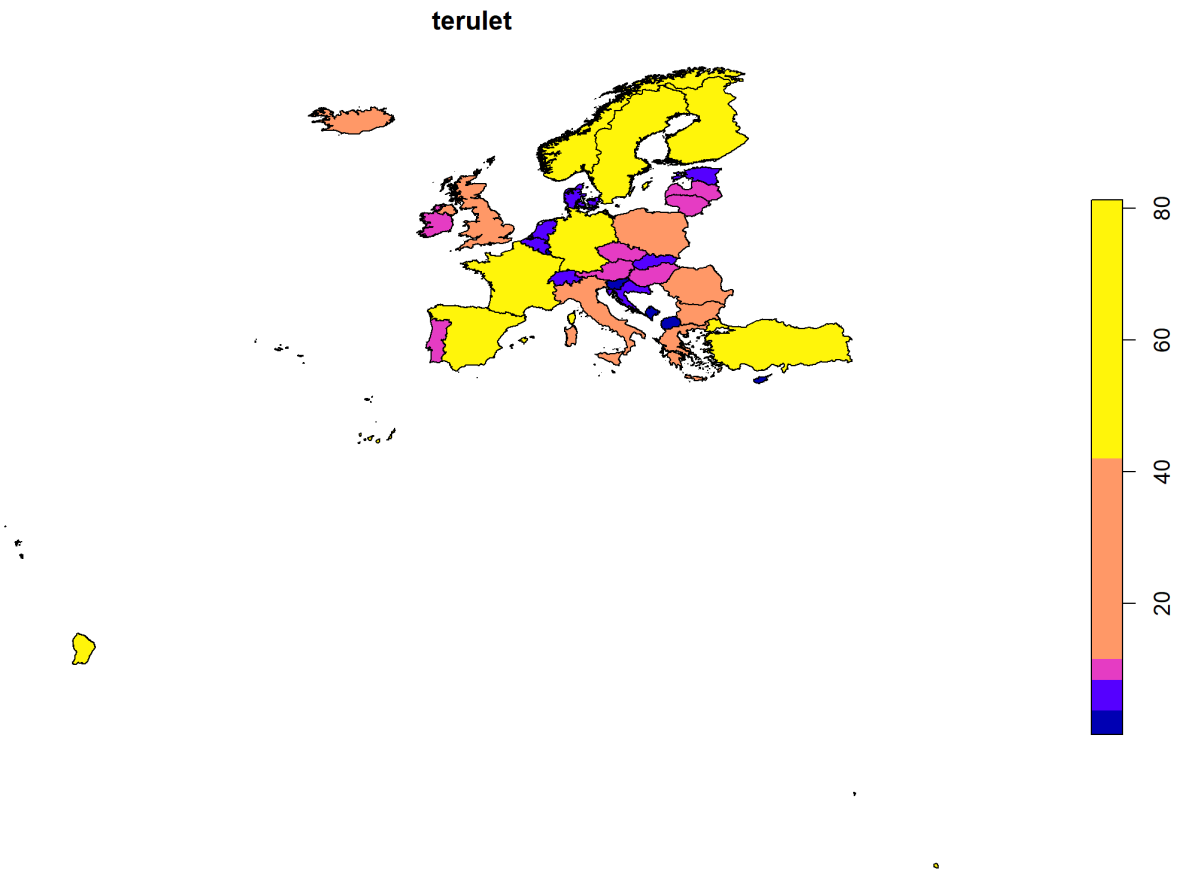
```
load("orszag_eu.RData")
```

```
plot(orszag_eu[, "terulet"], breaks = "equal", nbreaks = 10)
```



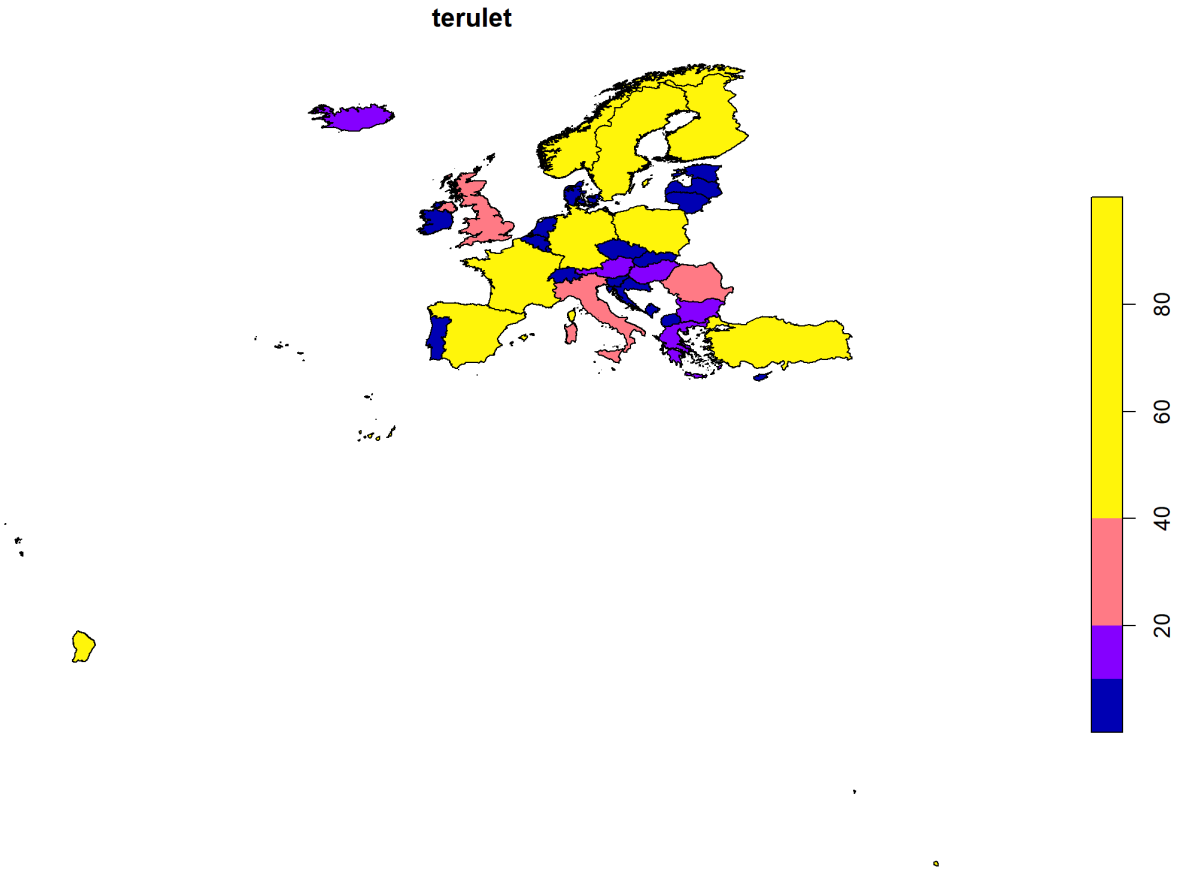
10. feladat (órai) – megoldás

```
plot(orszag_eu[, "terulet"], breaks = "quantile", nbreaks
 = 5)
```



10. feladat (órai) - megoldás

```
plot(orszag_eu[, "terulet"], breaks = c(0, 10, 20, 40, 100))
```

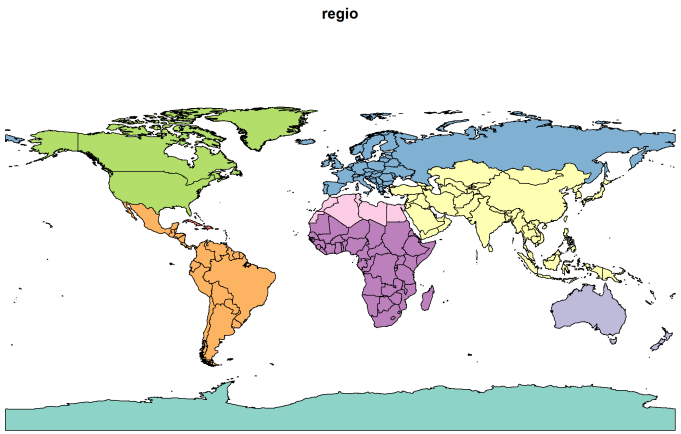


Színskála és jelmagyarázat

Alapértelmezett színskála:

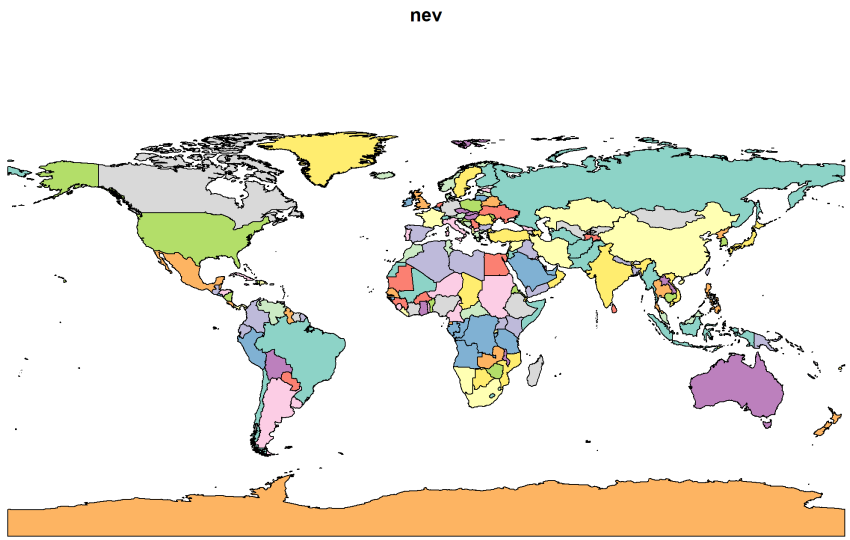
- kategorikus (`factor`): diszkrét színek
- szöveget kategorikusként kezeli
- számok: kék–lila–sárga átmenetes színskála
- felülírható a `pal` paraméterrel

```
load("orszagok_osszes.RData")
plot(orszagok_osszes[, "regio"], key.pos = NULL)
```



Színskála és jelmagyarázat

```
plot(orszagok_osszes[, "nev"])
```

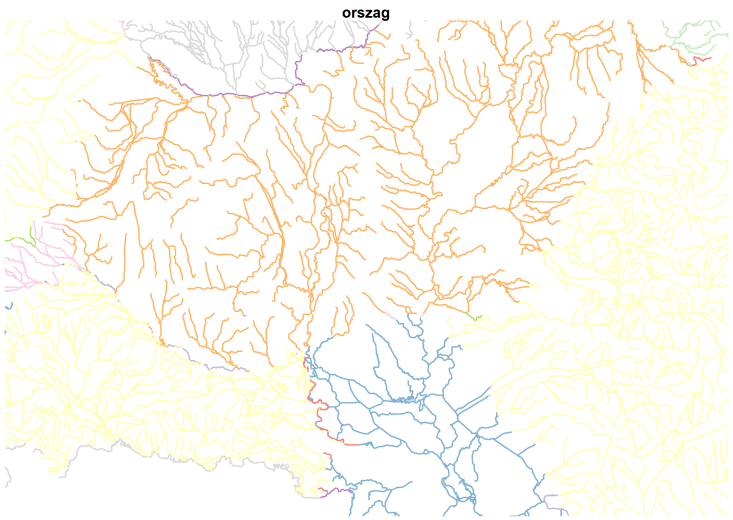


Ha túl sok a kategória, akkor több kategória kapja ugyanazt a színt (és nincs automatikus jelmagyarázat).
Ez nem szerencsés...
Ilyenkor jól jöhet a `randomcoloR` csomag.

Színskála és jelmagyarázat

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

```
plot(folyok[, "orszag"], xlim = c(16, 23), ylim = c(45,
 48), lwd = 2)
```

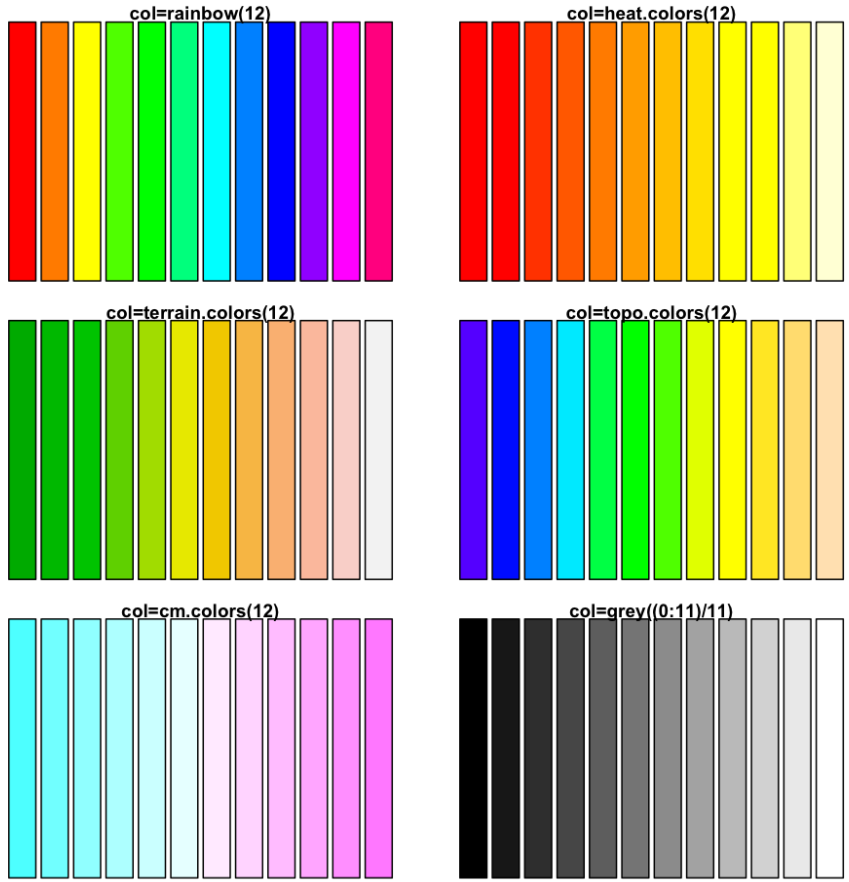


Hiába kevés a kategória az ábrázolt területen, összességében mégiscsak sok $\rightarrow$ nem készül jelmagyarázat.

Színskála és jelmagyarázat

A `pal` paraméter kétféle módon használható:

- színskála megadása konkrét színekkel (hexadecimális szövegként)
- ekkor pont `nbreaks` számú szín kell
- vagy egy olyan függvény nevét kell megadni, ami színskálát generál
- domborzathoz jók pl. a `topo.colors` vagy a `terrain.colors`



Színskála és jelmagyarázat

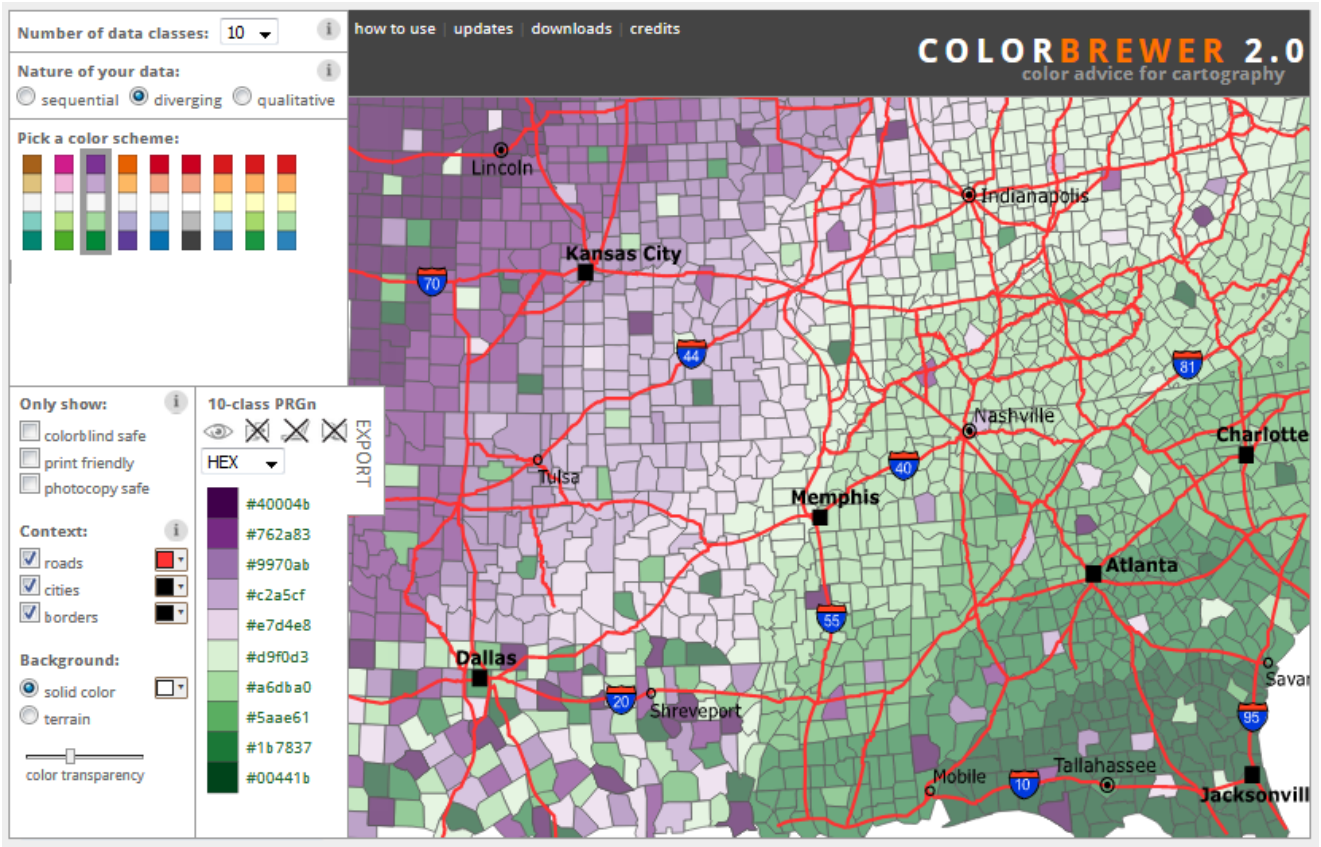
A beépített színskálákon túl sajátot is létrehozhatunk:

- a `colorRampPalette()` függvénnyel tetszőleges színekből
- az `RColorBrewer` csomag segítségével, sokféle palettából

És persze még iszonatosan sok színskála létezik a különböző R-csomagokban.
Részletes lista: itt.
És az összes paletta elérhető a `paletteer` csomaggal...

Színskála és jelmagyarázat

A következő honlapon interaktívan is próbálgathajuk: colorbrewer2.org

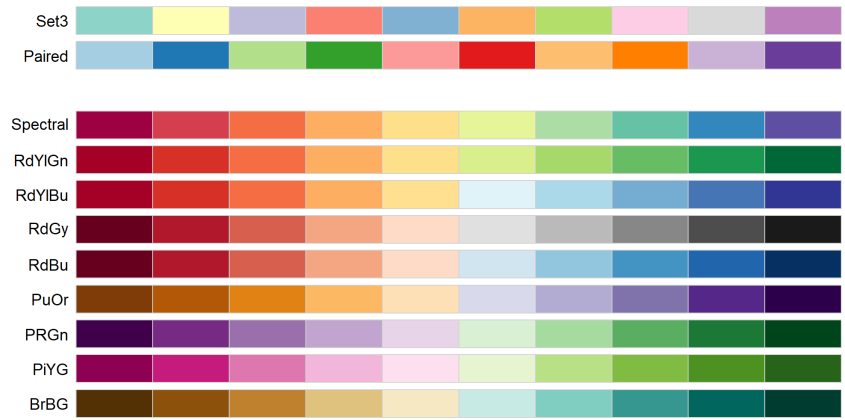


Színskála és jelmagyarázat

Olvassuk ki a kategorikus oszlop szintjeinek számát: `nlevels(x)`
Majd jelenítsük meg azon palettákat, amik képesek éppen ennyi színt ábrázolni: `display.brewer.all(n)`

```
regiok_szama <- nlevels(orszagok_osszes$regio)
library(RColorBrewer)
```

```
display.brewer.all(n = regiok_szama)
```



Színskála és jelmagyarázat

Ha kiválasztottuk a `name` nevű palettát, az `n` darab színt a `brewer.pal(n, name)` függvénnyel tudjuk létrehozni.

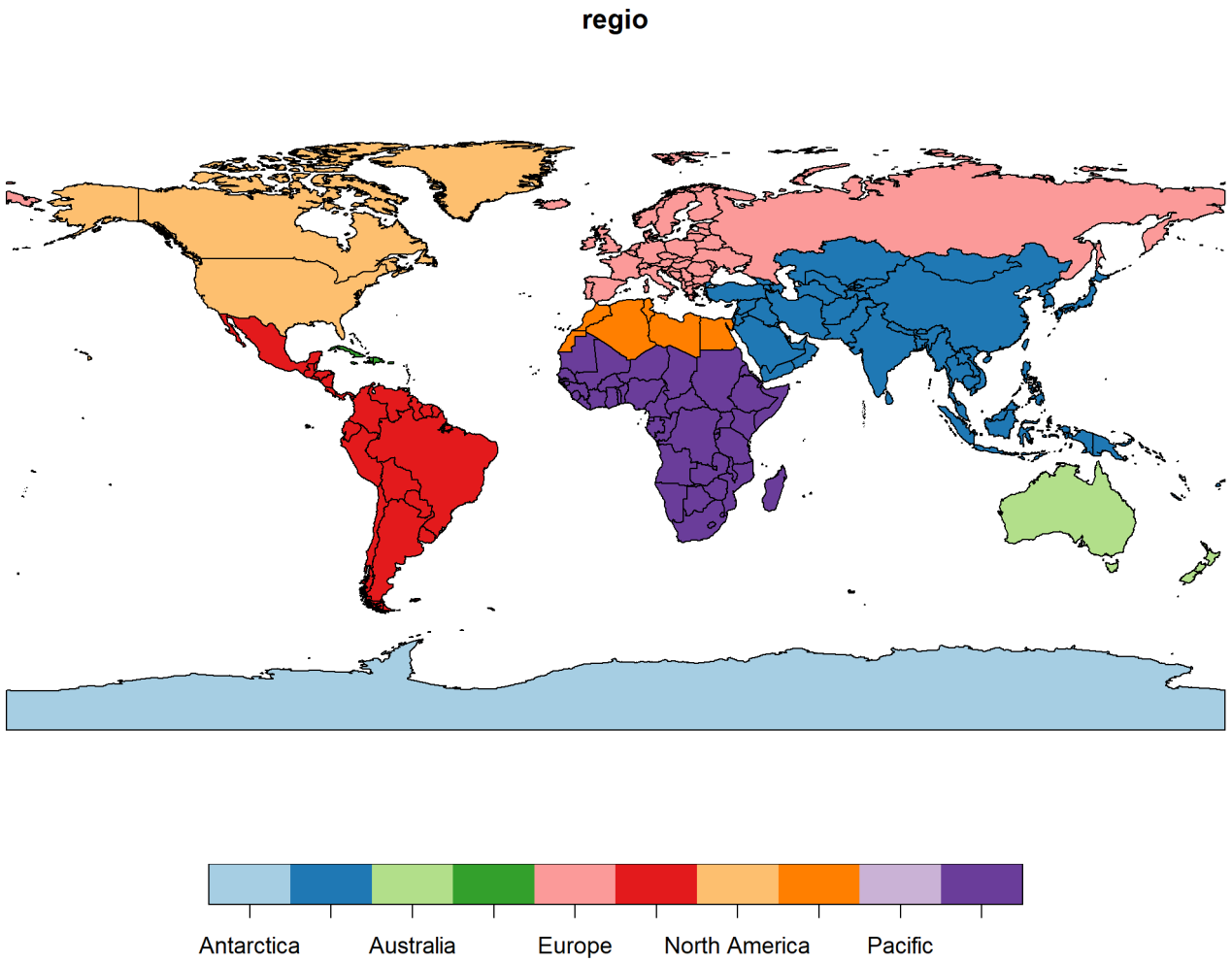
```
szinskala <- brewer.pal(n = regiok_szama, name = "Paired")
szinskala
```

```
 [1] "#A6CEE3" "#1F78B4" "#B2DF8A" "#33A02C" "#FB9A99"
 [6] "#E31A1C" "#FDBF6F" "#FF7F00" "#CAB2D6" "#6A3D9A"
```

A színskála egy szövegvektor, ami hexadecimális (`#RRGGBB`) színeket tartalmaz.

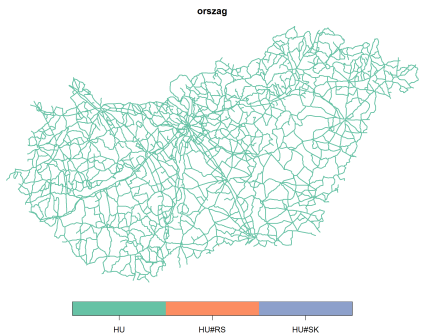
Színskála és jelmagyarázat

```
plot(orszagok_osszes[, "regio"], pal = szinskala)
```



11. feladat (órai)

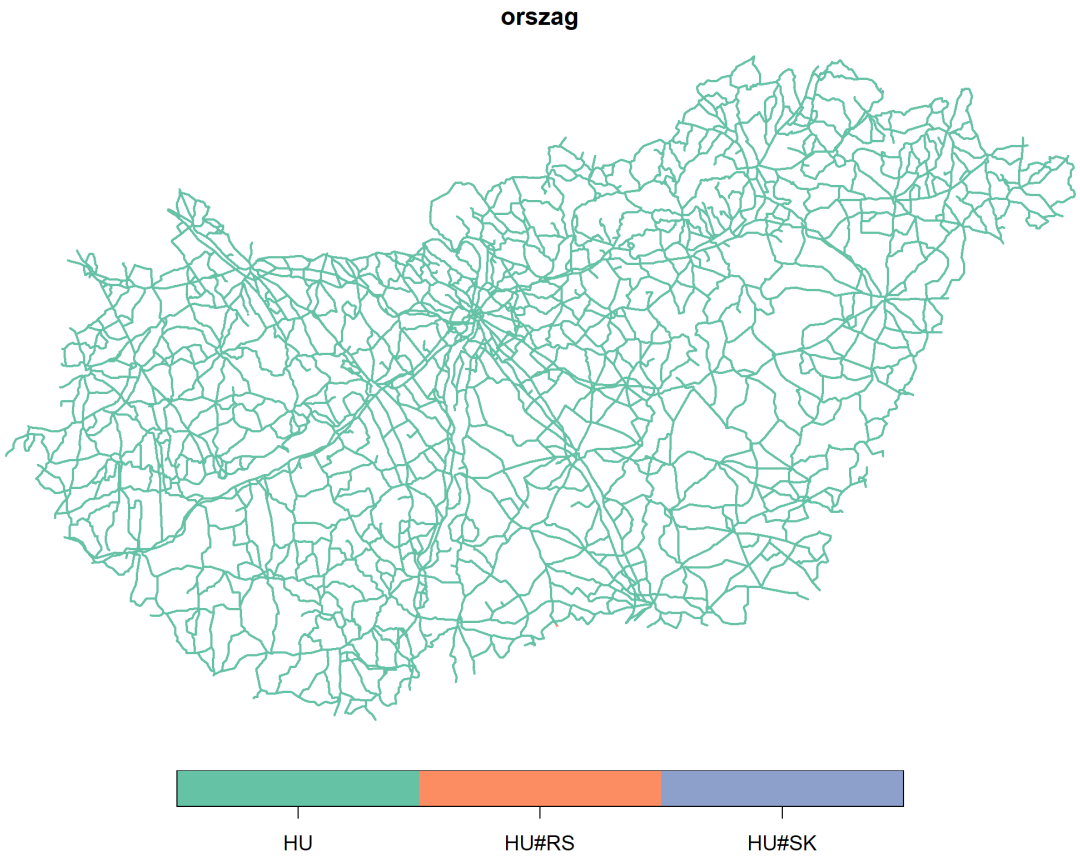
- Olvasd be az utak.RDatát,
- majd jelenítsd meg azt az “orszag” tulajdonsága szerint színezve, kétszeres vastagságú vonallal, a jelmagyarázat alulra kerüljön.
- Rögzítsd az orszagok_szama változóba, hogy ennek a tulajdonságnak mint kategorikus változónak hány szintje van.
- Hozz létre egy színskálát a “Reds” nevű RColorBrewer-palettából, aminek pont ennyi színe van.
- Mi a színskála osztálya (típusa)?
- Jelenítsd meg újra ezt a tulajdonságot, immáron az új színskála megfordítottját használva (a jelmagyarázat megint alulra kerüljön).



11. feladat (órai) – megoldás

```
load("utak.RData")
```

```
plot(utak[, "orszag"], key.pos = 1, lwd = 2)
```

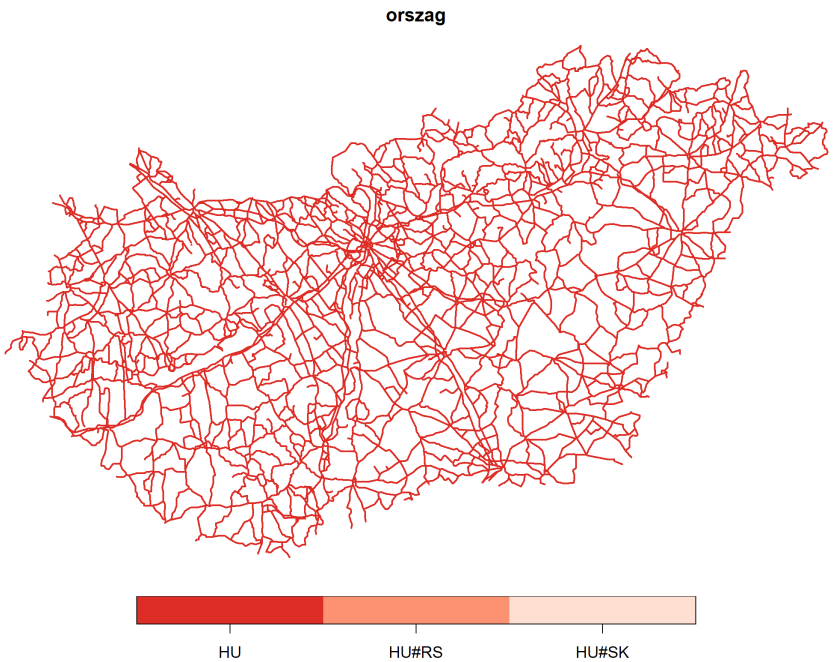


11. feladat – megoldás

```
orszagok_szama <- nlevels(utak$orszag)
szinskala <- brewer.pal(n = orszagok_szama, name = "Reds")
class(szinskala)
```

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

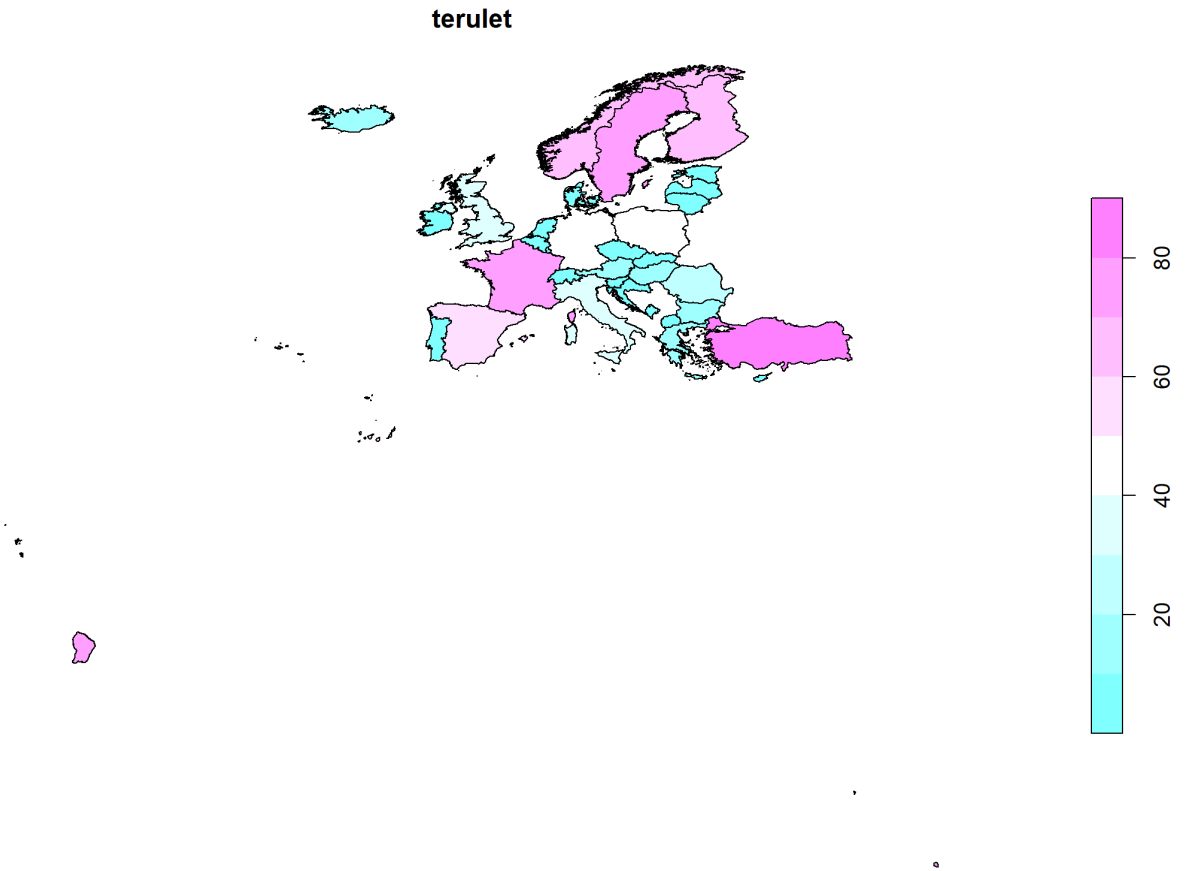
```
plot(utak[, "orszag"], pal = rev(szinskala), key.pos = 1,
 lwd = 2)
```



Színskála és jelmagyarázat

Színskálakészítő függvény átadása a `pal` paraméternek:

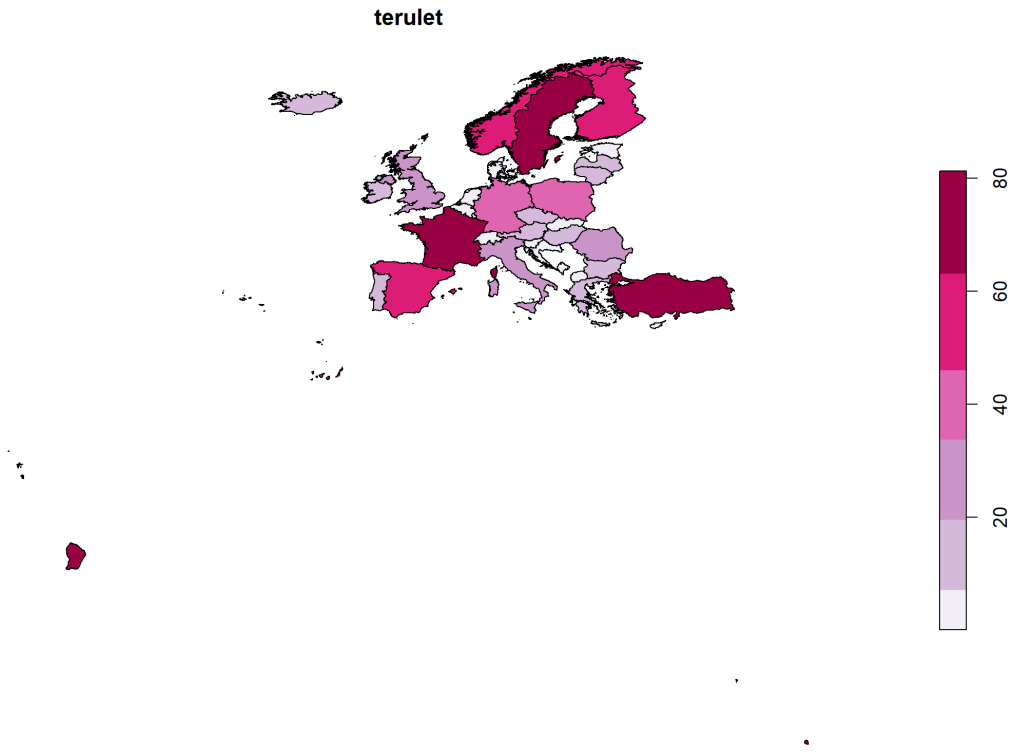
```
plot(orszag_eu[, "terulet"], pal = cm.colors)
```



Színskála és jelmagyarázat

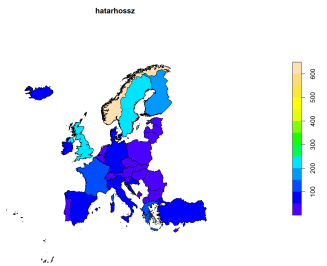
A `pal` paraméter természetesen kombinálható az `nbreaks`-szel és a `breaks`-szel.

```
szinskala <- brewer.pal(n = 6, name = "PuRd")
plot(orszag_eu[, "terulet"], pal = szinskala, nbreaks = 6,
 breaks = "jenks")
```



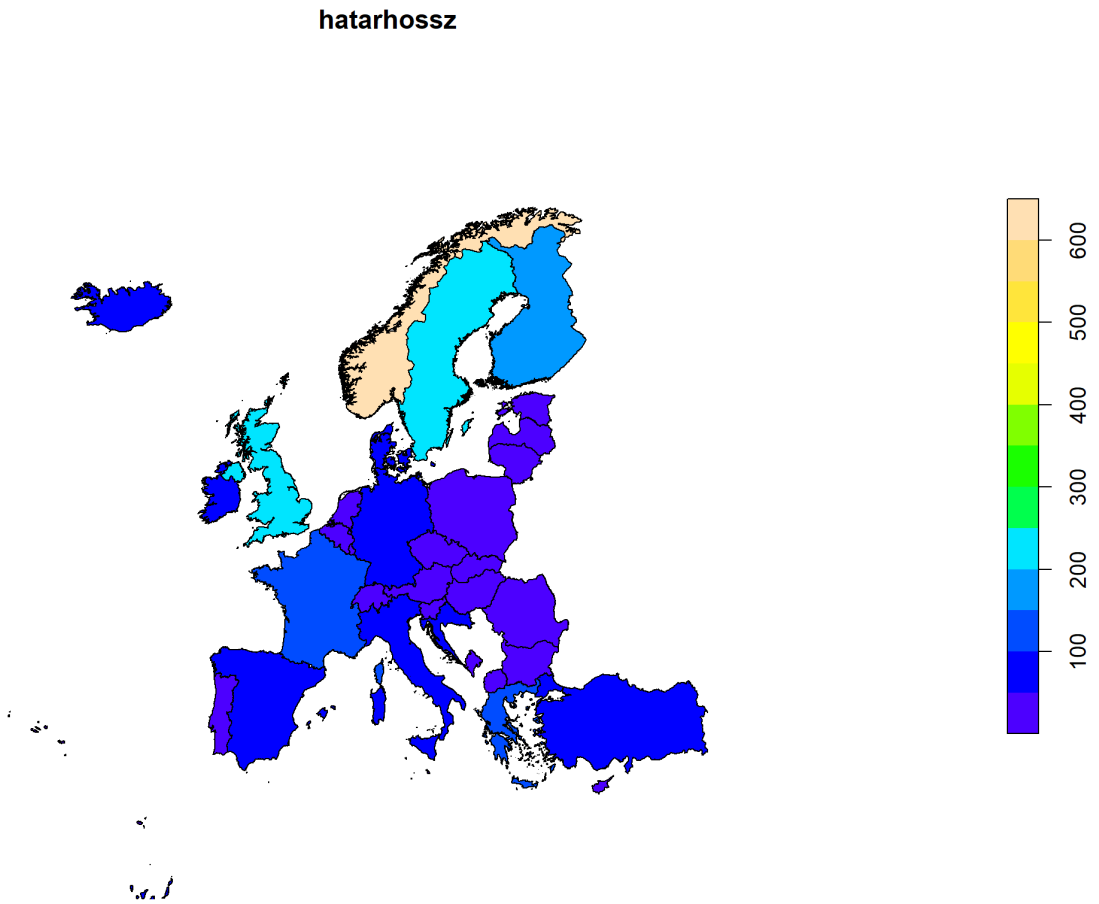
12. feladat (házi)

- Ábrázold az EU országait (orszag_eu) a határok hossza (“hatarhossz” oszlop) alapján színezve a “topo.colors” színskála színeivel a következő hosszúsági (x) és szélességi (y) fokok között:
  - hosszúság: –20 – 40,
  - szélesség: 30 – 80.
- Hozz létre 8 színből álló skálát az “RdYIGn” (Red, Yellow, Green) nevű RColorBrewer-palettából,
- majd újra ábrázold a megadott koordináta-tartományon belül a határhosszokat. Az ábrázolandó tartományt az eloszlásfüggvény kvantilisei alapján vágjad.
- Ne felejtsd el megadni, hogy hány részre bontanád a tartományt!



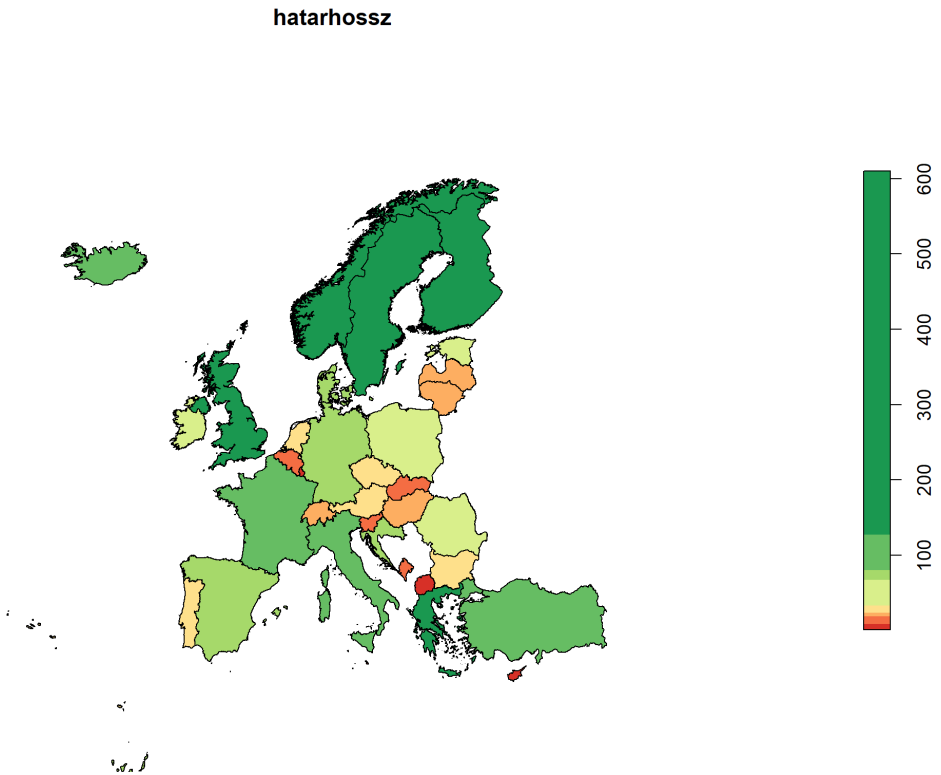
12. feladat (házi) – megoldás

```
plot(orszag_eu[, "hatarhossz"], pal = topo.colors, xlim =
 c(-20, 40), ylim = c(30, 80))
```



12. feladat (házi) – megoldás

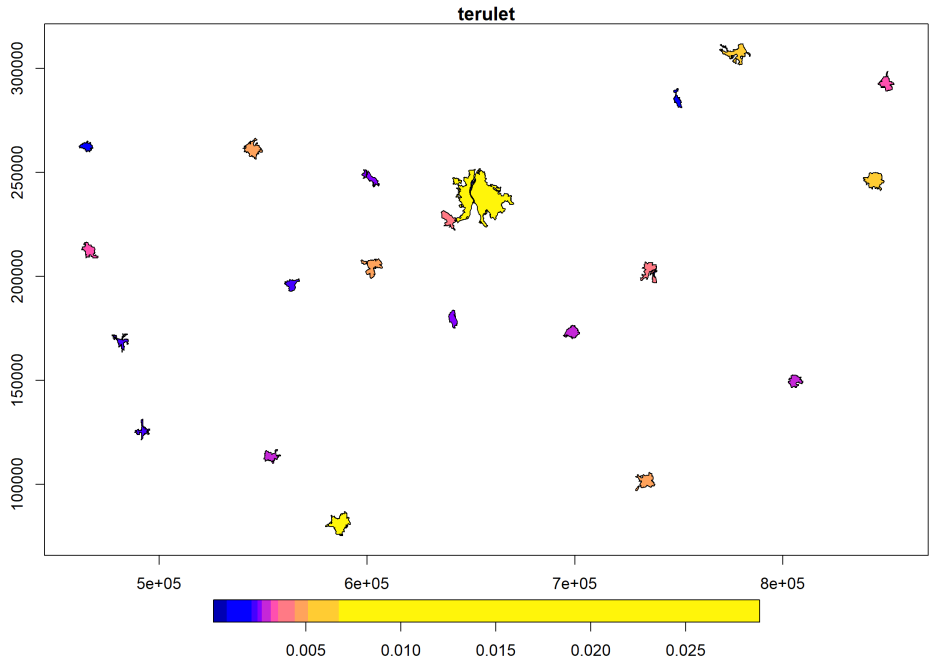
```
szinskala <- brewer.pal(n = 8, name = "RdYlGn")
plot(orszag_eu[, "hatarhossz"], pal = szinskala, nbreaks =
 8, breaks = "quantile", xlim = c(-20, 40), ylim = c(30,
 80))
```



Tulajdonságok és geometriák ábrázolása együtt

Tulajdonságok mellett geometriákat is ábrázolhatunk.

```
plot(varosok[, "terulet"], breaks = "quantile", axes =
  TRUE, key.pos = 1)
plot(utak_geometria, add = TRUE)
```

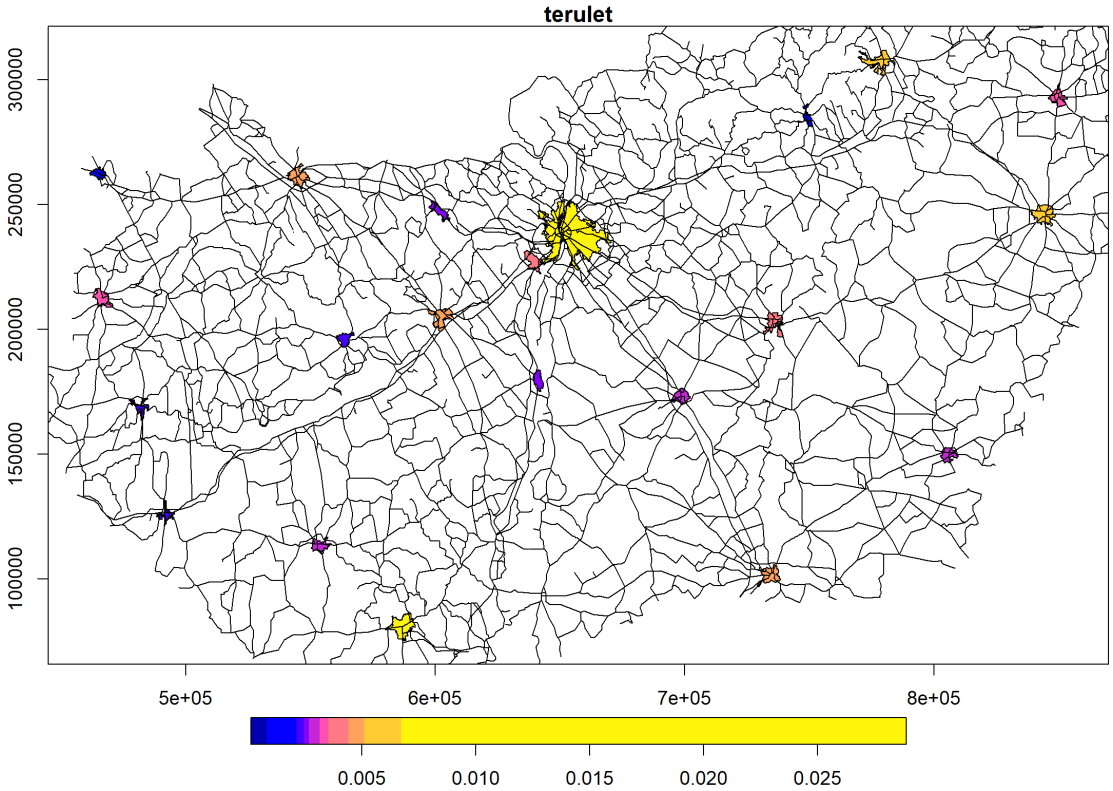


De a tulajdonságok ábrázolásakor a képvászon alapértelmezetten lezár...

Tulajdonságok és geometriák ábrázolása együtt

Ezért a `reset` paramétert `FALSE`-ra kell állítani.

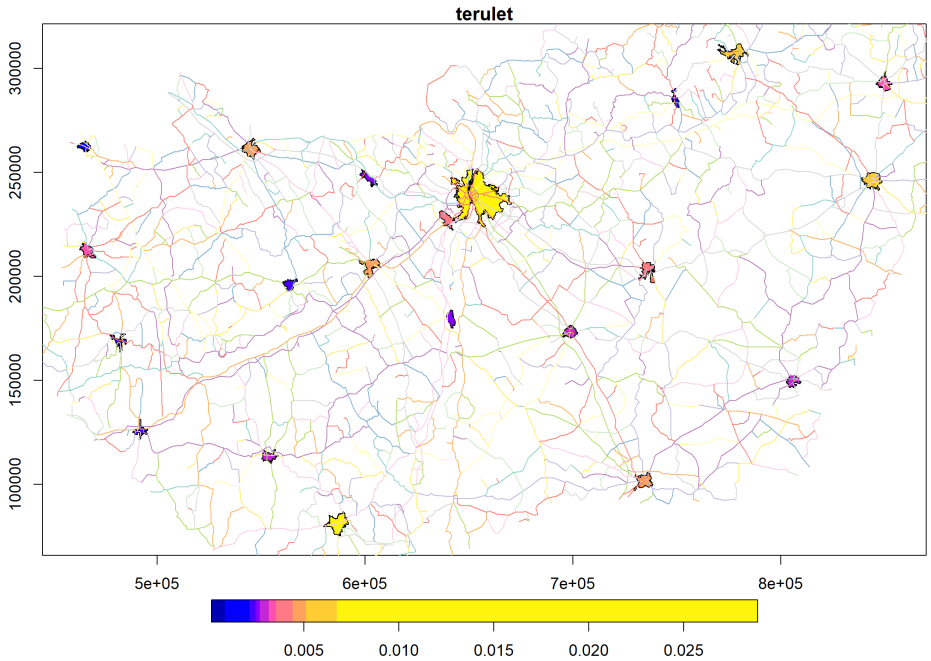
```
plot(varosok[, "terulet"], breaks = "quantile", axes =
  TRUE, key.pos = 1, reset = FALSE)
plot(utak_geometria, add = TRUE)
```



Tulajdonságok és geometriák ábrázolása együtt

Több, tulajdonság szerint színezett térképet is egymásra vetíthetünk:

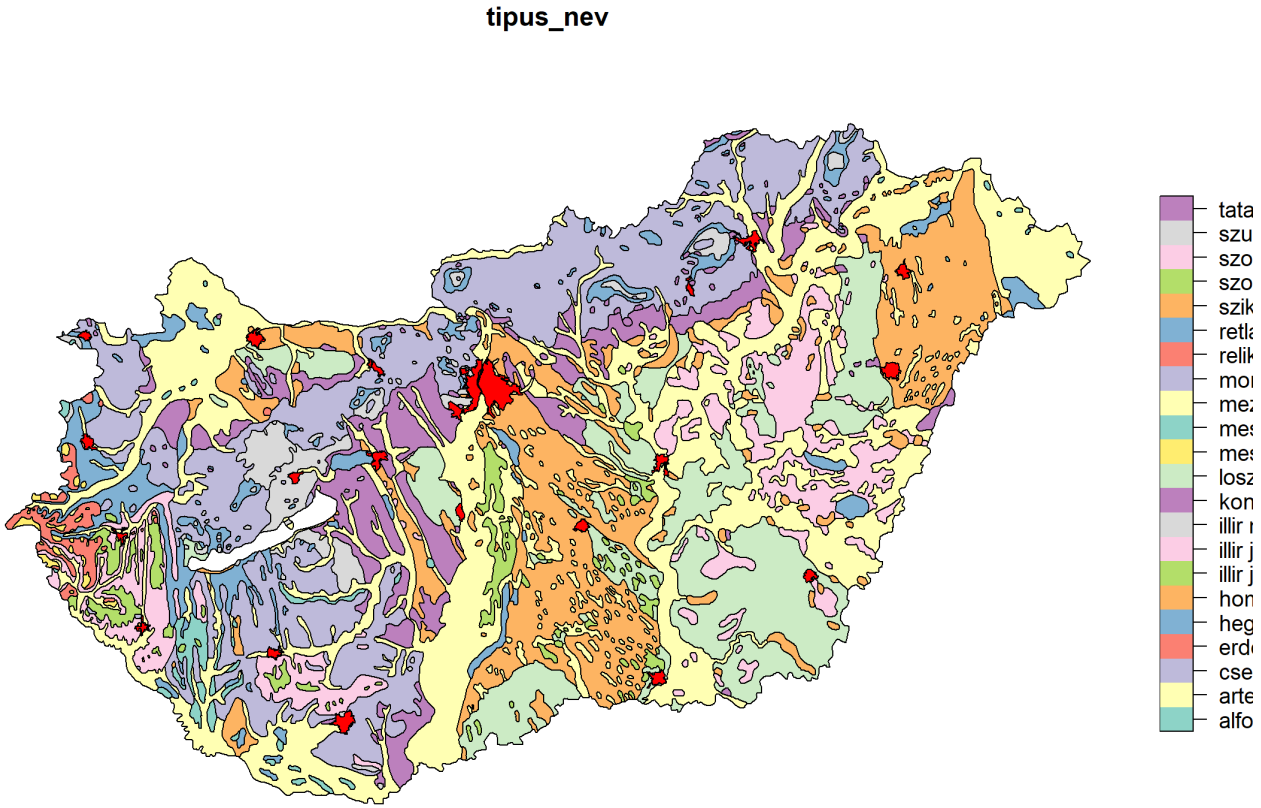
```
plot(varosok[, "terulet"], breaks = "quantile", axes =
  TRUE, key.pos = 1, reset = FALSE)
plot(utak[, "nev"], add = TRUE)
```



De csak az első kap jelmagyarázatot!

13. feladat (órai)

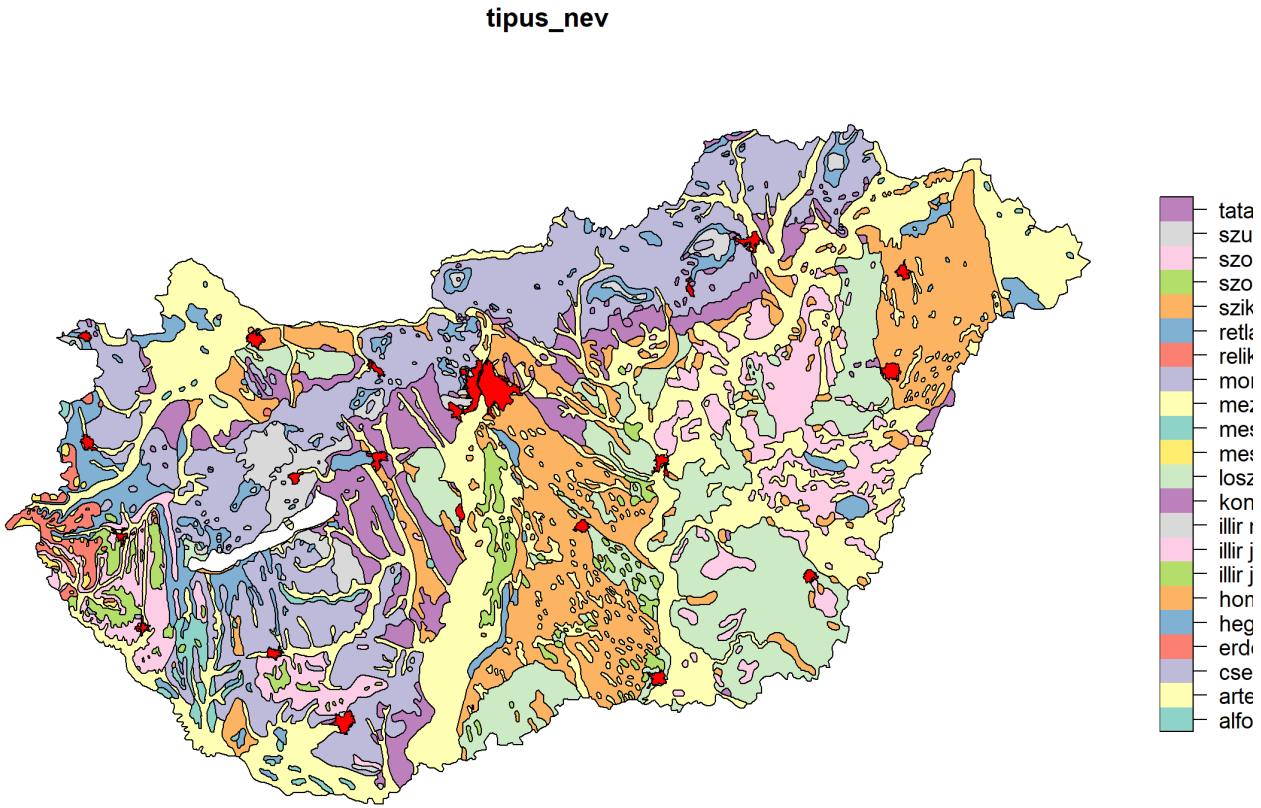
- Olvasd be a zolyomi.RData fájlt,
- és jelenítsd meg a poligonokat a “tipus_nev” oszlop szerint színezve.
- Add ehhez az ábrához a városok geometriáját piros kitöltő színnel.



13. feladat (órai) – megoldás

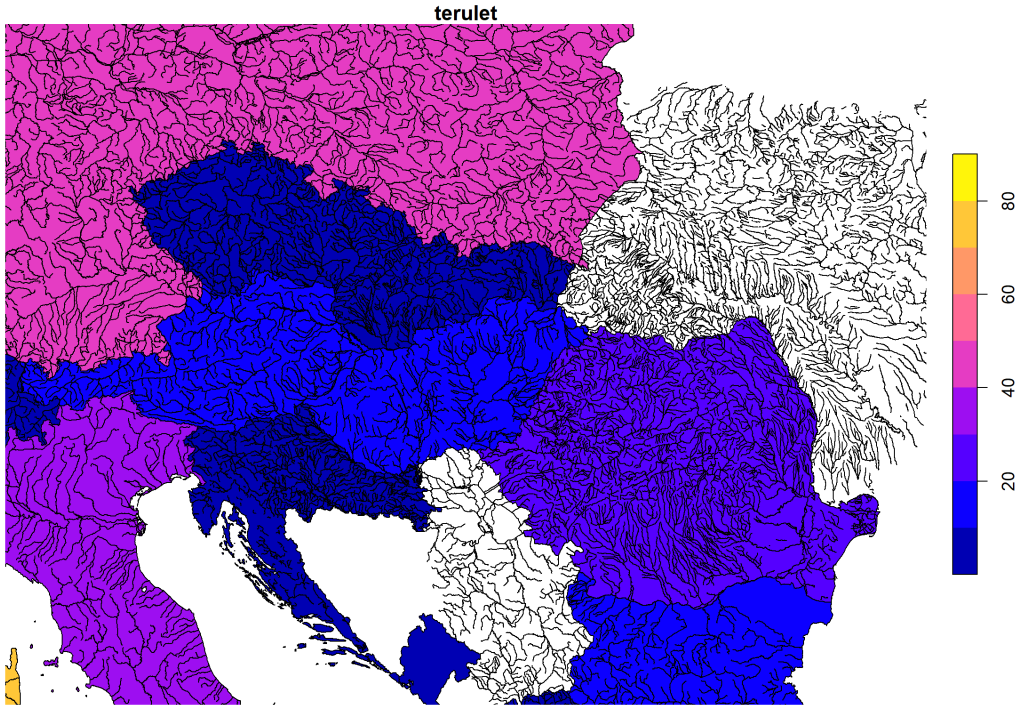
```
load("zolyomi.RData")
```

```
plot(zolyomi[, "tipus_nev"], reset = FALSE)
plot(varosok_geometria, col = "red", add = TRUE)
```



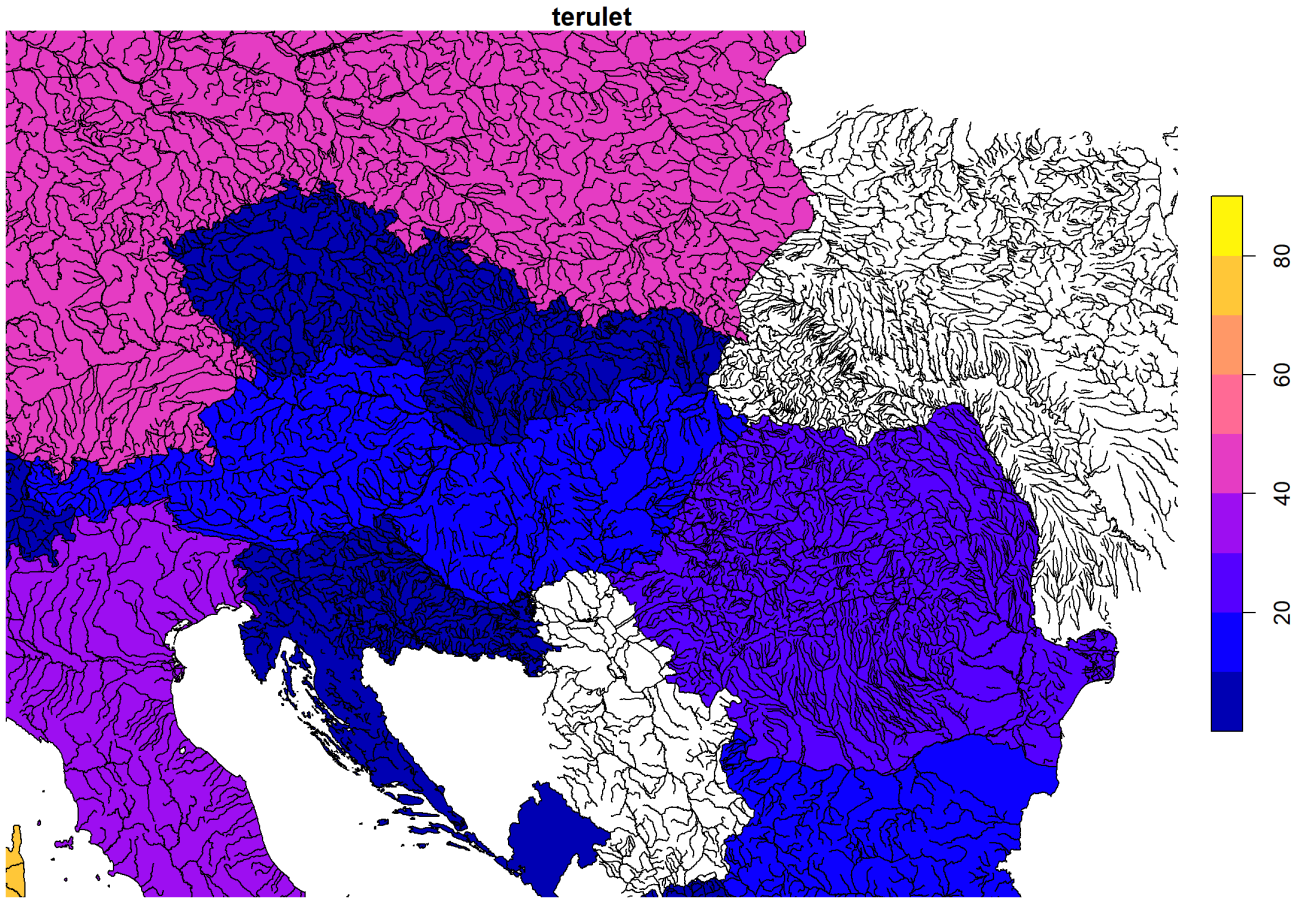
14. feladat (házi)

- Ábrázold az EU országait (orszag_eu) a terület szerint színezve, a következő hosszúsági (x) és szélességi (y) fokok között:
  - hosszúság: 10 – 30,
  - szélesség: 45 – 50.
- Jelenítsd meg rajtuk a folyók geometriáját.



14. feladat (házi) – megoldás

```
plot(orszag_eu[, "terulet"], xlim = c(10, 30), ylim =
  c(45, 50), reset = FALSE)
plot(st_geometry(folyok), add = TRUE)
```



Section 3

Kiegészítő térképi jelek

Kiegészítő térképi jelek

A kiegészítő térképi jelek segítik a térkép értelmezését, a rajta való kiigazodást.
Bizonyos alkalmazási helyzetekben bizonyos kiegészítő jelek használata elengedhetetlen!

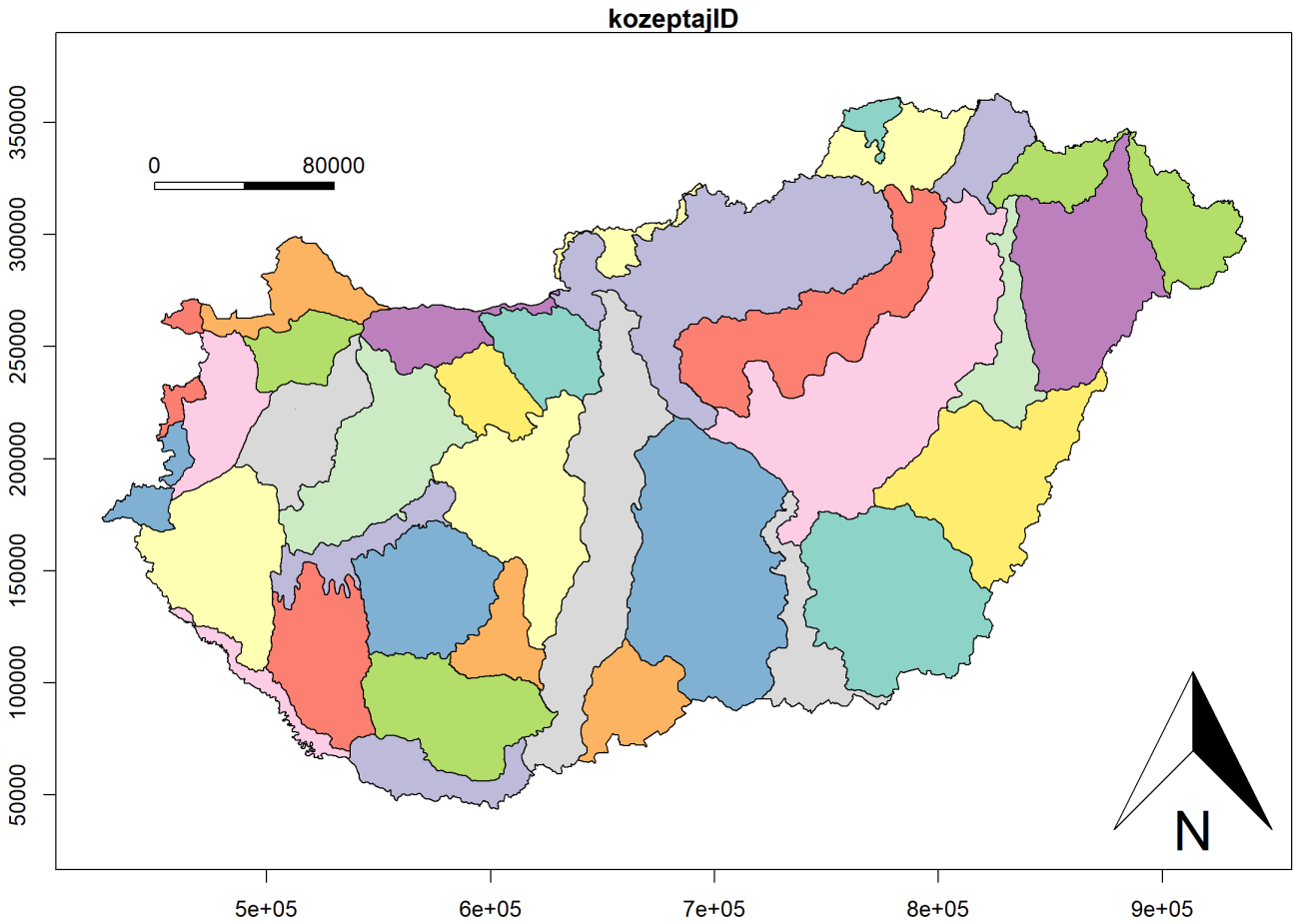
- északjel: `prettymapr::addnortharrow()` vagy `GISTools::north.arrow()`
- léptékrúd: `raster::scalebar()` vagy `prettymapr::addscalebar()`
- rácsháló: `plot(, graticule = TRUE)`

A `GISTools` csomagot sajnos már nem fejlesztik.

```
library(prettymapr)
library(raster)
load("kozeptajak.RData")
```

Kiegészítő térképi jelek

```
plot(kozeptajak, axes = TRUE, reset = FALSE)
addnortharrow(pos = "bottomright", scale = 2)
scalebar(d = 80000, xy = c(450000, 320000), type = "bar")
```



Kiegészítő térképi jelek

```
plot(kozeptajak, axes = TRUE, reset = FALSE)
addnortharrow(pos = "bottomright", scale = 2)
scalebar(d = 80000, xy = c(450000, 320000), type = "bar")
```

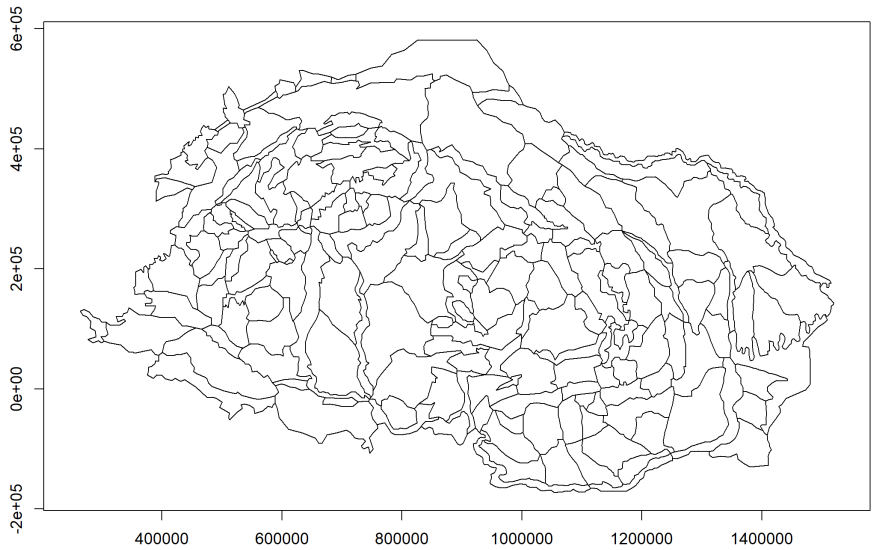
- `pos`, `xy`: helye a térképen
  - északjel: “bottomleft”, “bottomright”, “topleft” vagy “topright” (alapértelmezett)
  - léptékrúd: koordinátapár
- `scale`, `d`: nagysága/hossza
  - északjel: arányszám (alapértelmezett: 1)
  - léptékrúd: térképi egység (EOV esetén méter, WGS-84 esetén fokban - értelmetlen)
- `type`: “line” vagy “bar”

És még néhány további paraméter, amit most nem mutatok be...

Kiegészítő térképi jelek

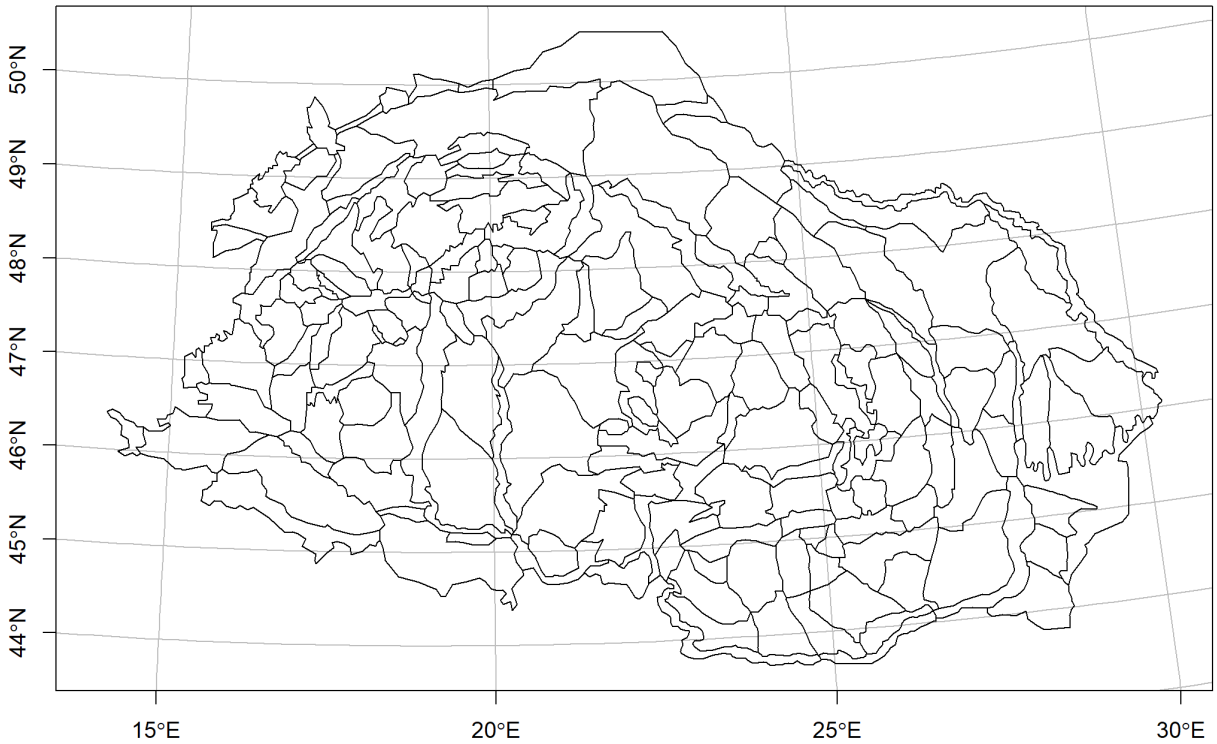
A rácshálózatot alapértelmezetten WGS-84 szerint jeleníti meg, szép, kerek fokoknál.
(Az `st_graticule()` függvénnyel tetszőlegesen testreszabott rácshálózatot is létrehozhatunk, és azt hozzáadhatjuk a térképhez.)

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



Kiegészítő térképi jelek

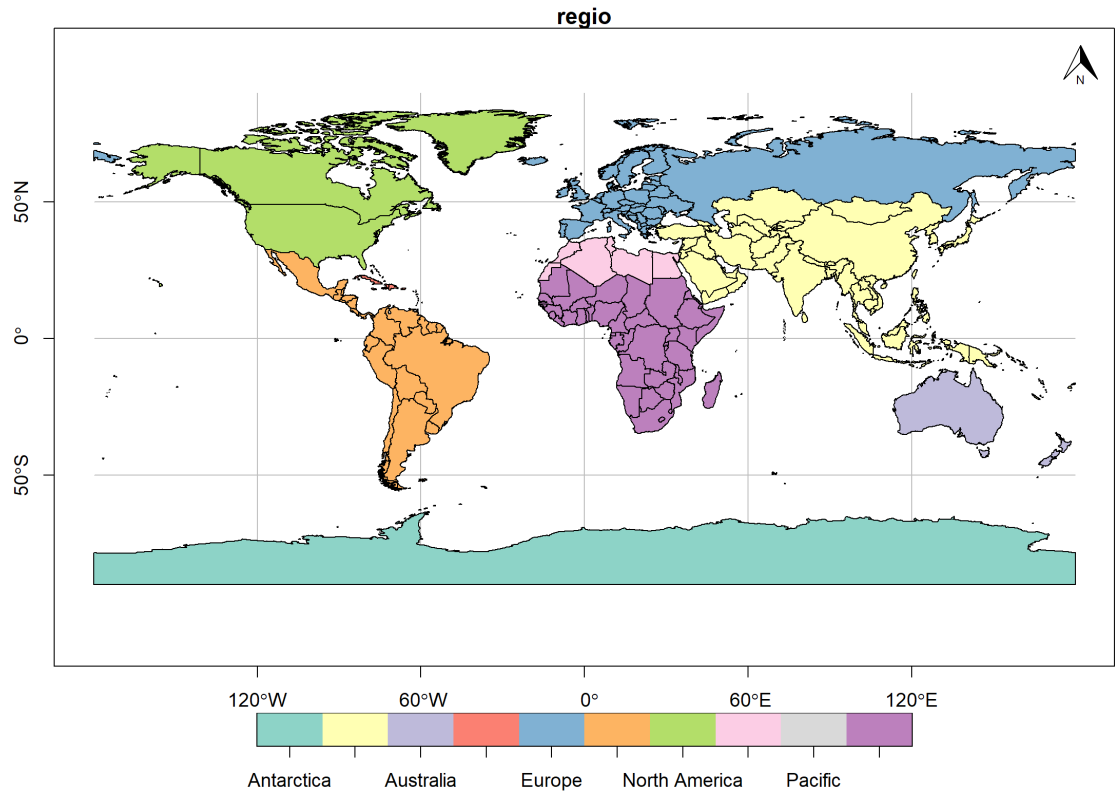
```
plot(tajbeosztas_geometria, axes = TRUE, graticule = TRUE)
```



Ilyenkor a tengelyfeliratokat is átrakja WGS-84-be.

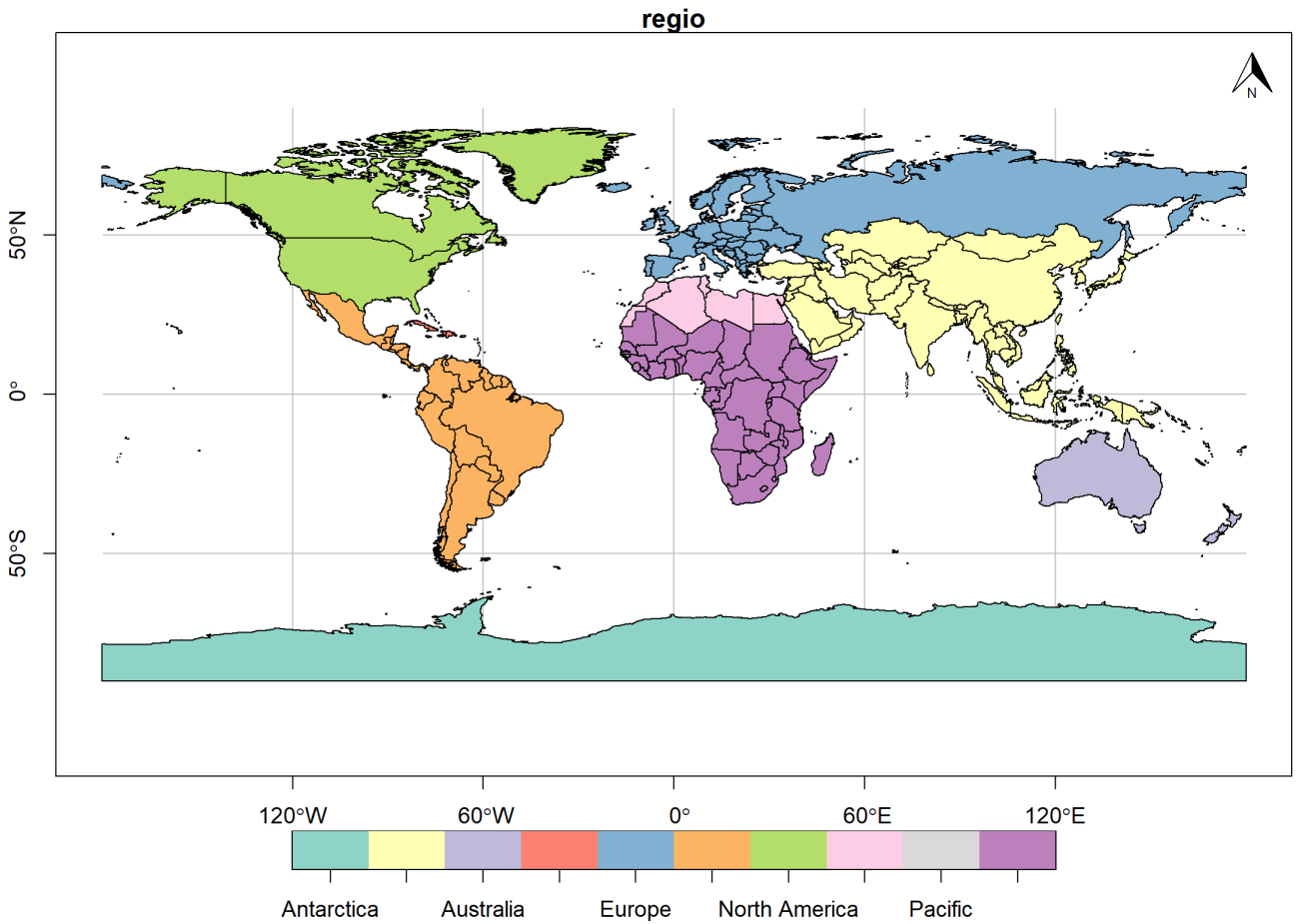
15. feladat (órai)

- Jelenítsd meg a világ országait (orszagok_osszes) a régiók szerint színezve.
- A térkép tartalmazzon tengelyt (tüskéket és koordináta-feliratokat), valamint rácshálózatot.
- Rakj az ábra jobb felső sarkába egy feles méretű északjelet.



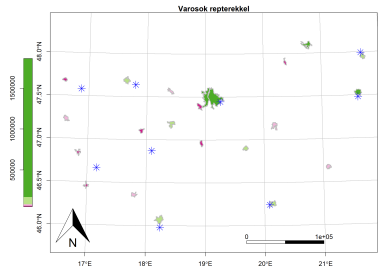
15. feladat (órai) – megoldás

```
plot(orszagok_osszes[, "regio"], axes = TRUE, graticule =
 TRUE, reset = FALSE)
addnortharrow(pos = "topright", scale = 0.5)
```



16. (összefoglaló) feladat (házi)

- Hozz létre egy 4 elemű színskálát a “PiYG” nevű, rózsaszínből zöldbe hajló RColorBrewer-paletta segítségével.
- Jelenítsd meg a városokat lakosságszám alapján színezve e színskála szerint úgy, hogy minden színkategóriába a városok egynegyede essen.
- Adj címet az ábrának, illetve lásd el tengelytüskékkel és -feliratokkal, valamint rácshálózattal.
- A jelmagyarázat bal oldalra kerüljön.
- Új rétegként add hozzá a repterek geometriáját másfélszeres méretű, kék csillaggal jelölve.
- A térkép bal alsó sarkába kerüljön háromszoros méretű északjel, míg a vízszintes 700000 és függőleges 50000 koordinátájú pontba 100 km hosszú léptékrúd.



16. (összefoglaló) feladat (házi) – megoldás

```
szinskala <- brewer.pal(n = 4, name = "PiYG")
plot(varosok[, "lakossag"], main = "Varosok repterekkel",
 border = "gray", lwd = 2, breaks = "quantile", nbreaks =
 4, axes = TRUE, graticule = TRUE, pal = szinskala,
 key.pos = 2, reset = FALSE)
plot(st_geometry(repterek), pch = 8, cex = 2, col =
 "blue", add = TRUE)
addnortharrow(pos = "bottomleft", scale = 1.5)
scalebar(d = 100000, xy = c(700000, 50000), type = "bar")
```