# Raszter-raszter és raszter-vektor műveletek

## Térinformatika R-ben

2023.11.27.

Section 1

# Raszter-raszter műveletek: vágás és maszkolás

# Raszter körbevágása

## Raszter szélének levágása

- sokszor a raszter szélén csupa `NA` vagy 0 érték szerepel
- ezekre nincs szükségünk, fölöslegesen növelik a raszter kiterjedését
- olyan sorok/oszlopok elhagyása a cél
- amelyek azonos értéket tartalmaznak

## `trim(x, values = NA)`

- `x`: a bemeneti raszter
- `values`: egy vagy több számérték, vagy NA (alapértelmezett)
- eredmény: csökkentett méretű raszter, amelynek a szélén már nincsenek csupa `values`-beli értéket tartalmazó sorok/oszlopok

# Raszter körbevágása

## Raszter körbevágása másik raszter alapján

- van egy raszterünk, ami túl nagy kiterjedésű
- szeretnénk csökkenteni a méretét
- a kívánt méretet egy másik rasztertől kölcsönözzük

## `crop(x, y)`

- `x`: ezt a rasztert szeretnénk kisebbíteni
- `y`: ez a raszter adja a kívánt kivágatot
- az `y` nem csak raszter lehet, hanem bármi, aminek van befoglaló doboza (később...)
- az `y` tartalma lényegtelen, csak az a téglalap fontos, amin belül elhelyezkedik
- eredmény: csökkentett méretű raszter

# Raszter körbevágása

```
library(sf)
library(raster)
domborzatmodell <- raster("domborzatmodell.tif")
csapadek <- raster("csapadek.tif")
felszinboritas <- raster("felszinboritas.tif")
felszinhomerseklet <- raster("felszinhomerseklet.tif")
load("varosok.RData")
load("repterek.RData")
load("nuts_regiok.RData")
load("orszag_eu.RData")
load("utak.RData")
load("eghajlat.RData")
```

# Raszter körbevágása

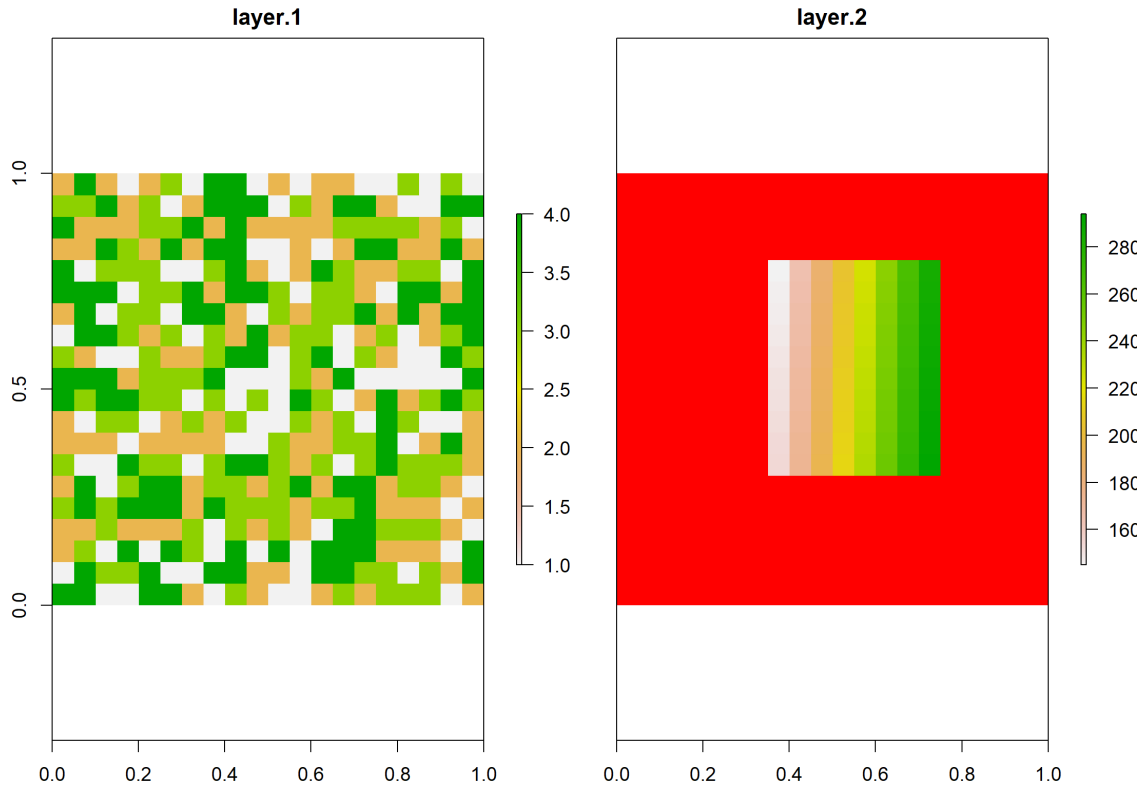
Létrehozunk két kis egészszám-rasztert:

```
set.seed(12345)
szamok1 <- sample(x = 1:4, size = 20 * 20, replace = TRUE)
szammatrix1 <- matrix(data = szamok1, ncol = 20, nrow = 20)
szamraszter1 <- raster(x = szammatrix1)
```

```
szamok2 <- 1:(20 * 20)
szammatrix2 <- matrix(data = szamok2, ncol = 20, nrow = 20)
szamraszter2 <- raster(x = szammatrix2)
szamraszter2[c(1:4, 15:20), ] <- NA
szamraszter2[, c(1:7, 16:20)] <- NA
```

# Raszter körbevágása

```
plot(stack(szamraszter1, szamraszter2), colNA = "red")
```




A második raszter szélén ismeretlen értékű cellák sorakoznak.
Tüntessük el őket!

# Raszter körbevágása

```
dim(szamraszter2)
```

```
[1] 20 20  1
```

```
res(szamraszter2)
```

```
[1] 0.05 0.05
```

```
extent(szamraszter2)
```

```
class       : Extent
xmin        : 0
xmax        : 1
ymin        : 0
ymax        : 1
```

# Raszter körbevágása

Az alapértelmezett `values`-érték most tökéletes lesz nekünk.

```
szamraszter2_korbevagott <- trim(x = szamraszter2, values
 = NA)
```

De ha mondjuk csupa 0 érték szerepelne a szélén, akkor a `values = 0` lenne a megoldás.

# Raszter körbevágása

```
dim(szamraszter2_korbevagott)
```

```
[1] 10  8  1
```

```
res(szamraszter2_korbevagott)
```

```
[1] 0.05 0.05
```

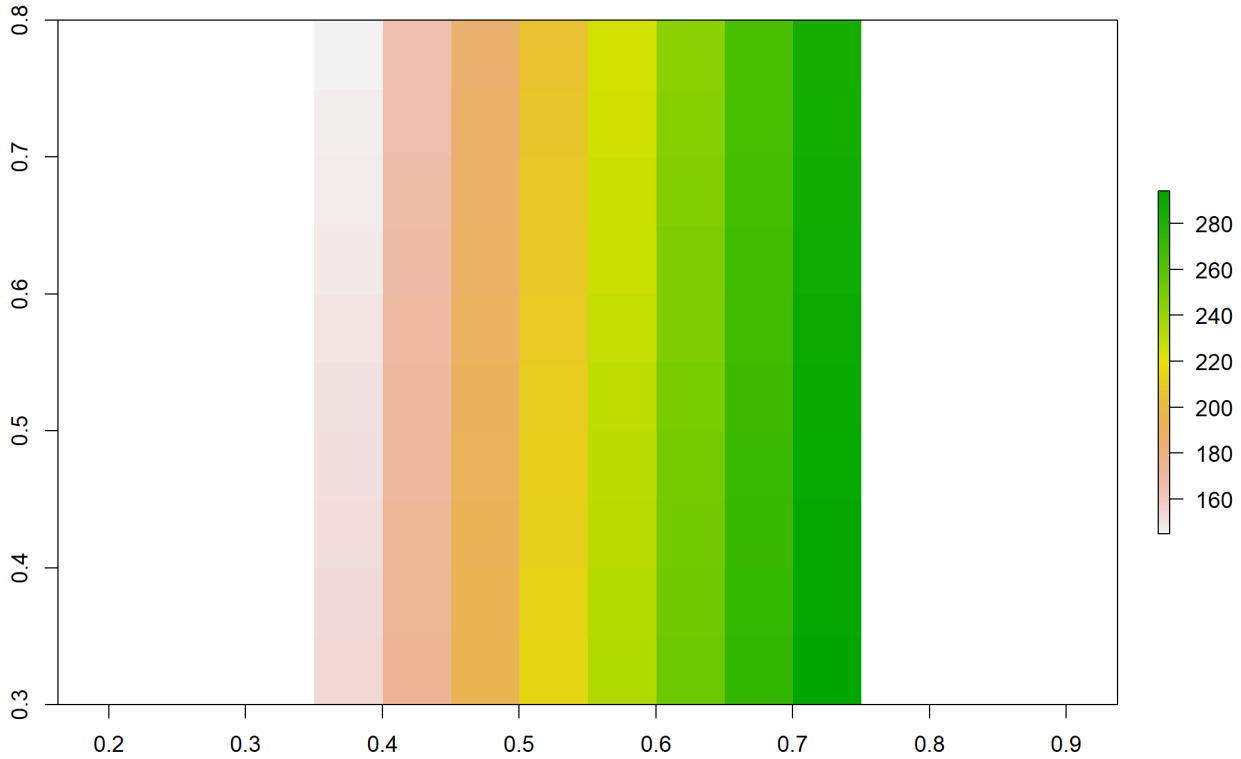
```
extent(szamraszter2_korbevagott)
```

```
class       : Extent
xmin        : 0.35
xmax        : 0.75
ymin        : 0.3
ymax        : 0.8
```

Megváltozott a cellák száma és a kiterjedés, de a felbontás változatlan.

# Raszter körbevágása

```
plot(szamraszter2_korbevagott)
```



# Raszter körbevágása

Néha ugyanolyan méretű raszterre van szükségünk:

```
plot(stack(szamraszter1, szamraszter2_korbevagott))
```

```
Error in h(simpleError(msg, call)): error in evaluating
 the argument 'x' in selecting a method for function
 'plot': different extent
```

Ilyenkor jól jöhet a `crop()`: vágjuk körbe az egyik rasztert a másik alapján!
Persze a tökéletes megoldás a `projectRaster(from, to)`.

# Raszter körbevágása

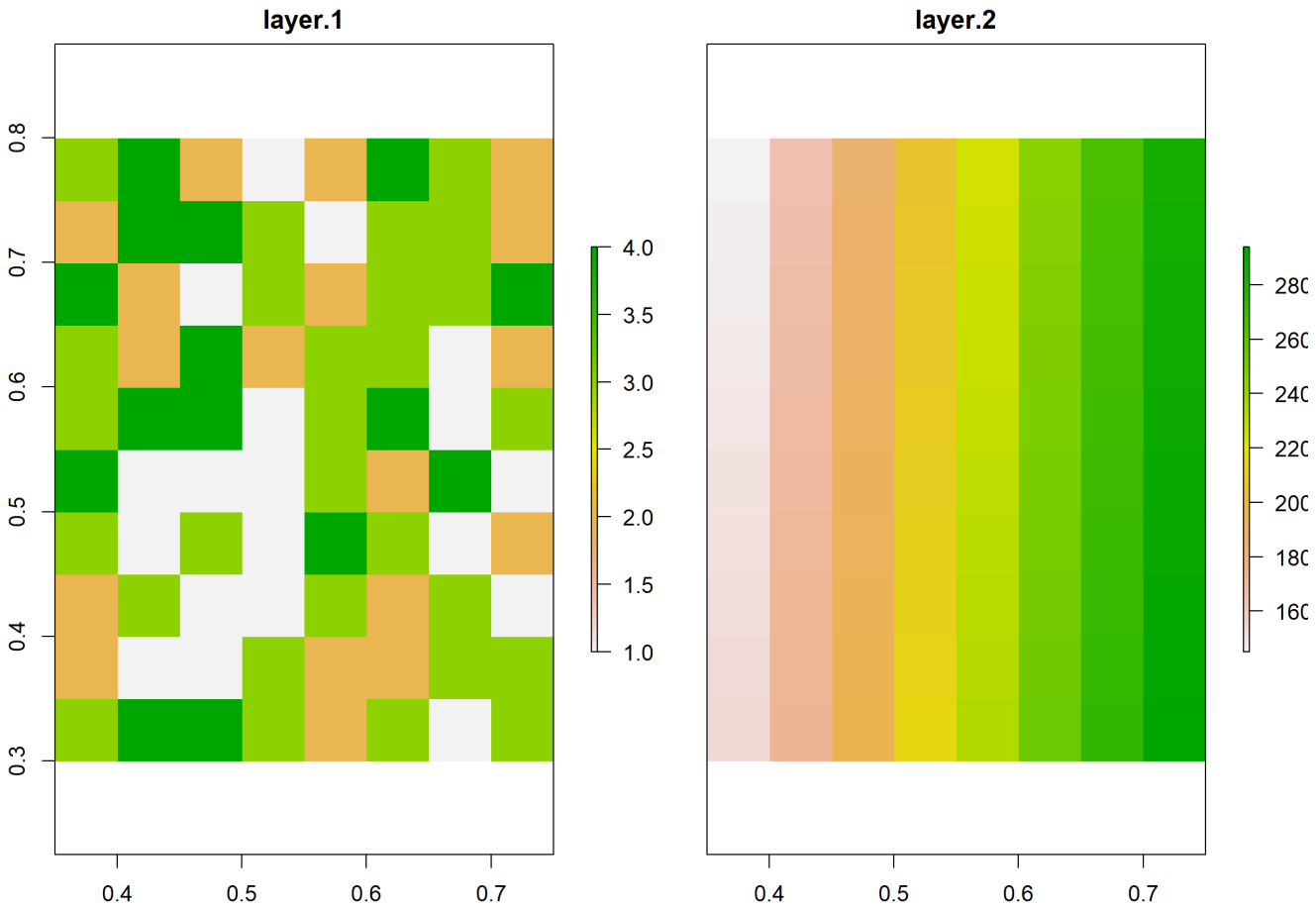
```
szamraszter1_korbevagott <- crop(x = szamraszter1, y =
 szamraszter2_korbevagott)
extent(szamraszter1_korbevagott)
```

```
class       : Extent
xmin        : 0.35
xmax        : 0.75
ymin        : 0.3
ymax        : 0.8
```

Most már ez sem a (0; 0) és (1; 1) pontok közti dobozt tölti ki.

# Raszter körbevágása

```
plot(stack(szamraszter1_korbevagott,
 szamraszter2_korbevagott))
```



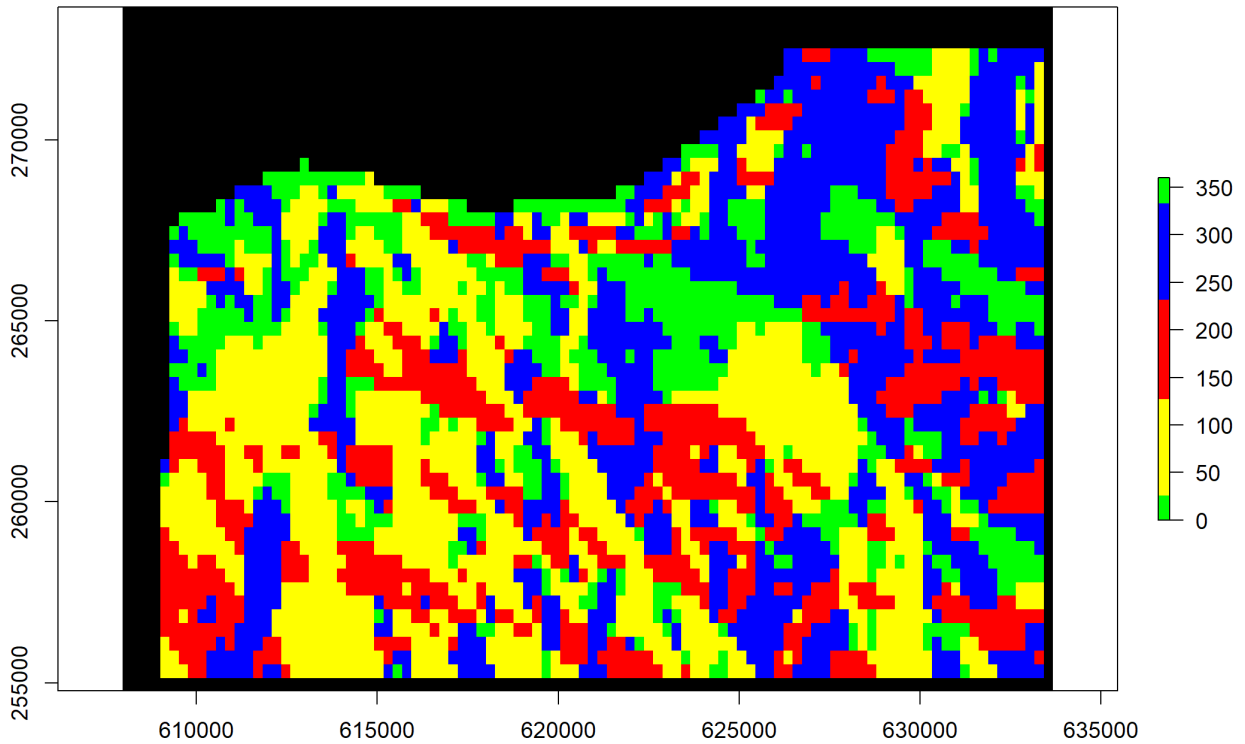
# Raszter körbevágása

Nézzük meg a `trim()` és `crop()` függvényeket valós adatokon!

```
esztergom <- domborzatmodell[1:150, 1:300, drop = FALSE]
esztergom <- aggregate(x = esztergom, fact = 3)
kitettseg <- terrain(x = esztergom, opt = "aspect", unit =
 "degrees")
```

# Raszter körbevágása

```
plot(kitettseg, col = c("green", "yellow", "yellow",
 "red", "red", "blue", "blue", "green"), colNA = "black")
```




A `terrain()` eredménye mindig `NA`-cellákat tartalmaz a széleken.

# Raszter körbevágása

```
ncell(kitettseg)
```

```
[1] 5000
```

```
kitettseg_korbevagott <- trim(kitettseg)
ncell(kitettseg_korbevagott)
```

```
[1] 4370
```

# Raszter körbevágása

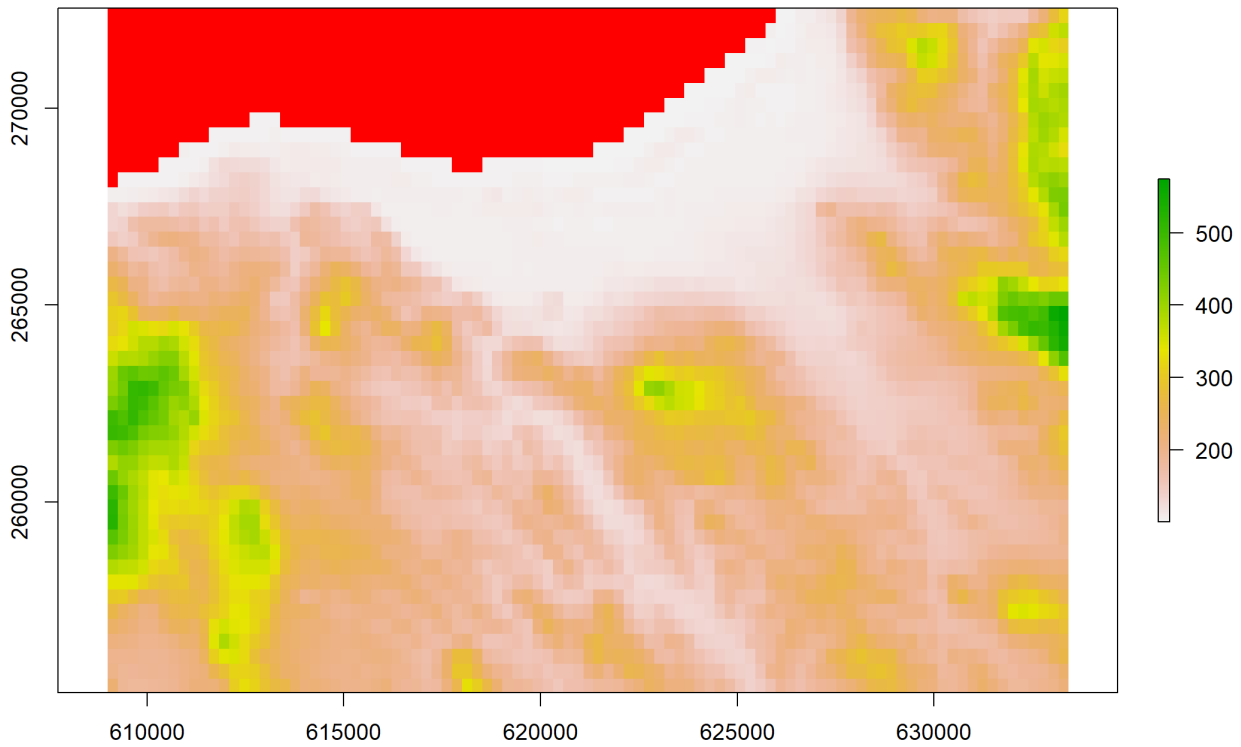
```
esztergom_korbevagott <- crop(x = esztergom, y =
 kitettseg_korbevagott)
ncell(esztergom_korbevagott)
```

```
[1] 4370
```

Most már ennek is kevesebb cellája van.

# Raszter körbevágása

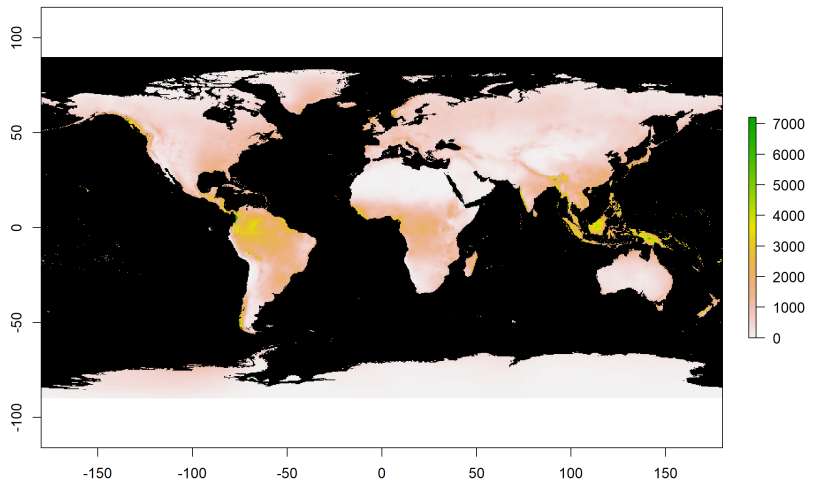
```
plot(esztergom_korbevagott, colNA = "red")
```




És nincsenek a szélén csupa-NA-s sorok/oszlopok.

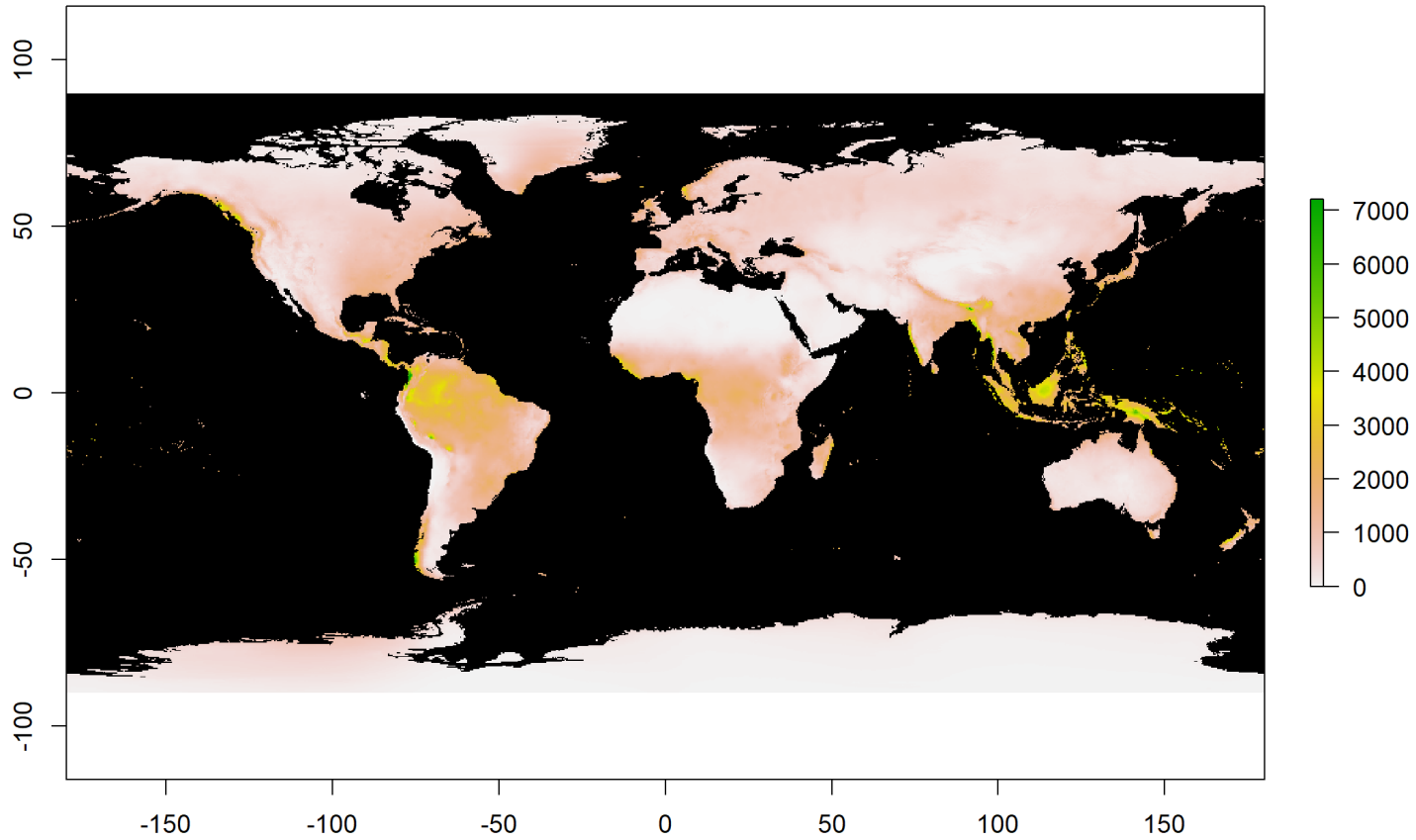
# 1. feladat (házi)

- Jelenítsd meg a “csapadek” nevű rasztert úgy, hogy az ismeretlen értékű cellák fekete színt kapjanak.
- Mi a kiterjedése ennek a raszternek?
- Ugye téged is zavar a sok fölös cella az északi sark közelében? (költői kérdés)
- Készíts csökkentett méretű rasztert úgy, hogy a szélén lévő, csupa ismeretlen értékű cellákból álló sorokat és oszlopokat elhagyod.
- Ellenőrzésképpen kérd le a kiterjedését a csökkentett raszternek is.



# 1. feladat (házi) – megoldás

```
plot(csapadek, colNA = "black")
```



# 1. feladat (házi) – megoldás

```
extent(csapadek)
```

```
class       : Extent
xmin        : -180
xmax        : 180
ymin        : -90
ymax        : 90
```

```
csapadek_korbevagott <- trim(csapadek)
extent(csapadek_korbevagott)
```

```
class       : Extent
xmin        : -180
xmax        : 180
ymin        : -90
ymax        : 83.66667
```

# 2. feladat (órai)

- Vágd körbe a “felszinboritas” nevű rasztert a “felszinhomerseklet” nevű raszter kiterjedése alapján.
- Miért kaptad a hibaüzenetet? Hogyan lehetséges, hogy a kiterjedésük nem lapol át?
- Vetítsd át a felszínborítást a felszínhőmérséklet vetületébe a legközelebbi szomszéd módszerét alkalmazva, “felszinboritas_eov” néven.
- Újfent próbálj meg egy kisebb területű rasztert létrehozni “felszinboritas_korbevagott” néven, immáron az átvetített raszterből kiindulva.
- Jelenítsd meg az eredményt.
- Miben lenne más (több) ennél a vetületmódosítós+körbevágós megoldásnál az, ha a `projectRaster(from, to)` függvényt használnánk?

# 2. feladat (órai) – megoldás

```
felszinboritas_korbevagott <- crop(x = felszinboritas, y =
 felszinhomerseklet)
```

```
Error in .local(x, y, ...): extents do not overlap
```

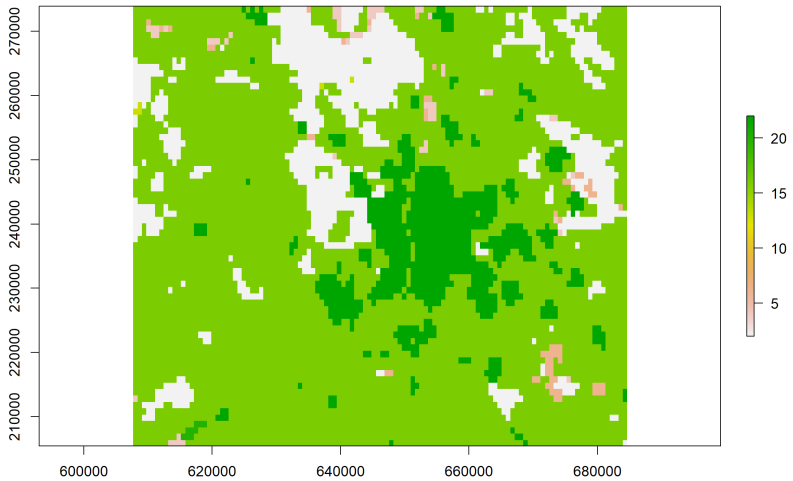
Más a vetületük, ezért a kiterjedések teljesen más nagyságrendű koordinátaértékekkel jellemezhetőek.

```
felszinboritas_eov <- projectRaster(from = felszinboritas,
 crs = projection(felszinhomerseklet), method = "ngb")
```

```
felszinboritas_korbevagott <- crop(x = felszinboritas_eov,
 y = felszinhomerseklet)
```

# 2. feladat (órai) – megoldás

```
plot(felszinboritas_korbevagott)
```




A `projectRaster(from, to)` esetén a felbontást és a dimenziókat (cellaszámot) is igazítottuk volna, nem csak a kiterjedést.

# Maszkolás

## Maszkolás célja

- szeretnénk egy raszter celláit törölni/megváltoztatni
- bizonyos helyeken

## Maszkolás függvényei

`cover(x, y)`

- egyszerű függvény
- egyik raszterbe bemásolja a másikból az értékeket
- "bizonyos hely": ahol az első raszter cellaértéke ismeretlen
- gyakorlatilag hiányzó adatot pótolhatunk így

`mask(x, mask, inverse, maskvalue, updatevalue)`

- sokváltozós, rugalmas függvény
- egyik raszter celláit törli vagy megadott értékre módosítja
- "bizonyos hely": ott (vagy éppen mindenütt másutt), ahol egy másik raszterben valamilyen érték szerepel

# Maszkolás

## `cover(x, y)`

- `x`: az adathiányos raszter
- `y`: ebből raszterből vesszük az értékeket
- eredmény: az `x`-hez hasonló raszter, de az ismeretlen értékű cellákban már ismert értékek szerepelnek (feltéve, hogy `y` értéke ott ismert volt)

# Maszkolás

`mask(x, mask, inverse = FALSE, maskvalue = NA, updatevalue = NA)`

- `x`: módosítandó raszter
- `mask`: az a raszter, ami meghatározza, hogy `x`-et hol módosítsuk (maszk)
- `maskvalue`: a `mask` ilyen értékű cellái határozzák meg, hogy hol módosítsunk ("ott", alapértelmezett: ismerelen értékű cellák)
- `inverse`: inkább ne "ott", hanem mindenütt másutt módosítsunk? (alapértelmezett: ott)
- `updatevalue`: mire módosítsuk `x` cellaértékeit? (alapértelmezett: töröljük)
- tehát a `maskvalue` a `mask` celláira vonatkozik, az `updatevalue` pedig az eredményraszter celláira

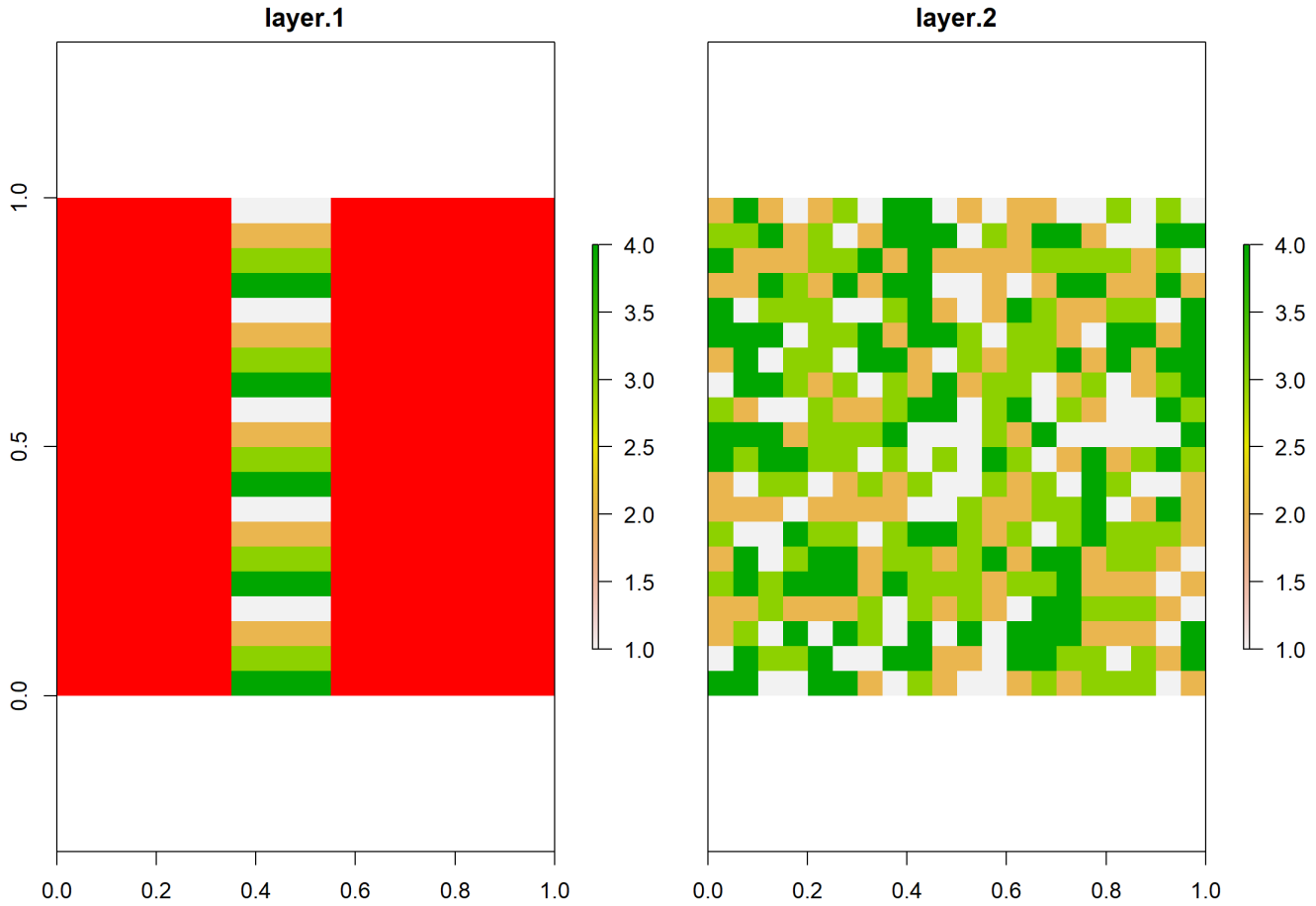
# Maszkolás

Létrehozunk egy újabb rasztert, sok üres oszloppal.

```
szammatrix3 <- matrix(data = 1:4, ncol = 20, nrow = 20)
szamraszter3 <- raster(x = szammatrix3)
szamraszter3[, c(1:7, 12:20)] <- NA
```

# Maszkolás

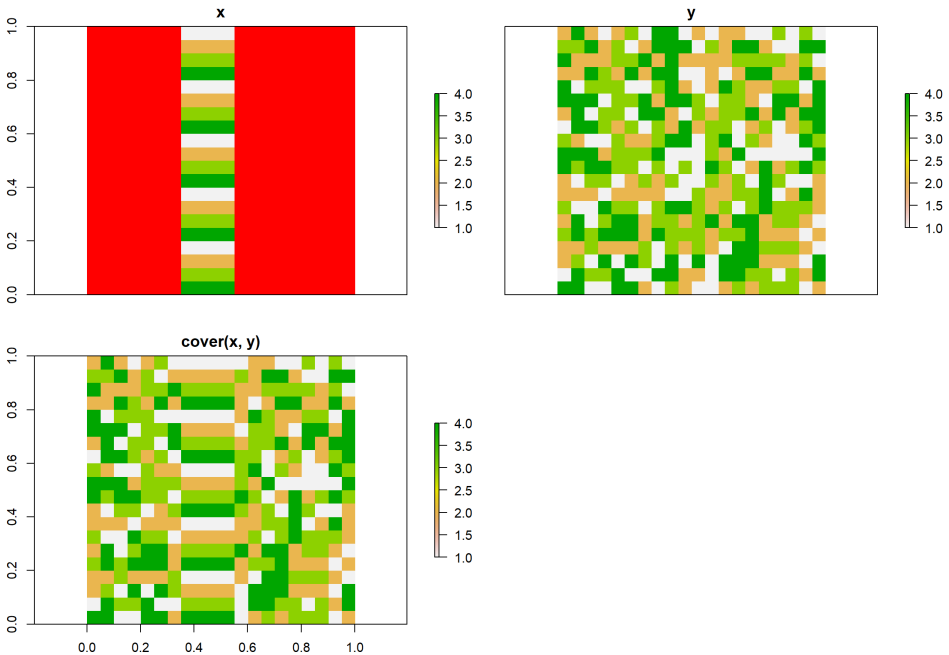
```
plot(stack(szamraszter3, szamraszter1), colNA = "red")
```




Pótoljuk a szamraszter3 üres celláit a szamraszter1-ből vett értékekkel!

# Maszkolás

```
szamraszter3_felulirt <- cover(x = szamraszter3, y =
 szamraszter1)
plot(stack(szamraszter3, szamraszter1,
 szamraszter3_felulirt), colNA = "red", main = c("x", "y",
 "cover(x, y)"))
```



# Maszkolás

Nézzük meg most a `mask()` függvényt különböző paraméterezésekkel!

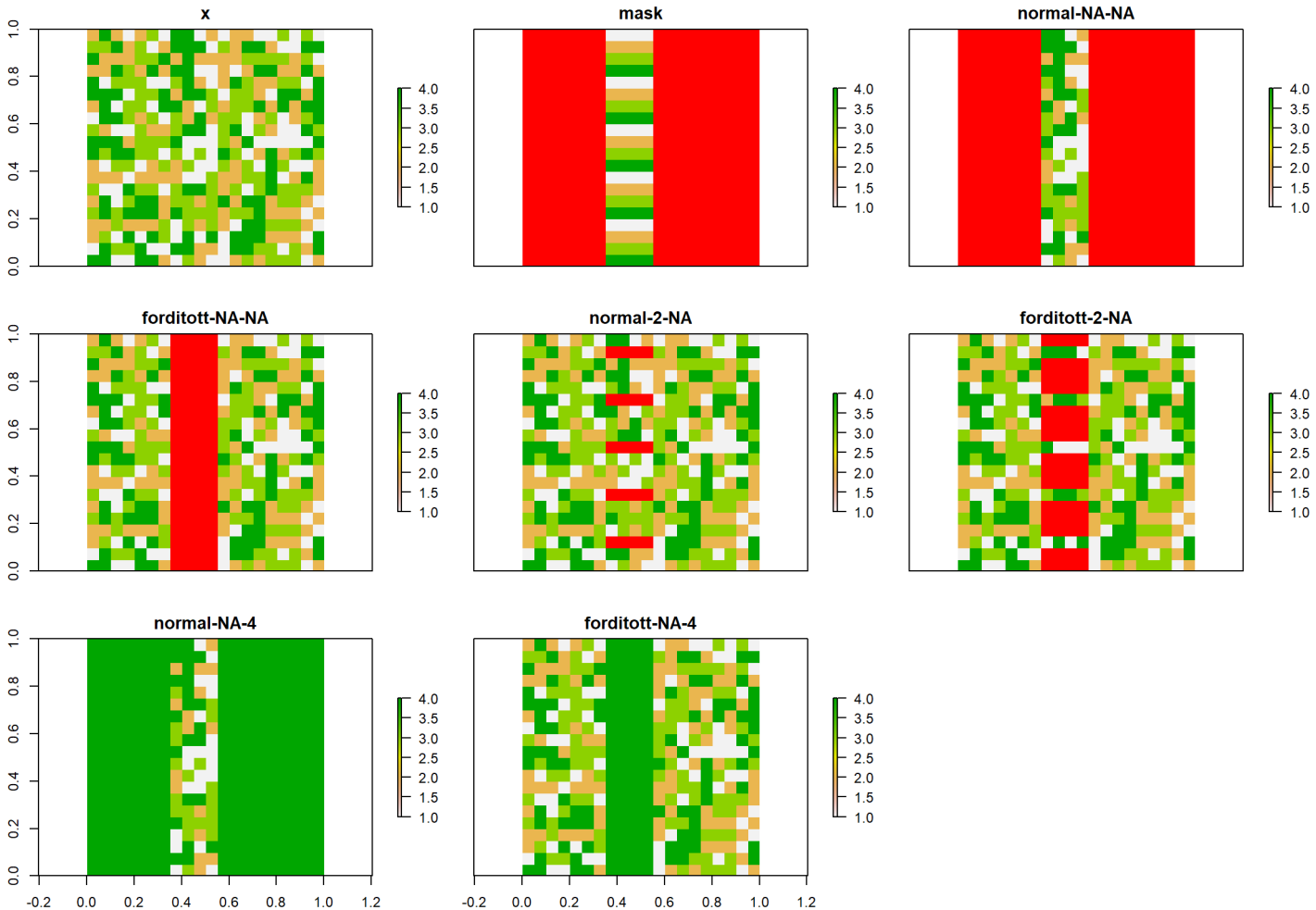
```
maszkolt1 <- mask(x = szamraszter1, mask = szamraszter3,
 inverse = FALSE, maskvalue = NA, updatevalue = NA)
maszkolt2 <- mask(x = szamraszter1, mask = szamraszter3,
 inverse = TRUE, maskvalue = NA, updatevalue = NA)
maszkolt3 <- mask(x = szamraszter1, mask = szamraszter3,
 inverse = FALSE, maskvalue = 2, updatevalue = NA)
maszkolt4 <- mask(x = szamraszter1, mask = szamraszter3,
 inverse = TRUE, maskvalue = 2, updatevalue = NA)
maszkolt5 <- mask(x = szamraszter1, mask = szamraszter3,
 inverse = FALSE, maskvalue = NA, updatevalue = 4)
maszkolt6 <- mask(x = szamraszter1, mask = szamraszter3,
 inverse = TRUE, maskvalue = NA, updatevalue = 4)
```

# Maszkolás

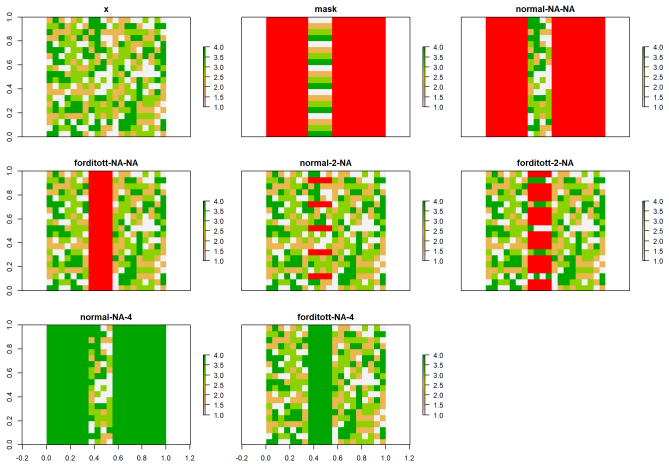
```
plot(stack(szamraszter1, szamraszter3, maszkolt1,
 maszkolt2, maszkolt3, maszkolt4, maszkolt5, maszkolt6),
 colNA = "red", main = c("x", "mask", "normal-NA-NA",
 "forditott-NA-NA", "normal-2-NA", "forditott-2-NA",
 "normal-NA-4", "forditott-NA-4"))
```

# Maszkolás

`inverse - maskvalue - updatevalue`



# Maszkolás




`inverse = FALSE` (alapértelmezett, "normál") maszkolás:

- alapértelmezetten az ismeretlen helyeken (`maskvalue = NA`) ismeretlenre módosít (`updatevalue = NA`)
- de a maszkoló érték lehet más is (`maskvalue = 2`)
- vagy a maszkolt helyen módosíthatunk másra is (`updatevalue = 4`)

Az `inverse = TRUE` esetben a `!is.na(mask[])` (`maskvalue = NA`) vagy a `!is.na(mask[]) & mask[] != 2` (`maskvalue = 2`) helyeken módosítunk.

# Maszkolás

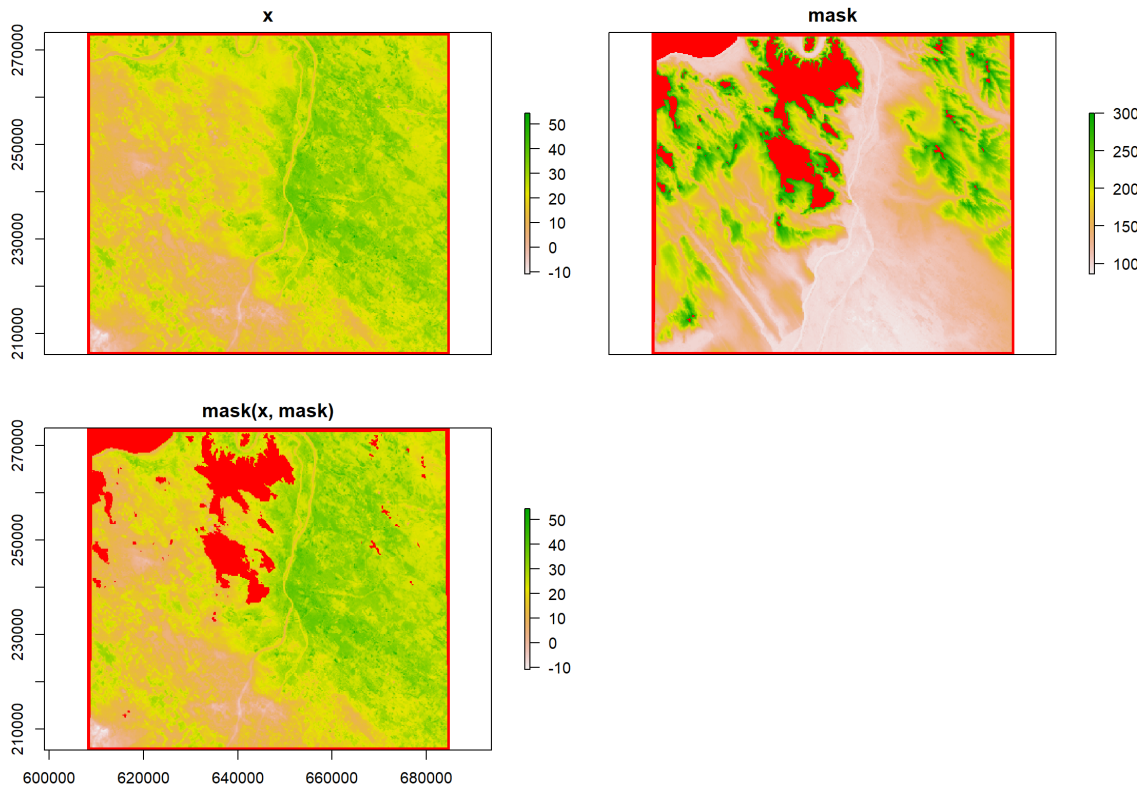
Valós adatokkal is próbáljuk ki!

```
dem_maszk <- domborzatmodell
dem_maszk[dem_maszk[] > 300] <- NA
homerseklet_maszkolt <- mask(x = felszinhomerseklet, mask
 = dem_maszk)
```

# Maszkolás

A felszínhőmérséklet ott, ahol a magasság 300 méternél nem nagyobb:

```
plot(stack(felszinhomerseklet, dem_maszk,
 homerseklet_maszkolt), main = c("x", "mask", "mask(x,
 mask)"), colNA = "red")
```



# 3. feladat (házi)

- Hozz létre másolatot a felszínhőmérsékleti raszterből, és e másolatban a 20 °C-nál melegebb cellákat töröld.
- A domborzatmodellt felhasználva készítsd el a következő eredményekkel járó maszkolásokat, majd jelenítsd meg őket a bemeneti raszterekkel egyetemben, kék színnel jelölve az ismeretlen cellákat.
  - A: magassági érték csak a meleg helyeken
  - B: hőmérséklet a hideg helyeken, magasság a meleg helyeken
  - C: az eredeti magassági érték a hideg helyeken, –100 m a meleg helyeken
  - D: az eredeti magasság mindenütt, kivéve ott, ahol 200 m volt a magasság, mert ott legyen inkább 0

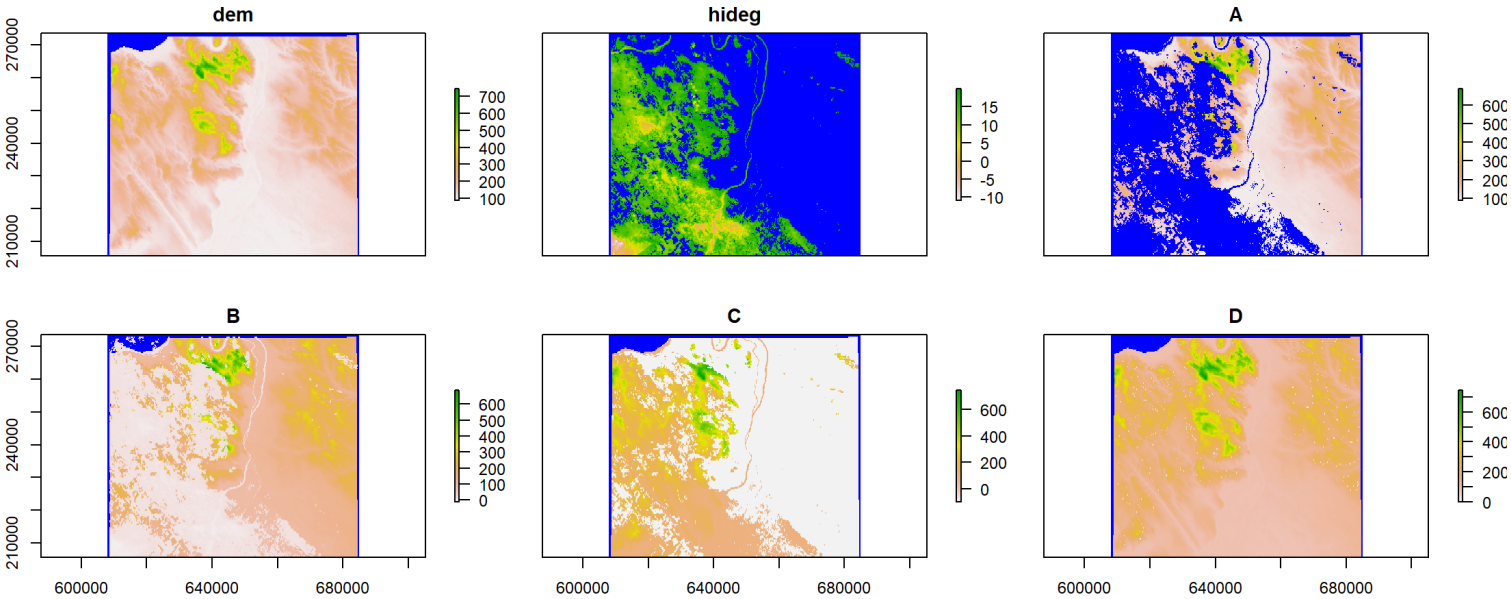
(A D pontra van egyszerűbb megoldás is a maszkolásnál (`[] <-`), de most a bonyolultabb megoldásra vagyok kíváncsi.)

# 3. feladat (házi) – megoldás

```
csak_hidegek <- felszinhomerseklet
csak_hidegek[csak_hidegek[] > 20] <- NA
eredmeny_a <- mask(x = domborzatmodell, mask =
 csak_hidegek, inverse = TRUE)
eredmeny_b <- cover(x = csak_hidegek, y = domborzatmodell)
eredmeny_c <- mask(x = domborzatmodell, mask =
 csak_hidegek, updatevalue = -100)
eredmeny_d <- mask(x = domborzatmodell, mask =
 domborzatmodell, maskvalue = 200, updatevalue = 0)
```

```
plot(stack(domborzatmodell, csak_hidegek, eredmeny_a,
 eredmeny_b, eredmeny_c, eredmeny_d), colNA = "blue", main
 = c("dem", "hideg", "A", "B", "C", "D"))
```

# 3. feladat (házi) – megoldás



Section 2

# Raszter-raszter műveletek: kategóriák kezelése

# Kategóriák felbontása

## Kategóriák felbontása

- adott egy raszter, amelynek a cellái diszkrét értékeket vesznek fel
- pl. felszínborítási kategóriák
- szeretnénk annyi különálló raszterréteget kapni, ahány kategóriánk volt
- minden réteg mutassa, hogy az adott cella az adott kategóriába esett-e (logikai raszter)
- emlékeztető: a "logikai raszter" valójában 0-t (`FALSE` helyett) és 1-et (`TRUE` helyett) tartalmaz

## `layerize(x, classes)`

- `x`: szétbontandó raszter
- `classes`: a szétbontandó kategóriák (opcionális)
- alapértelmezett esetben az összes kategóriát kibontja külön rétegbe
- eredmény: egy `RasterBrick` annyi réteggel, ahány kategóriát kibontottunk

# Kategóriák felbontása

```
nlayers(szamraszter1)
```

```
[1] 1
```

```
unique(szamraszter1[])
```

```
[1] 2 4 1 3
```

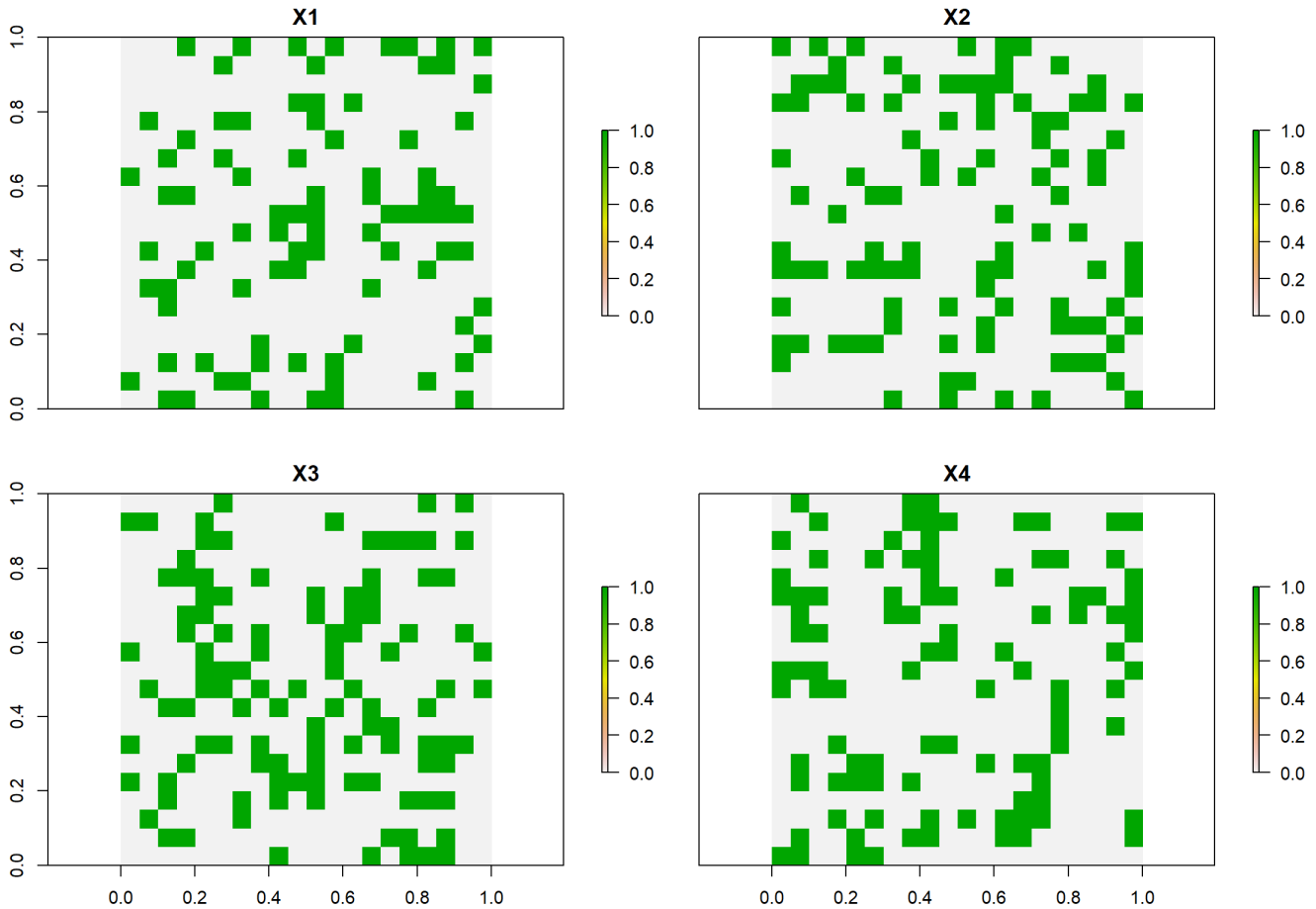
Az összes kategória felbontása:

```
kategoriakra_bontva <- layerize(x = szamraszter1)
nlayers(kategoriakra_bontva)
```

```
[1] 4
```

# Kategóriák felbontása

```
plot(kategoriakra_bontva)
```



# Kategóriák felbontása

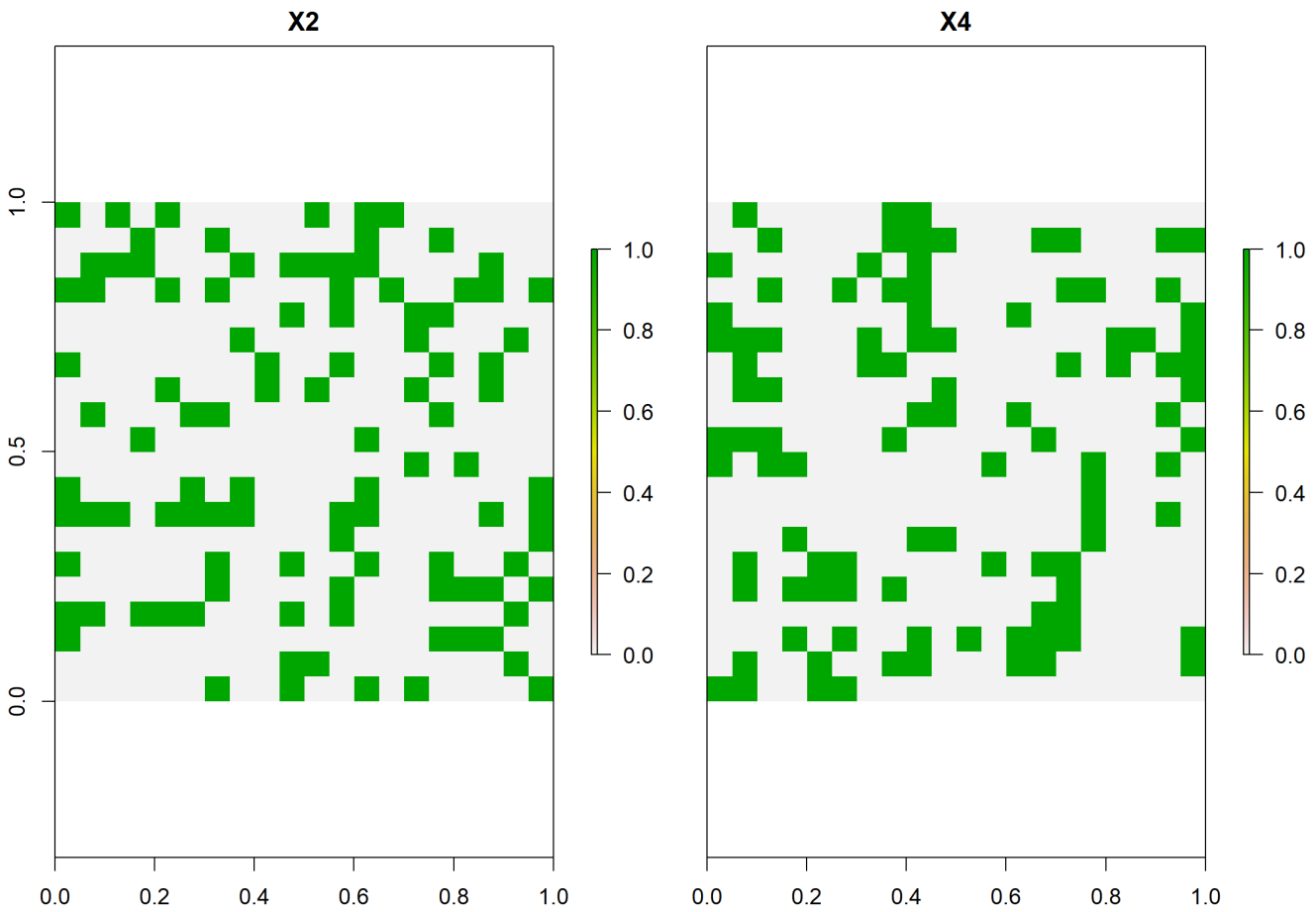
Csak néhány, kiválasztott kategória felbontása:

```
csak_fontos_kategoriak <- layerize(x = szamraszter1,
 classes = c(2, 4))
nlayers(csak_fontos_kategoriak)
```

```
[1] 2
```

# Kategóriák felbontása

```
plot(csak_fontos_kategoriak)
```



# Statisztikák kategóriánként

## Statisztikák kategóriánként (zonal statistics)

- adott két raszter
- az egyik értékeket tartalmaz, amiket csoportokba rendezünk
- e csoportokra statisztikát akarunk számolni (pl. átlagot)
- a másik kategóriákat (diszkrét számértékeket, "zónákat") tartalmaz
- ez meghatározza, hogy miképpen csoportosítsuk az első raszter celláit

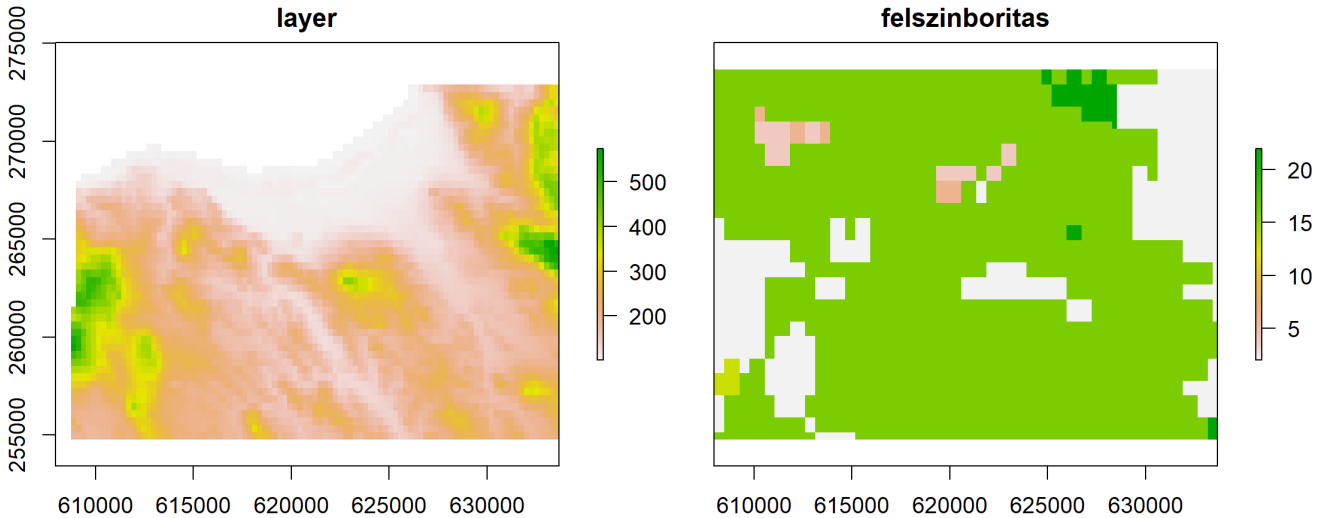
## `zonal(x, z, fun = "mean", na.rm = TRUE)`

- `x`: az értékeket tartalmazó raszter
- `z`: a kategóriákat (zónákat) tartalmazó raszter
- `fun`: a statisztikai függvény vagy a neve (alapértelmezett: átlagszámítás)
- `na.rm`: figyelmen kívül hagyja-e a hiányzó értékeket (alapértelmezett: igen)
- eredmény: egy számmátrix két oszloppal, elsőben a kategóriákkal, másodikban a kiszámolt statisztikákkal

# Statisztikák kategóriánként

```
esztergomi_felszinboritas <- projectRaster(from =
 felszinboritas, to = esztergom, method = "ngb")
```

```
plot(stack(esztergom, esztergomi_felszinboritas))
```



# Statisztikák kategóriánként

Minden kategóriára képezzük az átlagot:

```
atlagmagassag_kategoriankent <- zonal(x = esztergom, z =
 esztergomi_felszinboritas, fun = mean, na.rm = TRUE)
class(atlagmagassag_kategoriankent)
```

```
[1] "matrix" "array"
```

```
print(atlagmagassag_kategoriankent)
```

```
     zone    value
[1,]    2 296.4123
[2,]    4 103.3689
[3,]    6 106.3556
[4,]   13 346.0000
[5,]   16 179.9117
[6,]   22 149.1292
```

# Statisztikák kategóriánként

Ugyanígy a szórást (**s**tandard **d**eviation) is kiszámolhatjuk az `sd()` függvénnyel:

```
zonal(x = esztergom, z = esztergomi_felszinboritas, fun =
 sd, na.rm = TRUE)
```

```
     zone      value
[1,]    2 89.231445
[2,]    4  2.323915
[3,]    6  1.667830
[4,]   13 61.384059
[5,]   16 54.669161
[6,]   22 48.084472
```

# 4. feladat (házi)

- Az ábrázolás megkönnyítése végett töröld a felszínborítási raszter beépített színpalettáját az alábbi paranccsal:

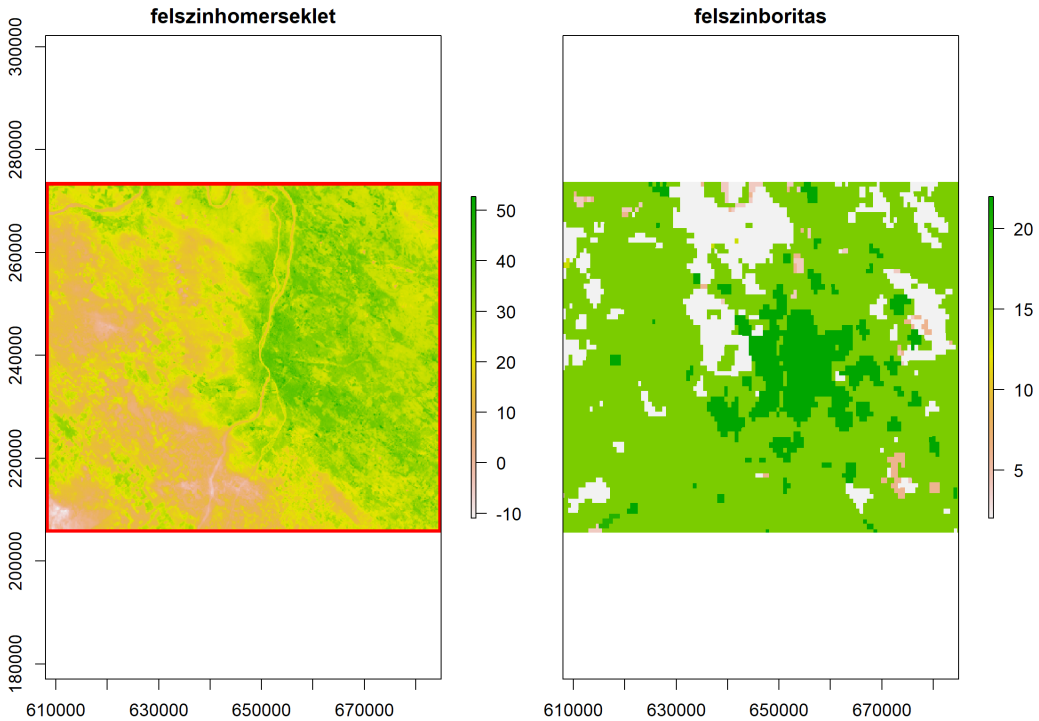
```
colortable(felszinboritas) <- logical()
```

- Milyen felszínborítási kategóriák léteznek (milyen diszkrét értékeket vesz fel a felszínborítási raszter)?
- Képezzél háromrétegű "logikai" rasztert, amely a 22-es, 16-os és 6-os felszínborítási kategóriákba esést tartalmazza.
- Jelenítsd meg az első két rétegét.
- Készíts a felszínborításból olyan másolatot, amit átvetítesz a legközelebbi szomszéd módszerét alkalmazva a felszínhőmérsékleti raszter rácshálójába.

...

# 4. feladat (házi)

- Jelenítsd meg egymás mellett a felszínhőmérsékletet és az átvetített rasztert, az ismeretlen értékű cellákat piros színnel jelöld.
- Készíts el a következő két területi statisztikát:
  - hőmérséklet minimuma minden felszínborítási kategóriára, az ismeretlen értékeket is figyelembe véve,
  - hőmérséklet maximuma minden felszínborítási kategóriára, az ismeretlen értékeket elhagyva

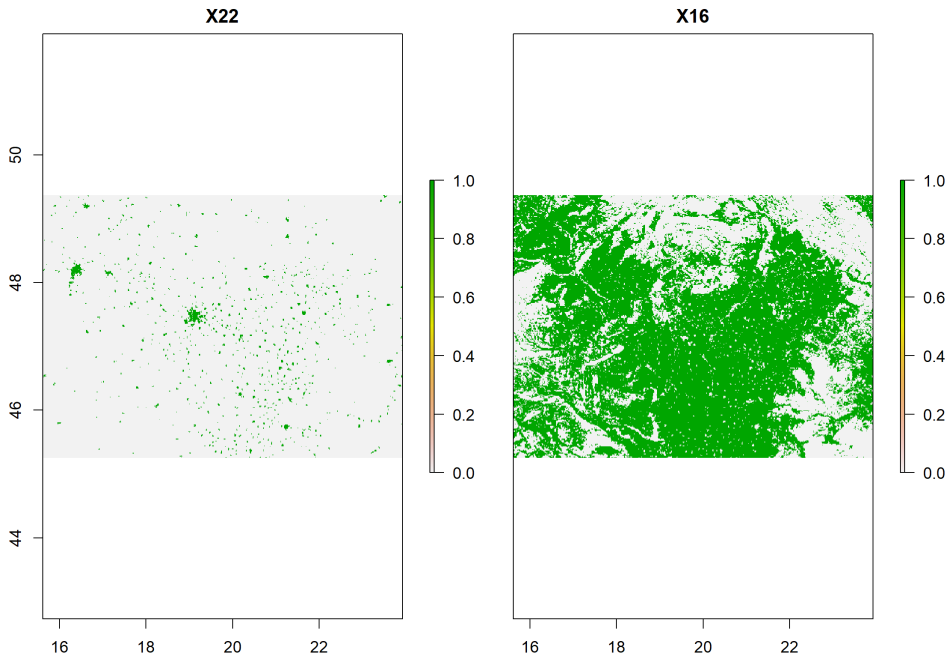


# 4. feladat (házi) – megoldás

```
colortable(felszinboritas) <- logical()
unique(felszinboritas[])
```

```
 [1] 16  2  6  4 22 13 20 17 14 15
```

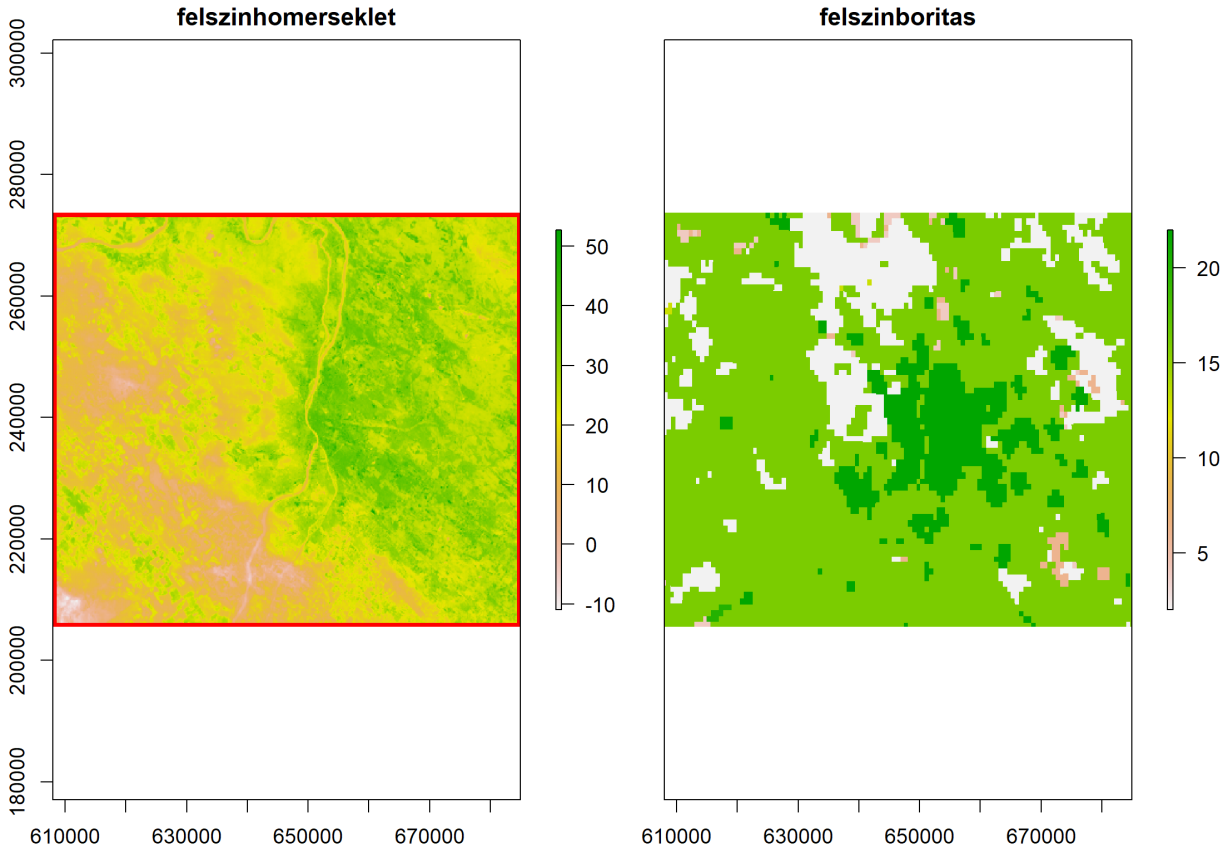
```
felszinboritas_kategoriakra_bontva <- layerize(x =
 felszinboritas, classes = c(22, 16, 6))
plot(felszinboritas_kategoriakra_bontva[[1:2]])
```



# 4. feladat (házi) – megoldás

```
felszinboritas_kivagat <- projectRaster(from =
 felszinboritas, to = felszinhomerseklet, method = "ngb")
```

```
plot(stack(felszinhomerseklet, felszinboritas_kivagat),
 colNA = "red")
```



# 4. feladat (házi) – megoldás

```
zonal(x = felszinhomerseklet, z = felszinboritas_kivagat,
 fun = min, na.rm = FALSE)
```

```
     zone      value
[1,]    2         NA
[2,]    4         NA
[3,]    6         NA
[4,]   13         NA
[5,]   16         NA
[6,]   20 0.5892578
[7,]   22         NA
```

# 4. feladat (házi) – megoldás

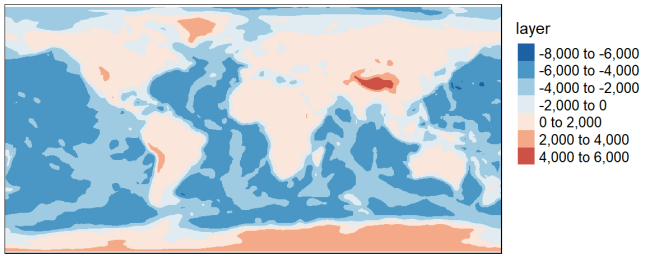
```
zonal(x = felszinhomerseklet, z = felszinboritas_kivagat,
 fun = max, na.rm = TRUE)
```

```
     zone     value
[1,]    2 41.091911
[2,]    4 41.625145
[3,]    6 40.684959
[4,]   13 22.695215
[5,]   16 54.444633
[6,]   20  9.381495
[7,]   22 52.744499
```

Section 3

# Raszter-raszter műveletek: mozgó ablak

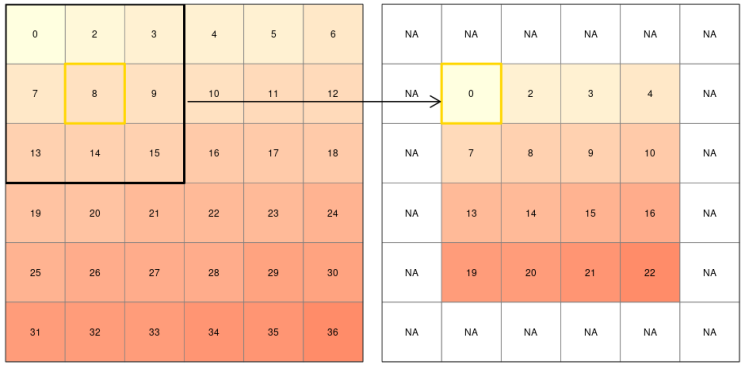
# Mozgó ablak




## Mozgó ablak (moving window, focal statistics)

- adott egy számértékeket tartalmazó raszterünk
- minden cellájába új értéket akarunk számolni
- amely során figyelembe vesszük a környező (nem feltétlenül közvetlenül szomszédos) cellák értékeit is
- megadott súlyokkal (pl. a távolsággal csökkenő módon)
- gyakorlatilag van egy súlyokkal definiált ablakunk, és azt tologatjuk végig a raszteren
- általában elkenjük, simítjuk így az értékeket (de nem csökken a felbontás!)

# Mozgó ablak




## `focal(x, w, fun, na.rm = FALSE)`

- `x`: egyrétegű raszter
- `w`: súlymátrix (ablak)
- `fun`: a függvény, ami az adott súlyozással az ablakba eső cellaértékeket aggregálja (alapértelmezett: összeg)
- `na.rm`: figyelmen kívül hagyja-e a hiányzó értékeket (alapértelmezett: nem)
- eredmény: egy ugyanolyan raszter, mint `x`, de más értékekkel

# Mozgó ablak

## Súlymátrix

- téglalap alakú számmátrix
- a sorainak, illetve oszlopainak száma legyen páratlan (hogy legyen középső elem)
- a mátrix elemei a súlyok, ahol a mátrix középső eleme a központi rasztercella súlya
- a súlyok lehetnek 0-k is (értsd: adott cellát ne vegyük figyelembe)
- a súlyok lehetnek negatívak is
- a súlyok összege jellemzően 1, vagy `nrow`×`ncol`

# Mozgó ablak

## Tipikus súlymátrixok

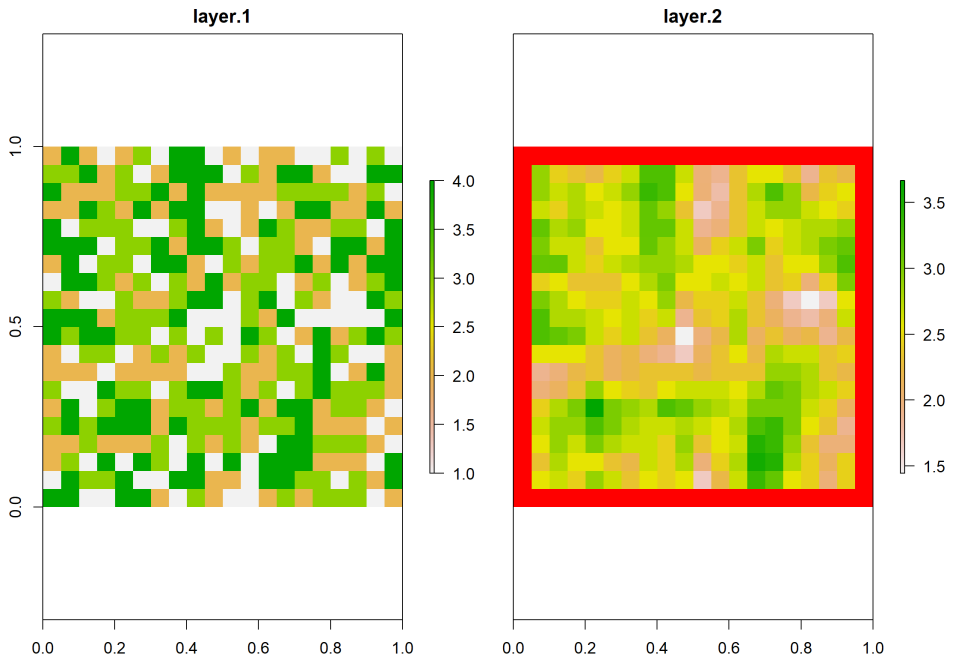
- 3×3-as mátrix, csupa 1-essel (átlaghoz) vagy 1/9-eddel (összeghez)
- 3×3-as mátrix, körben 1-essel vagy 1/8-dal, középen 0
- 3×3-as (vagy nagyobb) mátrix, ami szálkeresztet formáz
- tetszőleges méretű mátrix, Gauss-görbe szerű súlyokkal (ilyet létrehozhatunk a `focalWeight()` függvénnyel - nem mutatom be)
- tetszőleges méretű kör vagy téglalap alakú mátrix (ilyet létrehozhatunk a `focalWeight()` függvénnyel)
- 3×3-as Laplace-mátrix: szálkereszt elrendezés, a középső elem negatív súllyal

# Mozgó ablak

```
ablak_kicsi <- matrix(data = 1 / 9, ncol = 3, nrow = 3)
szamraszter1_elkent <- focal(x = szamraszter1, w =
 ablak_kicsi)
```

# Mozgó ablak

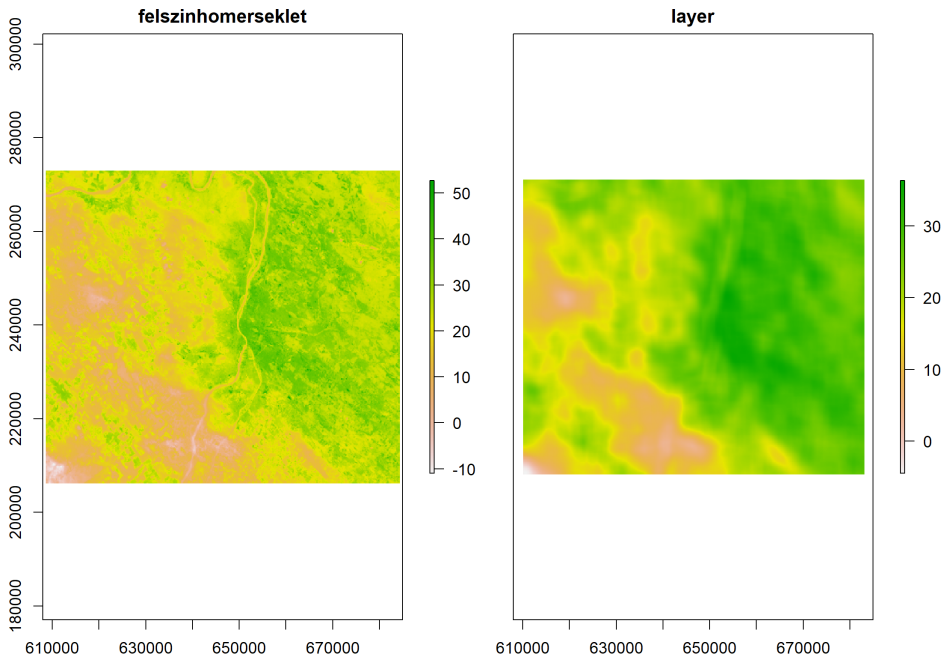
```
plot(stack(szamraszter1, szamraszter1_elkent), colNA =
 "red")
```




A szélső cellák ismeretlenek, hiszen az ablak kilóg a raszter területéről. Ezt orvosolhatjuk a `focal()` függvény `pad` és `padValue` paramétereivel (nem mutatom be).

# Mozgó ablak

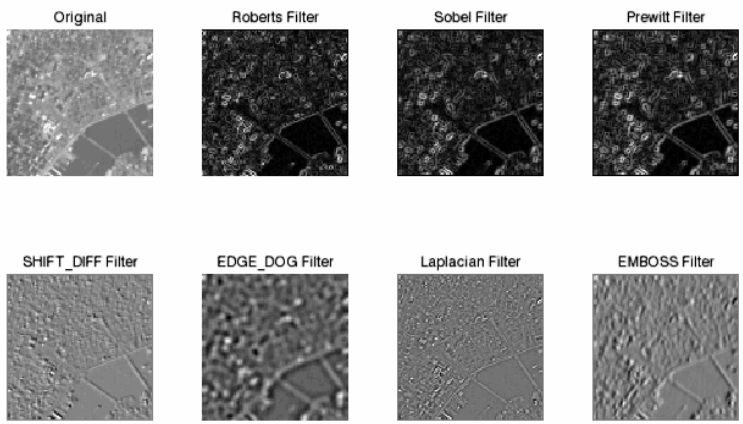
```
ablak_nagy <- matrix(data = 1 / (33 * 33), ncol = 33, nrow
 = 33)
felszinhomerseklet_elkent <- focal(x = felszinhomerseklet,
 w = ablak_nagy)
plot(stack(felszinhomerseklet, felszinhomerseklet_elkent))
```



# Mozgó ablak

Laplace-féle súlymátrix (Laplacian filter):

```
ablak_Laplace <- matrix(data = c(0, 1, 0, 1, -4, 1, 0, 1,
 0), ncol = 3, nrow = 3)
```

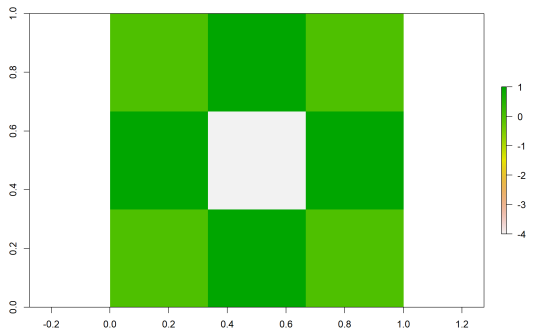


# Mozgó ablak

```
print(ablak_Laplace)
```

```
     [,1] [,2] [,3]
[1,]    0    1    0
[2,]    1   -4    1
[3,]    0    1    0
```

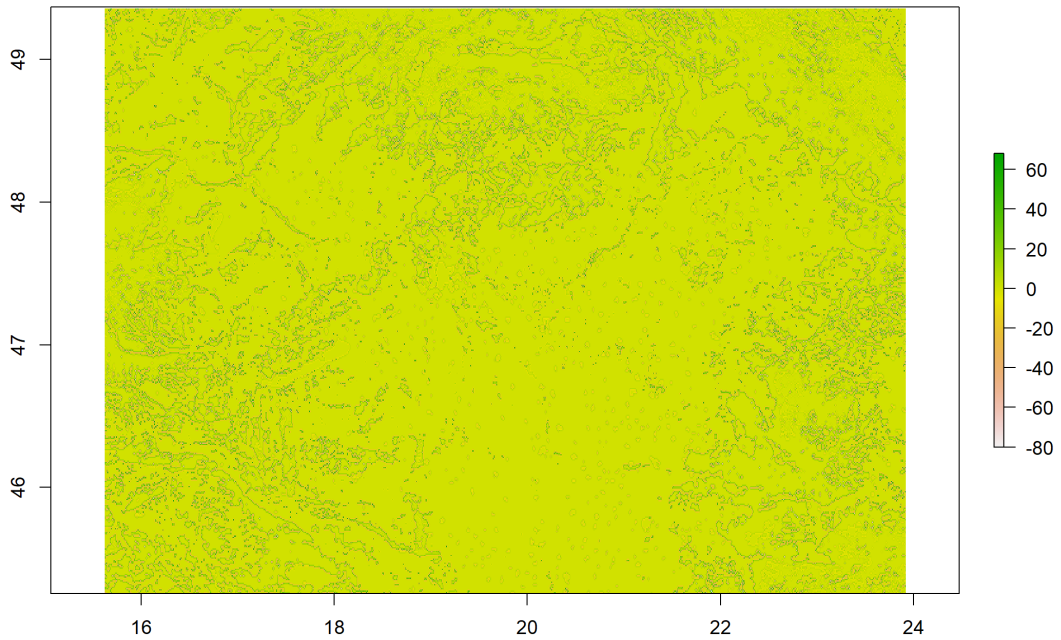
```
plot(raster(ablak_Laplace))
```



# Mozgó ablak

Kiemeli a különbségeket (elsősorban éles határok esetén működik):

```
felszinboritas_kiemelt <- focal(x = felszinboritas, w =
 ablak_Laplace)
plot(felszinboritas_kiemelt)
```



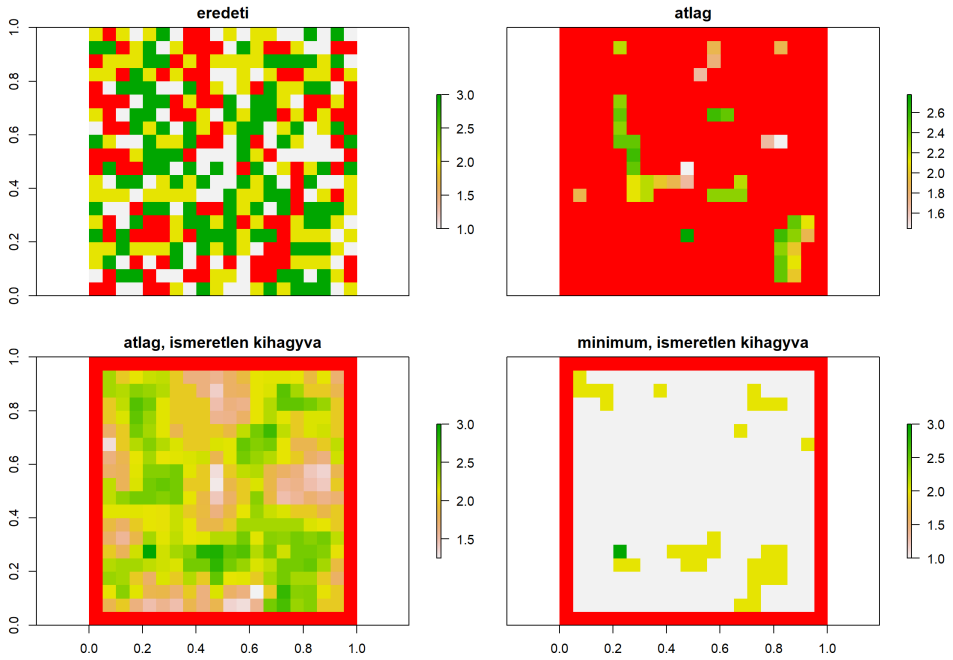
# Mozgó ablak

Lecseréljük a 4-es értékű cellákat ismeretlenre:

```
szamraszter1[szamraszter1[] == 4] <- NA
ablak_kicsi_egyesek <- matrix(data = 1, ncol = 3, nrow = 3)
atlag1 <- focal(x = szamraszter1, w = ablak_kicsi_egyesek,
 fun = mean, na.rm = FALSE)
atlag2 <- focal(x = szamraszter1, w = ablak_kicsi_egyesek,
 fun = mean, na.rm = TRUE)
minimum <- focal(x = szamraszter1, w =
 ablak_kicsi_egyesek, fun = min, na.rm = TRUE)
```

# Mozgó ablak

```
plot(stack(szamraszter1, atlag1, atlag2, minimum), main =
 c("eredeti", "atlag", "atlag, ismeretlen kihagyva",
 "minimum, ismeretlen kihagyva"), colNA = "red")
```




Ha nem hagyjuk el az ismeretlen értékeket (`na.rm = FALSE`), akkor – mivel szinte minden ablakba került `NA` – majdnem az egész raszter ismeretlen lesz.

# Mozgó ablak

A következő feladathoz töltsünk be egy, a `raster` csomaggal érkező mintarasztert!

```
logo <- stack(system.file("external/rlogo.grd", package =
 "raster"))
plotRGB(logo)
```

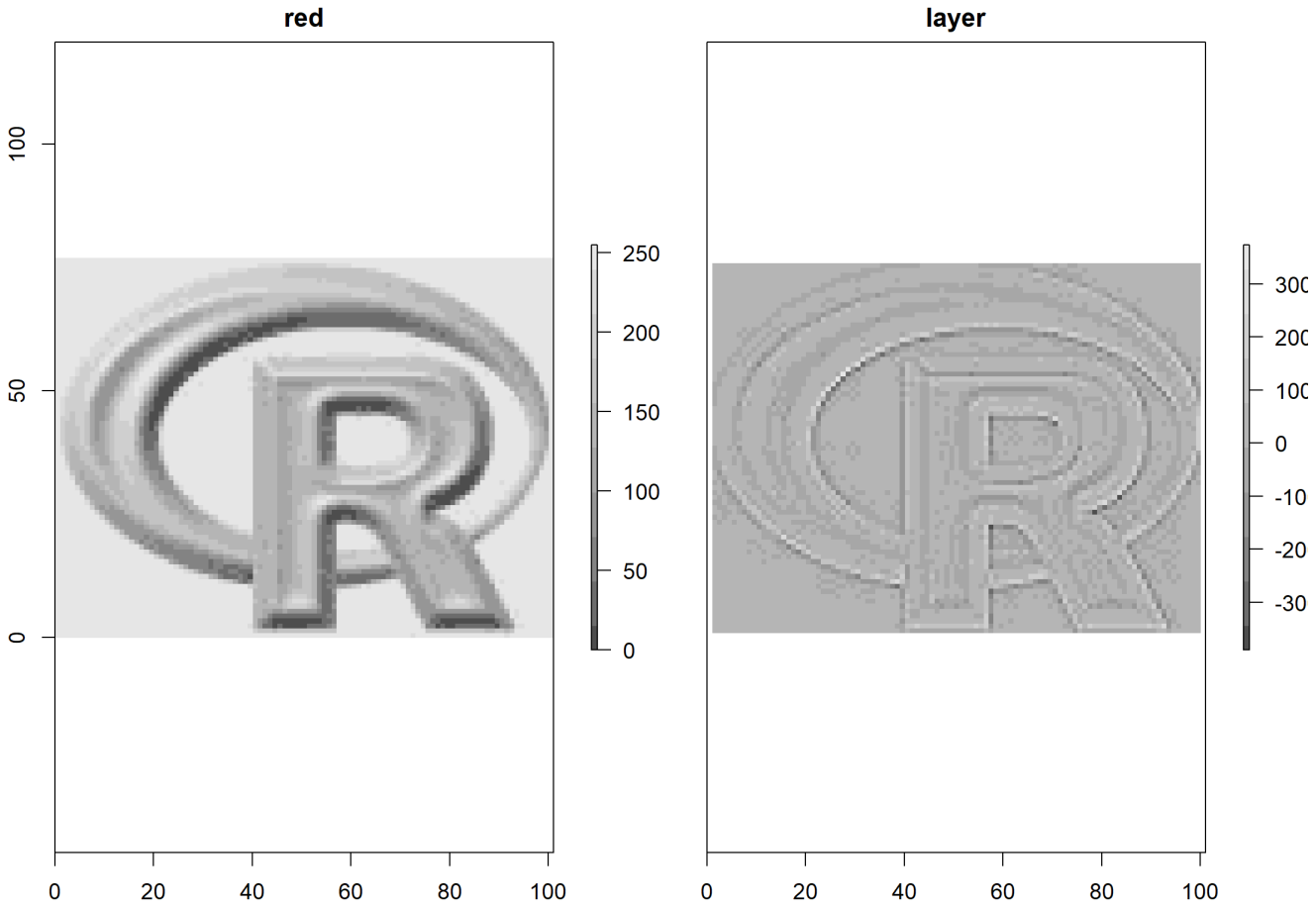
```
logo <- logo$red
```

# 5. feladat (órai)

- Mozgass végig egy Laplace-féle súlymátrixot a “logo” nevű raszter minden celláján, és a bemeneti rasztert, valamint a mozgóablak-elemzés eredményét jelenítsd meg egy ábrán, a szürke színsákla (`gray.colors()`) 10 színét alkalmazva.
- Mozgó ablakban számold ki a “szamraszter3” nevű raszter minden cellájára a 3×3-as ablakba eső cellák maximumát úgy, hogy az ismeretlen értékeket elhagyod. Használd bátran a korábban létrehozott, megfelelő mátrixunkat!
- Jelenítsd meg az eredményt és a bemeneti mátrixot úgy, hogy az ismeretlen értékű cellákat pirosra színezed.
- Készíts egy hosszú súlymátrixot, ami 7 oszlopot és 45 sort tartalmaz, és a súlyok összege 1.
- Készíts evvel a mátrixszal egy elkent domborzatmodellt, és az eredeti domborzatmodellel együtt jelenítsd meg az eredményt.

# 5. feladat (órai) – megoldás

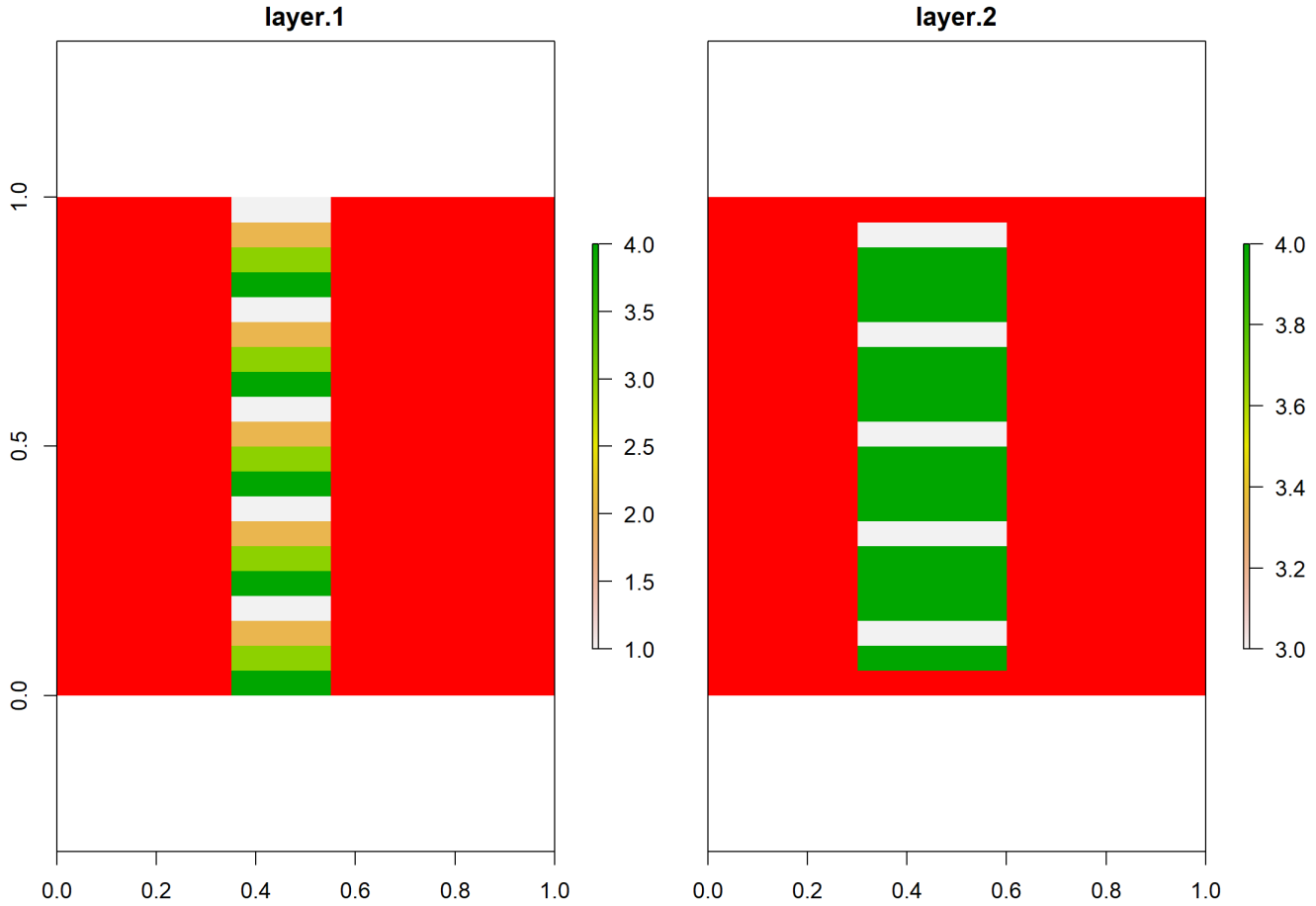
```
logo_kiemelt <- focal(x = logo, w = ablak_Laplace)
plot(stack(logo, logo_kiemelt), col = gray.colors(10))
```



# 5. feladat (órai) – megoldás

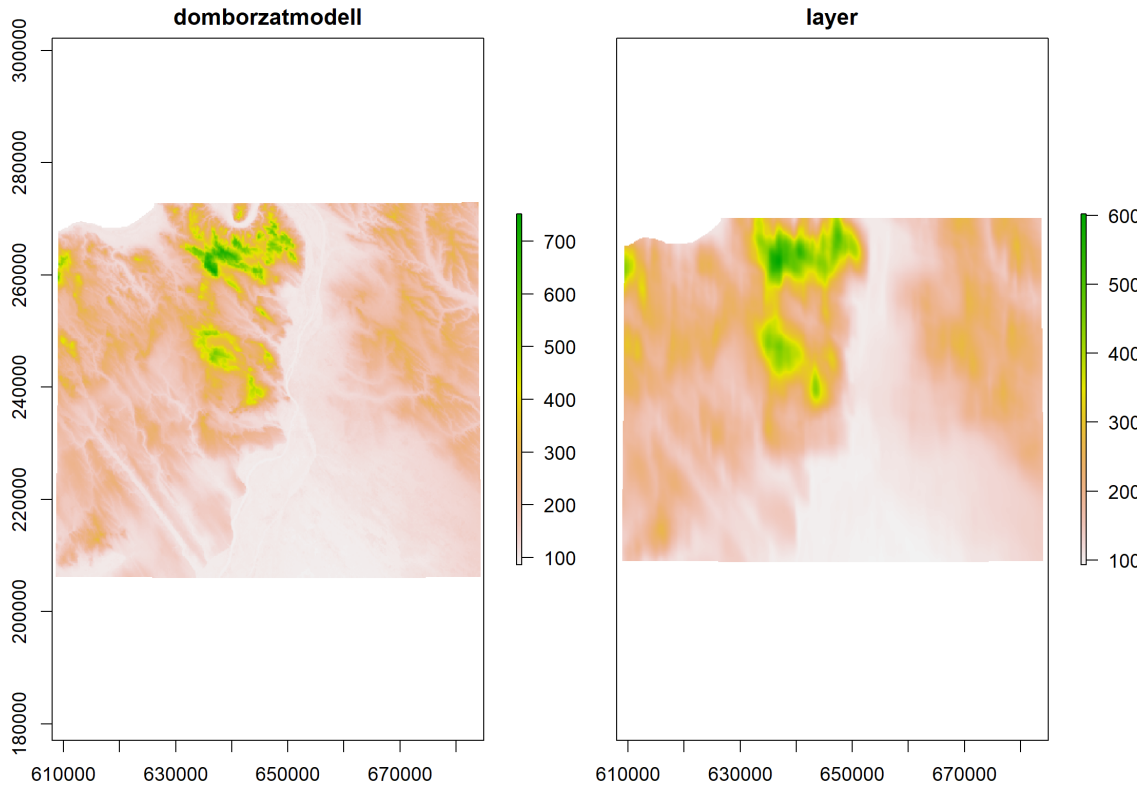
```
maximum <- focal(x = szamraszter3, w =
 ablak_kicsi_egyesek, fun = max, na.rm = TRUE)
```

```
plot(stack(szamraszter3, maximum), colNA = "red")
```



# 5. feladat (órai) – megoldás

```
ablak_hosszu <- matrix(data = 1 / (7 * 45), ncol = 7, nrow
 = 45)
domborzatmodell_elkent <- focal(x = domborzatmodell, w =
 ablak_hosszu)
plot(stack(domborzatmodell, domborzatmodell_elkent))
```



Section 4

# Raszter-vektor műveletek: vágás és maszkolás

# Vágás és maszkolás

## Vágás és maszkolás vektorral

- raszter vágásához (`crop()`) és maszkolásához (`mask()`) használhatunk Simple Features-t is
- előbbi a befoglaló dobozzal (`st_bbox()`) vág
- utóbbi az `sf`-objektum által takart cellákat tartja meg
- tipikusan `POLYGON` és `MULTIPOLYGON` típust használunk

## `crop(x, y)`

- `x`: a vágandó raszter
- `y`: a Simple Features, aminek a befoglaló dobozával vágunk
- eredmény: vágott raszer

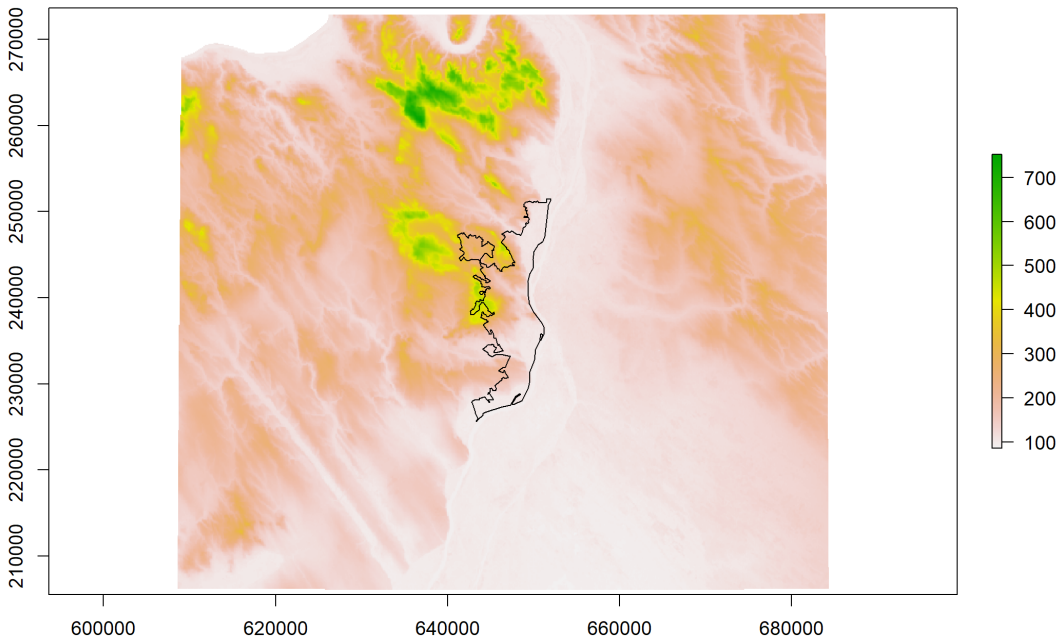
# Vágás és maszkolás

`mask(x, mask, inverse = FALSE, updatevalue = NA)`

- `x`: a maszkolandó (vágandó) raszter
- `mask`: a maszkoláshoz használni kívánt vektor (Simple Features)
- `inverse`: inkább a vektoron belül eső cellákat töröljük/módosítsuk? (alapértelmezett: nem)
- `updatevalue`: mire módosítsuk a cellaértéket? (alapértelmezett: töröljük)
- nincs `maskvalue` paraméter! (a vektor geomtriája határozza meg a maszkolást, nem az értéke)
- eredmény: egy `x`-hez hasonló raszter a maszkolási helyeken törölt/módosított értékekkel

# Vágás és maszkolás

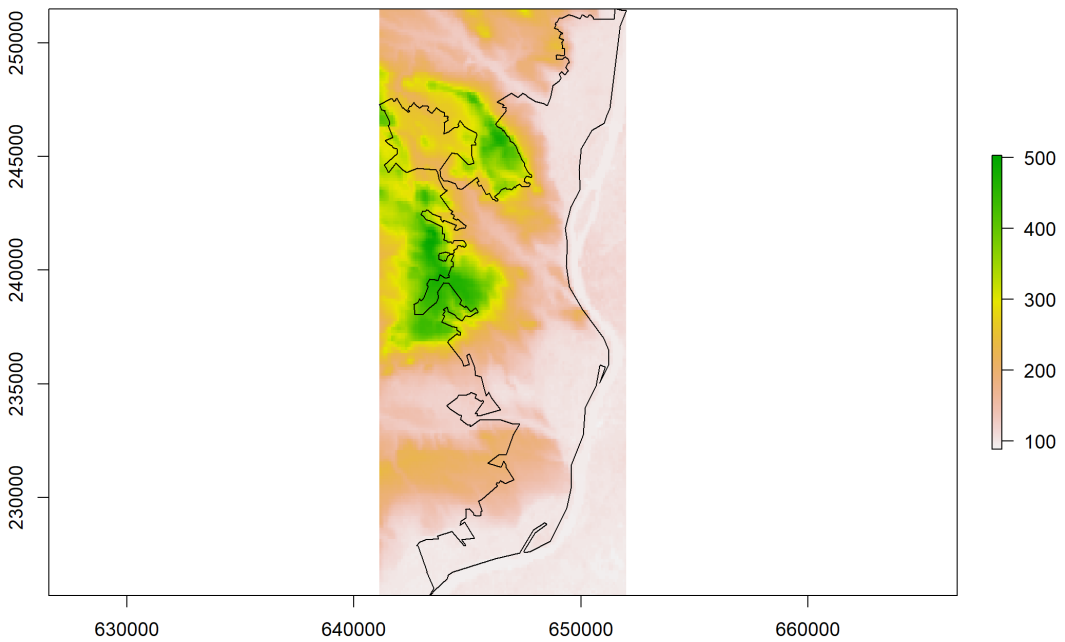
```
buda <- varosok[1, ]
plot(domborzatmodell)
plot(st_geometry(buda), add = TRUE)
```



# Vágás és maszkolás

Körbevágás befoglaló dobozzal:

```
dem_korbevagott1 <- crop(x = domborzatmodell, y = buda)
plot(dem_korbevagott1)
plot(st_geometry(buda), add = TRUE)
```



# Vágás és maszkolás

Körbevágás poligonnal (maszkolás):

```
dem_korbevagott2 <- mask(x = domborzatmodell, mask = buda,
 inverse = FALSE, updatevalue = NA)
dem_korbevagott3 <- mask(x = domborzatmodell, mask = buda,
 inverse = TRUE, updatevalue = NA)
dem_korbevagott4 <- mask(x = domborzatmodell, mask = buda,
 inverse = FALSE, updatevalue = 0)
dem_korbevagott5 <- mask(x = domborzatmodell, mask = buda,
 inverse = TRUE, updatevalue = 0)
```

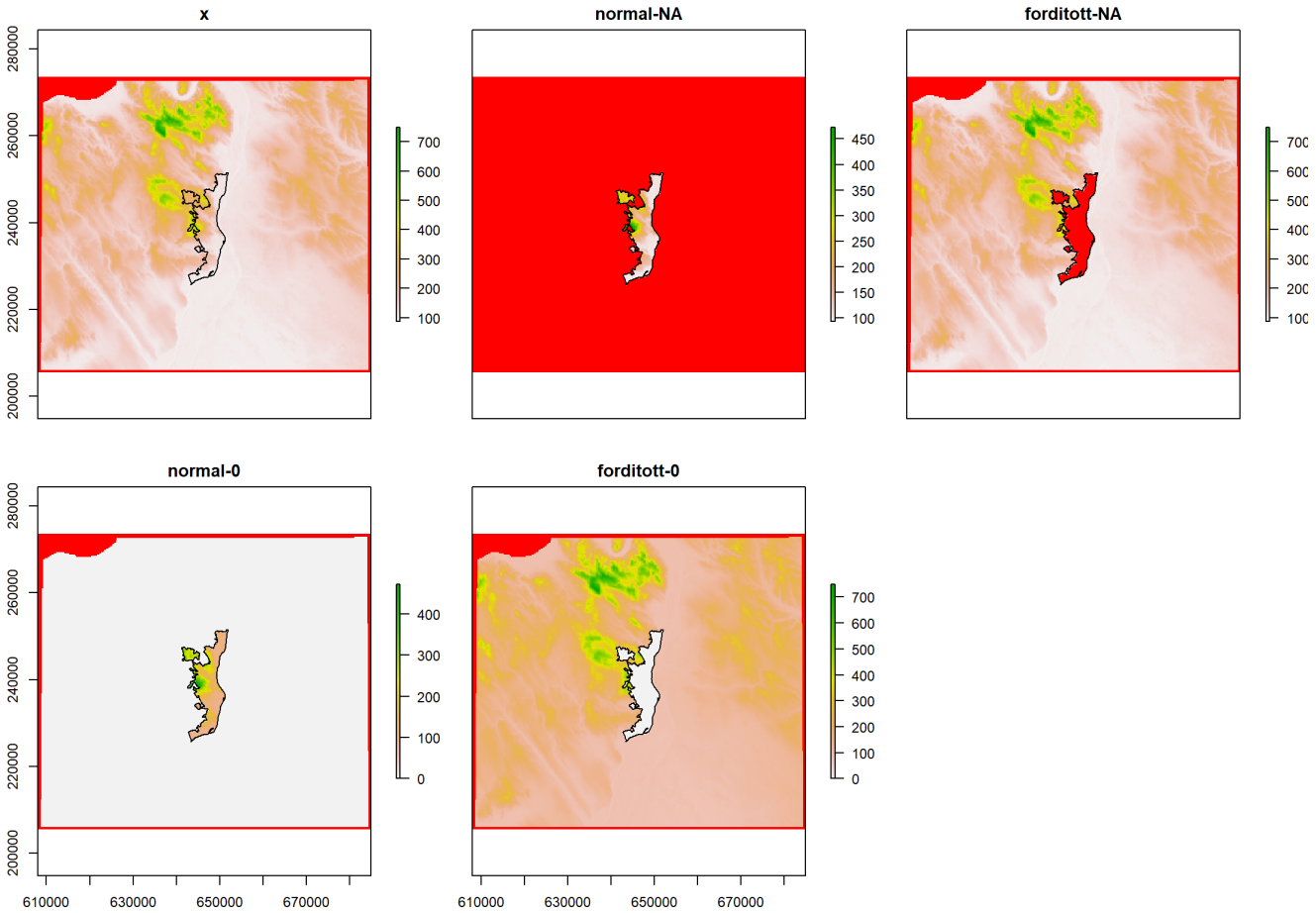
# Vágás és maszkolás

```
plot(stack(domborzatmodell, dem_korbevagott2,
 dem_korbevagott3, dem_korbevagott4, dem_korbevagott5),
 colNA = "red", main = c("x", "normal-NA", "forditott-NA",
 "normal-0", "forditott-0"), addfun = function()
 {plot(st_geometry(buda), add = TRUE)})
```

Az addfun paraméterrel nem találkoztunk még.

- De nem fogjuk később használni, elfelejthető.
- Minden egyes részábrához hozzá lehet adni vele további rétegeket.

# Vágás és maszkolás



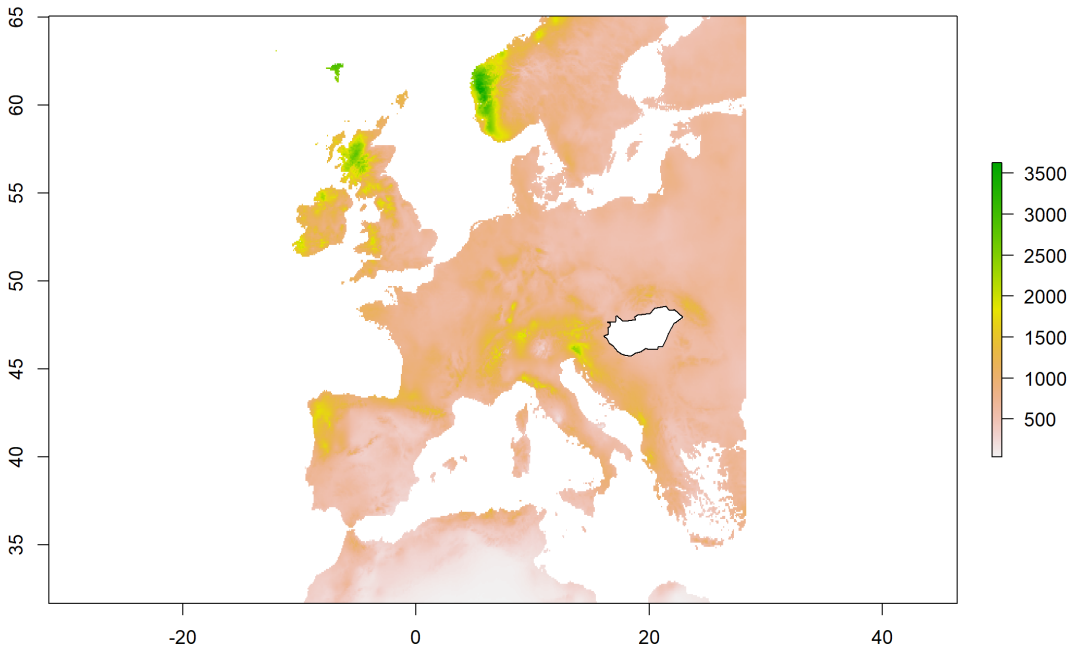

- alapértelmezett eset: a poligonon kívül eső rasztercellákat törli
- `inverse = TRUE` eset: a poligonba eső cellákat törli
- `updatevalue = 0` esetek: nem törli, hanem 0-ra módosítja

# 6. feladat (házi)

- Válogasd le a NUTS-régiók (“nuts_regiok”) közül Magyarországot, vagyis azt a poligont, ami a “regio_kod” nevű oszlopban a “HU” értéket tartalmazza.
- Készíts a “csapadek” nevű raszterből egy szűkítést, ami csak a 300–700. sorokat és 2000–2500. oszlopokat tartalmazza.
- Készítsd el a következő vágásokat/maszkolásokat a szűkített csapadékraszteren, és külön-külön jelenítsd meg őket, mindegyikre rárakva Magyarország körvonalát:
  - A: Magyarországon kívül maradjon a csapadék, azon belül törlődjön,
  - B: Magyarországon belül maradjon a csapadék, azon kívül legyen csupa 0,
  - C: Magyarországon belül maradjon a csapadék, azon kívül törlődjön, de úgy, hogy az eredményraszter fölösleges, csupa ismeretlen értéket tartalmazó sorait/oszlopait is elhagyod (a `mask()`-on kívül egy másik függvényt is használjál!),
  - D: Magyarország befoglaló dobozán belül maradjon a csapadék, a kívül eső részek kerüljenek levágásra

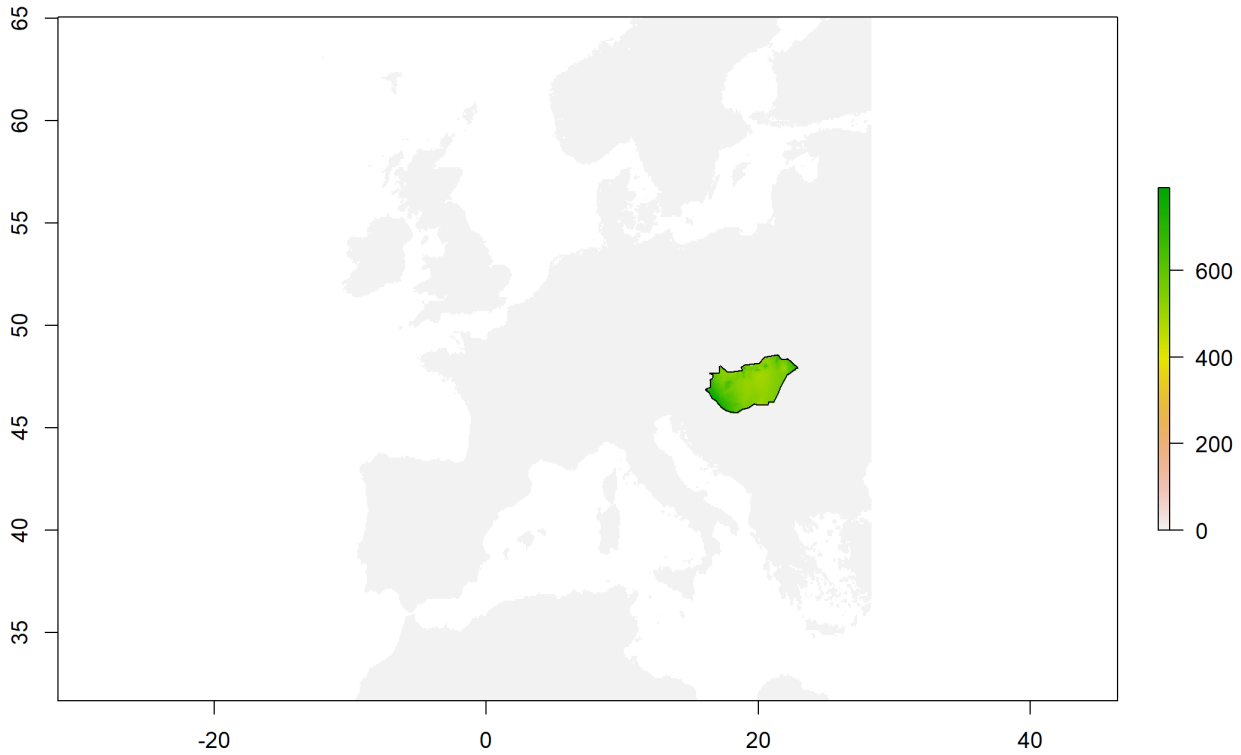
# 6. feladat (házi) – megoldás

```
magyaro <- nuts_regiok[nuts_regiok$regio_kod == "HU", ]
csapadek_eu <- csapadek[300:700, 2000:2500, drop = FALSE]
csapadek_korbevagott_a <- mask(x = csapadek_eu, mask =
 magyaro, inverse = TRUE, updatevalue = NA)
plot(csapadek_korbevagott_a)
plot(st_geometry(magyaro), add = TRUE)
```



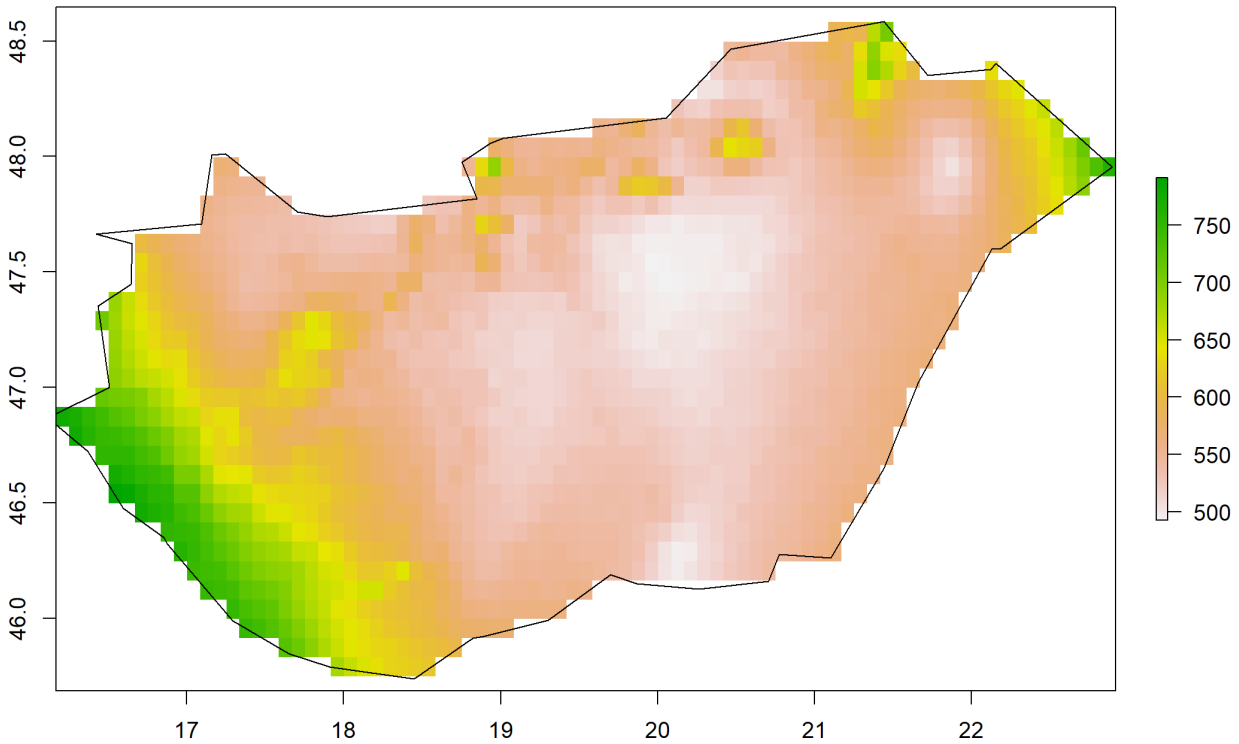
# 6. feladat (házi) – megoldás

```
csapadek_korbevagott_b <- mask(x = csapadek_eu, mask =
 magyaro, inverse = FALSE, updatevalue = 0)
plot(csapadek_korbevagott_b)
plot(st_geometry(magyaro), add = TRUE)
```



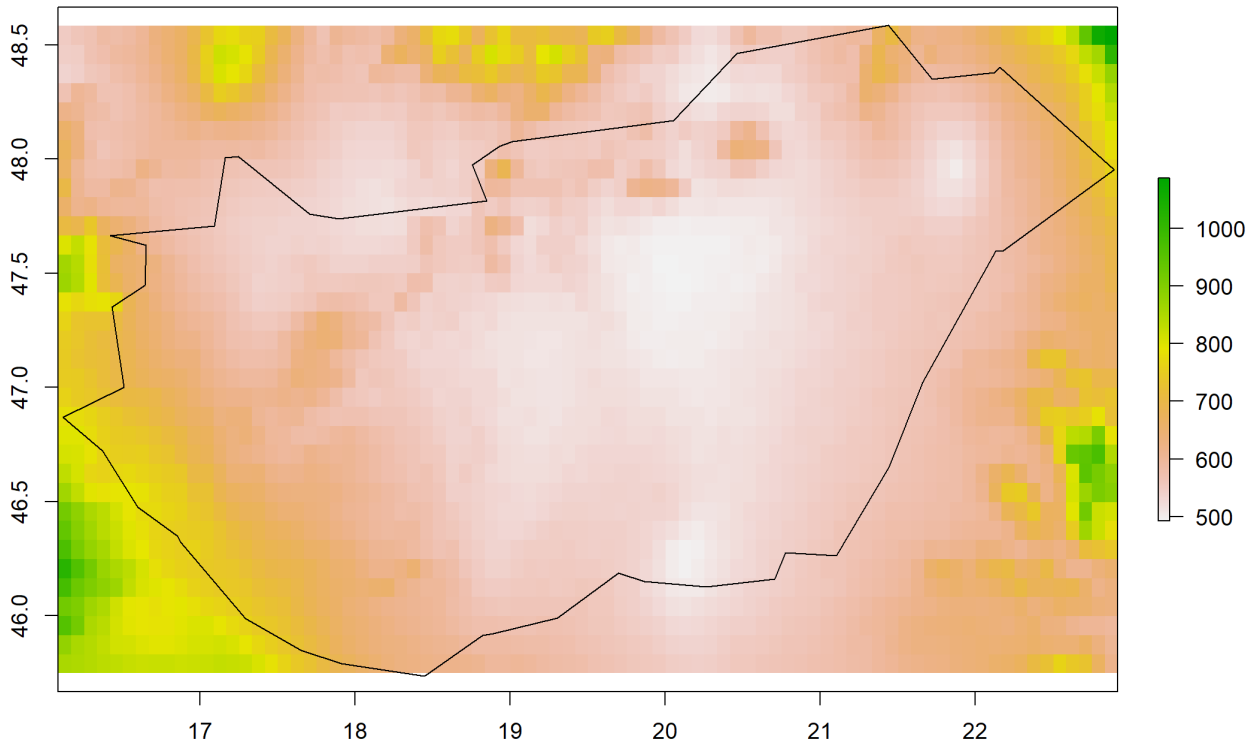
# 6. feladat (házi) – megoldás

```
csapadek_korbevagott_c <- trim(mask(x = csapadek_eu, mask
 = magyaro, inverse = FALSE, updatevalue = NA))
plot(csapadek_korbevagott_c)
plot(st_geometry(magyaro), add = TRUE)
```



# 6. feladat (házi) – megoldás

```
csapadek_korbevagott_d <- crop(x = csapadek_eu, y = magyaro)
plot(csapadek_korbevagott_d)
plot(st_geometry(magyaro), add = TRUE)
```



Section 5

# Raszter-vektor műveletek: értékkiemelés és raszterizálás

# Értékkiemelés

## Értékkiemelés

- adott egy raszter értékekkel
- és egy vektor (pont, vonal, poligon)
- minden vektorhoz várunk egy értéket, ami
  - pont esetén az alá eső cella értéke
  - vonal/poligon esetén az érintett/takart cellák értékeinek valamilyen statisztikája (pl. átlaga)
  - ha a poligonok hézagmentesen lefedik a síkot, akkor lényegileg zónastatisztikát képzünk (raszter helyett vektorral)

# Értékkiemelés

Ha poligonok alól szeretnénk értékeket kiemelni, akkor javaslom az `exactextract` csomag használatát.

- nem mutatom be
- jól dokumentált, részletesen paraméterezhető
- nagyon gyors
- figyelembe veszi a cellaátfedás mértékét is

```
extract(x, y, fun = NULL, na.rm = FALSE)
```

- `x`: a kiemelendő értékeket tartalmazó raszter
- `y`: a kiemelés helyét mutató vektor
- `fun`: a statisztikát képző függvény (vonal/poligon esetén van értelme)
- `na.rm`: elhagyja-e a statisztika számításához az ismeretlen értékeket (alapértelmezett: nem)
- eredmény pont esetén: olyan hosszú számvektor, ahány vektorunk volt
- eredmény vonal/poligon esetén: egyoszlopos mátrix annyi sorral, ahány vektorunk volt

# Értékkiemelés

Készítsük elő az adatokat!

- aggregáljuk a domborzatmodellt, hogy ne kelljen sokat nézni a homokórát, amíg az `extract()` dolgozik,
- átvetítjük a reptereket WGS-84-be

```
dem10 <- aggregate(x = domborzatmodell, fact = 10)
repterek_wgs <- st_transform(x = repterek, crs = 4326)
nrow(repterek_wgs)
```

```
[1] 9
```

# Értékkiemelés

Kiemeljük a repterek alá eső cellákból a csapadékértékeket:

```
csapadek_a_reptereken <- extract(x = csapadek, y =
 repterek_wgs)
class(csapadek_a_reptereken)
```

```
[1] "numeric"
```

```
length(csapadek_a_reptereken)
```

```
[1] 9
```

Éppen annyi értéket kaptunk, ahány repterünk van.

# Értékkiemelés

Adjuk hozzá új oszlopként a vektorhoz!

```
repterek_wgs$csapadek <- csapadek_a_reptereken
plot(repterek_wgs[, "csapadek"], pch = 16, cex = 3)
```

# Értékkiemelés

Ugyanez poligonokkal (`fun` és `na.rm` paramétereket is használjuk):

```
nehany_megye <- st_transform(x =
 nuts_regiok[nuts_regiok$regio_kod %in% c("HU101",
 "HU102", "HU312", "HU211", "HU212"), ], crs = 23700)
magassag_a_megyekben1 <- extract(x = dem10, y =
 nehany_megye, fun = mean, na.rm = TRUE)
```

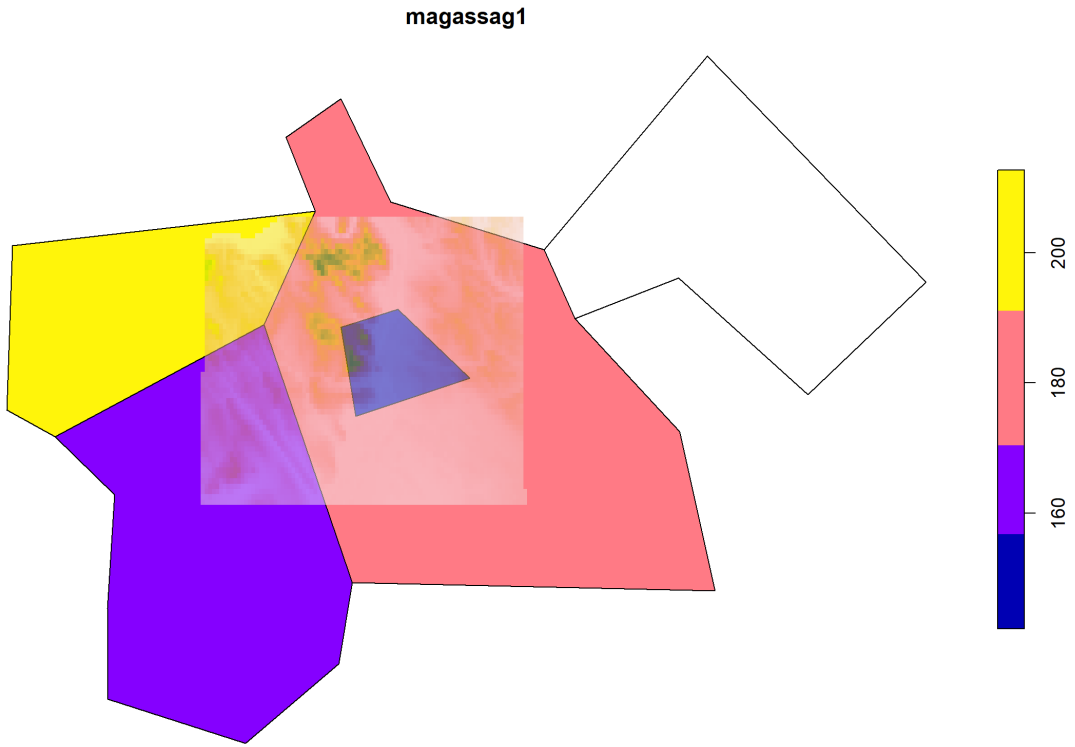
```
str(magassag_a_megyekben1)
```

```
 num [1:5, 1] 151 178 162 204 NA
```

Az eredmény egyoszlopos mátrix. Az első oszlopot a `[, 1, drop = TRUE]`-val emelhetjük ki.

# Értékkiemelés

```
nehany_megye$magassag1 <- magassag_a_megyekben1[, 1, drop
 = TRUE]
plot(nehany_megye[, "magassag1"], reset = FALSE)
plot(dem10, alpha = 0.5, add = TRUE, legend = FALSE)
```



# Értékkiemelés

Ha nem hagyjuk el az ismeretlen cellákat, akkor több `NA` vagy `NaN` értéket kapunk eredményként.

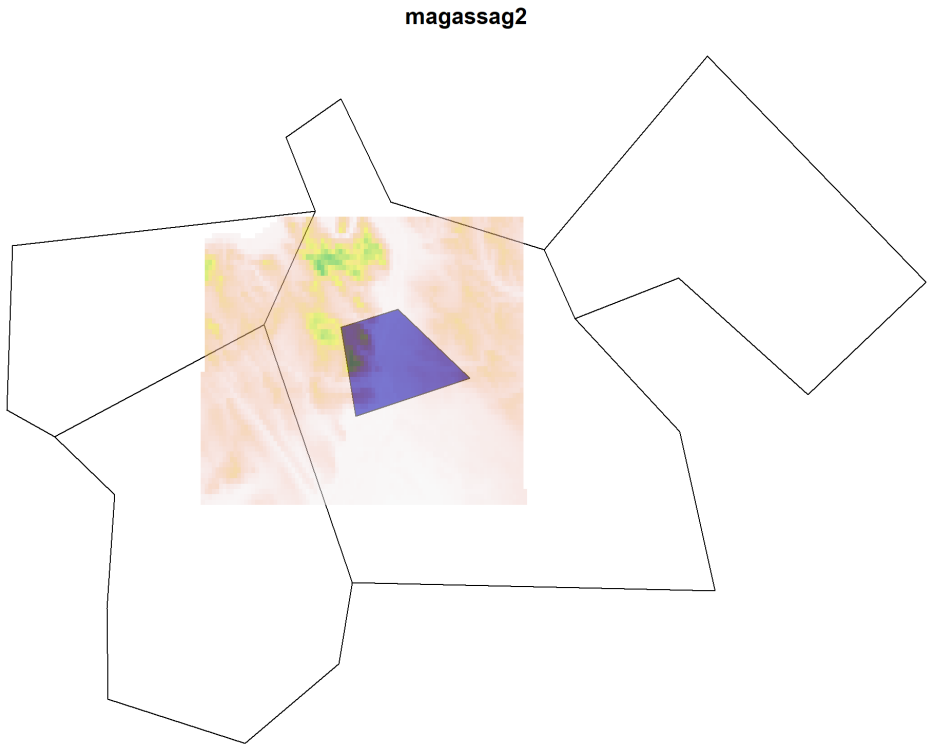
```
magassag_a_megyekben2 <- extract(x = dem10, y =
 nehany_megye, fun = mean, na.rm = FALSE)
```

```
str(magassag_a_megyekben2)
```

```
 num [1:5, 1] 151 NaN NaN NaN NA
```

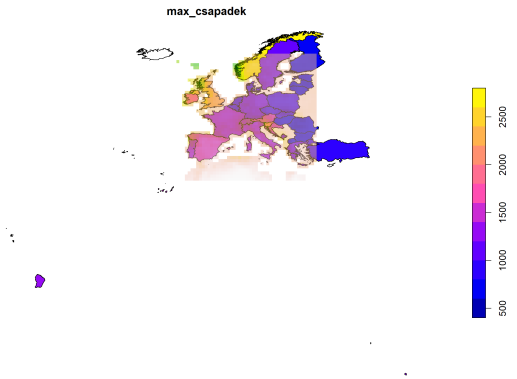
# Értékkiemelés

```
nehany_megye$magassag2 <- magassag_a_megyekben2[, 1, drop
 = TRUE]
plot(nehany_megye[, "magassag2"], reset = FALSE)
plot(dem10, alpha = 0.5, add = TRUE, legend = FALSE)
```



# 7. feladat (házi)

- A 6. feladatban létrehozott, Európára szűkített csapadékrasztert aggregáld 10×10-es cellablokkokkal, hogy gyorsabb legyen a számítás.
- Számold ki minden európai országra (“orszag_eu”) a csapadék országon belüli maximumát, az ismeretlen értékek elhagyásával.
- Mi az eredmény szerkezete?
- Add az országokhoz új oszlopként az eredményt, majd jelenítsd meg azt, hozzáadva 50%-os átlátszatlansággal, jelmagyarázat nélkül a rasztert is az ábrához.
- Válogasd le azt a repülőteret, amelynek a neve “Ferihegy”.
- Milyen magasan fekszik a ferihegyi repülőtér?



# 7. feladat (házi) – megoldás

```
csapadek_eu <- aggregate(x = csapadek_eu, fact = 10)
max_csapadek_az_orszagokban <- extract(x = csapadek_eu, y
 = orszag_eu, fun = max, na.rm = TRUE)
```

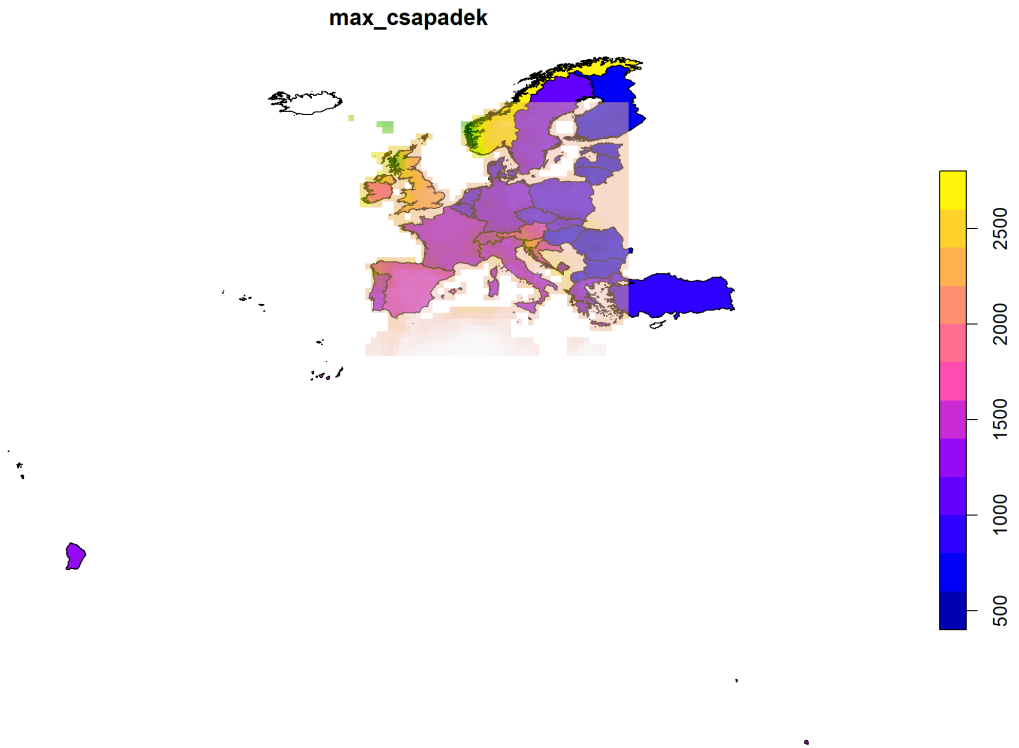
```
str(max_csapadek_az_orszagokban)
```

```
 num [1:35, 1] 1520 983 641 NA 862 ...
```

# 7. feladat (házi) – megoldás

```
orszag_eu$max_csapadek <- max_csapadek_az_orszagokban[, 1,
 drop = TRUE]
```

```
plot(orszag_eu[, "max_csapadek"], reset = FALSE)
plot(csapadek_eu, alpha = 0.5, add = TRUE, legend = FALSE)
```



# 7. feladat (házi) – megoldás

```
ferihegy <- repterek[repterek$nev == "Ferihegy", ]
```

```
extract(x = domborzatmodell, y = ferihegy)
```

```
147
```

# Vektorok raszterizálása

## Raszterizálás

- adott egy vektor (Simple Features) és egy raszter
- szeretnénk a vektort raszterré alakítani úgy, hogy követjük a bemeneti raszter szerkezetét (felbontás, dimenzió, kiterjedés)
- pont és vonal esetén beleesés/érintés, míg poligon esetén a cellaközéppont tartalmazása az irányadó
- az új raszter cellaértékeit a vektor egyik jellemzője (oszlopa) alapján képezzük
- hasonlít a `projectRaster(from, to)`-hoz, de ott a mintaraszterhez rasztert igazítunk, nem pedig vektort

# Vektorok raszterizálása

`rasterize(x, y, field, mask = FALSE, update = FALSE)`

- `x`: raszterizálandó vektor
- `y`: mintaraszter (lényegtelen, milyen értékek vannak benne, kivéve, ha `mask = TRUE` vagy `update = TRUE`)
- `field`: annak az oszlopnak a neve, amiből az értékek átkerülnek a rasztercellákba (opcionális)
- a `field` elhagyása esetén a vektor sorszáma kerül a cellákba
- `mask`: inkább ne a vektorból vegyük az értékeket, hanem a raszter eredeti értékeit tartsuk meg (maszkoljunk)? (alapértelmezett: de)
- `update`: a vektoron kívül eső cellák megtartsák az értéküket (alapértelmezett: nem, hanem legyenek ismeretlenek)
- eredmény: `y`-hoz hasonló raszter, `x` adott oszlopából vagy `y`-ból származó értékekkel az átfedő részeken, azon kívül ismeretlen vagy az eredeti értékekkel

# Vektorok raszterizálása

Ha nagy raszterrel kell dolgoznunk, akkor javaslom a `fasterize` csomagot!

- lényegileg ugyanazokat tudja, mint a `raster::rasterize()`
- de sokkal hatékonyabb

# Vektorok raszterizálása

Ugyanabba a vetületbe kell hozni az adatokat a raszterizálás előtt!
Ha betűre pontosan nem egyeznek meg, figyelmeztetést kapunk, de nyugodtan hagyjuk figyelmen kívül!

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

Városok raszterizálása a felszínborítást mintaraszterként használva, a lakosságszámot továbbörökítve:

```
varosok_raszterkent1 <- rasterize(x = varosok_wgs, y =
 felszinboritas, field = "lakossag")
class(varosok_raszterkent1)
```

```
[1] "RasterLayer"
attr(,"package")
[1] "raster"
```

# Vektorok raszterizálása

```
ncell(varosok_raszterkent1) == ncell(felszinboritas)
```

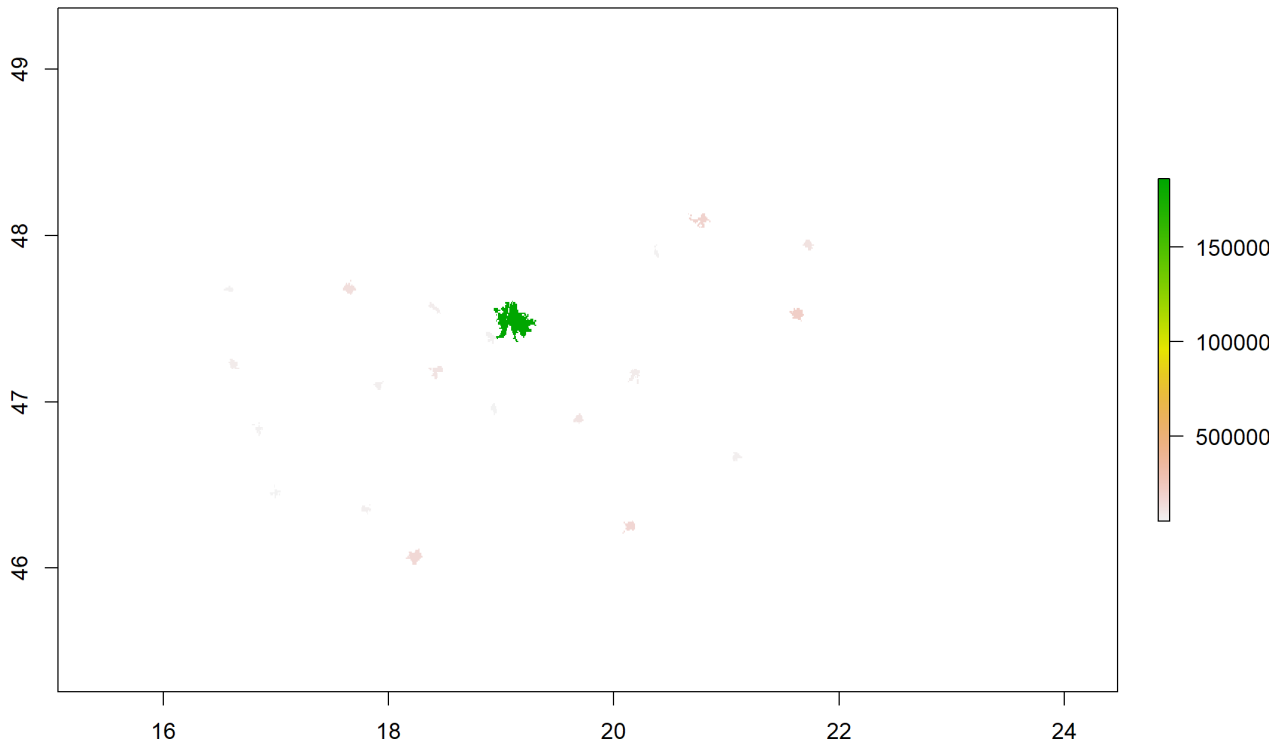
```
[1] TRUE
```

```
unique(varosok_raszterkent1[])
```

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

# Vektorok raszterizálása

```
plot(varosok_raszterkent1)
```



# Vektorok raszterizálása

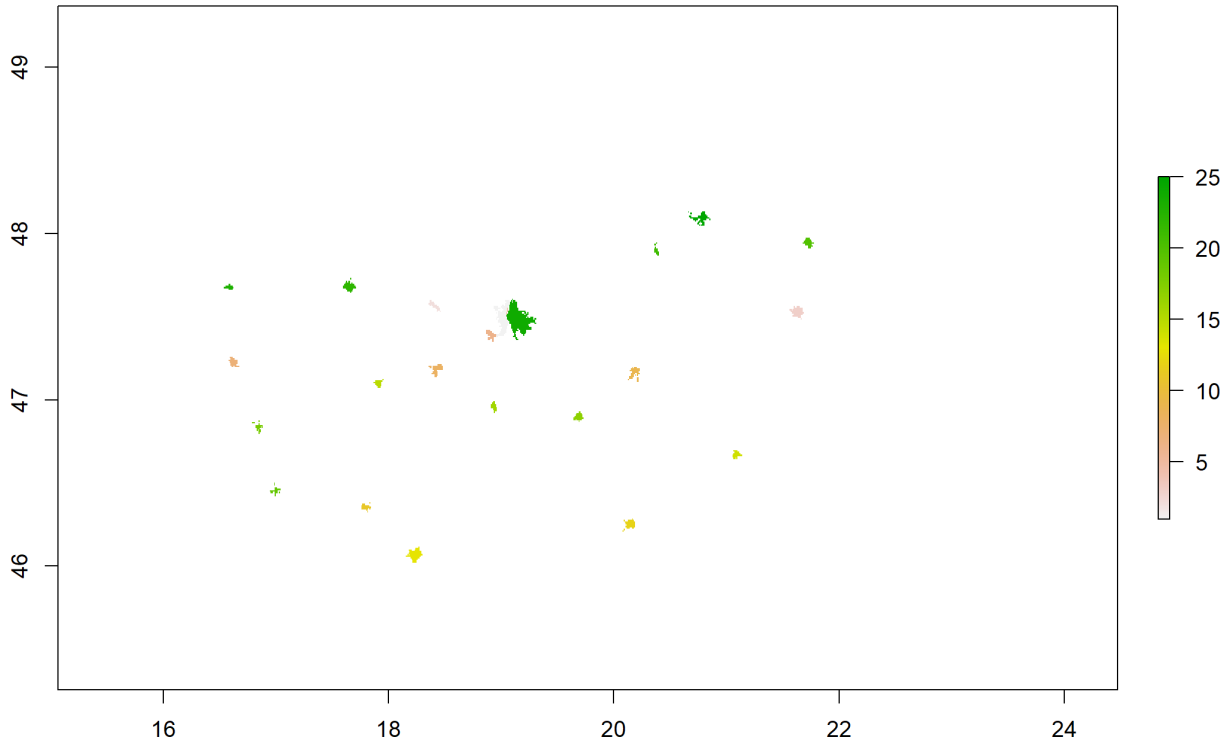
Ha a `field` paramétert elhagyjuk, akkor nem egy oszlopból kerülnek az eredményraszterbe az értékek, hanem a vektor sorszáma kerül át.

```
varosok_raszterkent2 <- rasterize(x = varosok_wgs, y =
 felszinboritas)
unique(varosok_raszterkent2[])
```

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

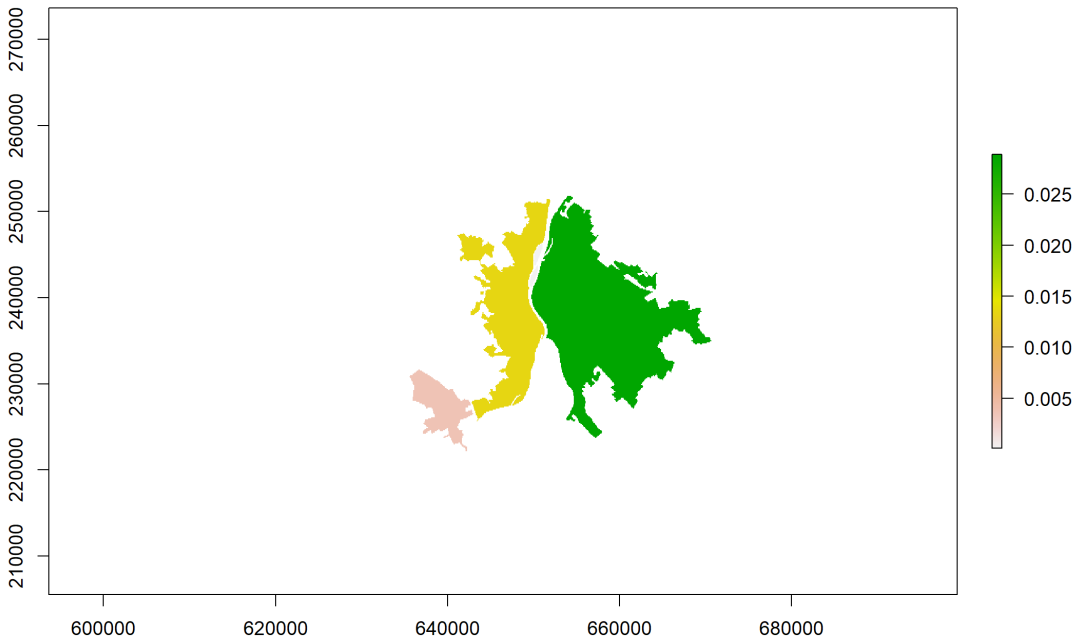
# Vektorok raszterizálása

```
plot(varosok_raszterkent2)
```



# Vektorok raszterizálása

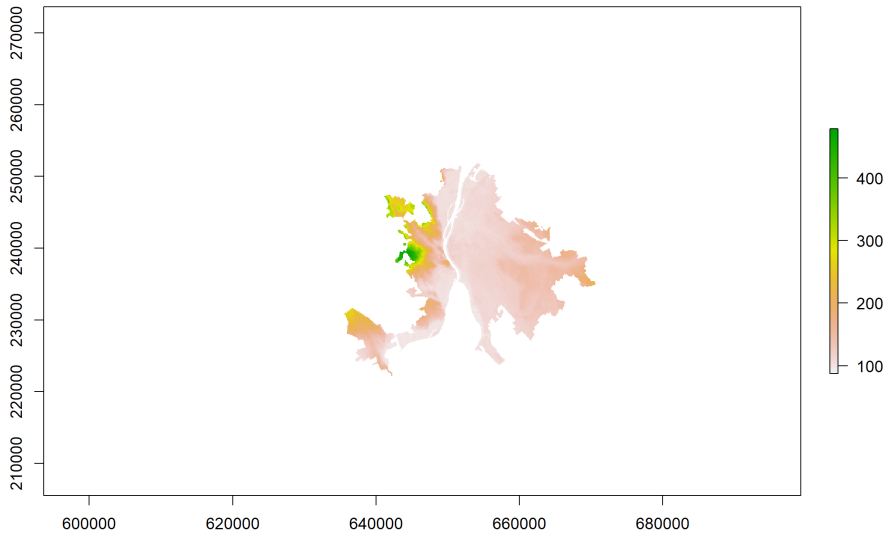
```
varosok_raszterkent3 <- rasterize(x = varosok, y =
 domborzatmodell, field = "terulet")
plot(varosok_raszterkent3)
```



# Vektorok raszterizálása

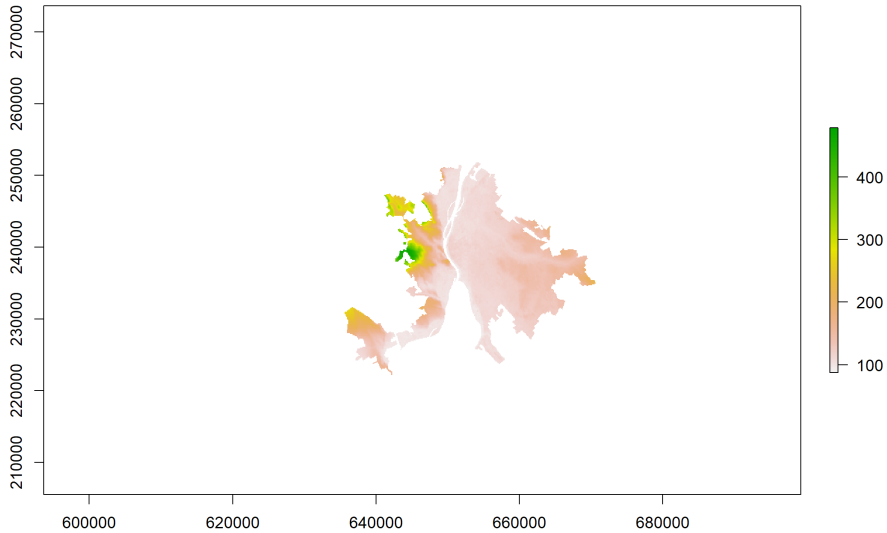
A `mask = TRUE` esetben ugyanaz történik, mintha a `mask()` függvényt használnánk.

```
varosok_raszterkent4 <- rasterize(x = varosok, y =
 domborzatmodell, mask = TRUE)
plot(varosok_raszterkent4)
```



# Vektorok raszterizálása

```
varosok_raszterkent4_mask <- mask(x = domborzatmodell,
 mask = varosok)
plot(varosok_raszterkent4_mask)
```




```
identical(varosok_raszterkent4, varosok_raszterkent4_mask)
```

```
[1] TRUE
```

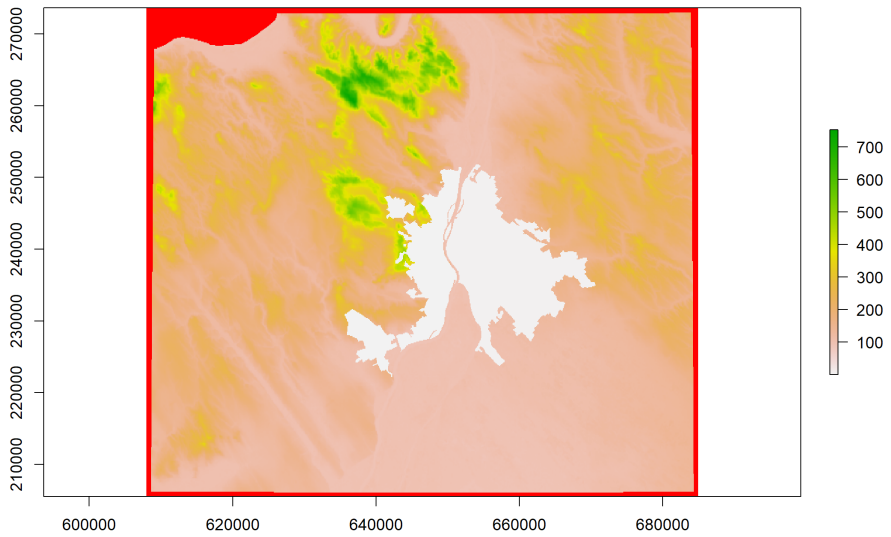
# Vektorok raszterizálása

Az `update = TRUE` eset arra való, hogy a raszter bizonyos (a vektor alá eső) értékeit frissítsük (a vektoron kívül eső cellák értéke megmarad). Most a magasságot frissítjük a városok határhosszával, aminek vajmi kevés értelme van...

```
varosok_raszterkent5 <- rasterize(x = varosok, y =
 domborzatmodell, field = "hatarhossz", update = TRUE)
```

# Vektorok raszterizálása

```
plot(varosok_raszterkent5, colNA = "red")
```




A határhosszak kicsik, ezért itt 0-nak látszanak.

# Vektorok raszterizálása

Vonalak raszterizálása:

```
nehany_ut <- utak[utak$nev %in% c("10", "11", "117"), ]
str(nehany_ut$nev)
```

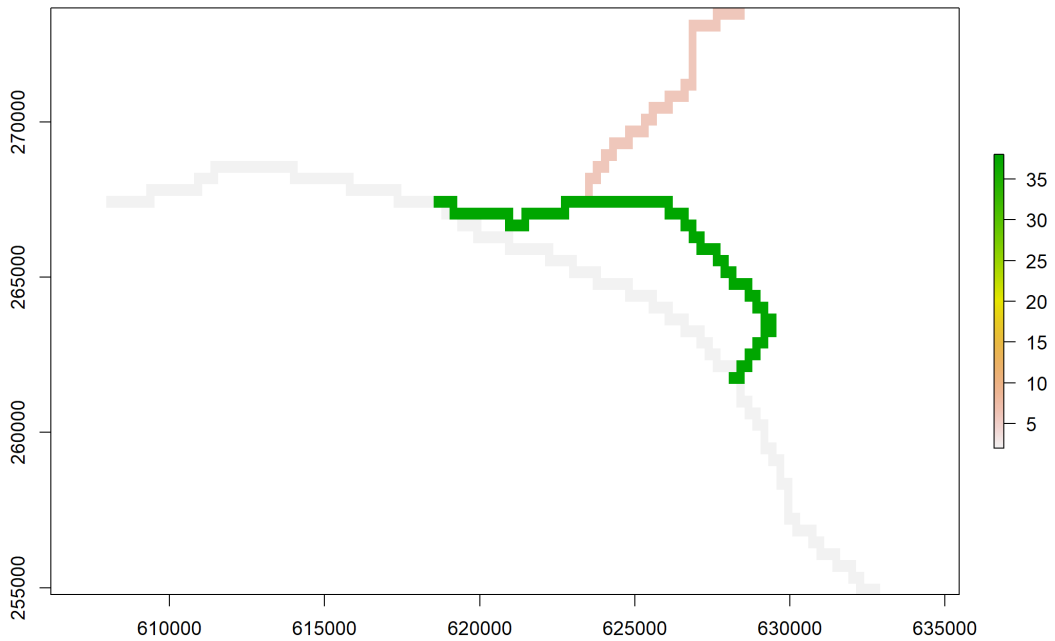
```
    Factor w/ 1048 levels "1","10","1011",..: 2 2 2 2
     2 2 2 2 2 2 ...
```

```
utak_raszterkent <- rasterize(x = nehany_ut, y =
 esztergom, field = "nev")
```

# Vektorok raszterizálása

Az eredményraszter szöveget/faktort nem tud tárolni, ezért a faktor szintjei kerültek át a "nev" oszlopból a cellákba az értékek.
Pl. a 10-es út a 2. szint, ezért itt a cellákban a 2-es érték szerepel.

```
plot(utak_raszterkent)
```

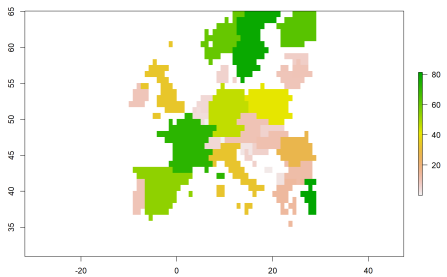


# Vektorok raszterizálása

Persze nem csak poligonokat és vonalakat, hanem pontokat is lehet raszterizálni.
Lásd a 8. feladatot.

# 8. feladat (órai)

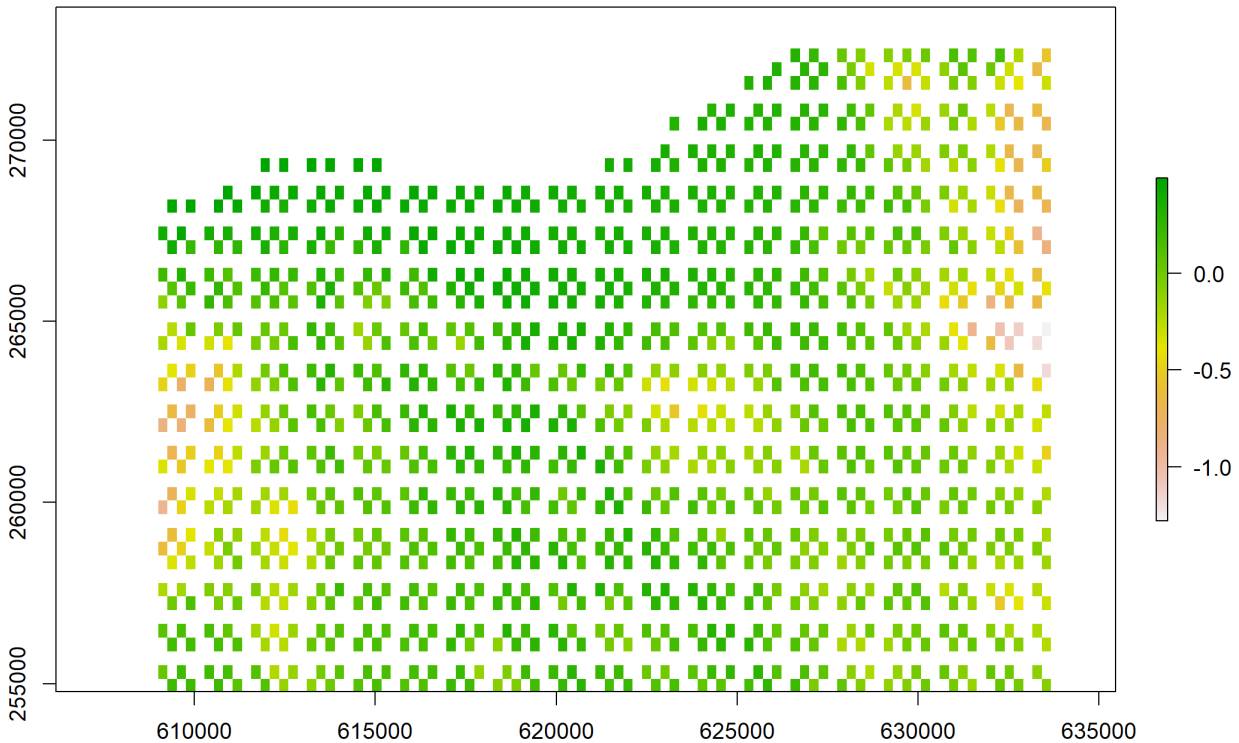
- Raszterizáld az “eghajlat” nevű Simple Features-t az “esztergom” nevű rasztert mintaként használva.
  - A téli középhőmérséklet (“T_DJF”) értékei kerüljenek a cellákba.
  - Jelenítsd meg az eredményt.
- Raszterizáld az európai országokat (“orszag_eu”) a 6. feladatban létrehozott európai csapadékrasztert mintaként használva.
  - A “terulet” nevű oszlopból kerüljenek át az értékek az új raszterbe.
  - Jelenítsd meg az eredményt.
- Készíts az európai csapadékraszterből egy frissített verziót,
  - ami az egyes országok alá eső cellákban az ország sorszámát tartalmazza.
  - Jelenítsd meg az eredményt!



# 8. feladat (órai) – megoldás

```
teli_homerseklet_raszterkent <- rasterize(x = eghajlat, y
 = esztergom, field = "T_DJF")
```

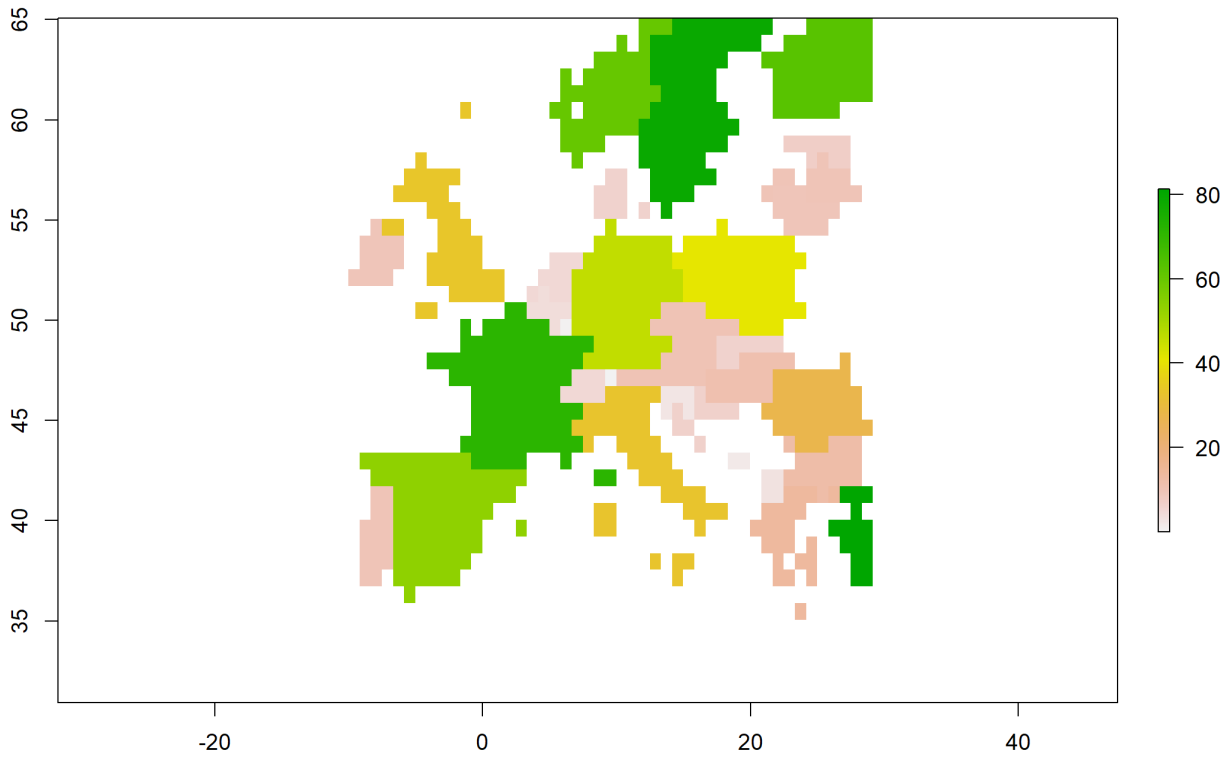
```
plot(teli_homerseklet_raszterkent)
```



# 8. feladat (órai) – megoldás

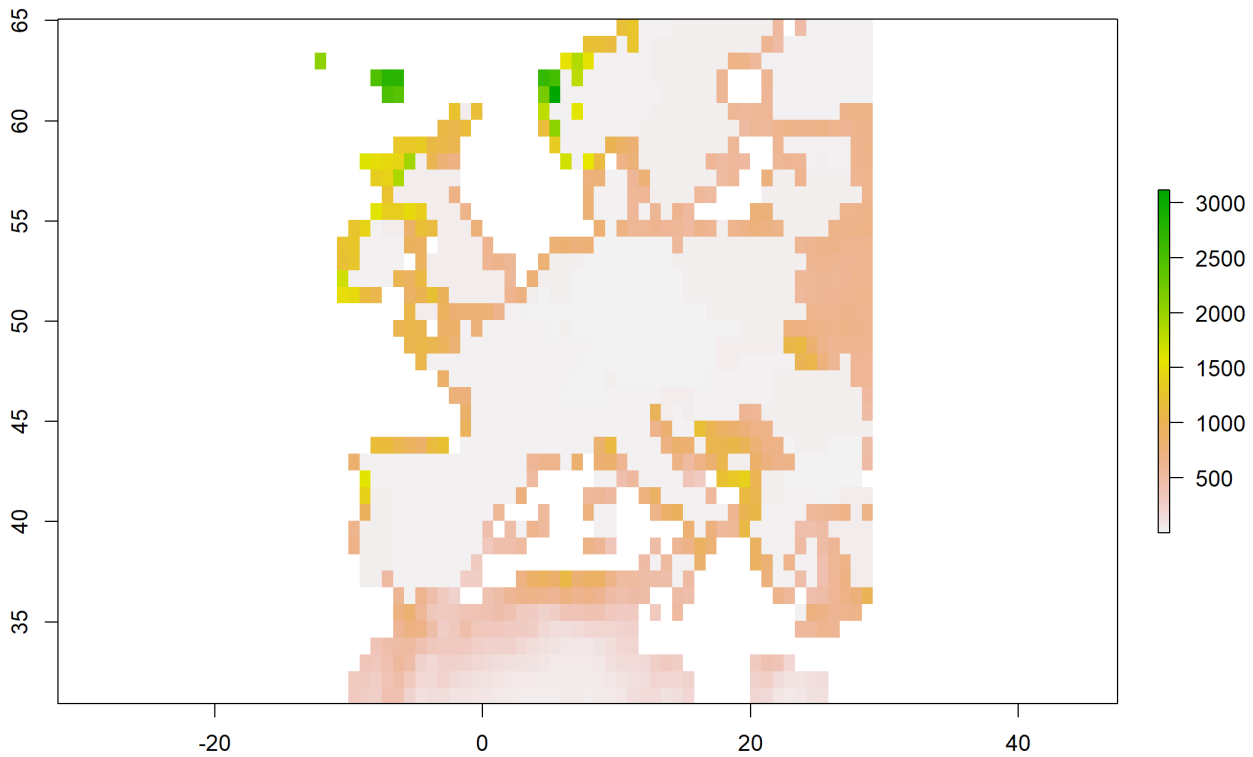
```
teruletek_raszterkent <- rasterize(x = orszag_eu, y =
 csapadek_eu, field = "terulet")
```

```
plot(teruletek_raszterkent)
```



# 8. feladat (órai) – megoldás

```
csapadek_es_sorszam <- rasterize(x = orszag_eu, y =
 csapadek_eu, update = TRUE)
plot(csapadek_es_sorszam)
```



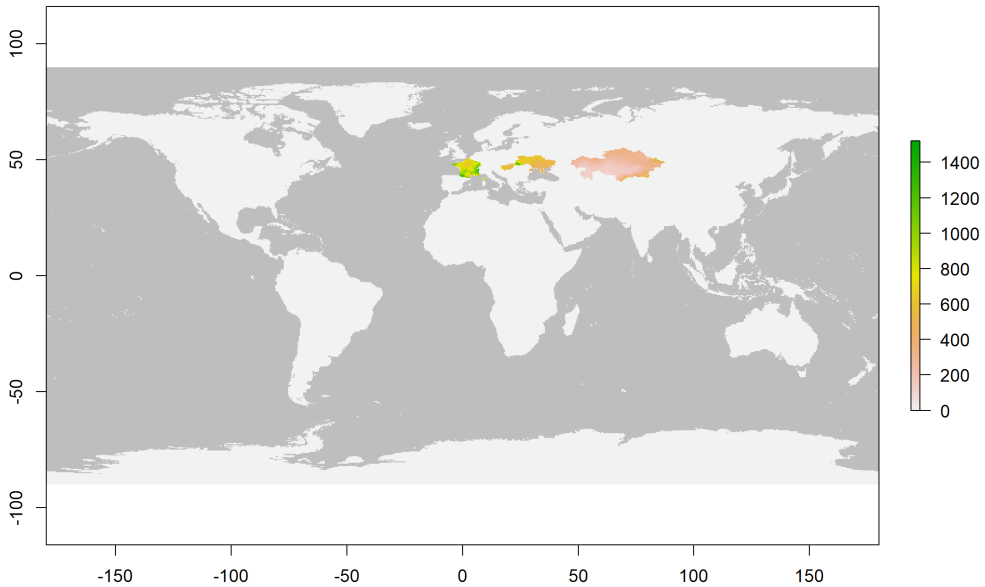
# 9. (összefoglaló) feladat (házi)

- A “csapadek” nevű raszterből készíts egy elkentebb változatot a 3×3-as súlymátrixot (“ablak_kicsi_egyesek”) használva. Az ismeretlen értékeket örökítsd tovább.
- Ahol ismeretlen érték képződött az elkenés miatt, ott pótold az értéket az eredeti csapadékraszterből.
- Töltsd be a világ országait tartalmazó “orszagok_osszes.RData” fájlt.
- Egy szövegvektorba rögzítsd Franciaország, Magyarország, Ukrajna és Kazahsztán kétbetűs országkódjait (“FR”, “HU”, “UA”, “KZ”).
- Válogasd le ezt a négy országot a fenti sorrendben (használd a `match(x, table)` függvényt!) a “kod_2betus” oszlop segítségével.
- Kétféle módon számold ki, hogy ezen országokban mennyi az átlagos csapadék:
  - egyrészt raszterizáld e négy országot, majd számolj zónastatisztikát;
  - másrészt közvetlenül a poligonokba eső rasztercellák átlagát is képezd.

...

# 9. (összefoglaló) feladat (házi)

- Mindkét eredmény esetén a sorokat nevezd át a négy országkódra, majd jelenítsd meg a képernyőn.
- Készíts egy rasztert a kiválasztott országok és az elkent csapadékraszter segítségével, amely a négy országon belül az elkent csapadékértékeket tartalmazza, de azokon kívül 0-t.
- Jelenítsd meg az eredményt, szürke színnel jelölve az ismeretlen értékeket.



# 9. (összefoglaló) feladat (házi) – megoldás

```
csapadek_elkent <- focal(x = csapadek, w =
 ablak_kicsi_egyesek, fun = mean, na.rm = FALSE)
csapadek_elkent <- cover(x = csapadek_elkent, y = csapadek)
load("orszagok_osszes.RData")
orszagkodok <- c("FR", "HU", "UA", "KZ")
nehany_orszag <- orszagok_osszes[match(x = orszagkodok,
 table = orszagok_osszes$kod_2betus), ]
```

```
orszagkodok_raszterkent <- rasterize(x = nehany_orszag, y
 = csapadek_elkent)
zonastatisztika <- zonal(x = csapadek_elkent, z =
 orszagkodok_raszterkent, fun = mean, na.rm = TRUE)
rownames(zonastatisztika) <- orszagkodok
```

# 9. (összefoglaló) feladat (házi) – megoldás

```
zonastatisztika
```

```
   zone    value
FR    1 848.4825
HU    2 568.4244
UA    3 575.8344
KZ    4 246.2405
```

```
poligonos_statisztika <- extract(x = csapadek_elkent, y =
 nehany_orszag, fun = mean, na.rm = TRUE)
rownames(poligonos_statisztika) <- orszagkodok
poligonos_statisztika
```

```
       [,1]
FR 848.4825
HU 568.4244
UA 575.8344
KZ 246.2405
```

# 9. (összefoglaló) feladat (házi) – megoldás

```
csapadek_mashol <- mask(x = csapadek_elkent, mask =
 nehany_orszag, updatevalue = 0)
plot(csapadek_mashol, colNA = "gray")
```