Egy- és kétváltozós geometriai műveletek

Térinformatika R-ben

2023.11.20.

Section 1

Egyváltozós logikai műveletek

Logikai aggregáló függvények

Ez a két függvény jól jön még a későbbiekben.

`any(...)`

- van-e köztük igaz (`TRUE`)?
- `...`: egy vagy több logikai vektor
- eredmény: egy darab logikai érték

`all(...)`

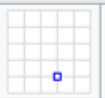
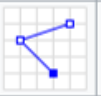
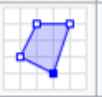
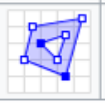
- mindegyik igaz-e (`TRUE`)?
- `...`: egy vagy több logikai vektor
- eredmény: egy darab logikai érték

Típus és tartalom lekérdezése

17 geometriatípus létezik
a fontosabbak:

- **Point**, MultiPoint
- **LineString**, **MultiLineString**
- Polygon, **MultiPolygon**
- GeometryCollection

Geometry primitives (2D)

Type	Examples	
Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

Multipart geometries (2D)

Type	Examples	
MultiPoint		`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
		`MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString		`MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))`
MultiPolygon		`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))`
		`MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))`

Típus és üresség lekérdezése

Üres geometriák lekérdezése

A kérdés

- nem az, hogy tartalmaz-e a Simple Features bármit,
- hanem hogy a benne lévő geometriák üres geometriák-e.

`st_is_empty(x)`

- eredmény: logikai vektor
- annyi elemű, ahány sora volt az `sf`-nek

Típus és üresség lekérdezése

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

```
st_is_empty(varosok_geometria)
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[10] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[19] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
any(st_is_empty(varosok_geometria))
```

```
[1] FALSE
```

Vagyis: egyik város sem üres geometriájú.

Típus és üresség lekérdezése

```
varosok_geometria[FALSE]
```

```
Geometry set for 0 features
Bounding box:  xmin: NA ymin: NA xmax: NA ymax: NA
Projected CRS: HD72 / EOV
```

```
st_is_empty(varosok_geometria[FALSE])
```

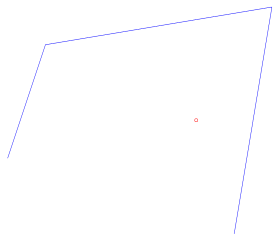
```
logical(0)
```

Nincs város, mely üres vagy nem üres geometriájú lehetne (0 darab logikai érték).

Típus és üresség lekérdezése

Létrehozunk mindenféle geometriát. (A használt függvényeket most nem mutatom be részletesen.)

```
mindenfele_geometriak <- st_sfc(
    st_point(c(6, 5)),
    st_point(),
    st_linestring(rbind(c(1, 4), c(2, 7), c(8, 8), c(7, 2))),
    st_linestring())
plot(mindenfele_geometriak, col = c("red", "orange",
 "blue", "green"))
```



Típus és üresség lekérdezése

```
st_is_empty(mindenfele_geometriak)
```

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

A narancssárga (2.) és zöld (4.) geometriák (amik a ploton se jelentek meg) üresek.

Típus és üresség lekérdezése

Geometriák típusának lekérdezése

`st_is(x, type)`

- `x`: az `sf` vagy `sfc`, aminek a geometriáira kíváncsiak vagyunk
- `type`: a típus szöveges, nagybetűs megnevezése, pl. `"POINT"` vagy `"LINESTRING"`
- eredmény: logikai vektor
- annyi elemű, ahány sora/eleme volt az `sf`-nek/`sfc`-nek

Típus és üresség lekérdezése

```
st_is(x = varosok_geometria, type = "MULTIPOINT")
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[10] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[19] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
st_is(x = varosok_geometria, type = "POLYGON")
```

```
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[11] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[21] TRUE TRUE TRUE TRUE TRUE
```

```
all(st_is(x = varosok_geometria, type = "POLYGON"))
```

```
[1] TRUE
```

```
any(st_is(x = mindenfele_geometriak, type = "POINT"))
```

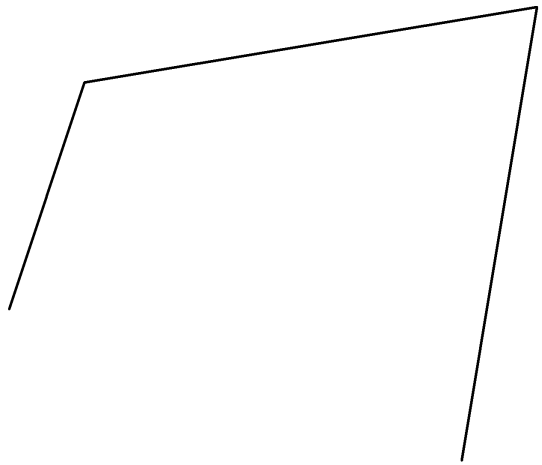
```
[1] TRUE
```

Típus és üresség lekérdezése

```
vonallancok_maszkja <- st_is(x = mindenfele_geometriak,
 type = "LINESTRING")
vonallancok_maszkja
```

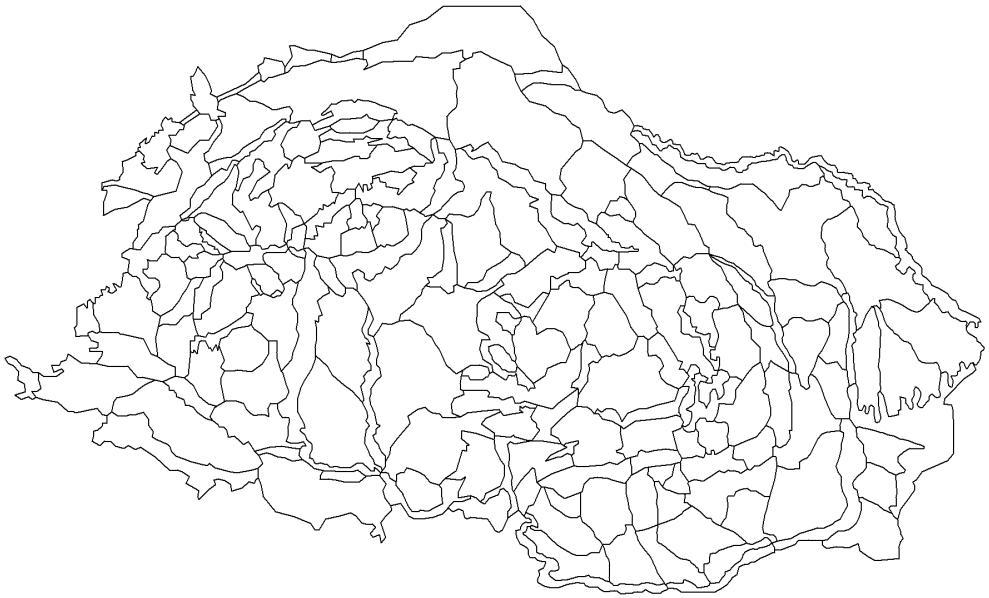
```
[1] FALSE FALSE  TRUE  TRUE
```

```
plot(mindenfele_geometriak[vonallancok_maszkja], lwd = 3)
```



1. feladat (házi)

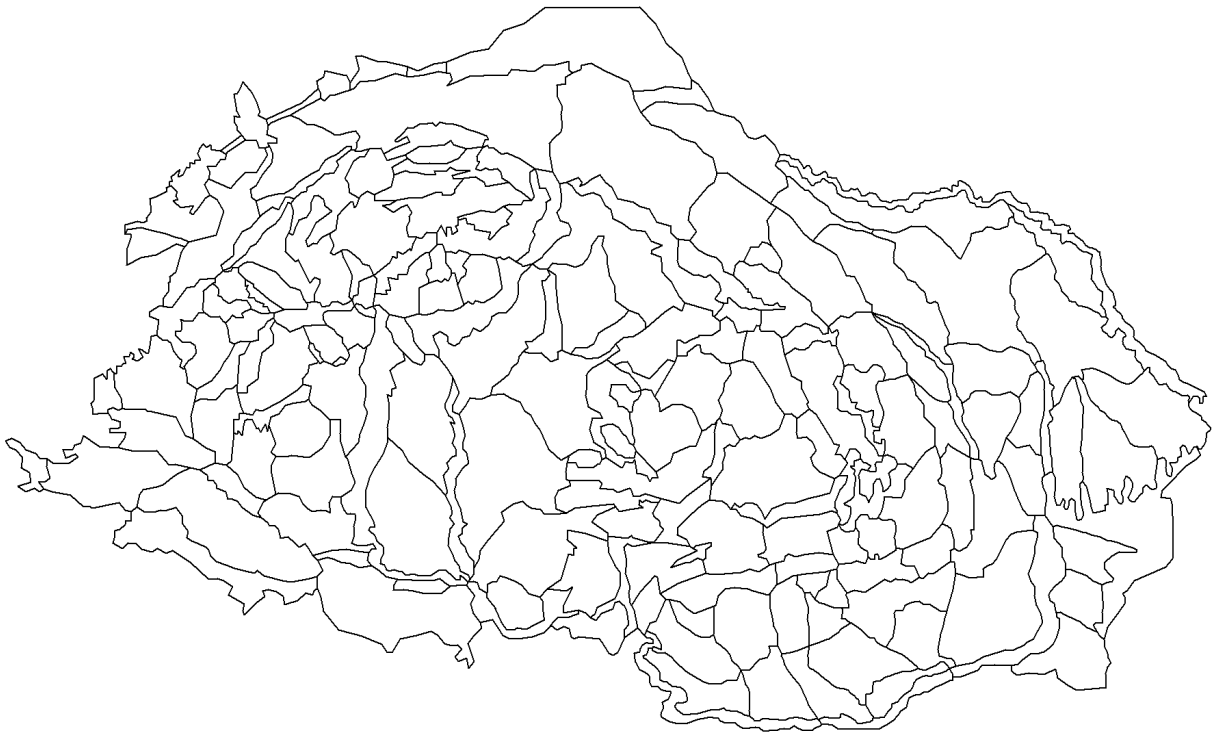
- Olvasd be a “tajbeosztas_geometria.RData” fájlt.
- Jelenítsd meg.
- Van a geometriák között “POLYGON” típusú? Ha nincs, vajon miért nincs?
- Hipotézis: “azért nincs, mert minden geometria ... típusú”. Egészítsd ki a hipotézist, és teszteld.



1. feladat (házi) – megoldás

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

```
plot(tajbeosztas_geometria)
```



1. feladat (házi) – megoldás

```
any(st_is(x = tajbeosztas_geometria, type = "POLYGON"))
```

```
[1] FALSE
```

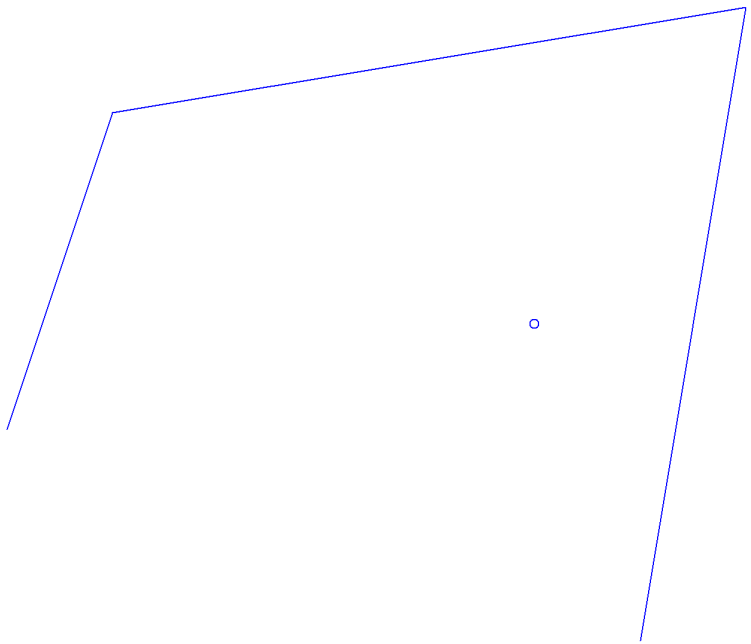
Hipotézis: “azért nincs, mert minden geometria MULTIPOLYGON típusú”

```
all(st_is(x = tajbeosztas_geometria, type = "MULTIPOLYGON"))
```

```
[1] TRUE
```

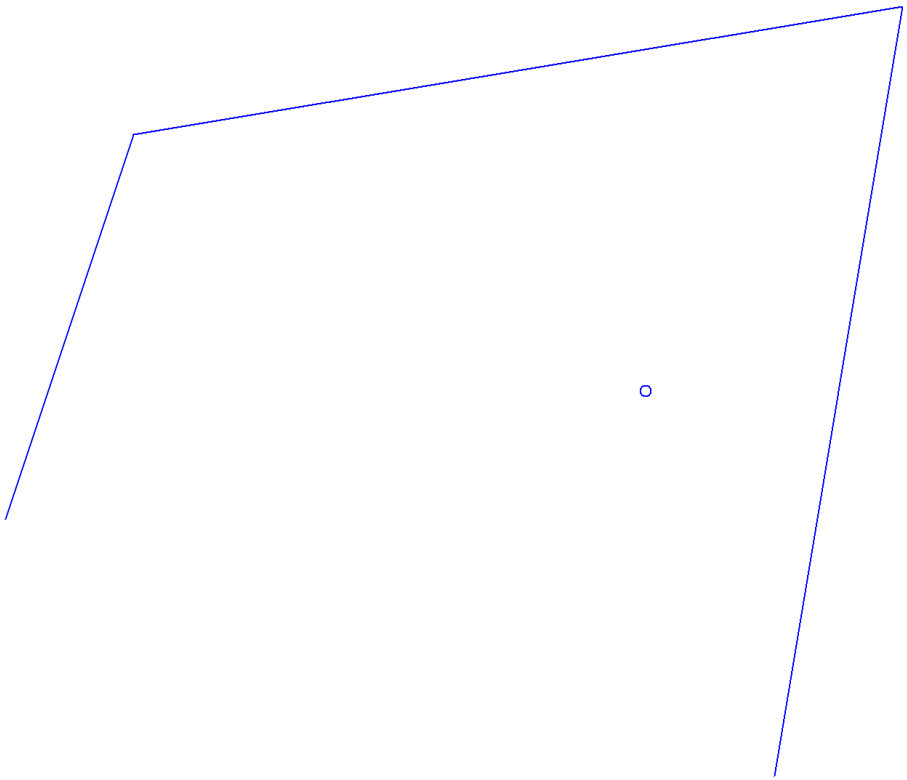
2. feladat (házi)

- Hozd létre “uresek_maszkja” néven a “mindenfele_geometriak” nevű `sfc` üres geometriáinak maszkját (logikai vektorát).
- Jelenítsd meg kék színnel a “mindenfele_geometriak” küzül azokat, amelyek nem üresek.



2. feladat (házi) – megoldás

```
plot(mindenfele_geometriak[!uresek_maszkja], col = "blue")
```



Section 2

Egyváltozós geometriai műveletek

Egyváltozós geometriai műveletek

Típuskonverzió

- `st_cast`: bármit bármivé
- `st_boundary`: poligon $\rightarrow$ vonal
- `st_polygonize`: vonal $\rightarrow$ poligon (később...)

Hossz- és területszámítás

- `st_length`: vonal hossza
- `st_area`: poligon területe

Új geometriák képzése

- `st_buffer`: puffer képzése
- `st_centroid`: középpont képzése
- `st_convex_hull`: legkisebb befoglaló konvex sokszög képzése
- `st_simplify`: vonallánc egyszerűsítése

Típuskonverzió

- kasztolás (casting), típusátalakítás
- elvben bármilyen geometriából bármivé (de a gyakorlatban nem...)
- néha több konverzió egymás után

Típuskonverzió

`st_cast(x, to)`

- `x`: amit át akarunk alakítani (`sf`/`sfc`)
- `to`: a céltípus szöveges, nagybetűs megnevezése, pl. `"POINT"` vagy `"LINESTRING"`
- `to`: opcionális, ha elhagyjuk, egyszerűsíteni próbál
- eredmény: `sf`/`sfc`, a sorok/elemek száma változhat!

`st_boundary(x)`

- poligon határvonalát képezi
- kb. megfelel egy `POLYGON`-on értelmezett `st_cast(x, to = "LINESTRING")`-nek

Típuskonverzió

```
tajbeosztas_geometria_poligon <- st_cast(x =
 tajbeosztas_geometria, to = "POLYGON")
all(st_is(x = tajbeosztas_geometria, type = "MULTIPOLYGON"))
```

```
[1] TRUE
```

```
all(st_is(x = tajbeosztas_geometria_poligon, type =
 "POLYGON"))
```

```
[1] TRUE
```

```
length(tajbeosztas_geometria)
```

```
[1] 179
```

```
length(tajbeosztas_geometria_poligon)
```

```
[1] 183
```

Típuskonverzió

```
load("utak.RData")
ut8 <- utak[utak$nev == "8", ]
ut8_torespontok <- st_cast(x = ut8, to = "POINT")
```

```
rownames(ut8)
```

```
 [1] "812"  "813"  "814"  "815"  "816"  "817"  "818"
 [8] "819"  "1783" "1787" "2554" "2555" "2556" "2557"
[15] "2558" "2559" "2560" "3176" "3177" "3178" "3179"
[22] "3180" "3181" "3182" "3183" "3184" "3185" "3186"
[29] "3187" "3188" "3189" "3190" "3191" "3192" "3193"
[36] "3194" "3195" "3196" "3197" "3198" "3538" "3541"
[43] "3542" "3831" "3832" "3833" "3834" "3835" "3836"
[50] "3939" "3940" "3941" "3942" "3984" "3985"
```

Az egyes útszakaszok (55 db.) sorszáma egyedi.

Típuskonverzió

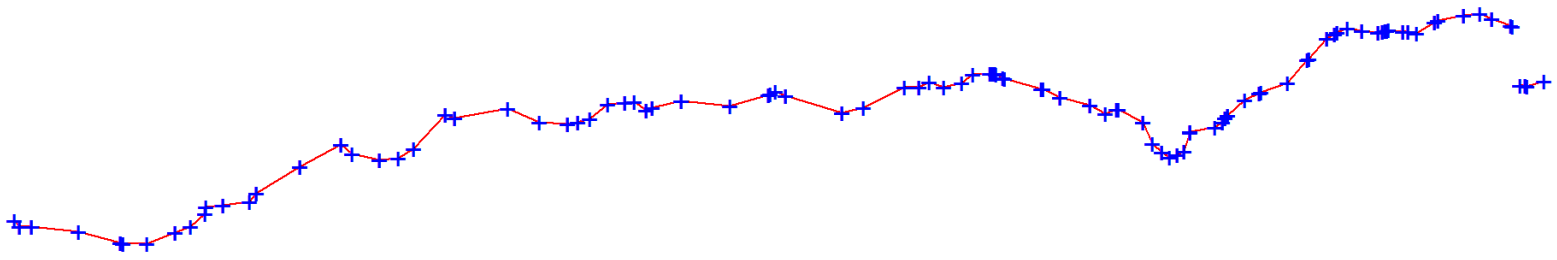
A töréspontok sorszáma utal a vonalszakasz sorszámára (duplikátum esetén .1 végződést kap).

```
rownames(ut8_torespontok)
```

```
  [1] "812"    "812.1"  "813"    "813.1"  "814"
  [6] "814.1"  "815"    "815.1"  "816"    "816.1"
 [11] "816.2"  "817"    "817.1"  "818"    "818.1"
 [16] "819"    "819.1"  "819.2"  "1783"   "1783.1"
 [21] "1787"   "1787.1" "2554"   "2554.1" "2555"
 [26] "2555.1" "2555.2" "2556"   "2556.1" "2556.2"
 [31] "2557"   "2557.1" "2557.2" "2558"   "2558.1"
 [36] "2559"   "2559.1" "2560"   "2560.1" "3176"
 [41] "3176.1" "3176.2" "3177"   "3177.1" "3178"
 [46] "3178.1" "3179"   "3179.1" "3180"   "3180.1"
 [51] "3180.2" "3181"   "3181.1" "3181.2" "3181.3"
 [56] "3181.4" "3182"   "3182.1" "3182.2" "3183"
 [61] "3183.1" "3184"   "3184.1" "3185"   "3185.1"
 [66] "3185.2" "3186"   "3186.1" "3186.2" "3186.3"
 [71] "3187"   "3187.1" "3188"   "3188.1" "3188.2"
```

Típuskonverzió

```
plot(st_geometry(ut8), col = "red")
plot(st_geometry(ut8_torespontok), col = "blue", pch =
 "+", add = TRUE)
```



Típuskonverzió

Az `st_boundary()`-t kicsit kényelmesebb használni, és könnyebb olvasni.

```
class(st_cast(x = varosok_geometria, to = "LINESTRING"))
```

```
[1] "sfc_LINESTRING" "sfc"
```

```
class(st_boundary(varosok_geometria))
```

```
[1] "sfc_LINESTRING" "sfc"
```

Típuskonverzió

A típuskonverzió van, hogy csak több lépésben végezhető el.

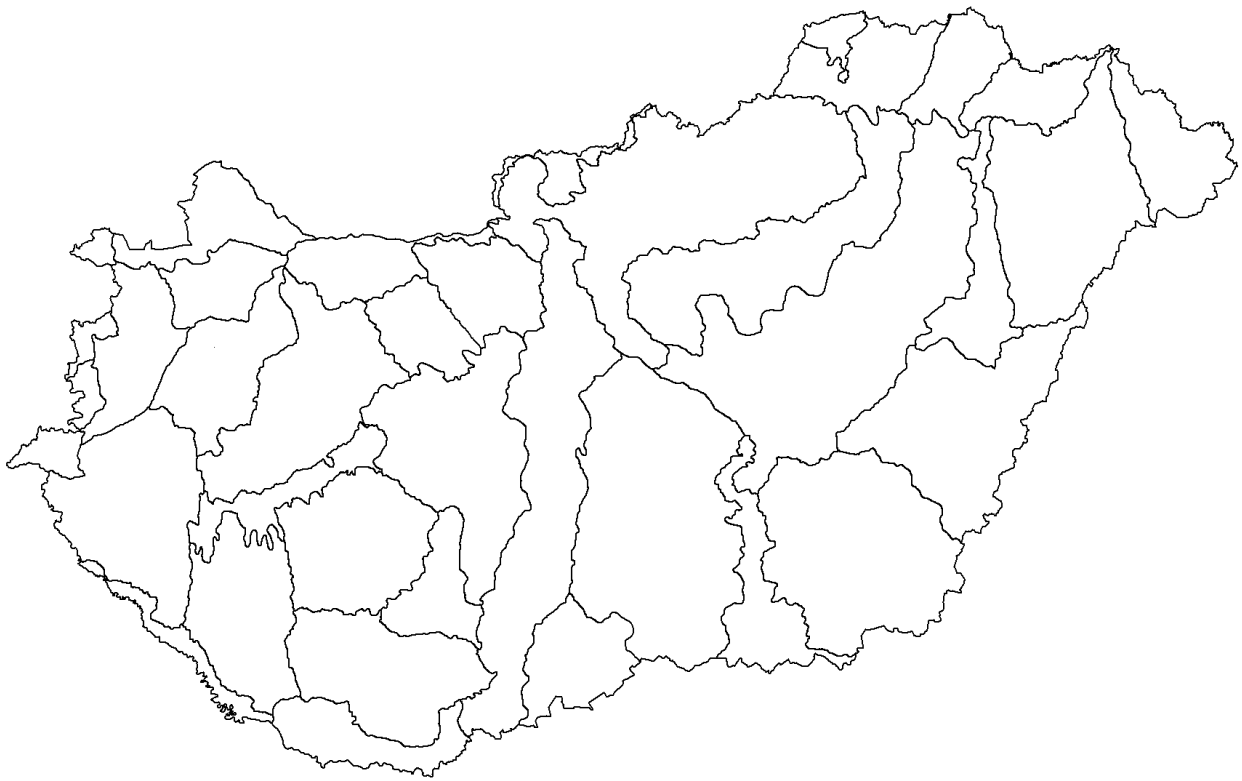
```
load("kozeptajak_geometria.RData")
```

```
st_cast(x = kozeptajak_geometria, to = "LINESTRING")
```

```
Error in st_cast.sfc(x = kozeptajak_geometria, to =
 "LINESTRING"): use smaller steps for st_cast; first cast
 to MULTILINESTRING or POLYGON?
```

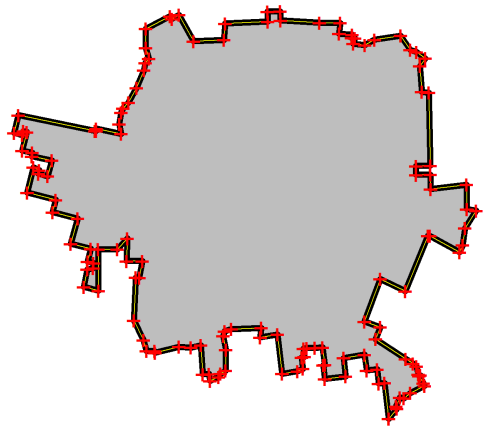
Típuskonverzió

```
plot(st_cast(x = st_cast(x = kozeptajak_geometria, to =
 "MULTILINESTRING"), to = "LINESTRING"))
```



3. feladat (órai)

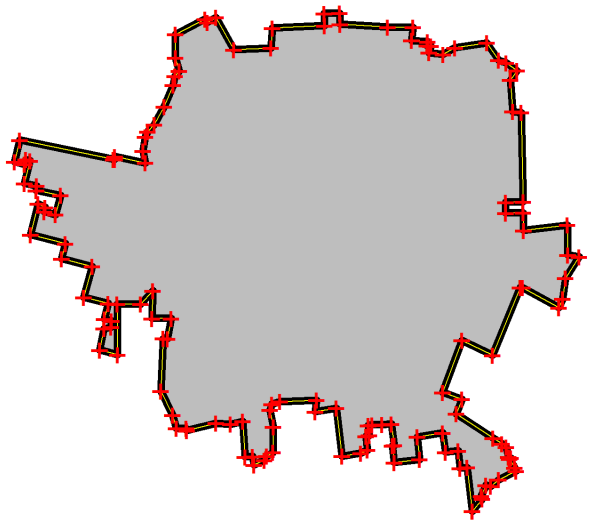
- Olvasd be a “varosok.RData” fájlt.
- Hozz létre egy “debrecen” nevű `sfc-t`, ami a “Debrecen” nevű város geometriáját tartalmazza.
- Jelenítsd meg hétszeres körvonalvastagsággal és szürke kitöltéssel Debrecent.
  - Add hozzá Debrecen körvonalát sárga színnel.
  - Add hozzá dupla méretű piros pluszjelekként Debrecen határának töréspontjait.



3. feladat (órai) – megoldás

```
load("varosok.RData")
debrecen <- st_geometry(varosok[varosok$nev == "Debrecen",
 ])
```

```
plot(debrecen, col = "gray", lwd = 7)
plot(st_boundary(debrecen), col = "yellow", add = TRUE)
plot(st_cast(x = debrecen, to = "POINT"), col = "red", pch
 = "+", cex = 2, add = TRUE)
```



Típuskonverzió

Vegyes geometriákat (geometriatípus neve: `GEOMETRY`) is létrehozhatunk.
`sfc`-k összefűzése egyszerű: `c()`

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

```
franciao <- st_geometry(nuts_regiok[nuts_regiok$regio_kod
 == "FR", ])
magyaro_ausztria <-
 st_geometry(nuts_regiok[nuts_regiok$regio_kod %in%
 c("HU", "AT"), ])
magyaro_ausztria <- st_cast(x = magyaro_ausztria, to =
 "POLYGON")
```

Típuskonverzió

```
nehany_orszag <- c(franciao, magyaro_ausztria)
nehany_orszag
```

```
Geometry set for 3 features
Geometry type: GEOMETRY
Dimension:     XY
Bounding box: xmin: -61.80593 ymin: -21.35013 xmax:
 55.8258 ymax: 51.08938
Geodetic CRS:  WGS 84

MULTIPOLYGON (((2.607036 50.91269, 2.863276 50....

POLYGON ((16.94028 48.61725, 16.94978 48.53579,...

POLYGON ((22.15531 48.4034, 22.89627 47.95412, ...
```

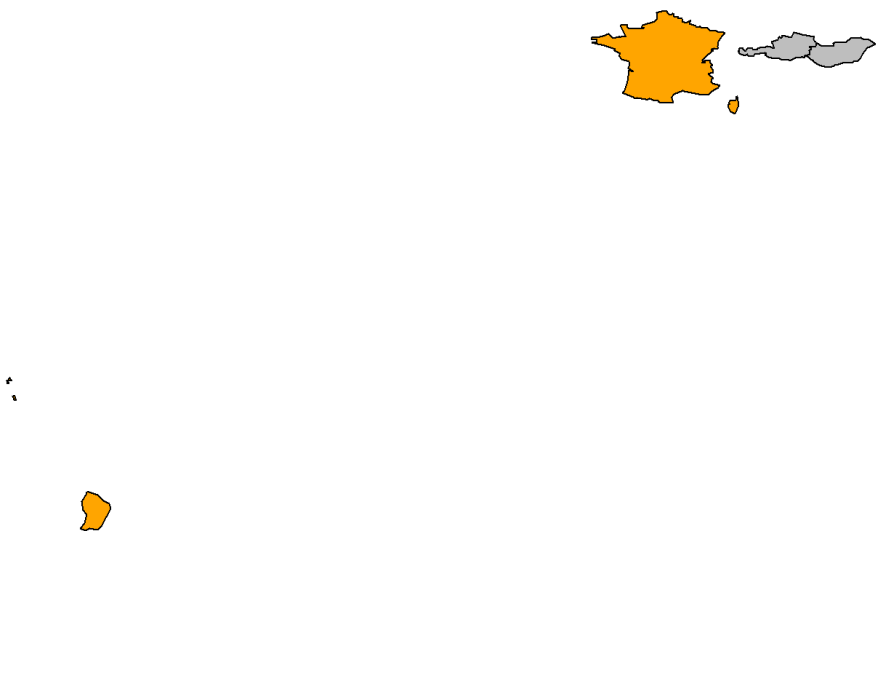
Van benne `MULTIPOLYGON` és `POLYGON` is, ezért az objektum geometriatípusa `GEOMETRY`.

4. feladat (házi)

- Hozz létre “multipoligonok_maszkja” néven egy logikai maszkot, amely azt mutatja, hogy a “nehany_orszag” nevű geometriák közül melyik `"MULTIPOLYGON"` típusú.
- Jelenítsd meg e geometriák közül a multipoligonokat narancssárga kitöltéssel.
  - Add hozzá e geometriák közül azokat szürke kitöltéssel, amelyek nem multipoligonok.

4. feladat (házi) – megoldás

```
multipoligonok_maszkja <- st_is(x = nehany_orszag, type =
 "MULTIPOLYGON")
plot(nehany_orszag[multipoligonok_maszkja], col = "orange")
plot(nehany_orszag[!multipoligonok_maszkja], col = "gray",
 add = TRUE)
```



Hossz- és területszámítás

Megkötések

- hossza a `LINESTRING` és `MULTILINESTRING` típusú geometriáknak van
- területe a `POLYGON` és `MULTIPOLYGON` típusú geometriáknak van
- a többinek 0 a hossza/területe
- ha mégis kíváncsiak vagyunk $\rightarrow$ előbb típuskonverzió!

Függvények

- `st_length(x)`
- `st_area(x)`
- eredmény: egy olyan hosszú vektor, ahány geometria van `x`-ben
- eredmény típusa: `units` (mértékegységet is tartalmazza)
- `as.numeric()`-kel számvektorrá alakítható

Hossz- és területszámítás

```
m15 <- st_geometry(utak[utak$nev == "M15", ])
st_length(m15)
```

```
Units: [m]
[1]  203.2024  530.1399 1110.2954 7524.7650 3390.4399
```

Hossz- és területszámítás

```
class(st_length(m15))
```

```
[1] "units"
```

```
str(st_length(m15))
```

```
 Units: [m] num [1:5] 203 530 1110 7525 3390
```

```
as.numeric(st_length(m15))
```

```
[1]  203.2024  530.1399 1110.2954 7524.7650 3390.4399
```

Hossz- és területszámítás

```
st_area(varosok_geometria)
```

```
Units: [m^2]
 [1] 112524969  21215927  49420661   1238370   1061841
 [6]  30282003  28825028  41303787  35540422   1656423
[11]  23671219  38114838  61955930  24487772  20747615
[16]  20846717  26755062  18305779  18325073  28946138
[21]  14805377  39857280  17300897 242026155  49703206
```

```
st_area(m15)
```

```
Units: [m^2]
[1] 0 0 0 0 0
```

Hossz- és területszámítás

```
st_length(varosok_geometria)
```

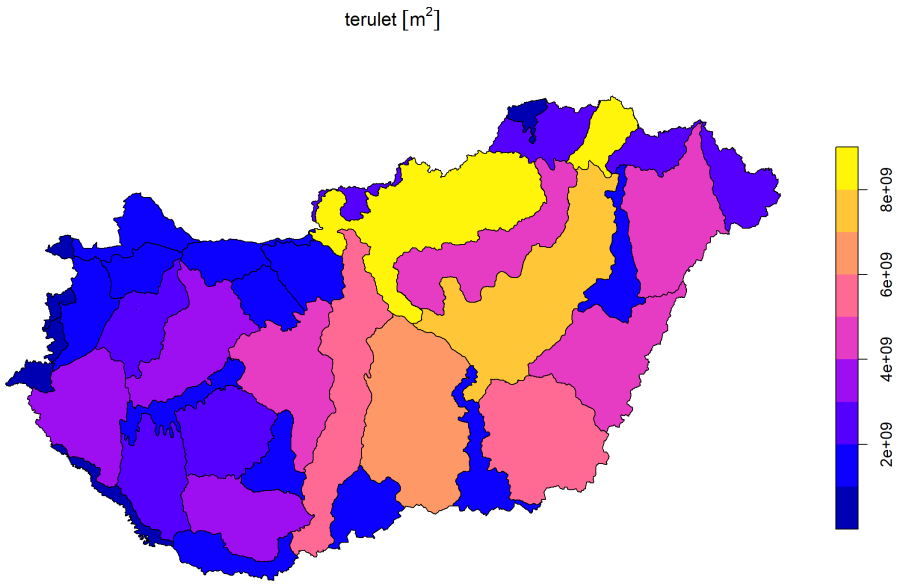
```
Units: [m]
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
st_length(st_boundary(varosok_geometria))
```

```
Units: [m]
 [1] 133579.726  49947.375  48421.166   5832.176
 [5]   5680.283  39080.561  41300.992  56438.099
 [9]  63963.770   9458.884  43250.235  51515.335
[13]  68233.400  34311.161  29366.331  27309.826
[17]  39052.663  70236.974  50673.528  46787.362
[21]  34281.890  55121.800  30368.875 153902.601
[25] 101628.465
```

5. feladat (házi)

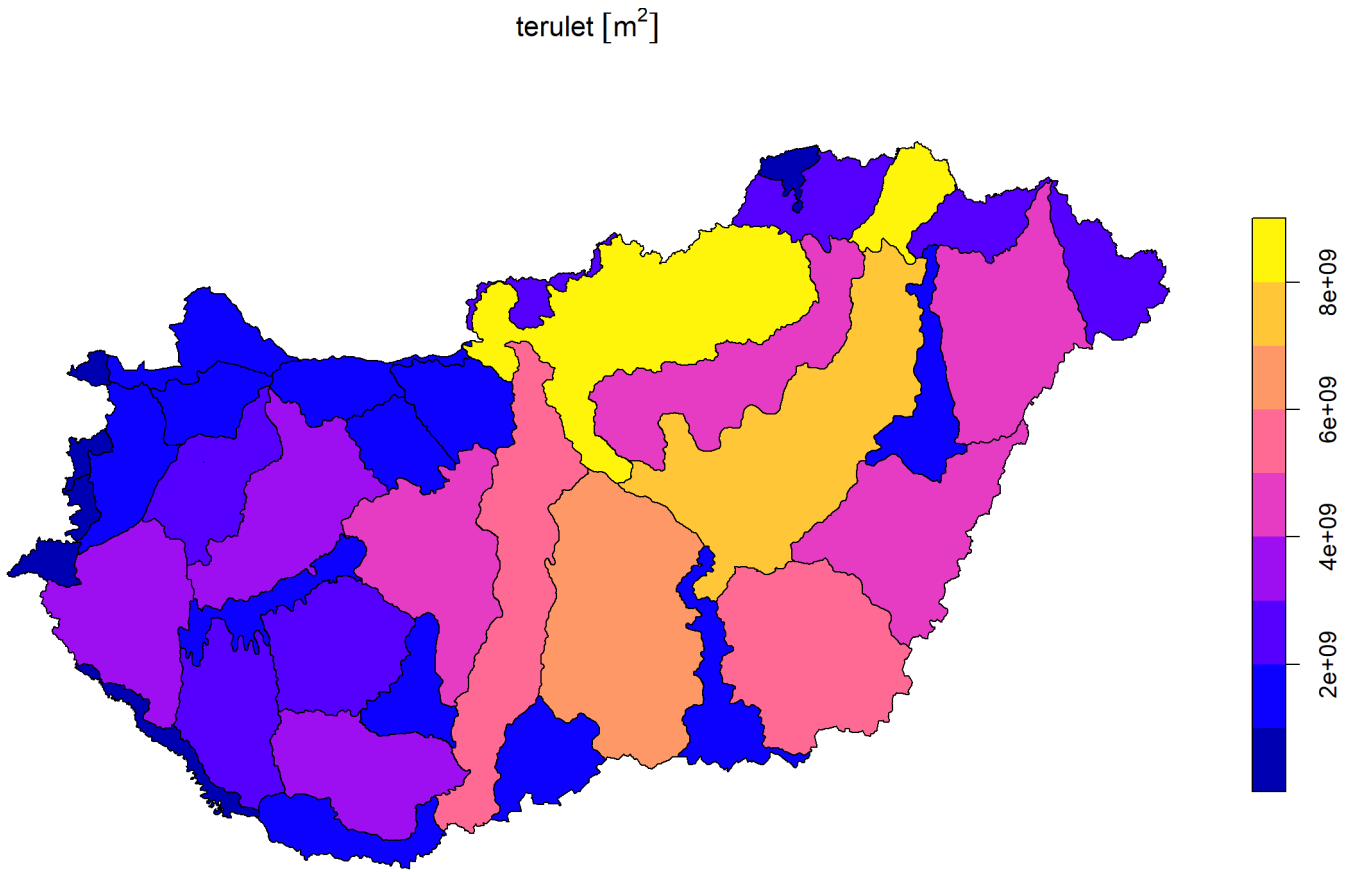
- Olvasd be a “kozeptajak.RData” fájlt.
- Hozz létre egy új oszlopot “terulet” néven, amelybe rögzíted a középtájak területét!
- Jelenítsd meg a középtájakat területük alapján színezve.
- Hozz létre egy új oszlopot “szettagoltsag” néven, amelybe rögzíted a középtájak széttagoltságát, amely alatt most értsük a határvonaluk hosszának és a területüknek a hányadosát.
- Jelenítsd meg a középtájakat széttagoltságuk alapján színezve.



5. feladat (házi) – megoldás

```
load("kozeptajak.RData")
kozeptajak$terulet <- st_area(kozeptajak)
```

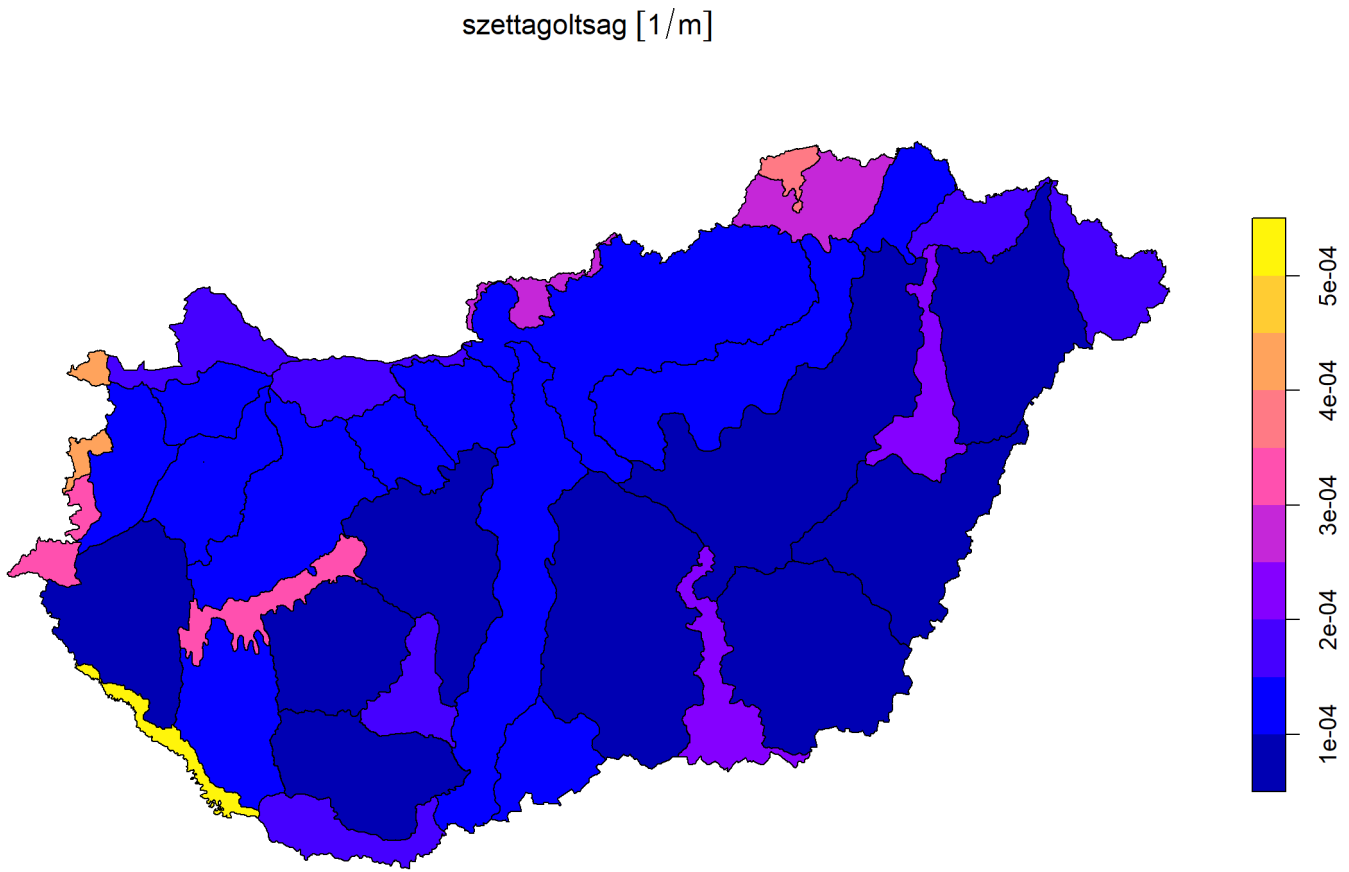
```
plot(kozeptajak[, "terulet"])
```



5. feladat (házi) – megoldás

```
kozeptajak$szettagoltsag <-
  st_length(st_boundary(kozeptajak)) / st_area(kozeptajak)
```

```
plot(kozeptajak[, "szettagoltsag"])
```



Puffer készítése

Puffer

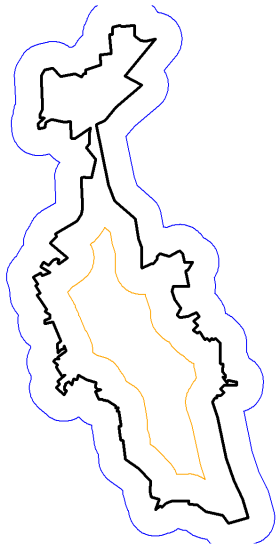
- ponttól/vonaltól/poligontól
- megadott távolságon belül lévő
- összes pont halmaza (vagyis egy poligon)
- összetartozó geometriáknak (`MULTI-`) közös a puffere
- ha külön puffereket szeretnék, előbb szedjük szét a geometriákat `st_cast()`-tal!

`st_buffer(x, dist)`

- `x`: aminek a pufferjét képezni szeretnénk
- `dist`: a vetületnek megfelelő mértékegységben (EOV: m, WGS-84: °) kifejezett távolság
- poligonok esetén negatív távolság is megadható (értsd: szűkít)
- eredményként annyi geometriát ad, amennyi geometriát `x` tartalmazott

Puffer készítése

```
eger <- st_geometry(varosok[varosok$nev == "Eger", ])
eger_novelt <- st_buffer(x = eger, dist = 500)
eger_csokkentett <- st_buffer(x = eger, dist = -500)
plot(eger, lwd = 3)
plot(eger_novelt, border = "blue", add = TRUE)
plot(eger_csokkentett, border = "orange", add = TRUE)
```



Puffer készítése

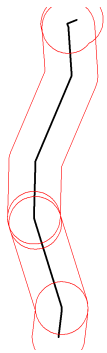
```
length(m15)
```

```
[1] 5
```

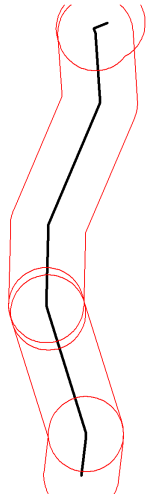
```
length(st_buffer(x = m15, dist = 1000))
```

```
[1] 5
```

```
plot(m15, lwd = 3)
plot(st_buffer(x = m15, dist = 1000), border = "red", add
 = TRUE)
```



Puffer készítése



Az öt útszakasznak öt különálló puffere van.
Ha egyet szeretnénk, a bemenetet vagy az eredményt egyesítenünk kell (később...).

6. feladat (órai)

- Válogasd le “olaszo” néven azon NUTS-régió (“nuts_regiok” nevű objektum) geometriáját, amely a “regio_kod” oszlopban “IT”-t tartalmaz.
- Készíts “olaszo_puffer” néven a leválogatott geometria köré 1,5°-os legnagyobb távolsággal puffert.
- Jelenítsd meg Olaszországot, majd rajta kék körvonalszínnel a létrehozott puffert.
- Ezután készítsd el ugyanezt az ábrát, de most úgy, hogy Olaszország három egységének három külön puffere legyen. Ehhez előbb típuskonverziót kell végezned.

6. feladat (órai) – megoldás

```
olaszo <- st_geometry(nuts_regiok[nuts_regiok$regio_kod ==
 "IT", ])
```

```
olaszo_puffer <- st_buffer(x = olaszo, dist = 1.5)
```

6. feladat (órai) – megoldás

```
plot(olaszo)
plot(olaszo_puffer, border = "blue", add = TRUE)
```

6. feladat (órai) – megoldás

```
olaszo_reszei <- st_cast(x = olaszo, to = "POLYGON")
plot(olaszo_reszei)
plot(st_buffer(x = olaszo_reszei, dist = 1.5), border =
 "blue", add = TRUE)
```

Középpont és befoglaló konvex sokszög

Középpont és befoglaló konvex sokszög

- bármilyen geometrián értelmezhetőek
- összetartozó geometriákat (`MULTI-`) ugyanúgy kezelik, mint a pufferképzés
- eredményként annyi geometriát adnak, amennyi geometriát `x` tartalmazott

Függvények

`st_centroid(x)`

- eredmény: pont(ok)
- kívül eshet az eredeti geometrián (ha az konkáv volt vagy több részből állt)!

`st_convex_hull(x)`

- eredmény: poligon(ok)

Középpont és befoglaló konvex sokszög

```
olaszo_kozeppontja <- st_centroid(olaszo)
```

```
length(olaszo_kozeppontja)
```

```
[1] 1
```

```
olaszo_reszei <- st_cast(x = olaszo, to = "POLYGON")
```

```
olaszo_reszeinek_kozeppontja <- st_centroid(olaszo_reszei)
```

```
length(olaszo_reszeinek_kozeppontja)
```

```
[1] 3
```

Középpont és befoglaló konvex sokszög

```
plot(olaszo)
plot(olaszo_kozeppontja, cex = 5, pch = "+", col = "red",
 add = TRUE)
plot(olaszo_reszeinek_kozeppontja, cex = 2, pch = "×", col
 = "orange", add = TRUE)
```

Középpont és befoglaló konvex sokszög

```
olaszo_befoglalo_sokszoge <- st_convex_hull(olaszo)
length(olaszo_befoglalo_sokszoge)
```

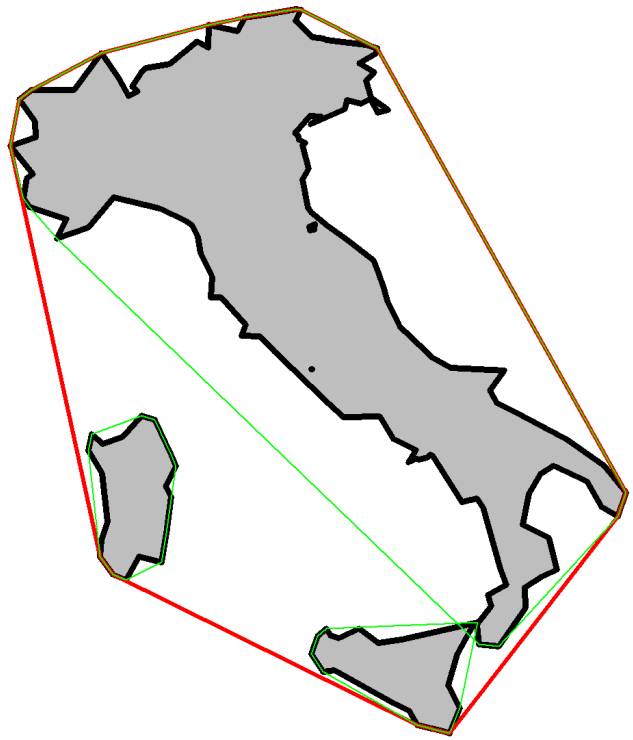
```
[1] 1
```

```
olaszo_reszeinek_befoglalo_sokszoge <-
 st_convex_hull(olaszo_reszei)
length(olaszo_reszeinek_befoglalo_sokszoge)
```

```
[1] 3
```

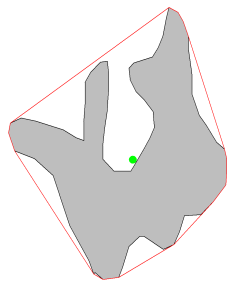
Középpont és befoglaló konvex sokszög

```
plot(olaszo, col = "gray", lwd = 4)
plot(olaszo_befoglalo_sokszoge, border = "red", lwd = 3,
 add = TRUE)
plot(olaszo_reszeinek_befoglalo_sokszoge, border =
 "green", add = TRUE)
```



7. feladat (házi)

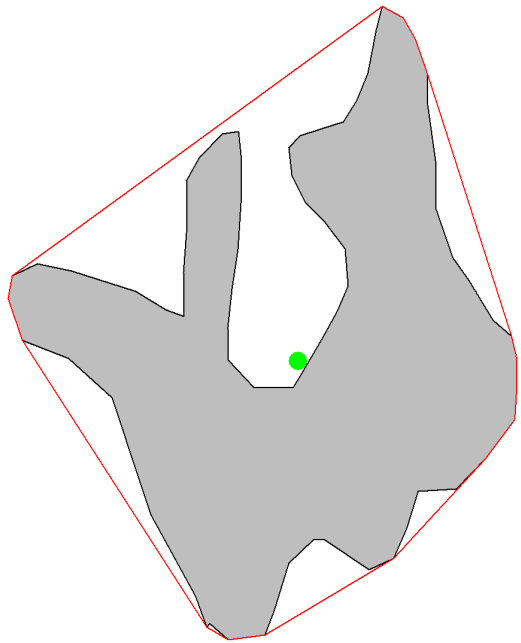
- Olvasd be a “zolyomi.RData” fájlt.
- Hozz létre egy “bukkerdo” nevű `sfc`-objektumot, ami a 93-as azonosítójú vegetációs foltot tartalmazza.
- Jelenítsd meg szürke kitöltőszínnel.
  - Add hozzá az ábrához kétszeres méretű, zöld, 16-os pontjellel a középpontját.
  - Add hozzá piros körvonalszínnel a befoglaló legkisebb konvex sokszöget.
- A zolyomi `sf`-objektumhoz adj hozzá egy új oszlopot “befoglalo_terulete” néven, amely minden poligonhoz a köré rajzolt legkisebb befoglaló konvex sokszög területét tartalmazza.
- Jelenítsd meg a Zólyomi-féle poligonokat a befoglaló terület alapján színezve.



7. feladat (házi) – megoldás

```
load("zolyomi.RData")
bukkerdo <- st_geometry(zolyomi[zolyomi$azonosito == 93, ])
```

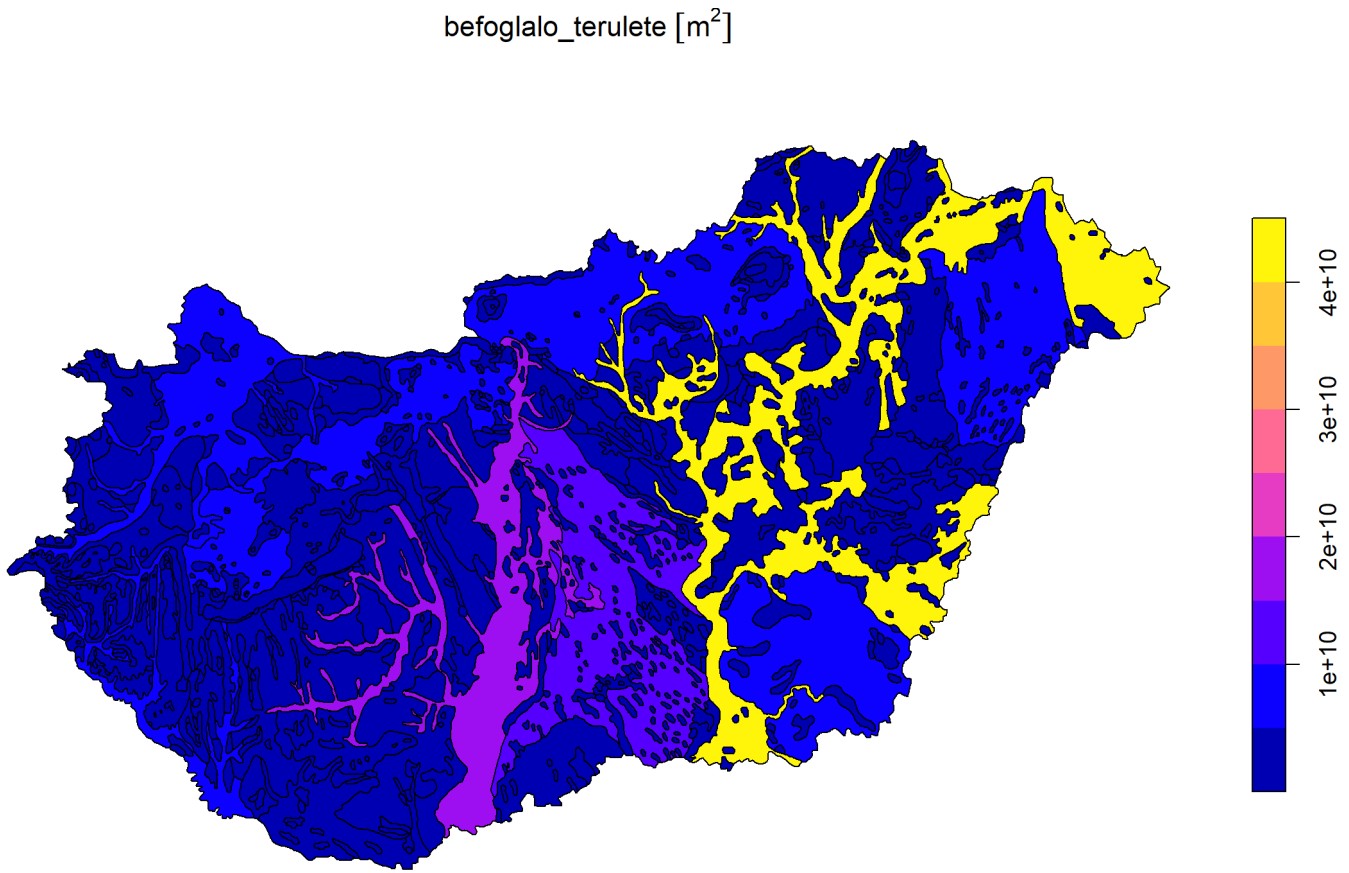
```
plot(bukkerdo, col = "gray")
plot(st_centroid(bukkerdo), col = "green", cex = 2, pch =
 16, add = TRUE)
plot(st_convex_hull(bukkerdo), border = "red", add = TRUE)
```



7. feladat (házi) – megoldás

```
zolyomi$befoglalo_terulete <-
 st_area(st_convex_hull(zolyomi))
```

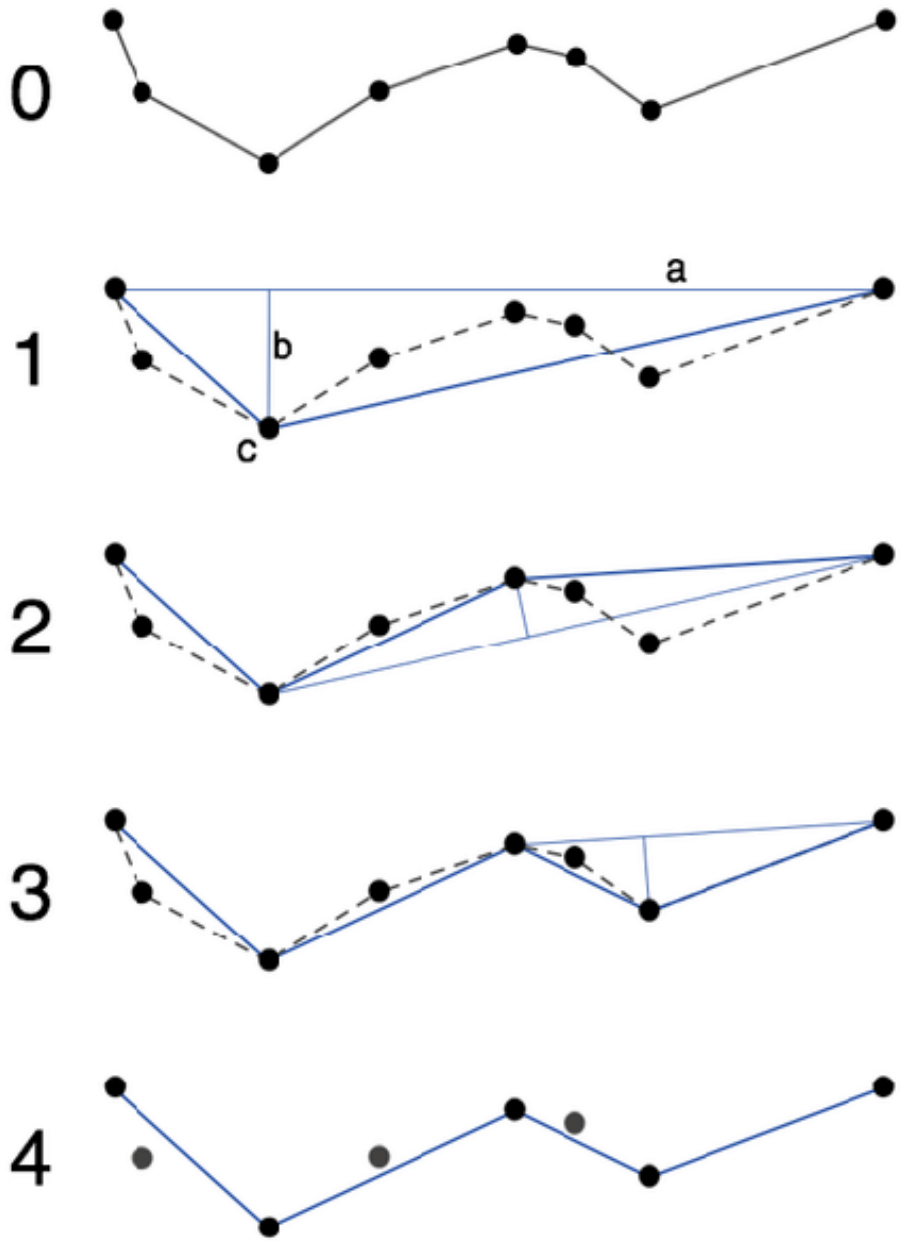
```
plot(zolyomi[, "befoglalo_terulete"])
```



Vonallánc egyszerűsítése

Vonallánc egyszerűsítése

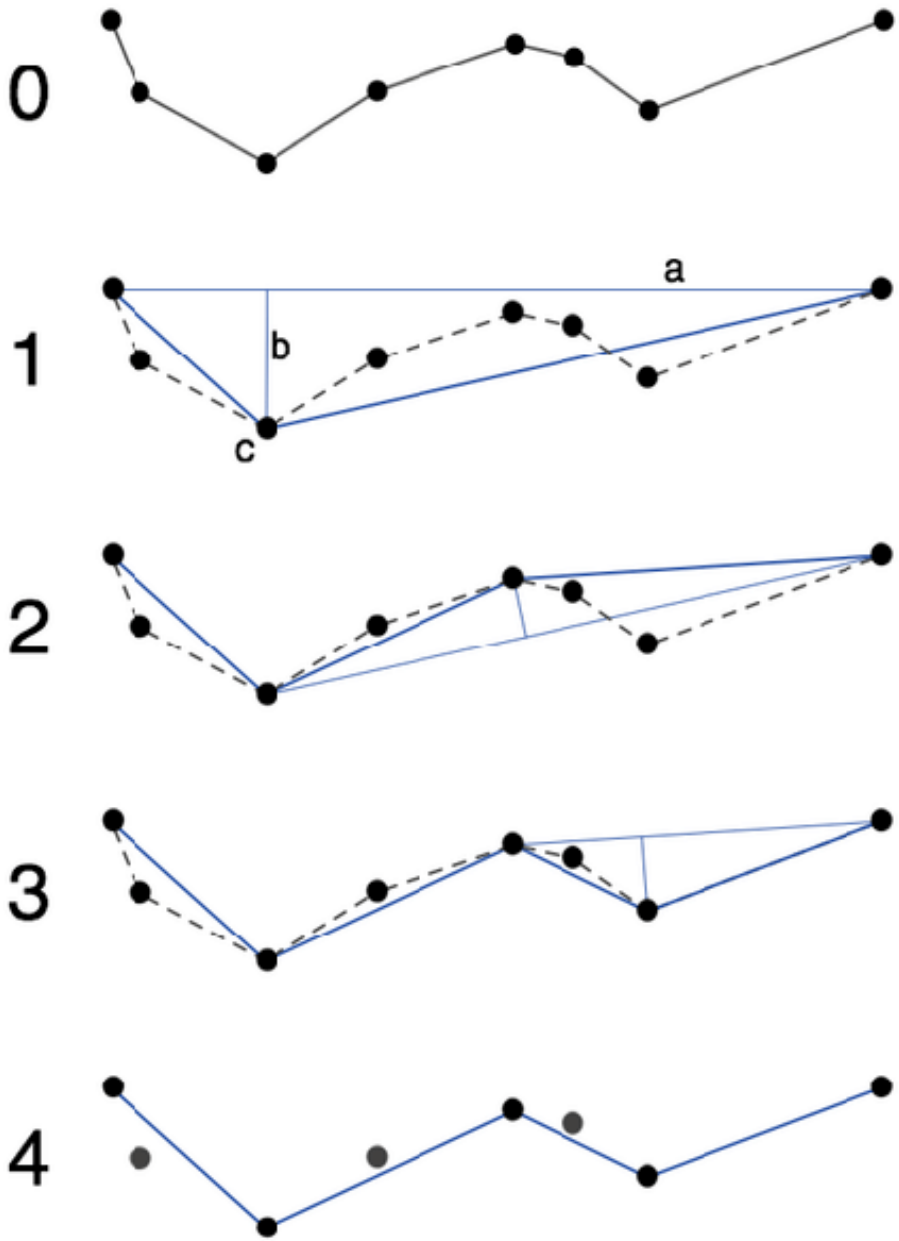
- Ramer–Douglas–Peucker-algoritmus
- sorban hozzáveszi a kezdő- és végponthoz azon közbülső töréspontokat, amelyek túlságosan kilógnak
- addig, amíg van a toleranciatávolságnál távolabb eső pont



Vonallánc egyszerűsítése

`st_simplify(x, dTolerance)`

- `x`: vonallánc
- `dTolerance`: toleranciatávolság

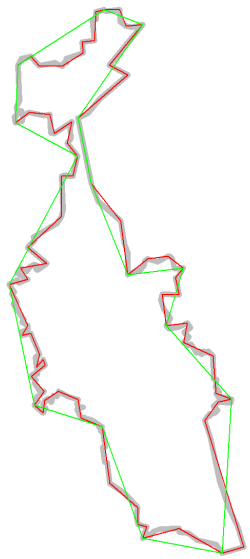


Vonallánc egyszerűsítése

```
eger_korvonala <- st_boundary(eger)
eger_egyszerusitett_100 <- st_simplify(x = eger_korvonala,
 dTolerance = 100)
eger_egyszerusitett_500 <- st_simplify(x = eger_korvonala,
 dTolerance = 500)
```

Vonallánc egyszerűsítése

```
plot(eger_korvonala, col = "gray", lwd = 5)
plot(eger_egyszerusitett_100, col = "red", add = TRUE)
plot(eger_egyszerusitett_500, col = "green", add = TRUE)
```

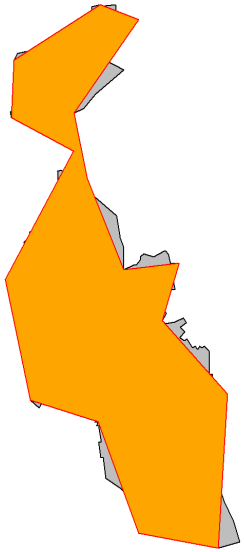


Vonallánc egyszerűsítése

Zárt vonallánc poligonná alakítása:

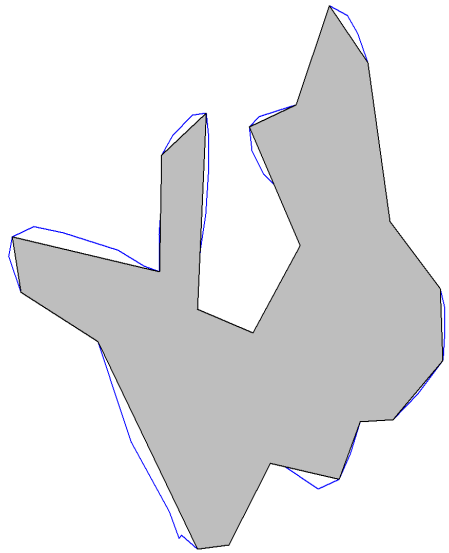
- `st_polygonize(x)`

```
eger_egyszerusitett_poligon <-
 st_polygonize(eger_egyszerusitett_500)
plot(eger, col = "gray")
plot(eger_egyszerusitett_poligon, col = "orange", border =
 "red", add = TRUE)
```



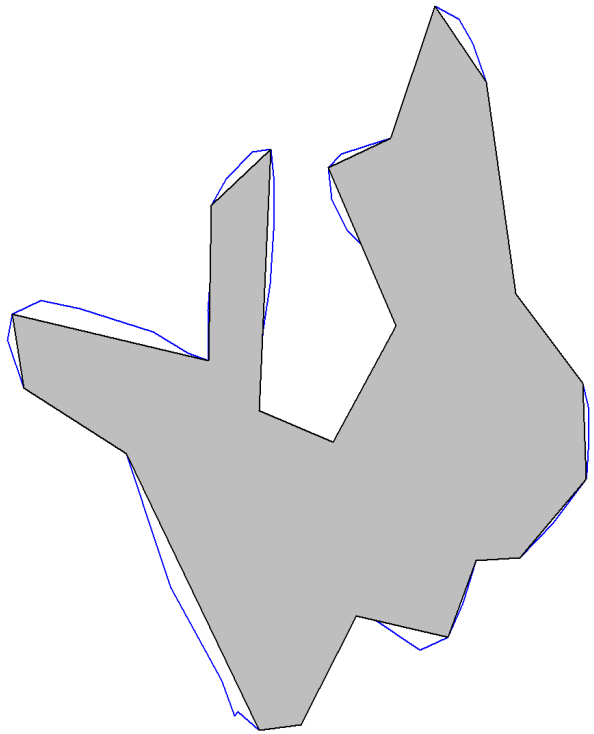
8. feladat (házi)

- Képezd a “bukkerdo” poligon határvonalát, egyszerűsítsd a Ramer–Douglas–Peucker-algoritmussal, 500 m-es toleranciatávolságot tartva az eredeti körvonaltól, és ezt az egyszerűsített körvonalat alakítsd vissza poligonná. Az új poligon neve legyen “bukkerdo_egyszerusitett”.
- Jelenítsd meg a bükkerdőt kék körvonallal.
  - Add hozzá szürke kitöltőszínnel az egyszerűsített poligont.



8. feladat (házi) – megoldás

```
bukkerdo_egyszerusitett <- st_polygonize(st_simplify(x =
 st_boundary(bukkerdo), dTolerance = 500))
plot(bukkerdo, border = "blue")
plot(bukkerdo_egyszerusitett, col = "gray", add = TRUE)
```



Section 3

Kétváltozós logikai műveletek

Kétváltozós logikai műveletek

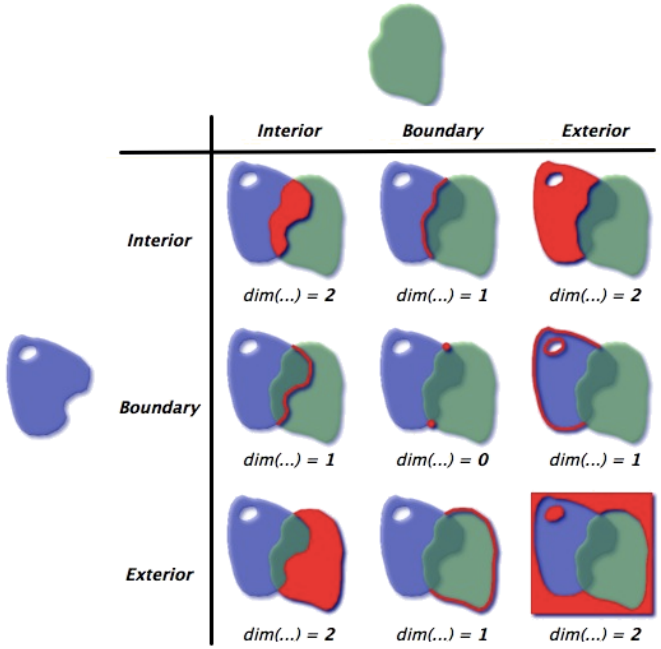
Műveletek

- predikátumok (predicates)
- logikai lekérdezések két geometria kapcsolatáról
- paraméterek: 2 geometria (`sfc`), eredmény: logikai mátrix (vagy lista)
- bemenet lehet `sf` is
- nagyon sokféle

Kétváltozós logikai műveletek

Lehetőségek

- **st_intersects**
- st_disjoint
- st_touches
- st_crosses
- **st_within**
- **st_contains**
- st_contains_properly
- st_overlaps
- st_equals
- st_covers
- st_covered_by
- st_equals_exact
- st_is_within_distance



2 elemnek 9 féle kapcsolata van. A predikátumfüggvények abban különböznek, hogy e 9 kapcsolattól milyen geometriatípust (üres/pont/vonal/poligon) várnak el.

Kétváltozós logikai műveletek

Függvények

```
st_intersects(x, y, sparse = TRUE)
st_within(x, y, sparse = TRUE)
st_contains(x, y, sparse = TRUE)
```

- összemetsz/benne található/tartalmazza
- paraméterezésük hasonló
- `x`: amit össze akarunk vetni
- `y`: amivel össze akarjuk vetni
- `sparse`: az eredményként előálló, `TRUE` értékeket ritkán tartalmazó logikai mátrixot inkább egy rövidebb listába szeretnénk tömöríteni

Kétváltozós logikai műveletek

Eredmény – `sparse = FALSE` eset

- inkább ezt fogjuk használni, számunkra most átláthatóbb
- logikai mátrix (sűrű, vagyis minden cellában tartalmaz értéket)
- annyi sora van, ahány eleme/sora `x`-nek
- annyi oszlopa van, ahány eleme/sora `y`-nak
- a cella megmutatja, hogy `x` adott eleme geometriai összefüggésben van-e (pl. összemetsződik-e) `y` adott elemével

Eredmény – `sparse = TRUE` eset

- alapértelmezett
- listaszerű eredmény
- annyi eleme van, ahány eleme/sora `x`-nek
- minden listaelem egy számvektor, azon `y`-beli elemek sorszámát tartalmazza, amelyekkel `x` adott eleme geometriai összefüggésben van

Kétváltozós logikai műveletek

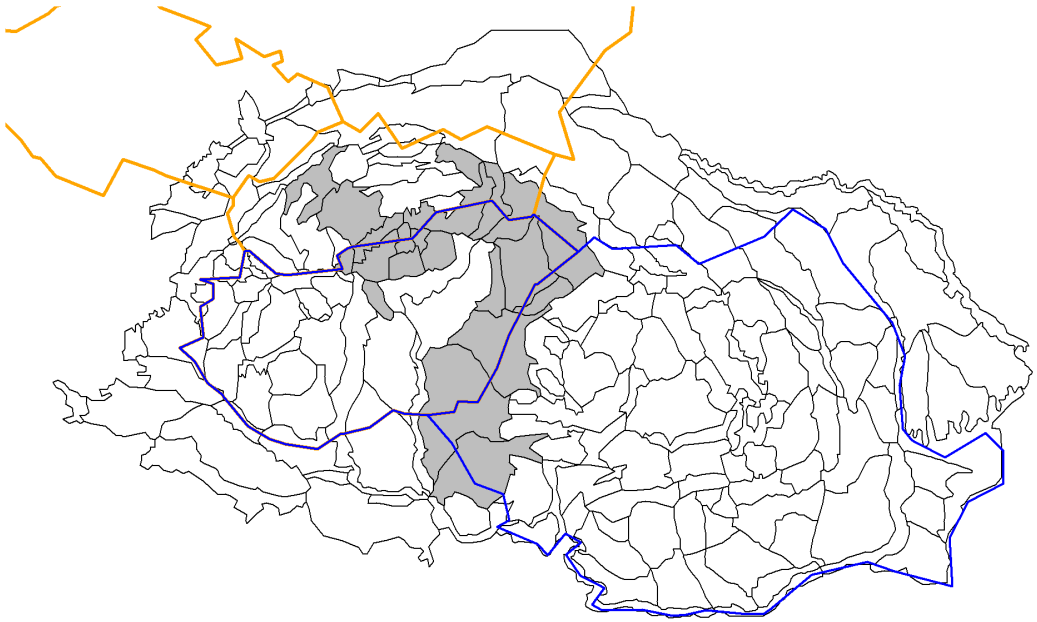
Készítsünk néhány geometriát, amikkel eljátszhatunk!

```
v4 <- st_transform(x = nuts_regiok[nuts_regiok$regio_kod
 %in% c("SK", "PL", "HU", "CZ"), ], crs = 23700)
magyaro_romania <-
 st_transform(nuts_regiok[nuts_regiok$regio_kod %in%
 c("RO", "HU"), ], crs = 23700)
tajbeosztas <- st_as_sf(tajbeosztas_geometria)
tajbeosztas$azonosito <- 1:nrow(tajbeosztas)
nehany_taj <- tajbeosztas[31:50, ]
```

`sfc`-ből `sf`-et egyszerűen készíthetünk: `st_as_sf()`

Kétváltozós logikai műveletek

```
plot(tajbeosztas_geometria)
plot(st_geometry(nehany_taj), col = "gray", add = TRUE)
plot(st_geometry(v4), border = "orange", lwd = 3, add =
 TRUE)
plot(st_geometry(magyaro_romania), border = "blue", lwd =
 2, add = TRUE)
```



Kétváltozós logikai műveletek

```
st_intersects(x = nehany_taj, y = magyaro_romania)
```

```
Sparse geometry binary predicate list of length 20,
where the predicate was `intersects'
first 10 elements:
 1: 2
 2: 1, 2
 3: 1, 2
 4: 1
 5: 1, 2
 6: 1, 2
 7: 1, 2
 8: 1
 9: 1
 10: 1
```

Kétváltozós logikai műveletek

Értsd: az 1. táj csak a 2. országgal (Románia) metsződik össze, a 2. táj mindkét országgal stb.

```
str(st_intersects(x = nehany_taj, y = magyaro_romania))
```

```
List of 20
 $ : int 2
 $ : int [1:2] 1 2
 $ : int [1:2] 1 2
 $ : int 1
 $ : int [1:2] 1 2
 $ : int [1:2] 1 2
 $ : int [1:2] 1 2
 $ : int 1
 $ : int 1
 $ : int 1
 $ : int 1
 $ : int 1
 $ : int 1
 $ : int 1
```

Kétváltozós logikai műveletek

```
osszemetszodesi_matrix <- st_intersects(x = nehany_taj, y
 = magyaro_romania, sparse = FALSE)
str(osszemetszodesi_matrix)
```

```
 logi [1:20, 1:2] FALSE TRUE TRUE TRUE TRUE TRUE ...
```

```
head(osszemetszodesi_matrix)
```

```
      [,1]  [,2]
[1,] FALSE  TRUE
[2,]  TRUE  TRUE
[3,]  TRUE  TRUE
[4,]  TRUE FALSE
[5,]  TRUE  TRUE
[6,]  TRUE  TRUE
```

Tájak (x) a sorok, országok (y) az oszlopok.

Kétváltozós logikai műveletek

Nevezzük el az oszlopokat, hogy átláthatóbb legyen!

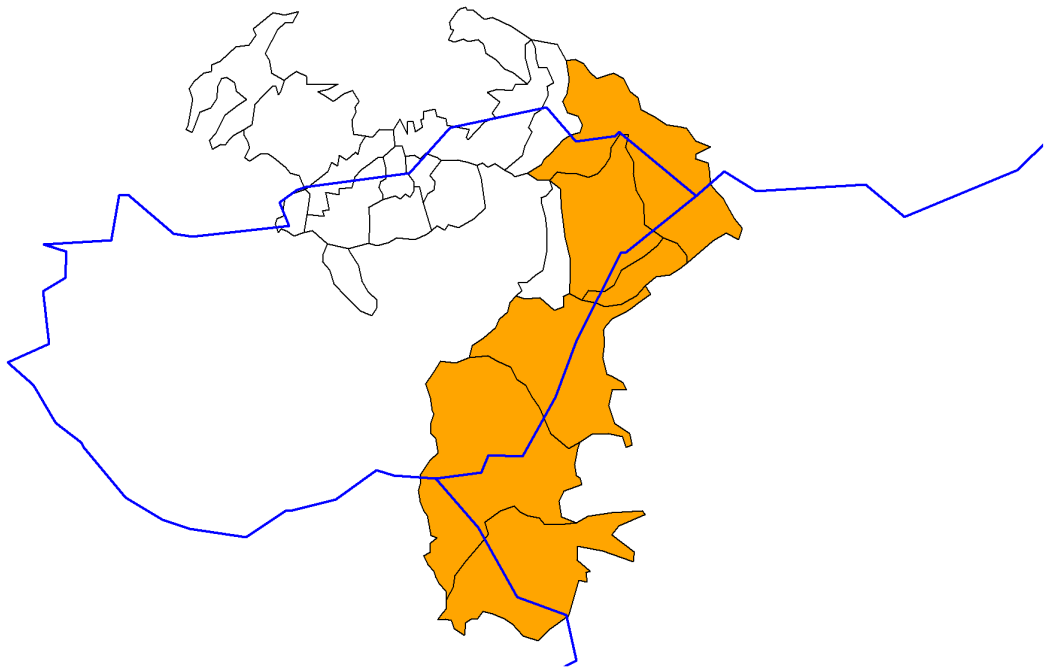
```
magyaro_romania$regio_kod
```

```
[1] HU RO
1951 Levels: AT AT1 AT11 AT111 AT112 AT113 AT12 ... UKN05
```

```
colnames(osszemetszodesi_matrix) <-
 magyaro_romania$regio_kod
nehany_taj$romaniaval_osszemetszodik <-
 osszemetszodesi_matrix[, "RO"]
```

Kétváltozós logikai műveletek

```
plot(st_geometry(nehany_taj))
plot(st_geometry(nehany_taj[
 nehany_taj$romaniaval_osszemetszodik, ]), col = "orange",
 add = TRUE)
plot(st_geometry(magyaro_romania), border = "blue", lwd =
 2, add = TRUE)
```

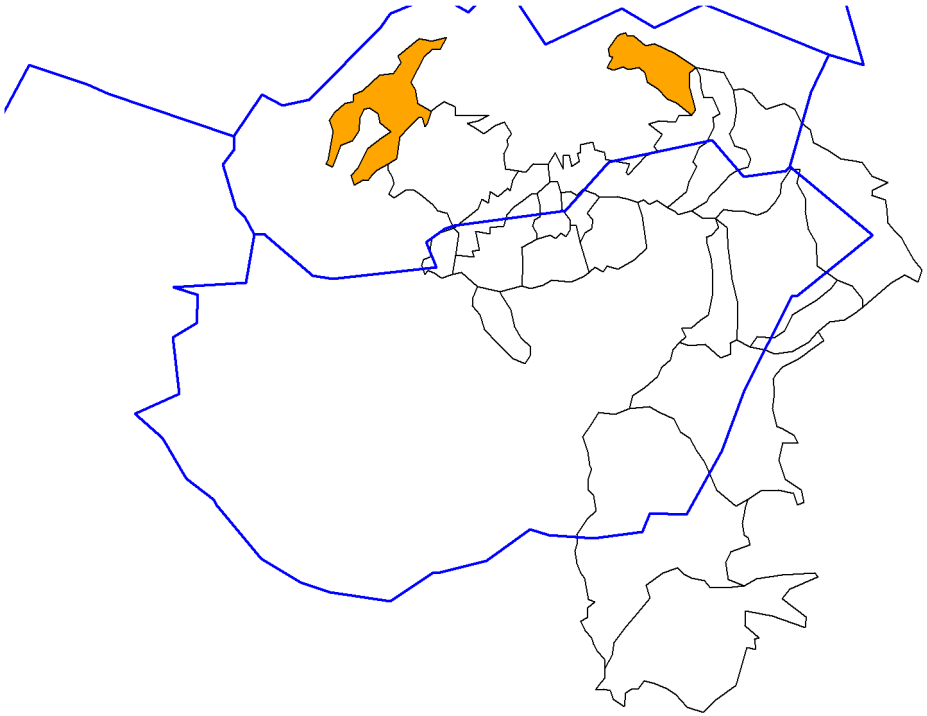


Kétváltozós logikai műveletek

```
beleesesi_matrix <- st_within(x = nehany_taj, y = v4,
 sparse = FALSE)
colnames(beleesesi_matrix) <- v4$regio_kod
nehany_taj$szlovakiaban_van <- beleesesi_matrix[, "SK"]
```

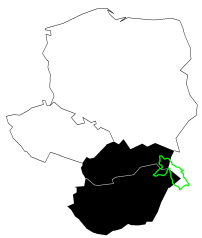
Kétváltozós logikai műveletek

```
plot(st_geometry(nehany_taj))
plot(st_geometry(nehany_taj[nehany_taj$szlovakiaban_van,
 ]), col = "orange", add = TRUE)
plot(st_geometry(v4), border = "blue", lwd = 2, add = TRUE)
```



9. feladat (órai)

- Metsszed össze a visegrádi négyeket (“v4”) a kiválasztott néhány tájegységgel (“nehany_taj”). Az eredményként kapott logikai mátrixot “osszemetszodesi_matrix” néven tárold.
- Nevezd el a mátrix oszlopait tájak azonosítói alapján.
- Adj egy új, “osszemetszodik_a_37essel” nevű oszlopot a visegrádi négyekhez, amely azt mutatja, hogy a “37”-es azonosítójú tájjal összemetsződik-e az adott ország.
- Jelenítsd meg a visegrádi négyek geometriáját.
  - Add hozzá az ábrához fekete kitöltőszínnel és fehér körvonalszínnel ezek közül azon országok geometriáját, amelyek összemetsződtek a kiválasztott tájjal.
  - Továbbá zöld, háromszoros vastagságú körvonallal add hozzá a 37-es azonosítójú tájat.

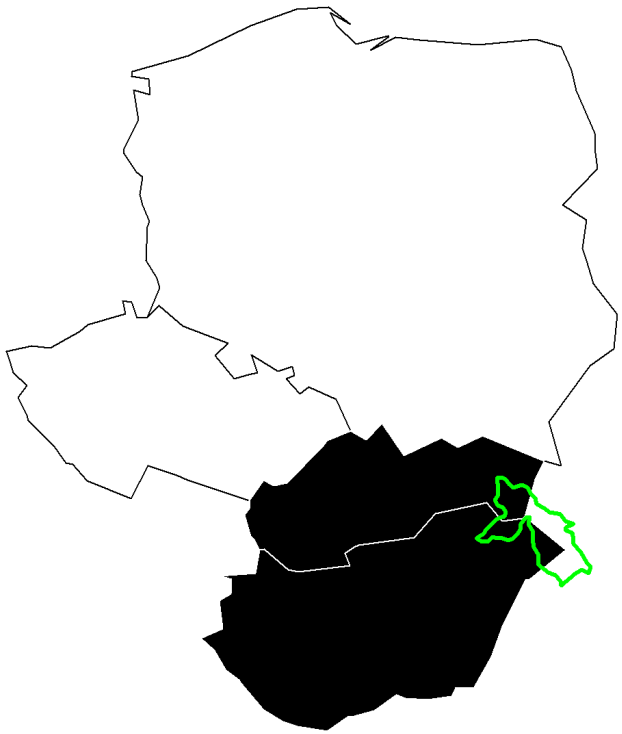


9. feladat (órai) – megoldás

```
osszemetszodesi_matrix <- st_intersects(x = v4, y =
 nehany_taj, sparse = FALSE)
colnames(osszemetszodesi_matrix) <-
 as.character(nehany_taj$azonosito)
v4$osszemetszodik_a_37essel <- osszemetszodesi_matrix[,
 "37"]
```

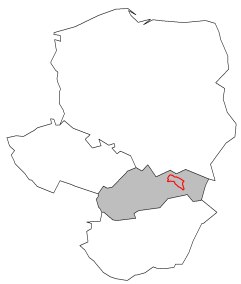
9. feladat (órai) – megoldás

```
plot(st_geometry(v4))
plot(st_geometry(v4[v4$osszemetszodik_a_37essel, ]), col =
 "black", border = "white", add = TRUE)
plot(st_geometry(nehany_taj[nehany_taj$azonosito == 37,
 ]), border = "green", lwd = 3, add = TRUE)
```



10. feladat (házi)

- Készíts “lefedesi_matrix” néven egy logikai mátrixot arról, hogy az egyes visegrádi országok (“v4”) lefedik-e (“covers”) a kiválasztott néhány tájegységet (“nehany_taj”).
- Nevezd el a mátrix oszlopait tájak azonosítói alapján.
- Adj egy új, “lefedi_a_48ast” nevű oszlopot a visegrádi négyekhez, amely azt mutatja, hogy a “48”-es azonosítójú tájat lefedi-e az adott ország.
- Jelenítsd meg a visegrádi négyek geometriáját.
  - Add hozzá az ábrához szürke kitöltéssel ezek közül azon országok geometriáját, amelyek lefedték a kiválasztott tájat.
  - Továbbá piros, kétszeres vastagságú körvonallal add hozzá a 48-es azonosítójú tájat.

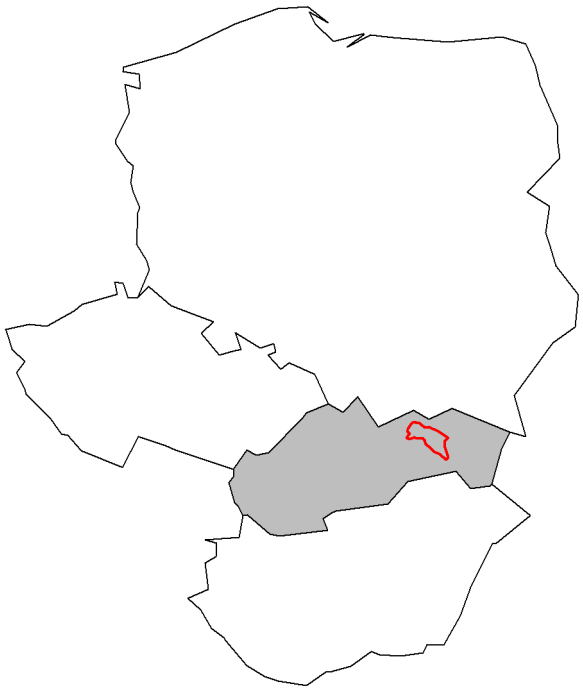


10. feladat (házi) – megoldás

```
lefedesi_matrix <- st_covers(x = v4, y = nehany_taj,
 sparse = FALSE)
colnames(lefedesi_matrix) <-
 as.character(nehany_taj$azonosito)
v4$lefedi_a_48ast <- lefedesi_matrix[, "48"]
```

10. feladat (házi) – megoldás

```
plot(st_geometry(v4))
plot(st_geometry(v4[v4$lefedi_a_48ast, ]), col = "gray",
 add = TRUE)
plot(st_geometry(nehany_taj[nehany_taj$azonosito == 48,
 ]), border = "red", lwd = 2, add = TRUE)
```



Összefűzés geometriai kapcsolat alapján

Összefűzés geometriai kapcsolat alapján

- spatial join
- nem kétváltozós logikai művelet, de azokra támaszkodik
- geometriai kapcsolat (pl. összemetsződés) alapján összerendezi két `sf` adatait

`st_join(x, y, join = st_intersects, left = TRUE)`

- `x`: amihez hozzá szeretnénk fűzni új oszlopokat
- `y`: amiből származnak majd az új adatok
- `join`: a geometriai kapcsolatot leíró függvény ("`x` *predikátum* `y`"-nak kell teljesülnie a hozzáfűzéshez)
- `left`: `x` összes elemét megtartsa-e?
- `left = FALSE` esetben ("inner join") csak azokat tarta meg, amihez talált geometriai kapcsolatot

Összefűzés geometriai kapcsolat alapján

```
nrow(nehany_taj)
```

```
[1] 20
```

```
tajak_orszaginfokkal <- st_join(x = nehany_taj, y = v4,
 join = st_within, left = TRUE)
nrow(tajak_orszaginfokkal)
```

```
[1] 20
```

Összefűzés geometriai kapcsolat alapján

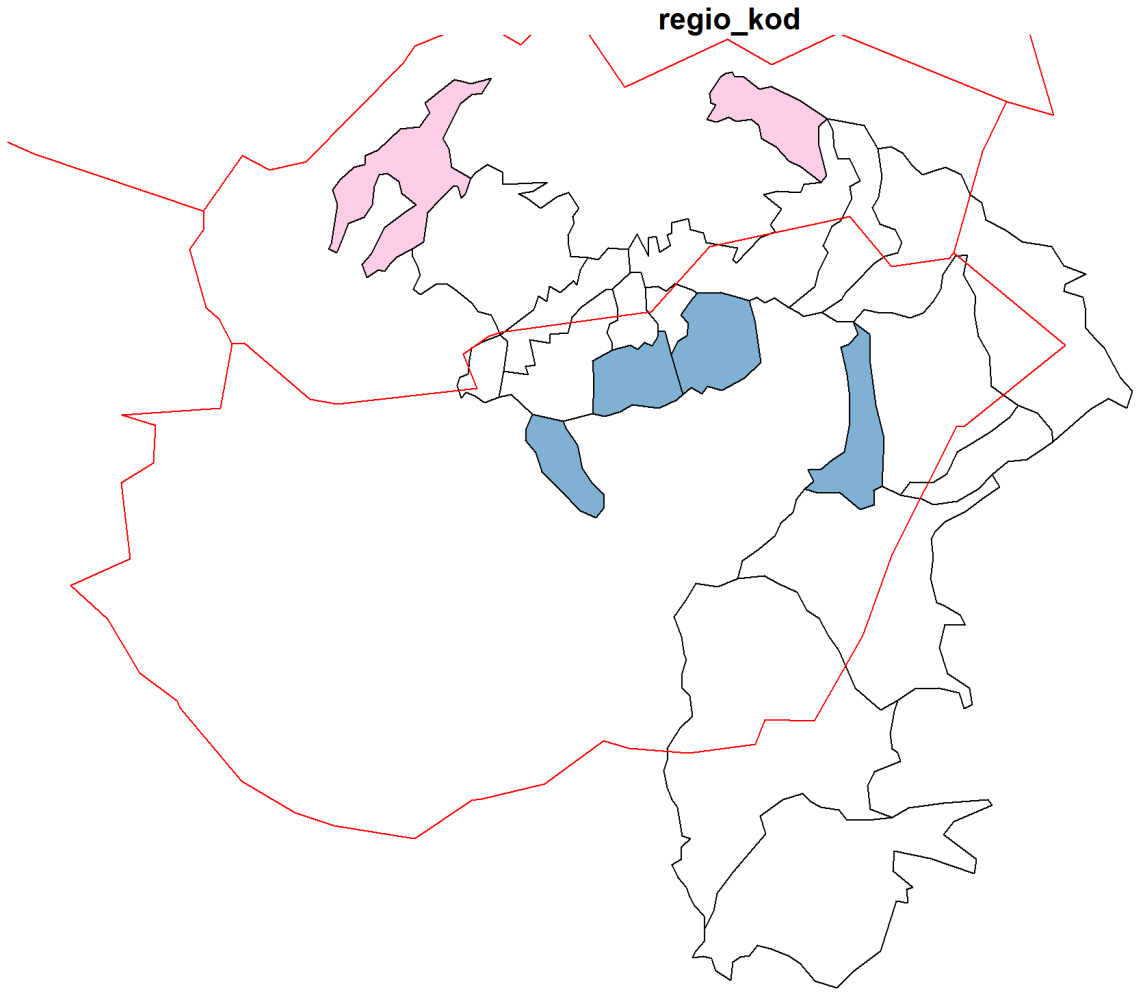
```
names(tajak_orszaginfokkal)
```

```
 [1] "azonosito"
 [2] "romaniaval_osszemetszodik"
 [3] "szlovakiaban_van"
 [4] "regio_kod"
 [5] "regio_szint"
 [6] "terulet"
 [7] "hatarhossz"
 [8] "osszemetszodik_a_37essel"
 [9] "lefedi_a_48ast"
[10] "x"
```

Mindkét adatsor (`x`, `y`) oszlopai megtalálhatóak az eredményben.

Összefűzés geometriai kapcsolat alapján

```
plot(tajak_orszaginfokkal[, "regio_kod"], reset = FALSE)
plot(st_geometry(v4), border = "red", add = TRUE)
```



Összefűzés geometriai kapcsolat alapján

`left = FALSE` esetben `x` részhalmazát kapjuk vissza.

```
tajak_orszaginfokkal <- st_join(x = nehany_taj, y = v4,
 join = st_within, left = FALSE)
nrow(tajak_orszaginfokkal)
```

```
[1] 6
```

11. feladat (házi)

- Töltsd be a “repterek.RData” fájlt.
- Válogass le “egy_ket_regio” néven néhány magyarországi régiót a “nuts_regiok” Simple Featuresből, pontosabban azokat, amelyeknek a kódja (“regio_kod” oszlop) ez: “HU10”, “HU21”, “HU22”, “HU30”, “HU32”, “HU33”. A leválogatott régiókat vetítsd át EOV vetületbe (23700).
- Hozz létre egy új Simple Featurest “repterek_regioinfokkal” néven, amely minden repteret tartalmaz, továbbá fennálló geometriai kapcsolat esetén a reptér adatain túl azon – előbb leválogatott – régiók adatait is tartalmazza, amely régióba a reptér beleesik (“within”).
- Ábrázold repülőtereket a tartalmazó régió területe alapján színezett 16-os pontjellel.
  - Add hozzá a repterek_regioinfokkal geometriáját. (Hogy lássuk az összefűzés összes elemét...)
  - Továbbá add hozzá az egy_ket_regio geometriáját kék körvonallal.
- Ne felejtsd el nyitva hagyni a képvásznat!

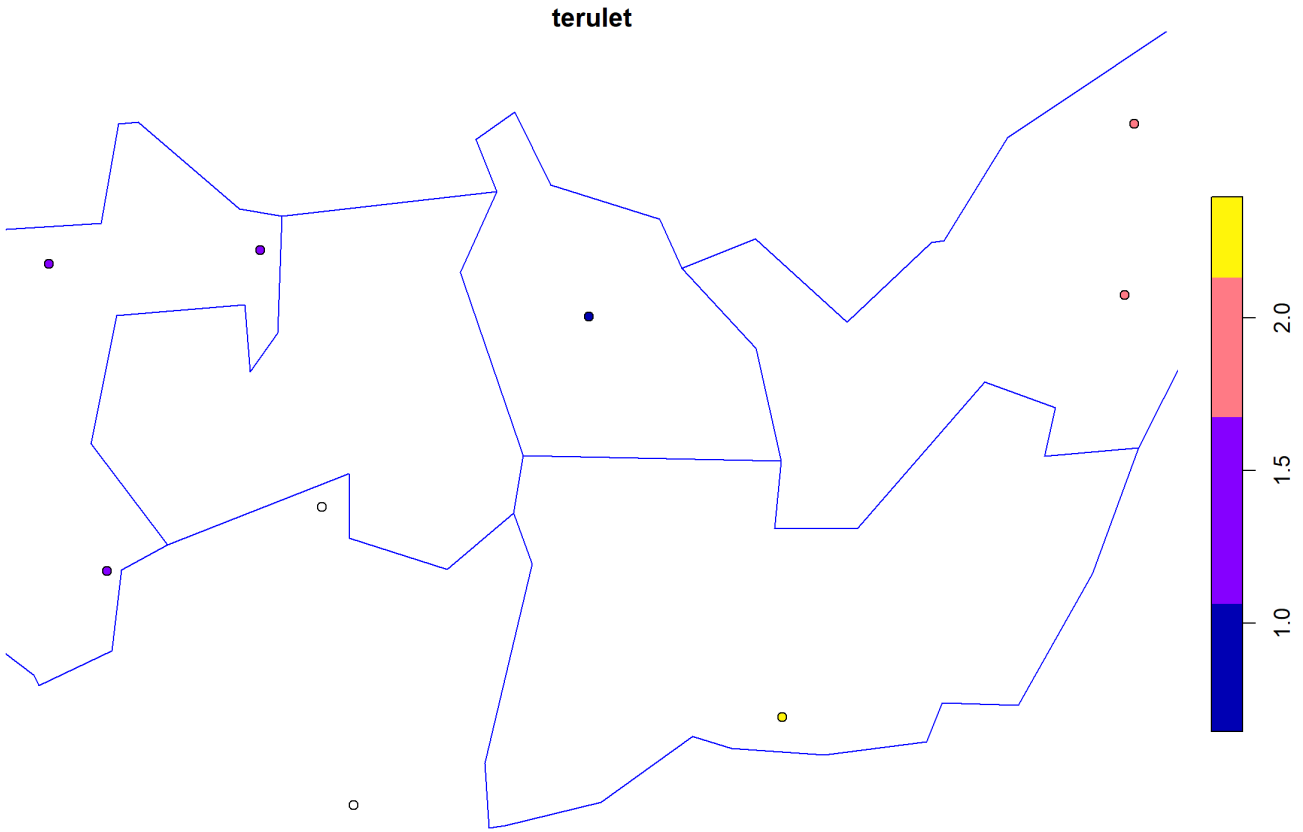
11. feladat (házi) – megoldás

```
load("repterek.RData")
```

```
egy_ket_regio <- st_transform(x =
 nuts_regiok[nuts_regiok$regio_kod %in% c("HU10", "HU21",
 "HU22", "HU30", "HU32", "HU33"), ], crs = 23700)
repterek_regioinfokkal <- st_join(x = repterek, y =
 egy_ket_regio, join = st_within, left = TRUE)
```

11. feladat (házi) – megoldás

```
plot(repterek_regioinfokkal[, "terulet"], pch = 16, reset
 = FALSE)
plot(st_geometry(repterek_regioinfokkal), add = TRUE)
plot(st_geometry(egy_ket_regio), border = "blue", add =
 TRUE)
```



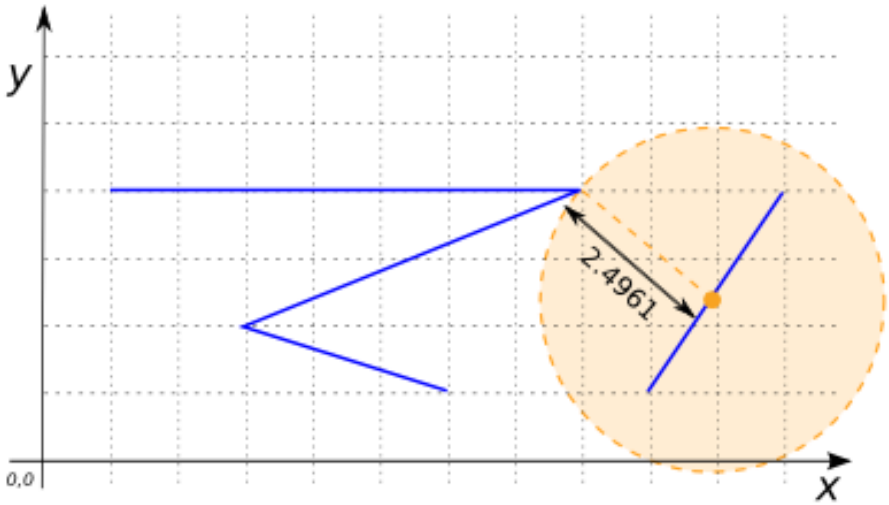
Section 4

Kétváltozós geometriai műveletek

Kétváltozós geometriai műveletek

Távolságszámítás

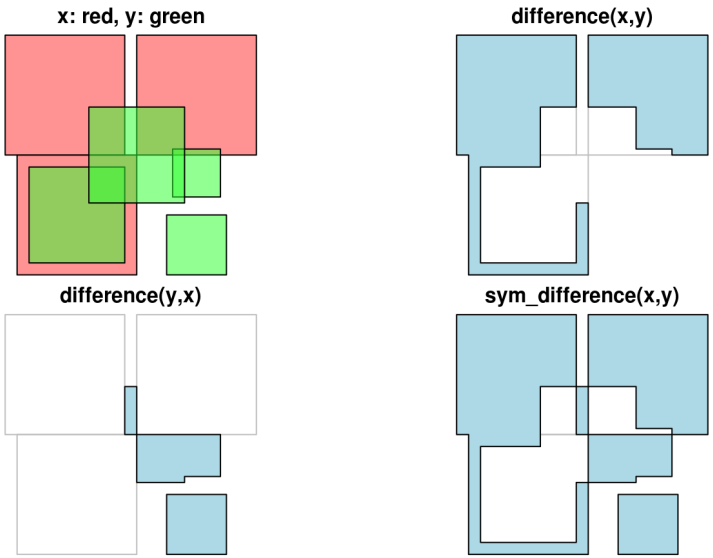
- nem igazi kétváltozós geometriai művelet
- két geometria alapján számot ad eredményül
- `st_distance(x, y)`



Kétváltozós geometriai műveletek

Valódi kétváltozós geometriai műveletek

- két geometria alapján geometriát adnak eredményül
- `st_union(x, y)`
- `st_intersection(x, y)`
- `st_difference(x, y)`
- `st_sym_difference(x, y)`



Távolságszámítás

`st_distance(x, y)`

- `x` és `y`: `sf` vagy `sfc` objektumok
- több elemet is tartalmazhatnak
- `x` minden elemének számítja az `y` elemeitől vett távolságát
- eredmény: mátrix, mértékegységgel (`units`)
- `y` elhagyható, akkor `x` elemeinek `x` elemeitől vett távolságát számítja (diagonális 0)

Távolságszámítás

```
szombathely_debrecen <- varosok[varosok$nev %in%
 c("Szombathely", "Debrecen"), ]
repter_varos_tav <- st_distance(x = repterek, y =
 szombathely_debrecen)
rownames(repter_varos_tav) <- repterek$nev
colnames(repter_varos_tav) <- szombathely_debrecen$nev
```

Elnevezzük a sorokat/oszlopokat, hogy könnyebb legyen értelmezni az eredményt.

Távolságszámítás

```
repter_varos_tav
```

```
Units: [m]
                  Debrecen Szombathely
Ferihegy         173567.995   196715.43
Debrecen           1712.672   374718.41
Gyor-Pér         282406.636    97186.95
Sármellék        347629.697    69632.04
Pécs-Pogány      306455.306   180983.71
Fertoszentmiklós 352624.998    40165.41
Nyíregyháza       47619.406   386717.99
Siófok-Kiliti    272991.550   113373.35
Szeged           181038.824   281496.08
```

Távolságszámítás

Önmagától vett távolság (y-t elhagyjuk, vagy ugyanazt adjuk meg, mint x-nél):

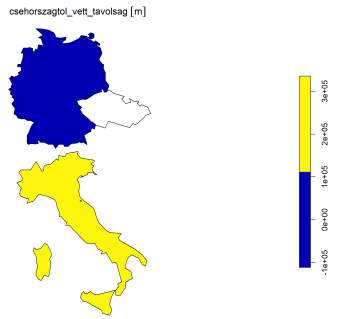
```
egymastol_vett_tavolsag <- st_distance(x = repterek)
colnames(egymastol_vett_tavolsag) <-
 rownames(egymastol_vett_tavolsag) <- repterek$nev
```

```
egymastol_vett_tavolsag[1:5, 1:3]
```

```
Units: [m]
             Ferihegy Debrecen Gyor-Pér
Ferihegy          0.0 178296.8 111523.4
Debrecen     178296.8      0.0 287832.0
Gyor-Pér     111523.4 287832.0      0.0
Sármellék    181370.5 350816.3 118398.5
Pécs-Pogány 180314.3 307525.1 187164.3
```

12. feladat (órai)

- Válogasd le a NUTS-régiók (“nuts_regiok”) közül az “IT” és “DE” kódú (“regio_kod” oszlop) régiókat “olaszo_nemeto” néven, és a “CZ” és “SK” kódú régiókat cseho_szlovakia néven.
- Számítsd ki a két országpáros elemeinek egymástól vett távolságá.
- Az eredménymátrix sorait és oszlopait nevezd el a régiókódoknak megfelelóen, majd a mátrixot írasd ki a képernyőre.
- Az “olaszo_nemeto” Simple Featureshöz adj egy új, “csehorszagtol_vett_tavolsag” nevű oszlopot, amibe kerüljön a Csehországtól vett távolság.
- Jelenítsd meg Olaszországot és Németországot a Csehországtól vett távolságuk alapján színezve.
  - Add ehhez az ábrához Csehország körvonalát.



12. feladat (órai) – megoldás

```
olaszo_nemeto <- nuts_regiok[nuts_regiok$regio_kod %in%
 c("IT", "DE"), ]
cseho_szlovakia <- nuts_regiok[nuts_regiok$regio_kod %in%
 c("CZ", "SK"), ]
tavolsagok <- st_distance(x = olaszo_nemeto, y =
 cseho_szlovakia)
rownames(tavolsagok) <- olaszo_nemeto$regio_kod
colnames(tavolsagok) <- cseho_szlovakia$regio_kod
```

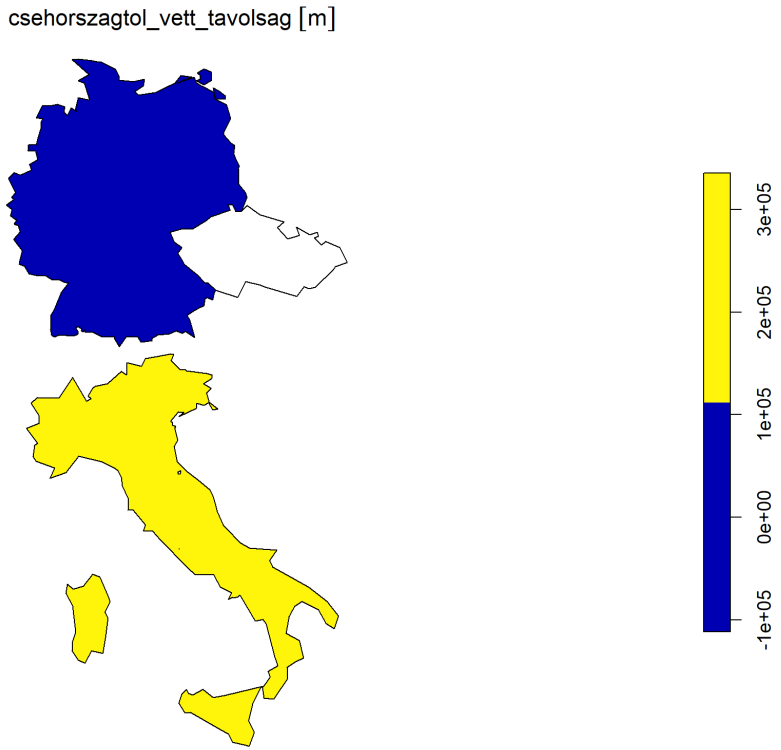
```
tavolsagok
```

```
Units: [m]
          CZ       SK
DE       0.0 224492.7
IT 223660.2 306617.3
```

12. feladat (órai) – megoldás

```
olaszo_nemeto$csehorszagtol_vett_tavolsag <- tavolsagok[,
 "CZ"]
```

```
plot(olaszo_nemeto[, "csehorszagtol_vett_tavolsag"], reset
 = FALSE)
plot(st_geometry(cseho_szlovakia[cseho_szlovakia$regio_kod
 == "CZ", ]), add = TRUE)
```



Unió, metszet és különbségek

Unió/egyesítés

- `st_union(x, y)`
- `y`: elhagyható, ekkor `x` elemeit egyesíti

Metszet

- `st_intersection(x, y)`

Különbség

- `st_difference(x, y)`
- `x` és `y` sorrendjének jelentősége van

Szimmetrikus különbség

- `st_sym_difference(x, y)`

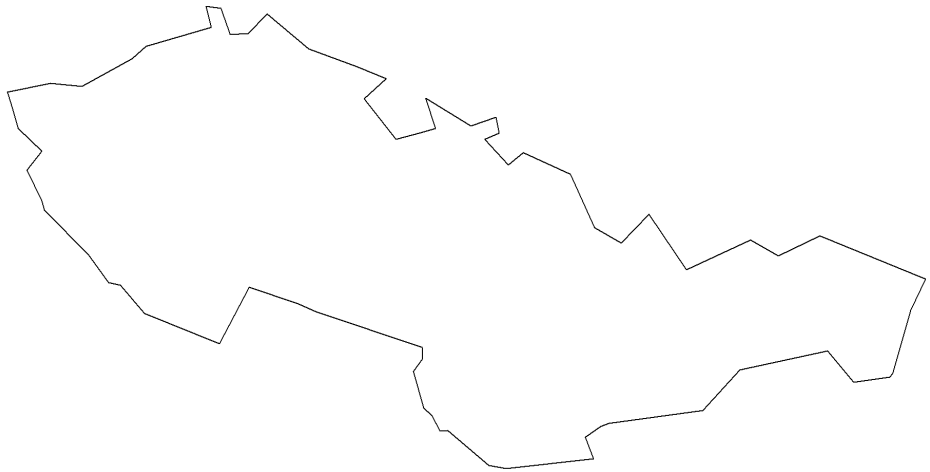
Unió, metszet és különbségek

A szimmetrikus különbség kiszámítható a többi halmazművelet segítségével:

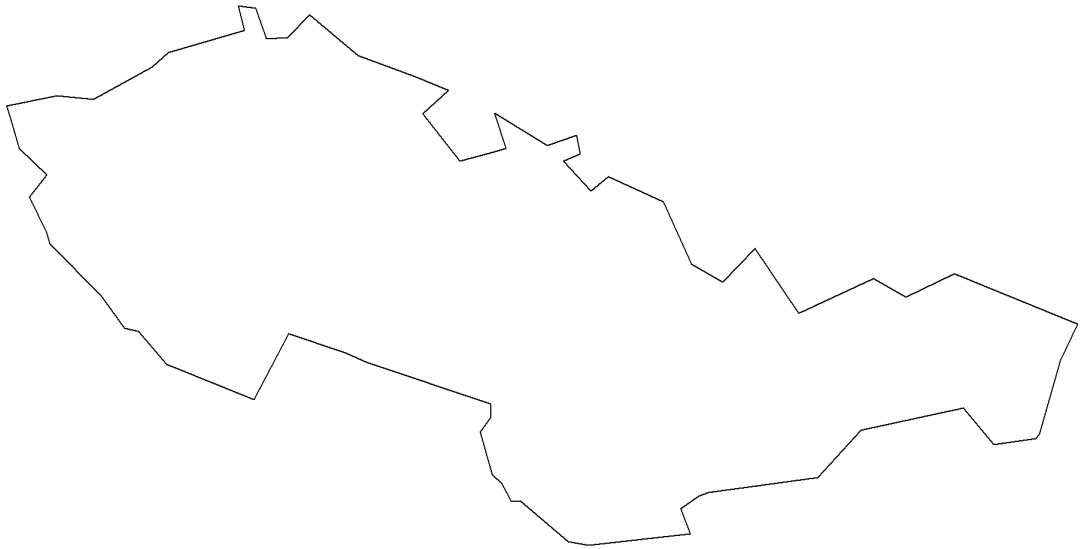
- `st_union(st_difference(x, y), st_difference(y, x))` vagy
- `st_difference(st_union(x, y), st_intersection(x, y))`.

Unió, metszet és különbségek

```
szlovakia <- st_transform(x =
 st_geometry(nuts_regiok[nuts_regiok$regio_kod == "SK",
 ]), crs = 23700)
csehorszag <- st_transform(x =
 st_geometry(nuts_regiok[nuts_regiok$regio_kod == "CZ",
 ]), crs = 23700)
csehszlovakia <- st_union(x = szlovakia, y = csehorszag)
plot(csehszlovakia)
```



Unió, metszet és különbségek



Most egy-egy elemet egyesítettünk. Ha `x` és/vagy `y` többelemű lenne, akkor az `st_union()` a `by_feature` logikai paraméterétől függően egy (alapértelmezett) vagy több elemet adna eredményül.

Unió, metszet és különbségek

```
class(magyaro_romania)
```

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

```
nrow(magyaro_romania)
```

```
[1] 2
```

```
magyaro_romania <- st_union(x = magyaro_romania)
```

Unió, metszet és különbségek

```
class(magyaro_romania)
```

```
[1] "sfc_POLYGON" "sfc"
```

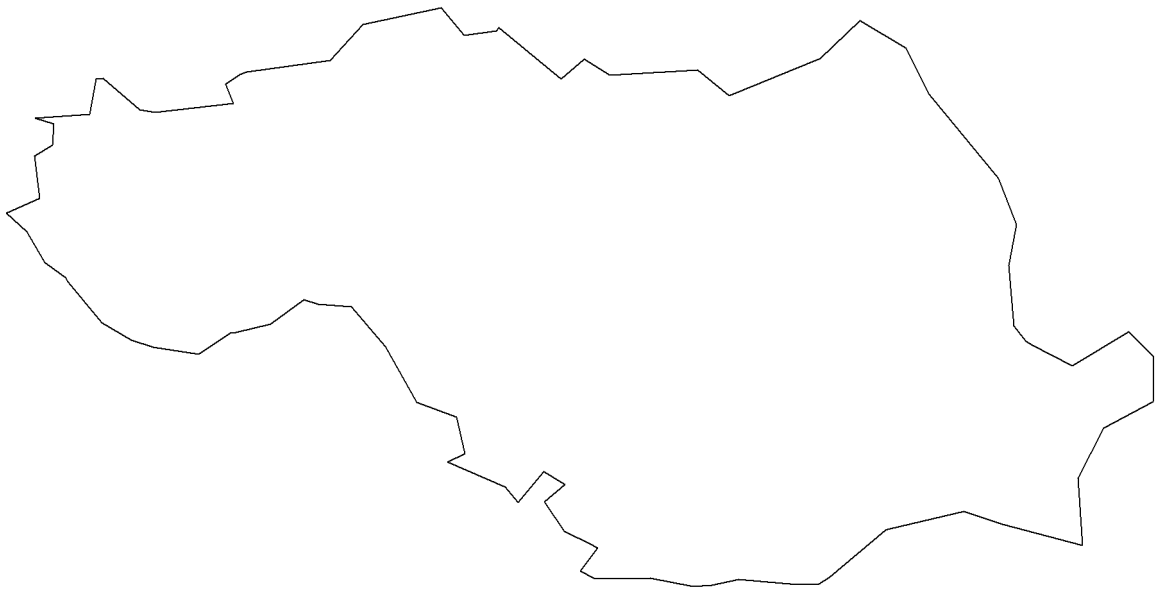
```
length(magyaro_romania)
```

```
[1] 1
```

A kétváltozós geometriai műveletek eredménye csak geometria.
Az adatok elvesznek (hiszen melyiket kellene megtartani?).

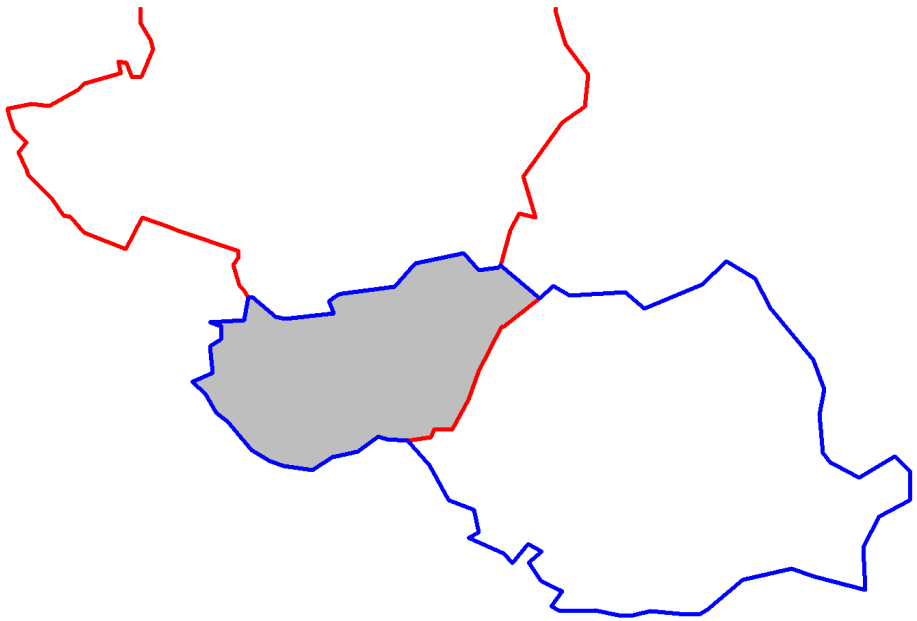
Unió, metszet és különbségek

```
plot(magyaro_romania)
```



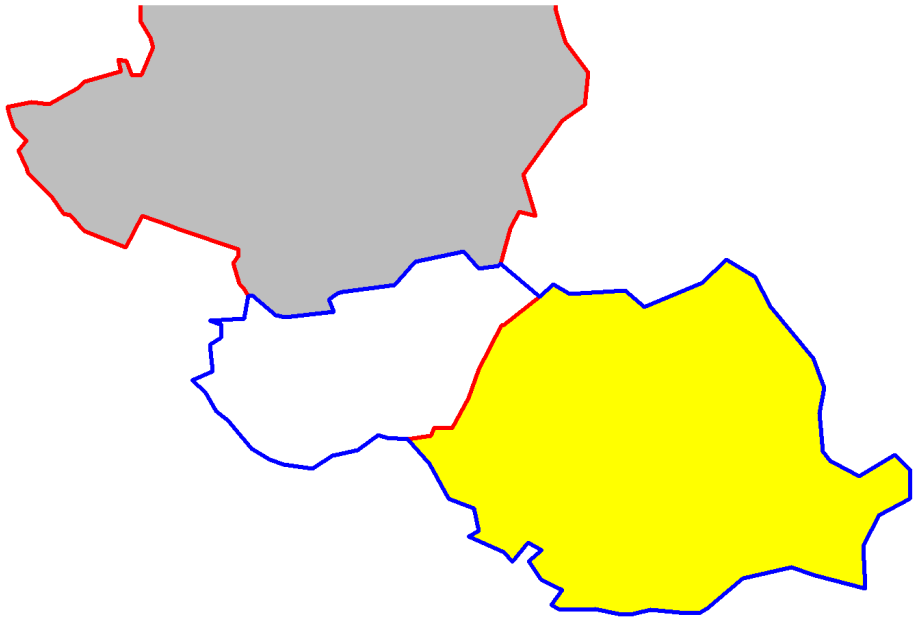
Unió, metszet és különbségek

```
v4 <- st_union(x = v4)
plot(st_intersection(x = v4, y = magyaro_romania), col =
 "gray", xlim = c(-200000, 1600000), ylim = c(-200000,
 500000))
plot(v4, border = "red", lwd = 3, add = TRUE)
plot(magyaro_romania, border = "blue", lwd = 3, add = TRUE)
```



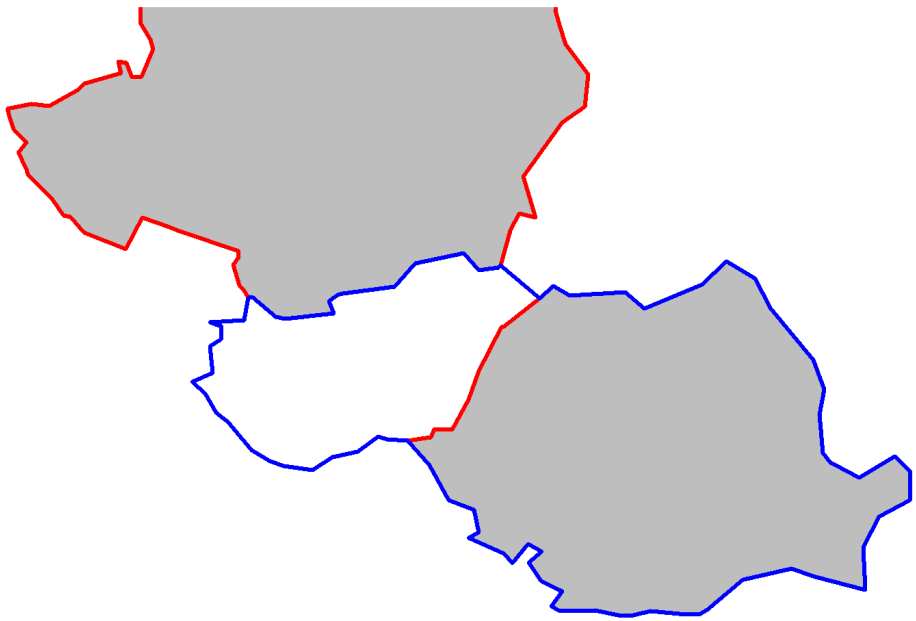
Unió, metszet és különbségek

```
plot(st_difference(x = v4, y = magyaro_romania), col =
 "gray", xlim = c(-200000, 1600000), ylim = c(-200000,
 500000))
plot(st_difference(x = magyaro_romania, y = v4), col =
 "yellow", add = TRUE)
plot(v4, border = "red", lwd = 3, add = TRUE)
plot(magyaro_romania, border = "blue", lwd = 3, add = TRUE)
```



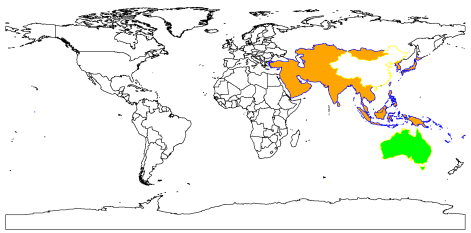
Unió, metszet és különbségek

```
plot(st_sym_difference(x = v4, y = magyaro_romania), col =
  "gray", xlim = c(-200000, 1600000), ylim = c(-200000,
  500000))
plot(v4, border = "red", lwd = 3, add = TRUE)
plot(magyaro_romania, border = "blue", lwd = 3, add = TRUE)
```



13. feladat (házi)

- Olvasd be az “orszagok_osszes.RData” fájlt.
- Kapcsold ki a gömbfelszíni geometriát az `sf_use_s2(FALSE)` függvényhívással.
- Válogasd le “azsia” néven azon országokat, amelyeknek a regio oszlopa “Asia” értéket tartalmaz, egyesítsd a poligonokat.
- Válogasd le “ausztralia_kina” néven az “Australia” és “China” nevű országokat, ezeket is egyesítsd.
- Jelenítsd meg az összes ország geometriáját.
  - Narancssárga kitöltéssel add ehhez az ábrához Ázsia, illetve Ausztrália–Kína kettősének különbségét.
  - Zöld kitöltéssel add hozzá Ausztrália–Kína, valamint Ázsia különbségét.
  - Kék határvonallal rakd rá Ázsiát, illetve sárga körvonallal az Ausztrália–Kína kettőst.



13. feladat (házi) – megoldás

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

```
sf_use_s2(FALSE)
```

```
Spherical geometry (s2) switched off
```

```
azsia <- st_union(x =
 orszagok_osszes[orszagok_osszes$regio == "Asia", ])
```

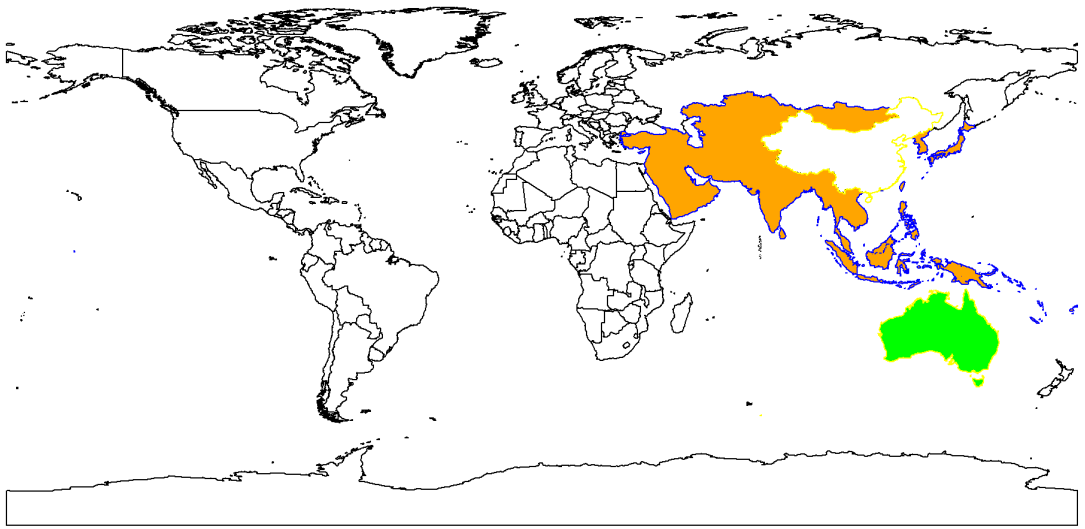
```
although coordinates are longitude/latitude, st_union
assumes that they are planar
```

```
ausztralia_kina <- st_union(x =
 orszagok_osszes[orszagok_osszes$nev %in% c("Australia",
 "China"), ])
```

```
although coordinates are longitude/latitude, st_union
assumes that they are planar
```

13. feladat (házi) – megoldás

```
plot(st_geometry(orszagok_osszes))
plot(st_difference(x = azsia, y = ausztralia_kina), col =
  "orange", add = TRUE)
plot(st_difference(x = ausztralia_kina, y = azsia), col =
  "green", add = TRUE)
plot(azsia, border = "blue", add = TRUE)
plot(ausztralia_kina, border = "yellow", add = TRUE)
```



13. feladat (házi) – megoldás

A végén kapcsoljuk vissza a gömbfelszíni geometriát:

```
sf_use_s2(TRUE)
```

```
Spherical geometry (s2) switched on
```

14. (összefoglaló) feladat (házi)

- Olvasd be az “orszag_eu.RData”-t.
- Van a “tajbeosztas_geometria” nevű `sfc`-objektumnak olyan eleme, ami `"POINT"` típusú?
- Képezd “taj” néven a tajbeosztas_geometria 11. és 12. elemeinek unióját! Majd 2000 méter sugárral képezd e tájegység pufferét, írd felül a meglévő “taj”-at.
- Mekkora a tájegység területe? És mi a vetülete? Vetítsd át WGS-84 koordináta-rendszerbe (4326).
- Mi ennek a tájegységnek az Olaszországtól (“olaszo” nevű objektum) vett távolsága?
- Jelenítsd meg az “orszag_eu” geometriáját a 15-30°K és 40-50°É koordináta-tartományban.
- Add hozzá piros kitöltőszínnel az átvetített tájegységet.
- Alakítsd át a “taj”-at Simple Features-zé. Majd kapcsold a tájegységhez azon ország adatait, amelyben található (“within”).
- Ellenőrzésképpen jelenítsd meg az eredmény “orszagkod” oszlopát.

14. (összefoglaló) feladat (házi) – megoldás

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

```
any(st_is(x = tajbeosztas_geometria, type = "POINT"))
```

```
[1] FALSE
```

```
taj <- st_union(tajbeosztas_geometria[c(11, 12)])
taj <- st_buffer(x = taj, dist = 2000)
st_area(taj)
```

```
5966070205 [m^2]
```

14. (összefoglaló) feladat (házi) – megoldás

```
st_crs(taj)
```

```
Coordinate Reference System:
  User input: EPSG:23700
  wkt:
PROJCRS["HD72 / EOV",
    BASEGEOGCRS["HD72",
        DATUM["Hungarian Datum 1972",
            ELLIPSOID["GRS 1967",6378160,298.247167427,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4237]],
    CONVERSION["Egyseges Orszagos Vetuleti",
        METHOD["Hotine Oblique Mercator (variant B)",
            ID["EPSG",9815]],
                                PARAMETER["Latitude
                                 of
                                 projection
                                 centre",47.1443937222222
```

14. (összefoglaló) feladat (házi) – megoldás

```
taj <- st_transform(x = taj, crs = 4326)
st_distance(x = taj, y = olaszo)
```

```
Units: [m]
          [,1]
[1,] 263742.7
```

14. (összefoglaló) feladat (házi) – megoldás

```
plot(st_geometry(orszag_eu), xlim = c(15, 30), ylim =
  c(40, 50))
plot(taj, col = "red", add = TRUE)
```

14. (összefoglaló) feladat (házi) – megoldás

```
taj <- st_as_sf(taj)
taj_orszaginfokkal <- st_join(x = taj, y = orszag_eu, join
 = st_within)
taj_orszaginfokkal$orszagkod
```

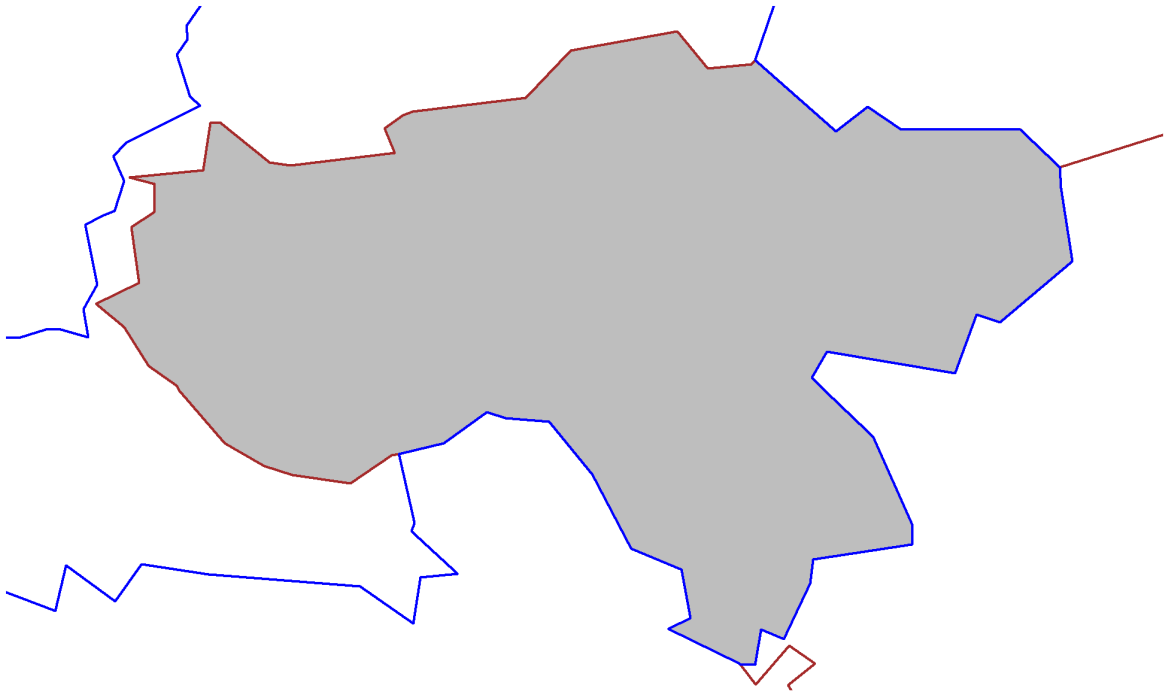
```
[1] HU
1951 Levels: AT AT1 AT11 AT111 AT112 AT113 AT12 ... UKN05
```

15. (összefoglaló) feladat (házi)

- Olvasd be a “magyarorszag.RData” fájlt.
- “regiok” néven válogasd le a “nuts_regiok”-ból azon régiókat, amelyek 2. szintűek (“regio_szint” oszlop).
- Metszd össze a régiókat Magyarországgal, az eredményként kapott logikai mátrixot “osszemetszodesi_matrix” néven tárold.
- “osszemetszo_regiok” néven kérd le azon régiókat, amelyek összemetsződtek Magyarországgal. Egyesítsd ezen régiókat.
- Vetítsd át EOV-ba (23700) a “magyaro_romania” nevű Simple Featurest. Tartalmaz üres geometriát?
- Hozd létre “metszet” néven “magyaro_romania” és “osszemetszo_regiok” metszetét. Képezd e metszet határvonalát, 0.5°-os toleranciatávolsággal egyszerűsítsd azt, majd alakítsd vissza poligonná. ...

15. (összefoglaló) feladat (házi)

- Ábrázold e poligon geometriáját szürke kitöltéssel.
  - Add hozzá “magyaro_romania” geometriáját barna, kétszeres vastagságú körvonalként.
  - Majd add hozzá az “osszemetszo_regiok”-at kék, kétszeres vastagságú körvonalként.



15. (összefoglaló) feladat (házi) – megoldás

```
load("magyarorszag.RData")
regiok <- nuts_regiok[nuts_regiok$regio_szint == 2, ]
osszemetszodesi_matrix <- st_intersects(x = regiok, y =
 magyarorszag, sparse = FALSE)
osszemetszo_regiok <- regiok[osszemetszodesi_matrix, ]
osszemetszo_regiok <- st_union(osszemetszo_regiok)
```

15. (összefoglaló) feladat (házi) – megoldás

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

```
any(st_is_empty(magyaro_romania))
```

```
[1] FALSE
```

```
metszet <- st_intersection(x = magyaro_romania, y =
 osszemetszo_regiok)
metszet <- st_polygonize(st_simplify(x =
 st_boundary(metszet), dTolerance = 0.5))
```

15. (összefoglaló) feladat (házi) – megoldás

```
plot(st_geometry(metszet), col = "gray")
plot(st_geometry(magyaro_romania), border = "brown", lwd =
 2, add = TRUE)
plot(osszemetszo_regiok, border = "blue", lwd = 2, add =
 TRUE)
```