Raszterek kezelése

Térinformatika R-ben

2023.11.20.

Section 1

Raszterekről általában

Raszterekről általában

Mi a raszter?

- cellák táblázatszerű elrendezésben
- egy cellához tartozhat egy vagy több érték (rétegek)

Csomagok

- **`raster`**: raszterek kezelése, megjelenítése, beolvasása, mentése
- `rgdal`: raszterek beolvasása, mentése
- `terra`: raszterek kezelése
- `stars`: időbeli dimenziót is tartalmazó raszterek kezelése

Raszterekről általában

Memóriakezelés

- a raszterek általában nagyok, feldolgozásuk sok időbe telhet (akár napok!)
- fájlmérettől függ, hogy a memóriába olvassa-e
- képes részletekben (chunk) feldolgozni
- menet közben módosulhat (beolvas memóriába vagy kiment ideiglenes fájlba)
- ha rasztert a `save()`-vel mentünk $\rightarrow$ mindenképp a memóriában legyen!
- összességében tehát nem fogyasztja el a RAM-ot!

Raszterekről általában

Adattípusok

- 3 típus létezik
- a csomag legtöbb függvénye mindhárom típust tudja kezelni (de nem feltétlenül azonos módon)
- "egyrétegű" = max. 1 rétegű
- "többrétegű" = akárhány (0, 1, sok) rétegű
- `RasterLayer`: egyrétegű raszter
- `RasterBrick`: többrétegű raszter, gyors kezelés, memóriában vagy egy fájlban
- `RasterStack`: többrétegű raszter, lassabb kezelés, rétegenként akár eltérő helyeken (külön fájlokban)

Section 2

Beolvasás, mentés, létrehozás

Egyrétegű raszter létrehozása mátrixból

A raszter hasonlít a `numeric` típusú mátrixra, csak térbeliséggel bír.
Ezért könnyen létrehozhatjuk egy mátrixból.
Ehhez előbb készítsünk véletlen egész számokat tartalmazó mátrixot véletlen mintavétellel!

`set.seed(seed)`

- rögzíti a véletlenszám-generátort

`sample(x, size, replace = FALSE)`

- véletlen mintát vesz
- `x`: a potenciális elemek vektora
- `size`: ennyi elemet választ `x`-ből
- `replace`: visszatevés nélküli legyen-e a mintavétel

Egyrétegű raszter létrehozása mátrixból

`matrix(data = NA, nrow = 1, ncol = 1)`

- létrehoz egy mátrixot
- `data`: a mátrixba kerülő elemek (pl. számok) vektora, oszlopfolytonosan
- `nrow`: sorok szama
- `ncol`: oszlopok száma

`raster(x)`

- egyrétegű raszter készít egy mátrixból (vagy egyéb dologból; később...)
- `x`: a mátrix, amit raszterré szeretnénk alakítani

Egyrétegű raszter létrehozása mátrixból

```
library(raster)
```

```
set.seed(12345)
szamok <- sample(x = 1:10, size = 6 * 8, replace = TRUE)
szammatrix <- matrix(data = szamok, ncol = 6, nrow = 8)
```

```
print(szammatrix)
```

Egyrétegű raszter létrehozása mátrixból

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    3    7    8    4    3    3
[2,]   10   10   10    9    1    1
[3,]    8    1    3    9    1   10
[4,]   10    8    9    4    5    8
[5,]    8    7    4    8   10    9
[6,]    2    6   10    6   10    8
[7,]    6    1    7    9    3    4
[8,]    6    4    2    5    3    2
```

Egyrétegű raszter létrehozása mátrixból

```
szamraszter <- raster(x = szammatrix)
class(szamraszter)
```

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

Tehát ez egy egyrétegű raszter (`RasterLayer`), és a típust a `raster` csomag definiálja.

Egyrétegű raszter létrehozása mátrixból

```
print(szamraszter)
```

```
class       : RasterLayer
dimensions  : 8, 6, 48  (nrow, ncol, ncell)
resolution  : 0.1666667, 0.125  (x, y)
extent      : 0, 1, 0, 1  (xmin, xmax, ymin, ymax)
crs         : NA
source      : memory
names       : layer
values      : 1, 10  (min, max)
```

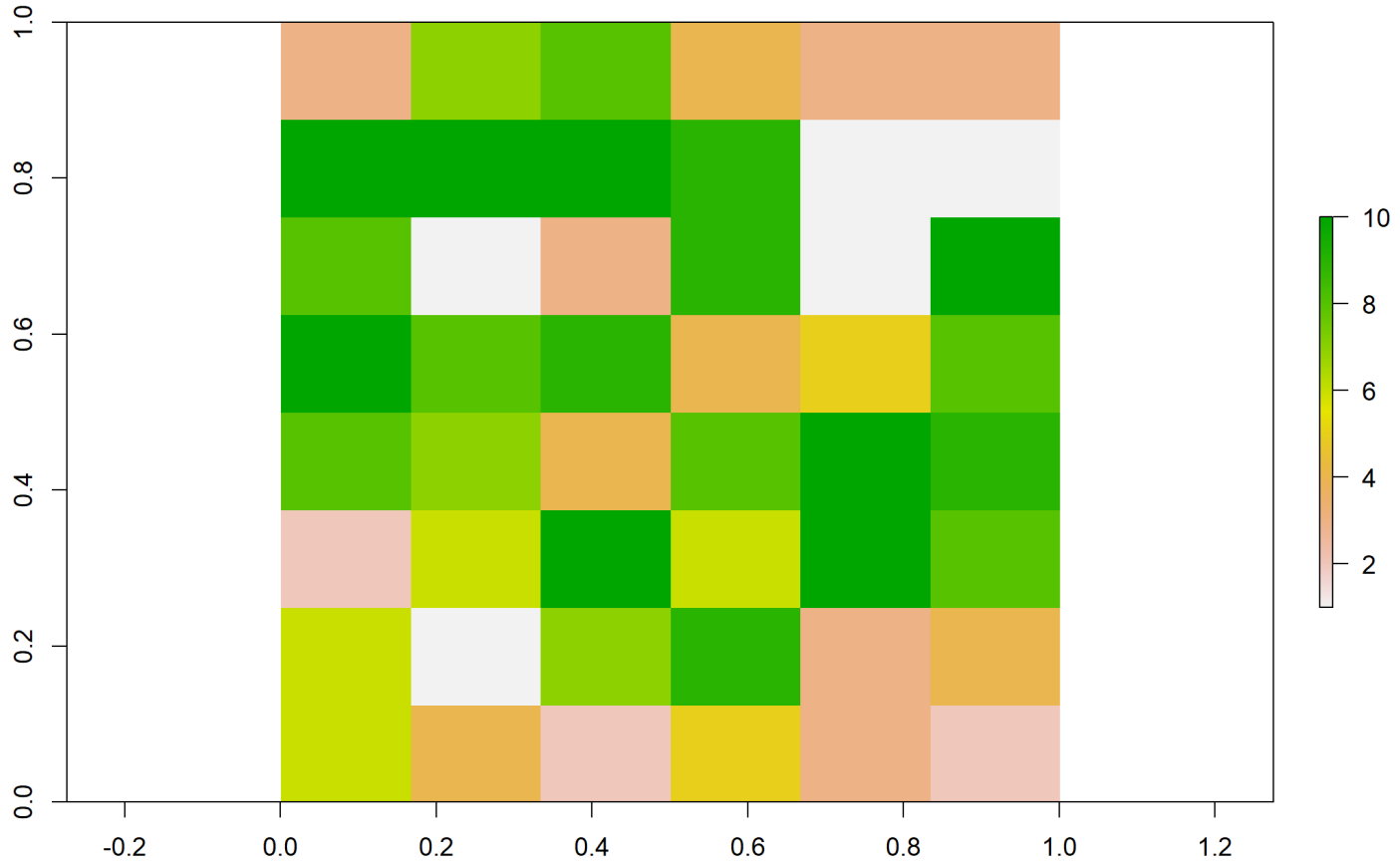
A `print()` a raszter legfontosabb jellemzőit írja ki.

- típus
- sorok/oszlopok/cellák(/rétegek) száma
- cellák felbontása (élhossza)
- kiterjedés, vetület, hely
- rétegnevek és a rétegek min./max.-értékei

Raszter gyors megjelenítése

A `plot()` függvénnyel a rasztert ábrázolhatjuk.

```
plot(szamraszter)
```



Egyrétegű raszter beolvasása fájlból

A `raster()` függvény nem csak mátrixot, hanem fájlnevet (egyelemű karaktervektort) is fogad.

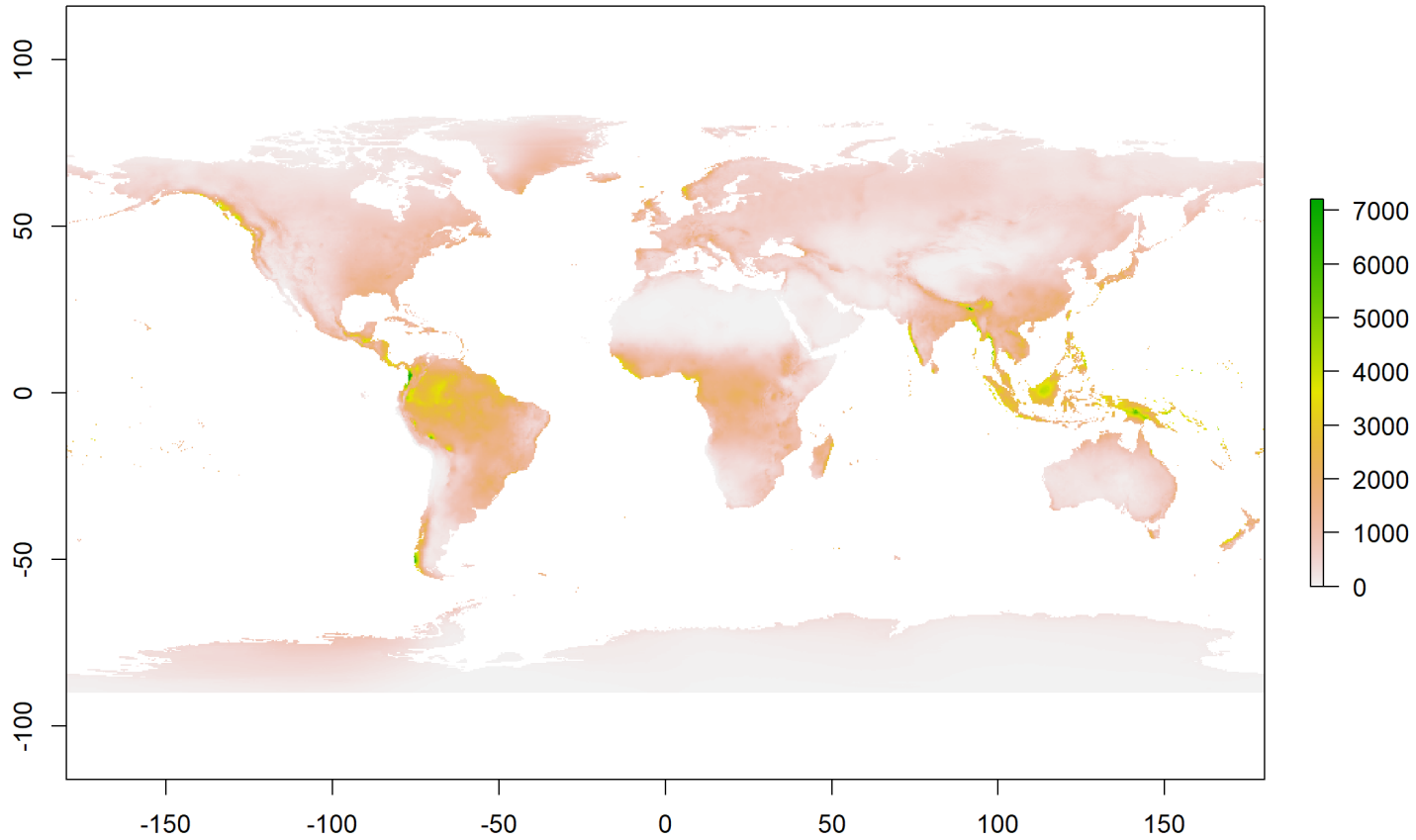
```
csapadek <- raster(x = "csapadek.tif")
```

```
print(csapadek)
```

```
class      : RasterLayer
dimensions : 2160, 4320, 9331200  (nrow, ncol, ncell)
resolution : 0.08333333, 0.08333333  (x, y)
extent     : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : csapadek.tif
names      : csapadek
values     : 0, 11191  (min, max)
```

Egyrétegű raszter beolvasása fájlból

```
plot(csapadek)
```

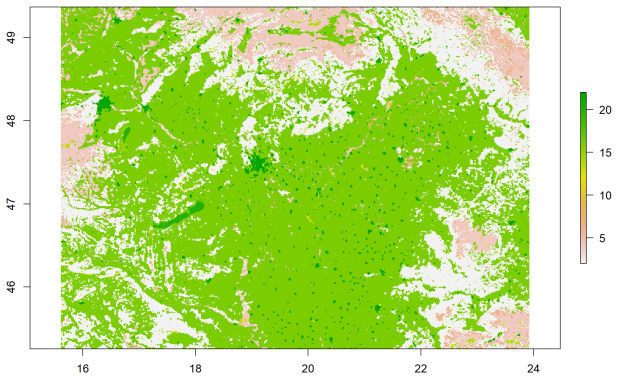


1. feladat (órai)

- Hozz létre “szammatrix2” néven egy 10×10-es mátrixot, amibe kerüljenek 1-től 100-ig az egész számok.
- Alakítsd e mátrixot `RasterLayerré` “szamraszter2” néven, és jelenítsd meg.
- Olvasd be a felszinboritas.tif fájlt a “felszinboritas” nevű raszterbe.
- Futtasd le a következő sort, (amivel törlöd a raszter beépített jelmagyarázatát (később...), segítve a megjelenítést):

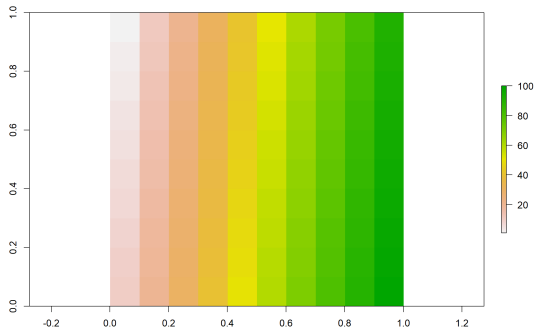
```
colortable(felszinboritas) <- logical()
```

- Jelenítsd meg.



1. feladat (órai) – megoldás

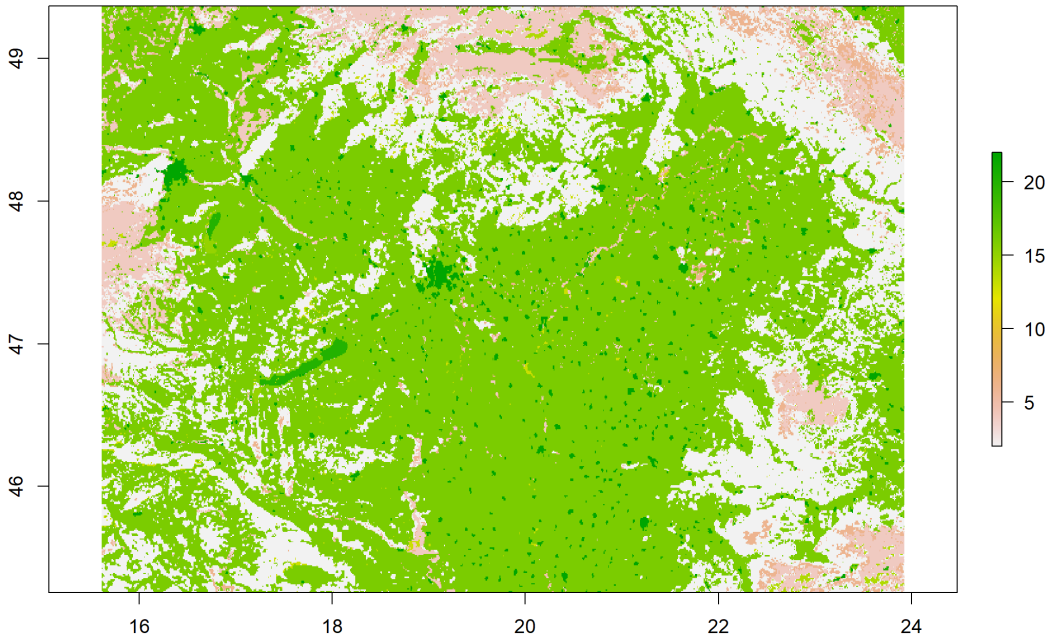
```
szammatrix2 <- matrix(data = 1:100, ncol = 10, nrow = 10)
szamraszter2 <- raster(x = szammatrix2)
plot(szamraszter2)
```



1. feladat (órai) – megoldás

```
felszinboritas <- raster(x = "felszinboritas.tif")
colortable(felszinboritas) <- logical()
```

```
plot(felszinboritas)
```



Többrétegű raszter beolvasása fájlból

Beolvasó függvények:

- `RasterLayer`: `raster()`
- `RasterBrick`: `brick()`
- `RasterStack`: `stack()`

```
ortofoto <- brick(x = "ortofoto.tif")
class(ortofoto)
```

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

Többrétegű raszter beolvasása fájlból

```
print(ortofoto)
```

Megjelenik az `nlayers` jellemző: rétegek száma.
Az első néhány réteg nevét, minimumát és maximumát felsorolja.

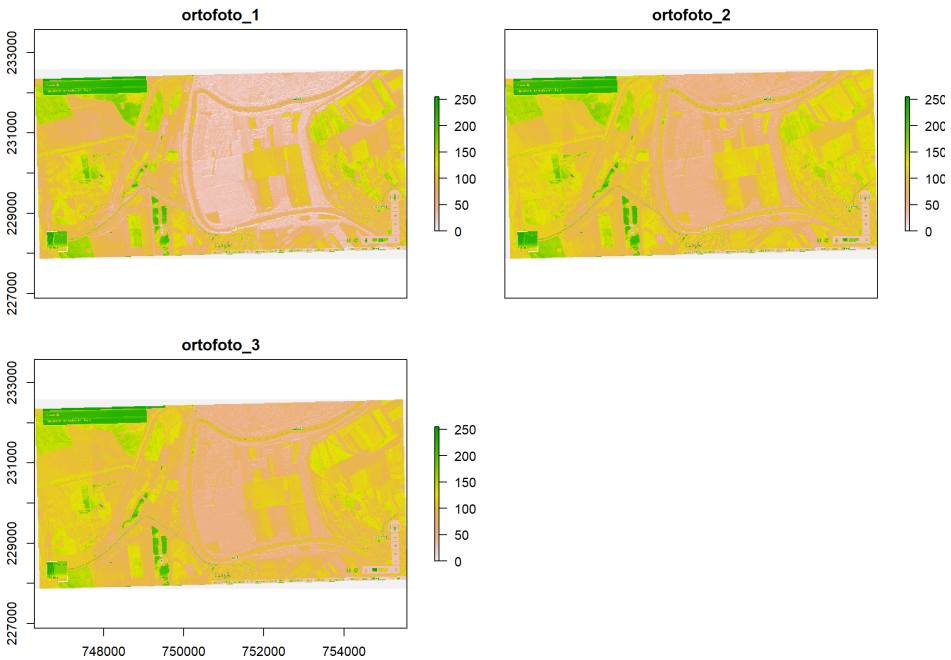
Többrétegű raszter beolvasása fájlból

```
class       : RasterBrick
dimensions  : 1314, 2591, 3404574, 3 (nrow, ncol, ncell,
 nlayers)
resolution  : 3.587778, 3.587778  (x, y)
extent      : 746278.1, 755574, 227871.8, 232586.2 (xmin, xmax,
 ymin, ymax)
crs         : +proj=somerc +lat_0=47.1443937222222
 +lon_0=19.0485717777778 +k_0=0.99993 +x_0=650000
 +y_0=200000 +ellps=GRS67 +units=m +no_defs
source      : ortofoto.tif
names       : ortofoto_1, ortofoto_2, ortofoto_3
min values  :          0,          0,          0
max values  :        255,        255,        255
```

Többrétegű raszter beolvasása fájlból

A `plot()` most több réteget külön jelenít meg.
Maximum 16-ot (módosítható).

```
plot(ortofoto)
```



Persze inkább színes képként szeretnénk ábrázolni, nem színcsatornánként külön (később...).

Többrétegű raszter beolvasása fájlból

```
ortofoto <- stack(x = "ortofoto.tif")
class(ortofoto)
```

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

Többrétegű raszter beolvasása fájlból

A `print()` most nem írja ki az adat helyét, mert az rétegenként eltérő lehet.

```
print(ortofoto)
```

```
class       : RasterStack
dimensions : 1314, 2591, 3404574, 3 (nrow, ncol, ncell,
 nlayers)
resolution : 3.587778, 3.587778  (x, y)
extent : 746278.1, 755574, 227871.8, 232586.2 (xmin, xmax,
 ymin, ymax)
crs : +proj=somerc +lat_0=47.1443937222222
 +lon_0=19.0485717777778 +k_0=0.99993 +x_0=650000
 +y_0=200000 +ellps=GRS67 +units=m +no_defs
names       : ortofoto_1, ortofoto_2, ortofoto_3
min values :           0,           0,           0
max values :         255,         255,         255
```

Többrétegű raszter létrehozása

`RasterStacket` létrehozhatunk

- külön fájlokból vagy
- már beolvasott/létrehozott raszterekből (tipikusan `RasterLayer`ekből, de akár többrétegűekből is).

```
dem_es_homerseklet <- stack("domborzatmodell.tif",
 "felszinhomerseklet.tif")
```

- dem: **D**igital **E**levation **M**odel, domborzatmodell (magasság méterben)
- homerseklet: távérzékelt felszíni hőmérséklet °C-ban

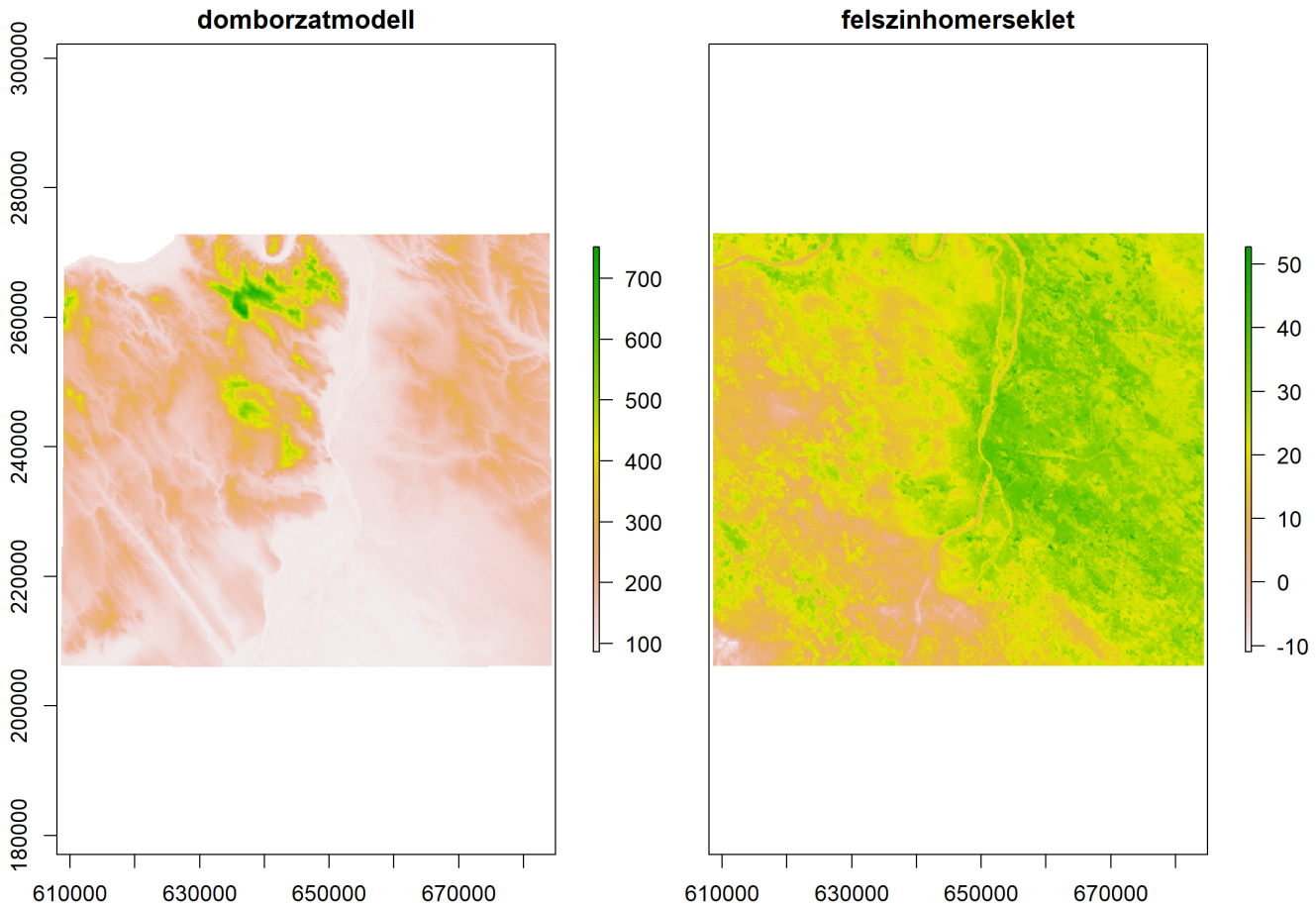
Többrétegű raszter létrehozása

```
print(dem_es_homerseklet)
```

```
class       : RasterStack
dimensions : 541, 898, 485818, 2 (nrow, ncol, ncell,
 nlayers)
resolution : 85.7, 126  (x, y)
extent : 607957.6, 684916.2, 205511.8, 273677.8 (xmin,
 xmax, ymin, ymax)
crs : +proj=somerc +lat_0=47.1443937222222
 +lon_0=19.0485717777778 +k_0=0.99993 +x_0=650000
 +y_0=200000 +ellps=GRS67 +units=m +no_defs
names       : domborzatmodell, felszinhomerseklet
min values :         86.00000,           -10.98939
max values :        752.00000,            54.44464
```

Többrétegű raszter létrehozása

```
plot(dem_es_homerseklet)
```



Többrétegű raszter létrehozása

Ugyanaz másképpen:

- beolvasunk két `RasterLayert`
- majd azokat fűzzük össze `RasterStackké`

```
dem <- raster(x = "domborzatmodell.tif")
homerseklet <- raster(x = "felszinhomerseklet.tif")
```

```
dem_es_homerseklet <- stack(dem, homerseklet)
```

Ugyanezt nem csak `RasterLayerekkel`, hanem `RasterStackekkel` és `RasterBrickekkel` is eljátszhatjuk (lásd: 2. feladat)

Többrétegű raszter létrehozása

```
print(dem_es_homerseklet)
```

```
class       : RasterStack
dimensions : 541, 898, 485818, 2 (nrow, ncol, ncell,
 nlayers)
resolution : 85.7, 126  (x, y)
extent : 607957.6, 684916.2, 205511.8, 273677.8 (xmin,
 xmax, ymin, ymax)
crs : +proj=somerc +lat_0=47.1443937222222
 +lon_0=19.0485717777778 +k_0=0.99993 +x_0=650000
 +y_0=200000 +ellps=GRS67 +units=m +no_defs
names       : domborzatmodell, felszinhomerseklet
min values :         86.00000,           -10.98939
max values :        752.00000,            54.44464
```

2. feladat (házi)

- Olvasd be a felszinhomerseklet.tif-et “homerseklet” néven `RasterStack`ként, valamint a domborzatmodell.tif fájlt “dem” néven `RasterBrick`ként.
- Hozz belőlük létre “homerseklet_es_dem” néven egy `RasterStacket`.
- A legfontosabb jellemzőit írasd ki a képernyőre.

2. feladat (házi) – megoldás

```
homerseklet <- stack(x = "felszinhomerseklet.tif")
dem <- brick(x = "domborzatmodell.tif")
```

```
homerseklet_es_dem <- stack(homerseklet, dem)
```

```
print(homerseklet_es_dem)
```

2. feladat (házi) – megoldás

```
class       : RasterStack
dimensions  : 541, 898, 485818, 2 (nrow, ncol, ncell,
 nlayers)
resolution  : 85.7, 126  (x, y)
extent      : 607957.6, 684916.2, 205511.8, 273677.8 (xmin,
 xmax, ymin, ymax)
crs         : +proj=somerc +lat_0=47.1443937222222
 +lon_0=19.0485717777778 +k_0=0.99993 +x_0=650000
 +y_0=200000 +ellps=GRS67 +units=m +no_defs
names       : felszinhomerseklet, domborzatmodell
min values  :          -10.98939,        86.00000
max values  :           54.44464,       752.00000
```

Raszter létrehozása koordinátát tartalmazó táblázatból

`rasterFromXYZ(xyz)`

- `xyz`: a koordinátát tartalmazó táblázat
- lehet `data.frame` vagy `matrix`
- első 2 oszlop a 2 koordináta
- 3 oszlopban a cellaértékek
- többrétegű raszter esetén 4., 5. stb. oszlopok is
- visszatérési értéke az oszlopszámtól függően `RasterLayer` vagy `RasterBrick`

Raszter létrehozása koordinátát tartalmazó táblázatból

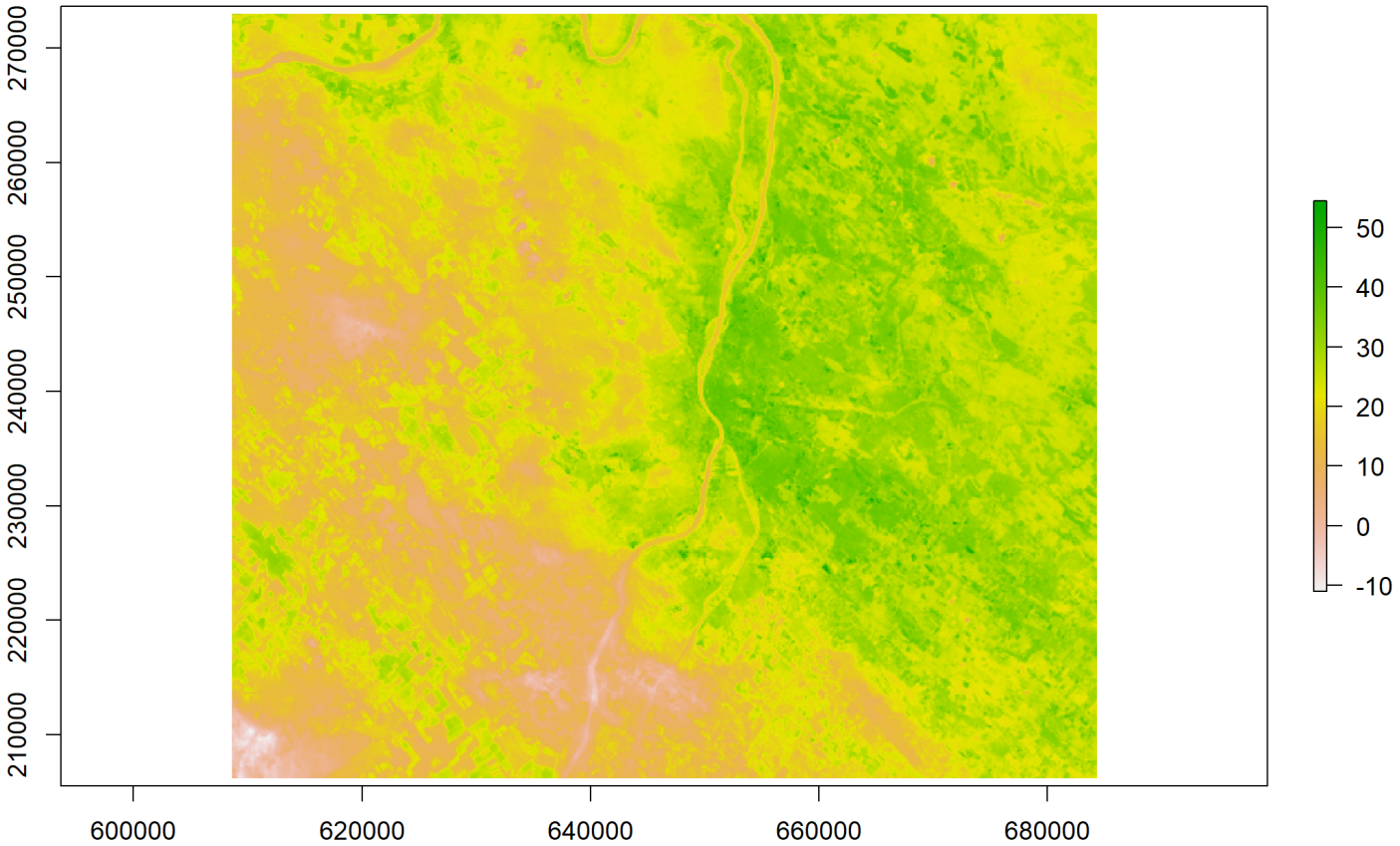
```
homerseklet_tablazatban <- read.table(file =
 "felszinhomerseklet.csv", sep = "", dec = ".", header =
 TRUE)
```

```
head(homerseklet_tablazatban)
```

```
     eov_y    eov_x homerseklet
1 608000.4 273614.8          NA
2 608086.1 273614.8          NA
3 608171.8 273614.8          NA
4 608257.5 273614.8          NA
5 608343.2 273614.8          NA
6 608428.9 273614.8          NA
```

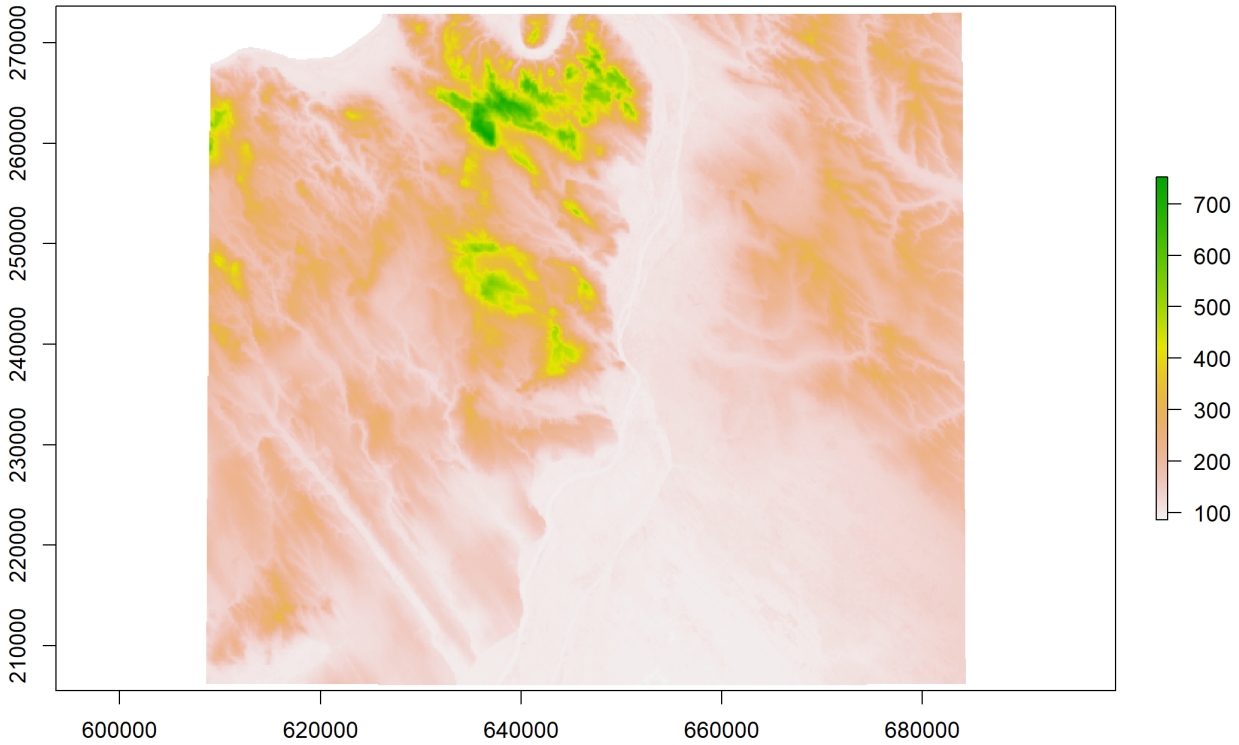
Raszter létrehozása koordinátát tartalmazó táblázatból

```
homerseklet <- rasterFromXYZ(xyz = homerseklet_tablazatban)
plot(homerseklet)
```



3. feladat (órai)

- Olvasd be “dem_tablazatban” néven a domborzatmodell.csv tagolt, fejlécet tartalmazó fájlt.
- Írd ki a képernyőre a beolvasott táblázat oszlopneveit.
- Hozz létre a táblázatból dem néven rasztert, majd jelenítsd meg.



3. feladat (órai) – megoldás

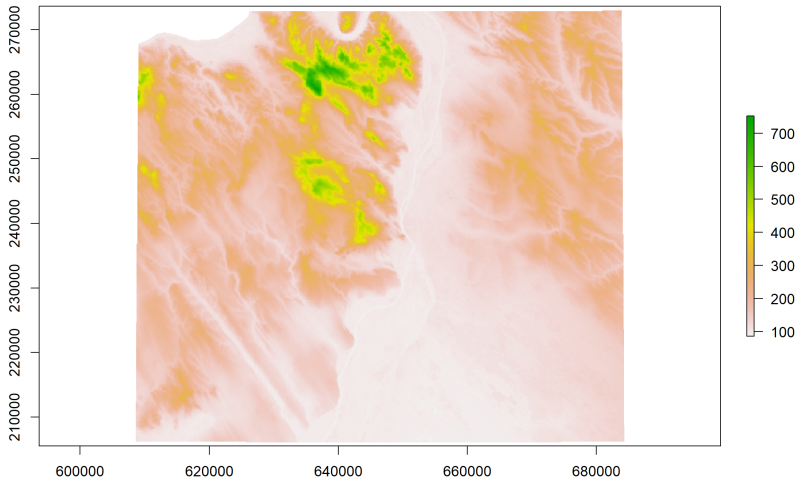
```
dem_tablazatban <- read.table(file =
 "domborzatmodell.csv", sep = "", dec = ".", header =
 TRUE)
```

```
colnames(dem_tablazatban)
```

```
[1] "eov_y"    "eov_x"    "magassag"
```

```
dem <- rasterFromXYZ(xyz = dem_tablazatban)
```

```
plot(dem)
```



Raszter mentése

Raszter menthető

- hagyományos raszterformátumokban (pl. geoTiff, NetCDF, ArcGIS ascii, KML) vagy a `raster` csomag saját formátumában a `writeRaster()`-rel
- RData fájlként a `save()`-vel

`writeFormats()`

- lekéri a `raster` csomag által kezelt formátumokat
- egy kétoszlopos táblázatot ad eredményül
- az első oszlopban szerepel a formátum rövid neve, amit a mentéskor használhatunk

Raszter mentése

`writeRaster(x, filename, format, overwrite = FALSE)`

- `x`: a mentendő `Raster*` objektum
- `filename`: a mentés helye
- `format`: a kezelt formátumok egyikének megnevezése (opcionális)
- ha hiányzik $\rightarrow$ a fájlnévből próbálja kitalálni, ha nem sikerül, saját formátumban ment
- `overwrite`: felülírja-e a meglévő fájlt (ha `FALSE` és létezik a fájl, hibát dob)
- nincs visszatérési értéke

Raszter mentése

```
writeFormats()
```

```
      name         long_name
 [1,] "raster"     "R-raster"
 [2,] "SAGA"       "SAGA GIS"
 [3,] "IDRISI"     "IDRISI"
 [4,] "IDRISIold"  "IDRISI (img/doc)"
 [5,] "BIL"        "Band by Line"
 [6,] "BSQ"        "Band Sequential"
 [7,] "BIP"        "Band by Pixel"
 [8,] "ascii"      "Arc ASCII"
 [9,] "CDF"        "NetCDF"
[10,] "AAIGrid"    "Arc/Info ASCII Grid"
```

Raszter mentése

Kiterjesztés alapján kitalálja, hogy geoTiff-ként kell mentenie:

```
dir.create("kimenet", showWarnings = FALSE, recursive =
 TRUE)
writeRaster(x = szamraszter, filename = paste0(getwd(),
 "/kimenet/szamraszter.tif"))
```

A `filename` paraméter teljes elérési utat vár (relatív elérési út esetén "`Dataset copy failed`" hibát kaphatunk).
A "`NOT UPDATED FOR PROJ >= 6`" figyelmeztetést - ha kapunk - figyelmen kívül hagyhatjuk.

Raszter mentése

Ha a kiterjesztés alapján nincs tippje a formátumra, a saját formátumában ment (gri/grd):

```
writeRaster(x = szamraszter, filename = paste0(getwd(),
 "/kimenet/szamraszter.bla"))
```

```
Warning in .getFormat(filename): extension .bla is
unknown. Using default format.
```

```
class       : RasterLayer
dimensions  : 8, 6, 48  (nrow, ncol, ncell)
resolution  : 0.1666667, 0.125  (x, y)
extent      : 0, 1, 0, 1  (xmin, xmax, ymin, ymax)
crs         : NA
source      : szamraszter.grd
names       : layer
values      : 1, 10  (min, max)
```

Raszter mentése

Ilyenkor inkább adjuk meg a `format` paramétert!

```
writeRaster(x = szamraszter2, filename = paste0(getwd(),
 "/kimenet/szamraszter2.bla"), format = "GTiff")
```

Ha már létezik a fájl, hibát kapunk:

```
writeRaster(x = szamraszter, filename = paste0(getwd(),
 "/kimenet/szamraszter.tif"))
```

```
Error: [writeStart] file exists. You can use
 'overwrite=TRUE' to overwrite it
```

Raszter mentése

Az `overwrite` paramétert kell ilyenkor `TRUE`-ra állítani:

```
writeRaster(x = szamraszter, filename = paste0(getwd(),
 "/kimenet/szamraszter.tif"), overwrite = TRUE)
```

Memóriába olvasás kényszerítése

Memóriában való tárolás lekérdezése

`inMemory(x)`

- logikai értéket ad vissza
- `TRUE`, ha a memóriában tárolja
- `FALSE`, ha külső fájlban tárolja
- R-ben létrehozott raszter is kerülhet ideiglenes fájlba
- külső fájlból beolvasott raszter is kerülhet memóriába

Memóriába olvasás kényszerítése

`readAll(object)`

- visszatérési értéke egy, az `object`-tel megegyező tartalmú raszter
- ami a memóriában van
- nem örökre!
- ha eleve a memóriában volt, akkor nem változik semmi
- `save()` előtt fontos lehet meghívni

Memóriába olvasás kényszerítése

```
inMemory(szamraszter)
```

```
[1] TRUE
```

```
inMemory(csapadek)
```

```
[1] FALSE
```

```
csapadek <- readAll(csapadek)
inMemory(csapadek)
```

```
[1] TRUE
```

Raszter mentése

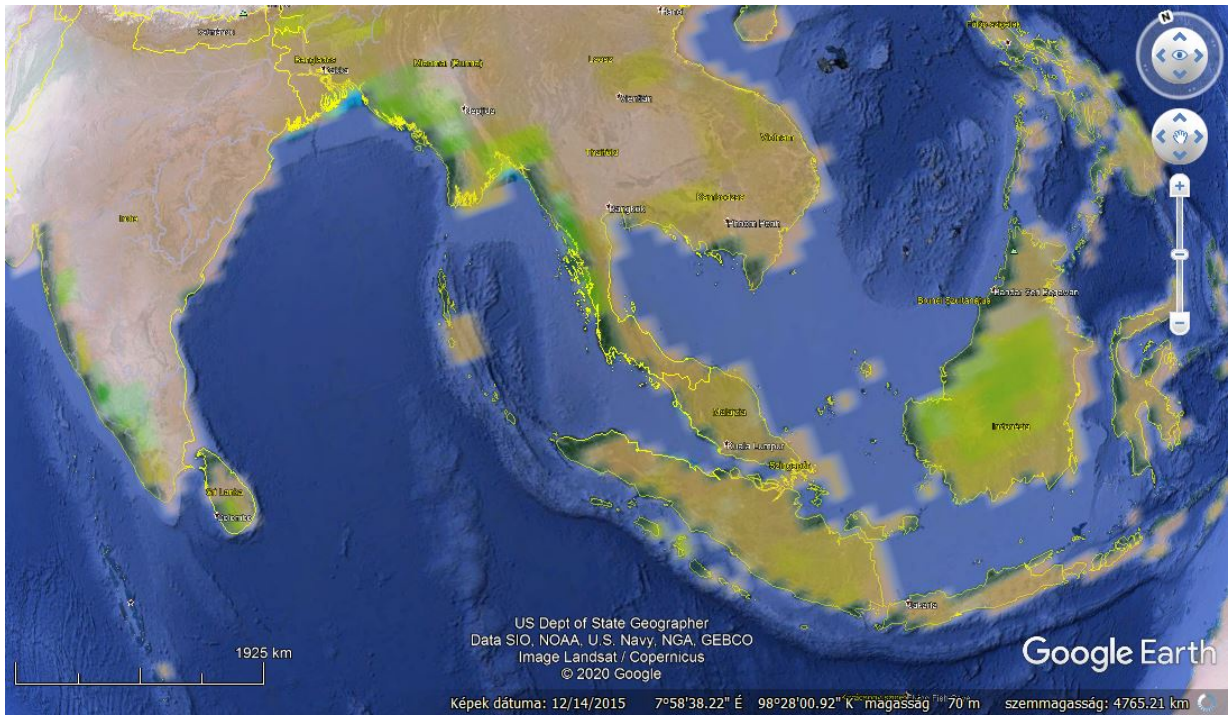
A `save()`/`load()` is jól használható.
Fontos: ha nincs a memóriában a raszter, akkor `save()` lényegileg csak egy hivatkozást ment.
Ha a fájl időközben megszűnik (töröltük/áthelyeztük/eleve ideiglenes volt), akkor a `load()` csak egy adattartalom nélküli torzót fog beolvasni.
→ Használjuk a `readAll()`-t a mentés előtt!

```
szamraszter <- readAll(szamraszter)
save(szamraszter, file = "kimenet/szamraszter.RData")
load("domborzatmodell.RData")
```

4. feladat (házi)

- Mentsd el a “csapadek” nevű `RasterLayer`t a “csapadek.kmz” fájlba, ha létezik, írd felül.
- Mentsd el a “szamraszter2” `RasterLayer`t a “szamraszter2.RData” fájlba (kényszerítsd előtte a memóriába olvasást, és ellenőrizd, hogy oda került-e).

Betöltve GoogleEarth-be így néz ki:



4. feladat (házi) – megoldás

```
writeRaster(x = csapadek, filename = paste0(getwd(),
 "/kimenet/csapadek.kmz"), overwrite = TRUE)
szamraszter2 <- readAll(szamraszter2)
inMemory(szamraszter2)
save(szamraszter2, file = "kimenet/szamraszter2.RData")
```

Section 3

Megjelenítés

Megjelenítés

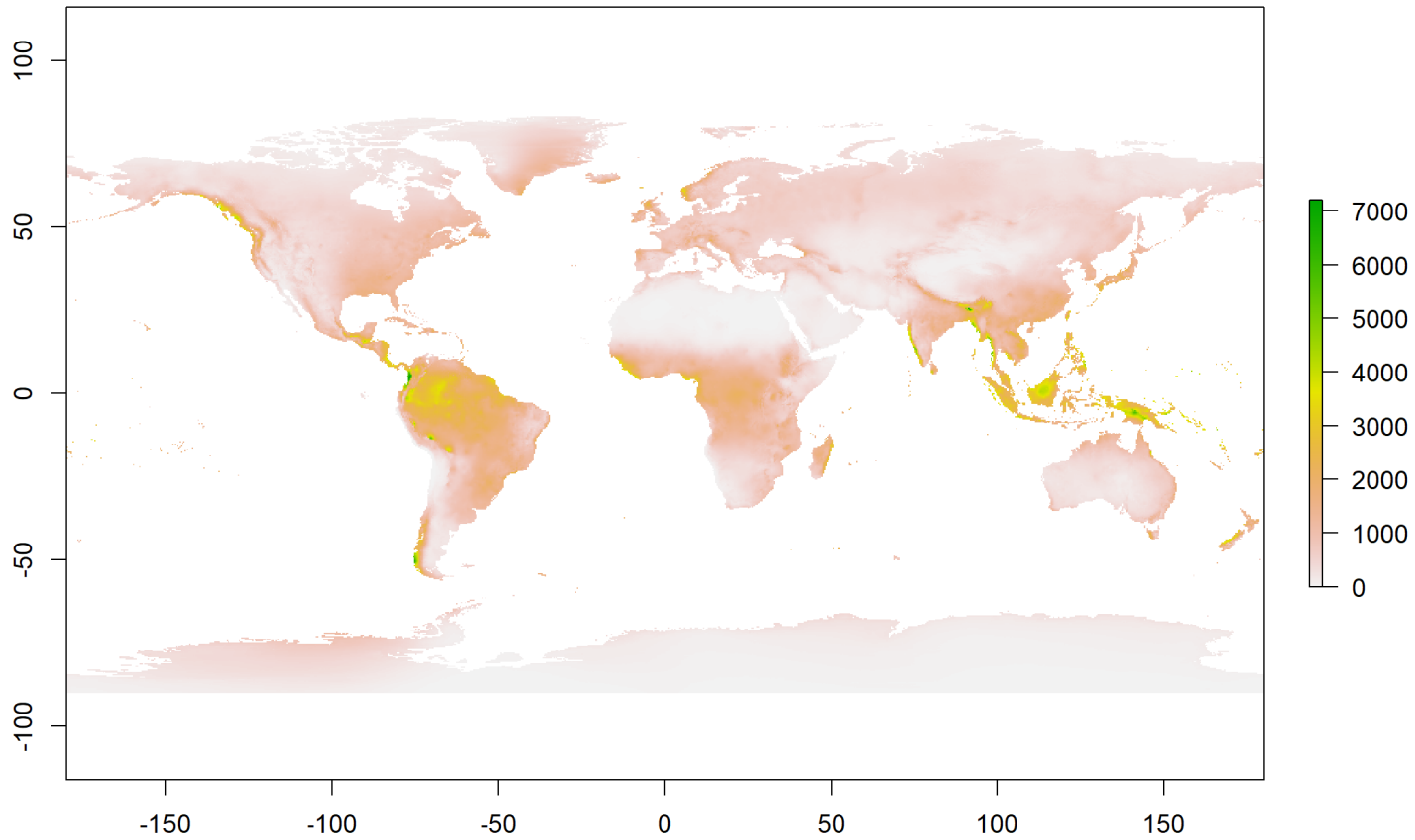
```
plot(x, maxpixels = 500000, col =
rev(terrain.colors(255)), colNA = "transparent", alpha
= 1, interpolate = FALSE, legend = TRUE, axes = TRUE)
```

- `x`: megjelenítendő `Raster*`
- `maxpixels`: az ábrázoláshoz maximum hány cellát mintavételezzen a raszterből? (Nagy raszterek esetén jól jön...)
- `col`: színskála
- `colNA`: az ismeretlen értékű cellák színe
- `alpha`: átlátszatlanság (0–1, ahol 0 a teljesen átlátszó)
- `interpolate`: elkenje-e a cellák határát (interpoláljon-e)
- `legend`: megjelenítse-e a jelmagyarázatot
- `axes`: megjelenítse-e a tengelytüskéket koordinátákkal

Természetesen a többi, jól ismert paraméter (`main`, `xlab`, `ylab`, `xlim`, `ylim` stb.) is használható.

Megjelenítés

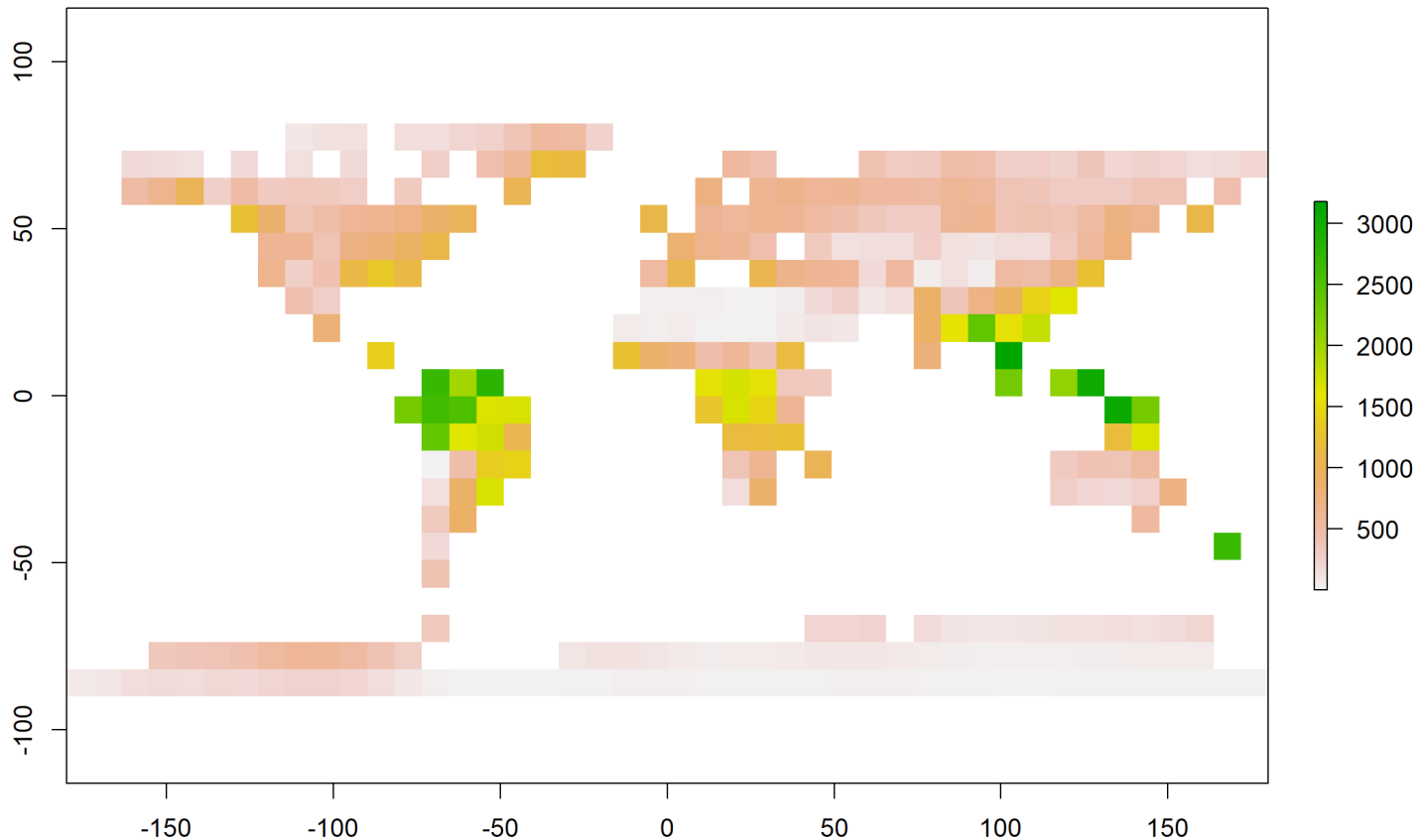
```
plot(x = csapadek)
```



Megjelenítés

Csak ezer cellát veszünk:

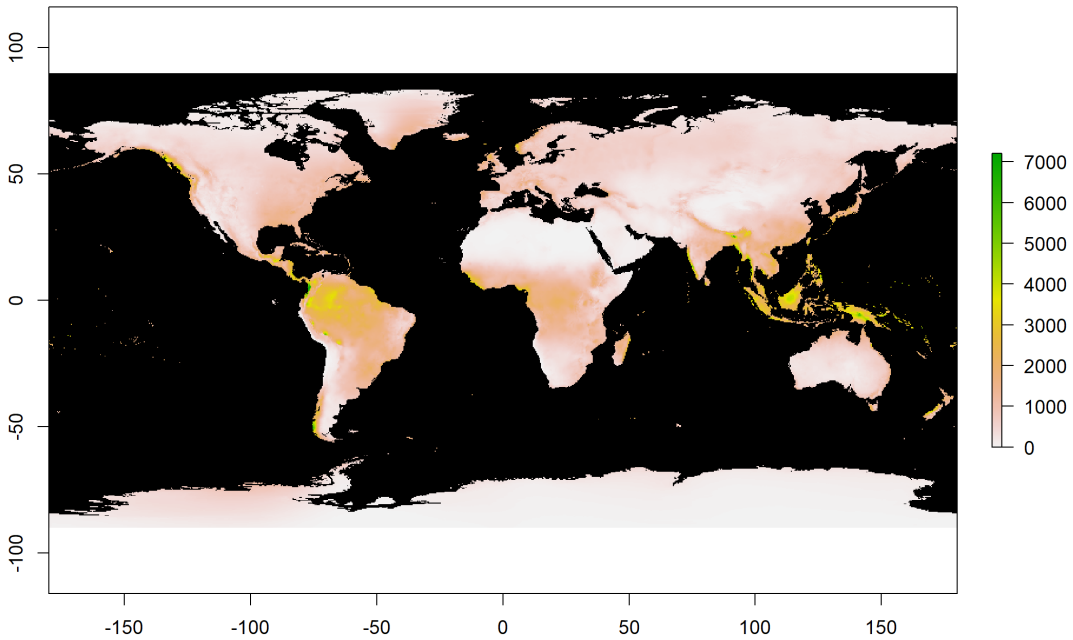
```
plot(x = csapadek, maxpixels = 1000)
```



Megjelenítés

Ismeretlen értékek feketével:

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

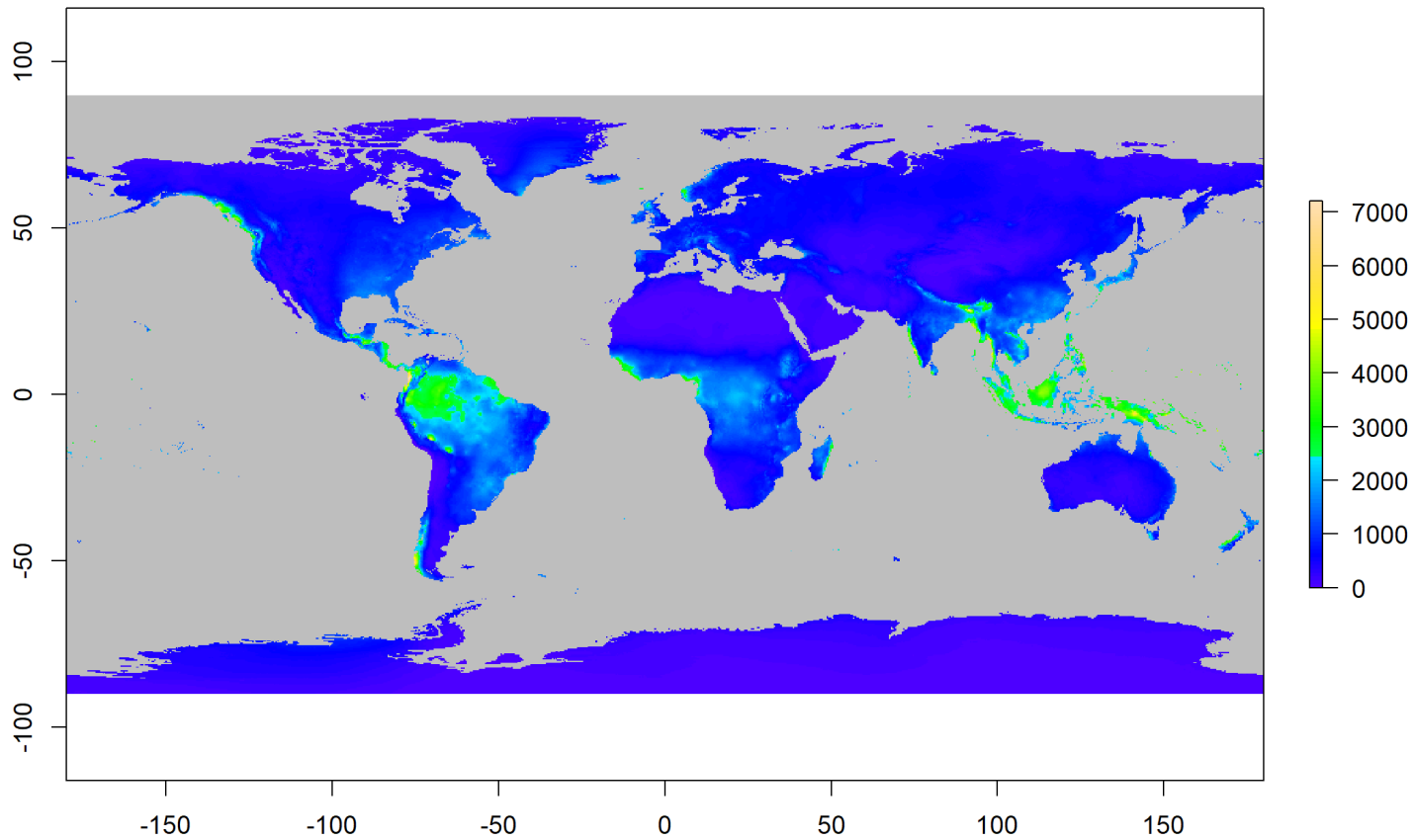


Az alapértelmezett átlátszó szín csak akkor praktikus, ha van a raszter alatt más réteg, vagy ha a színskála nem tartalmaz fehér színt.

Megjelenítés

A `topo.colors` 100 elemű színskálája:

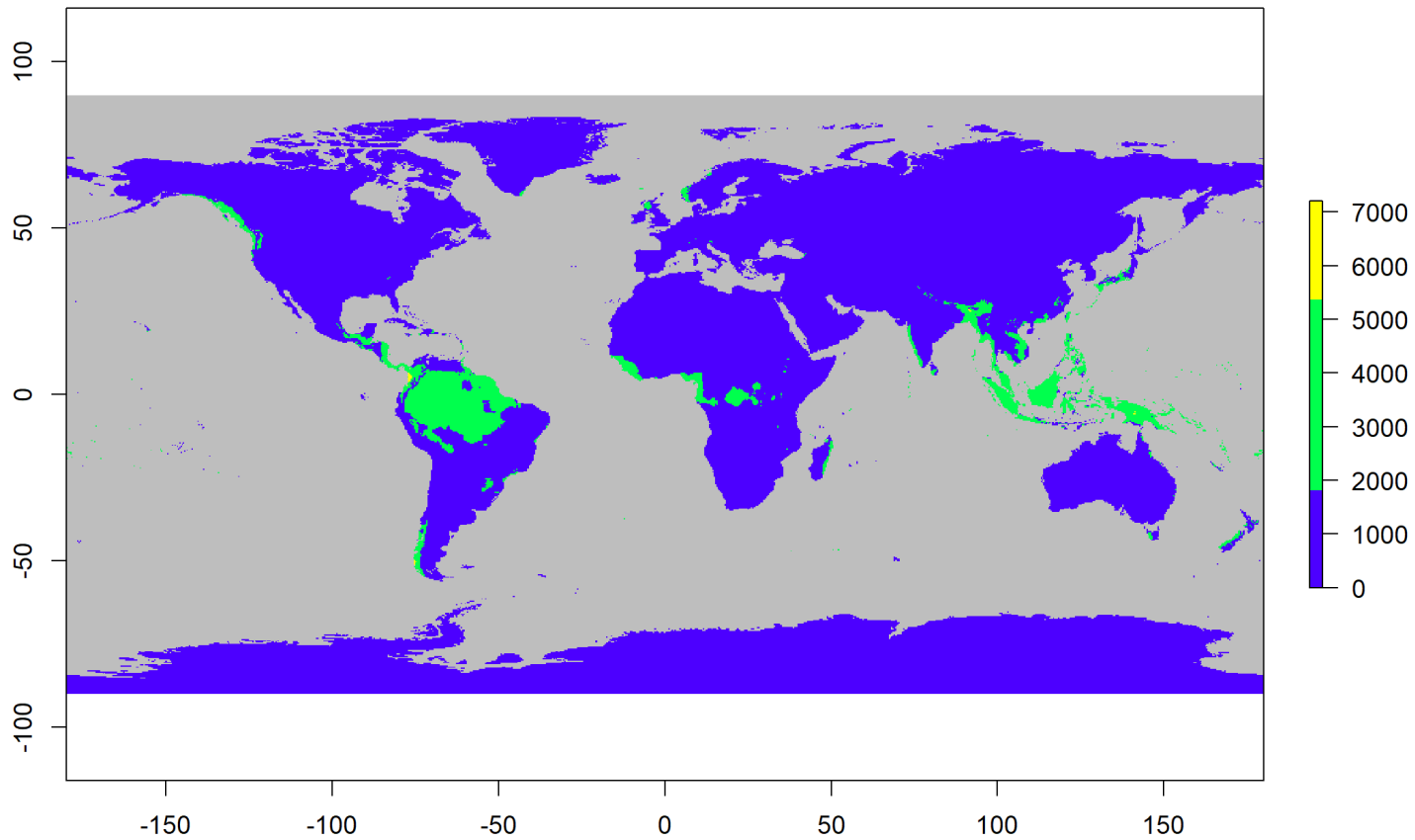
```
plot(x = csapadek, colNA = "gray", col = topo.colors(100))
```



Megjelenítés

Ugyanez a színskála 3 színnel:

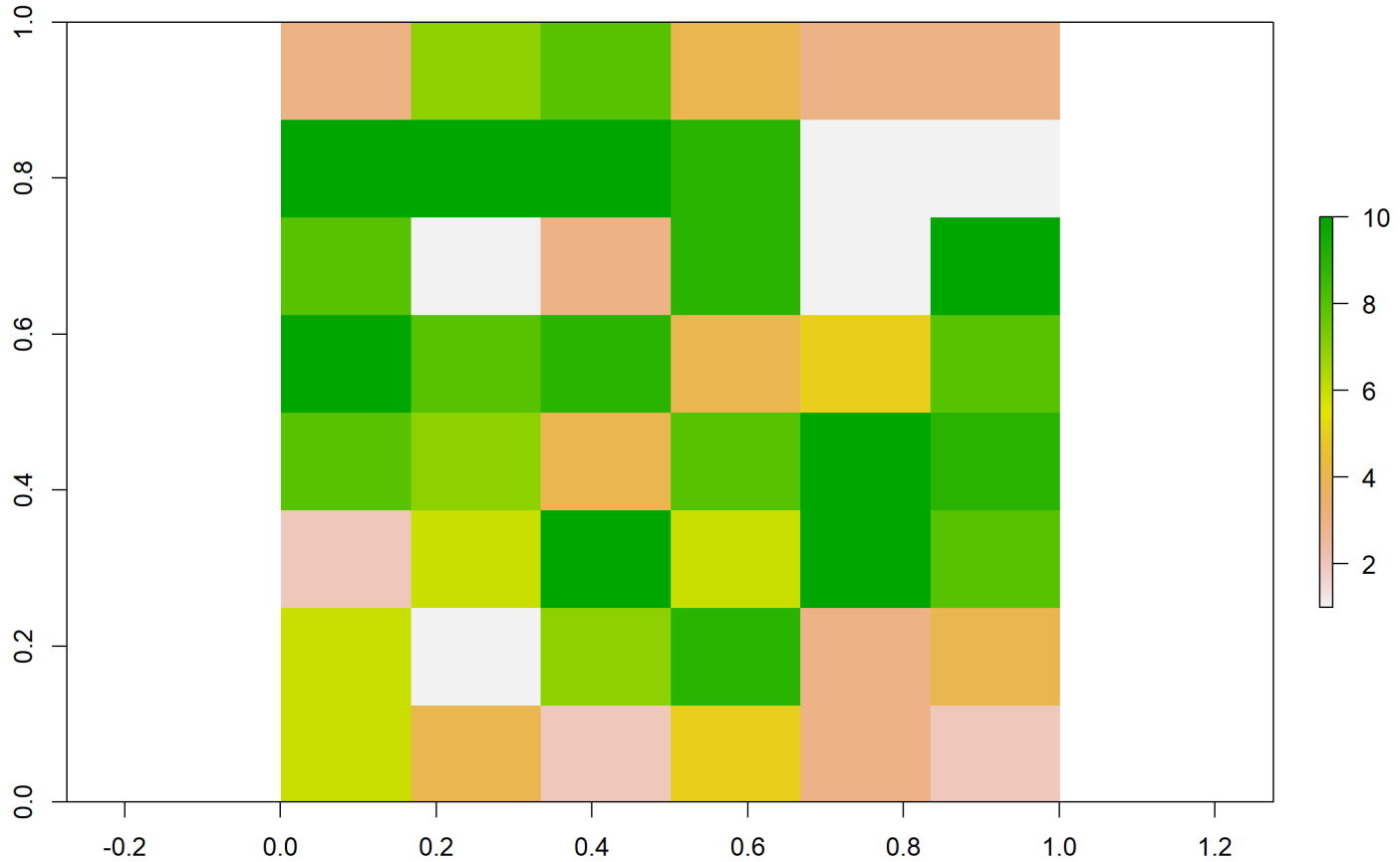
```
plot(x = csapadek, colNA = "gray", col = topo.colors(3))
```



Megjelenítés

Nézzünk meg egy kisebb rasztert, ahol a cellák jól elkülönülnek!

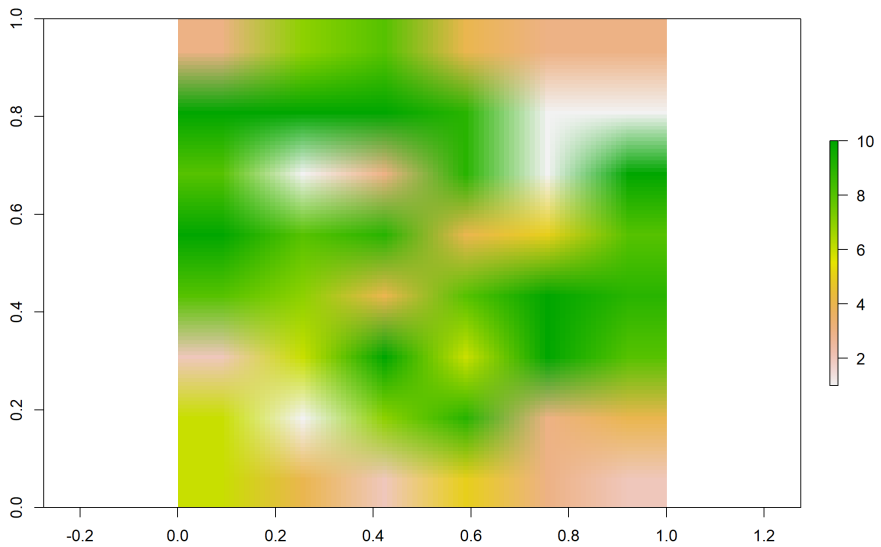
```
plot(x = szamraszter)
```



Megjelenítés

Elkenés/interpolálás:

```
plot(x = szamraszter, interpolate = TRUE)
```

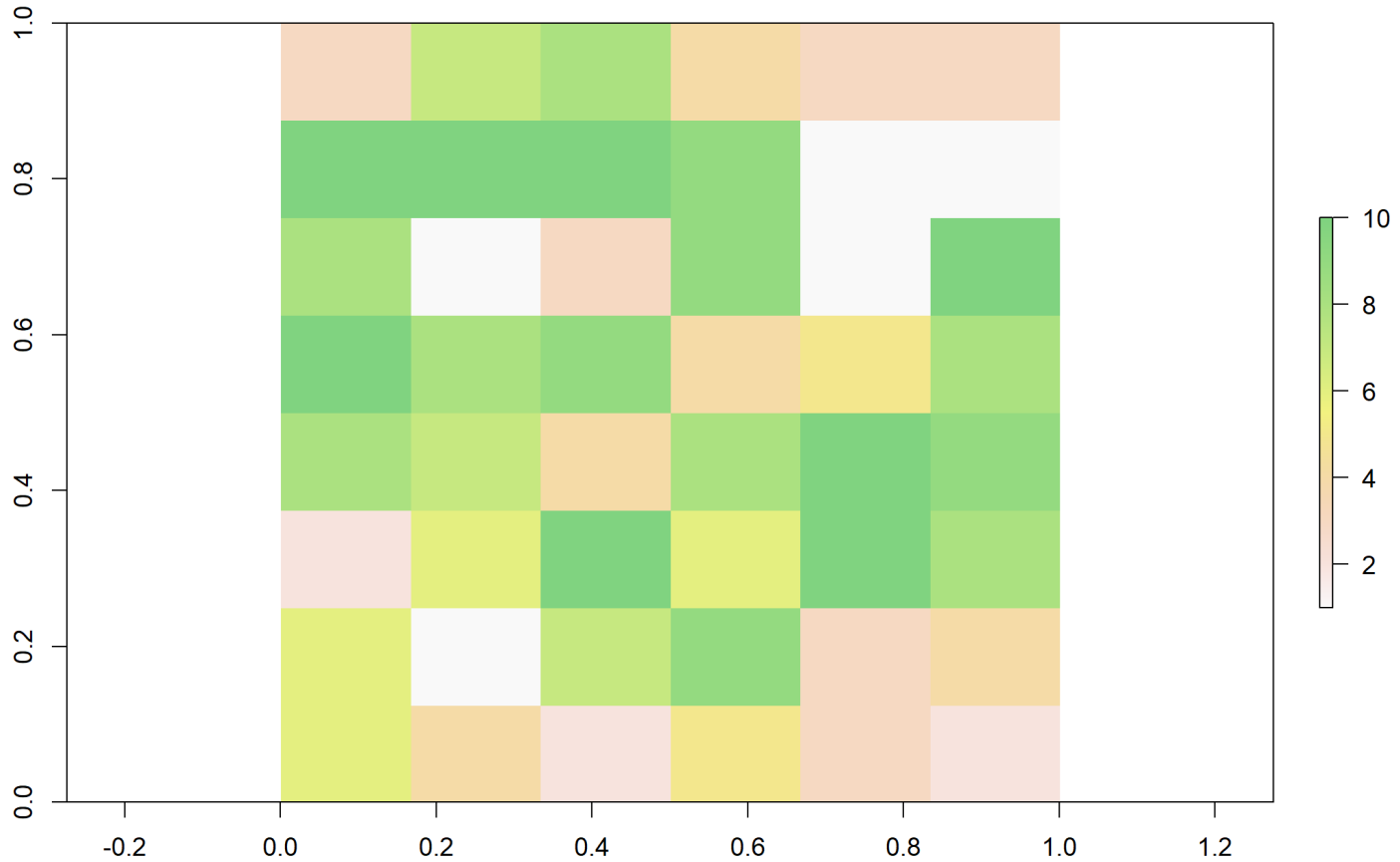


Ez csak a megjelenítést befolyásolja, az eredeti raszter változatlan.
A valódi interpoláció kicsit másképp működik (és jobban testreszabható).

Megjelenítés

50%-os áttetszőség:

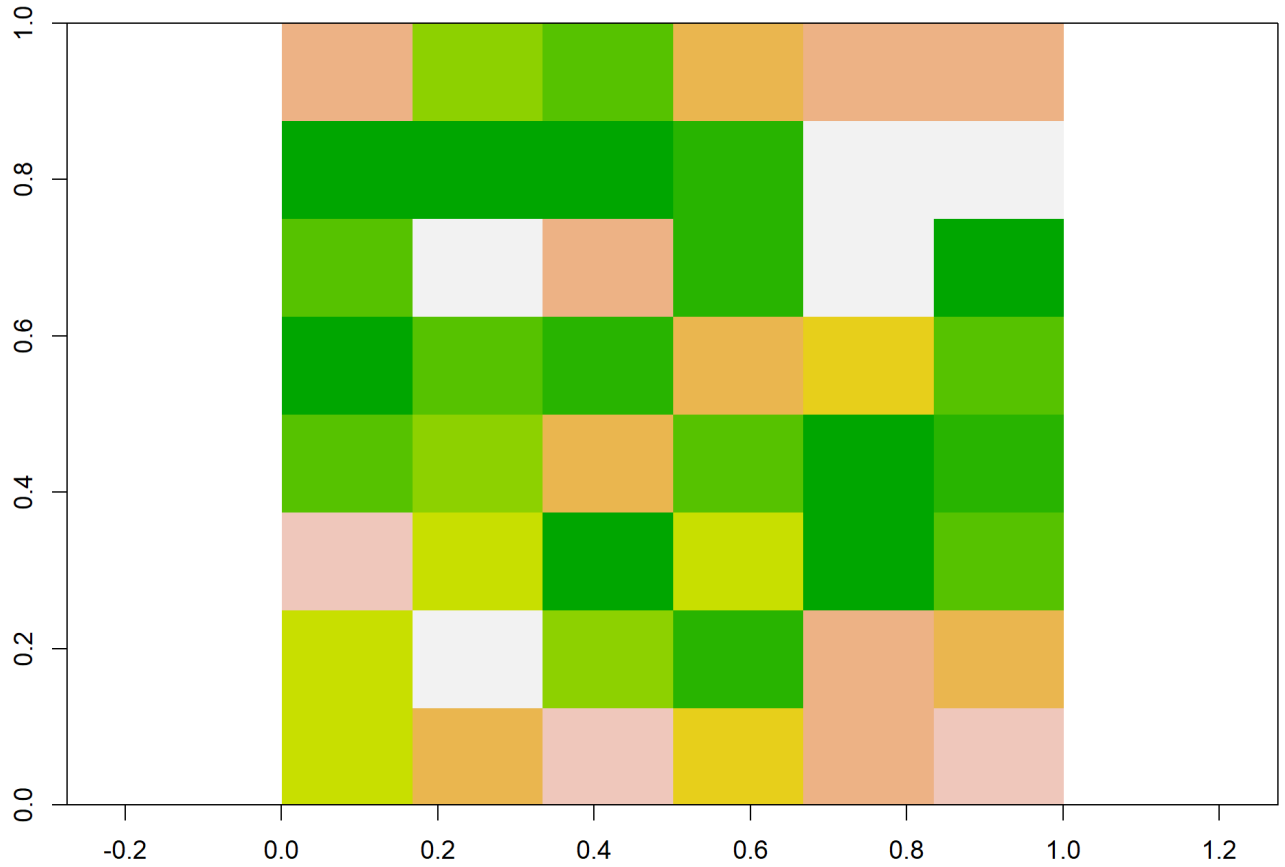
```
plot(x = szamraszter, alpha = 0.5)
```



Megjelenítés

Jelmagyarázat eltüntetése:

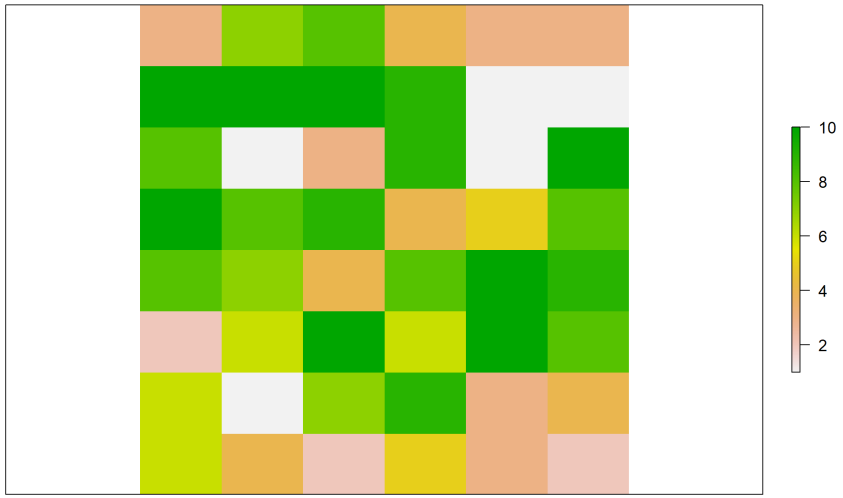
```
plot(x = szamraszter, legend = FALSE)
```



Megjelenítés

Koordináták és tengelytüskék eltüntetése:

```
plot(x = szamraszter, axes = FALSE)
```



A bemutatott paraméterek természetesen tetszőlegesen kombinálhatóak.

Megjelenítés

`plotRGB(x, r = 1, g = 2, b = 3)`

- színes kompozitképet jelenít meg három réteg (színcsatorna) alapján
- `x`: megjelenítendő többrétegű (>=3) raszter
- `r`: piros színcsatornát tartalmazó réteg sorszáma
- `g`: zöld színcsatornát tartalmazó réteg sorszáma
- `b`: kék színcsatornát tartalmazó réteg sorszáma

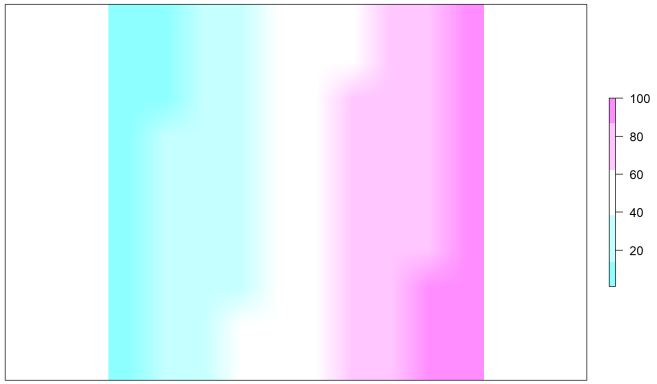
Megjelenítés

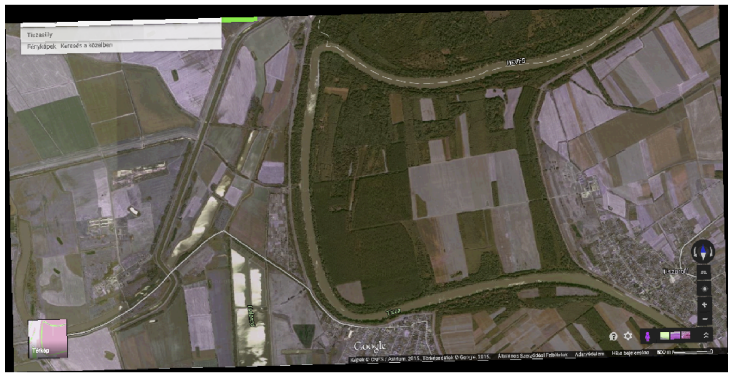
Az alapértelmezett `r`, `g` és `b` paraméterek nekünk most tökéletesek lesznek:

```
plotRGB(x = ortofoto)
```

5. feladat (órai)

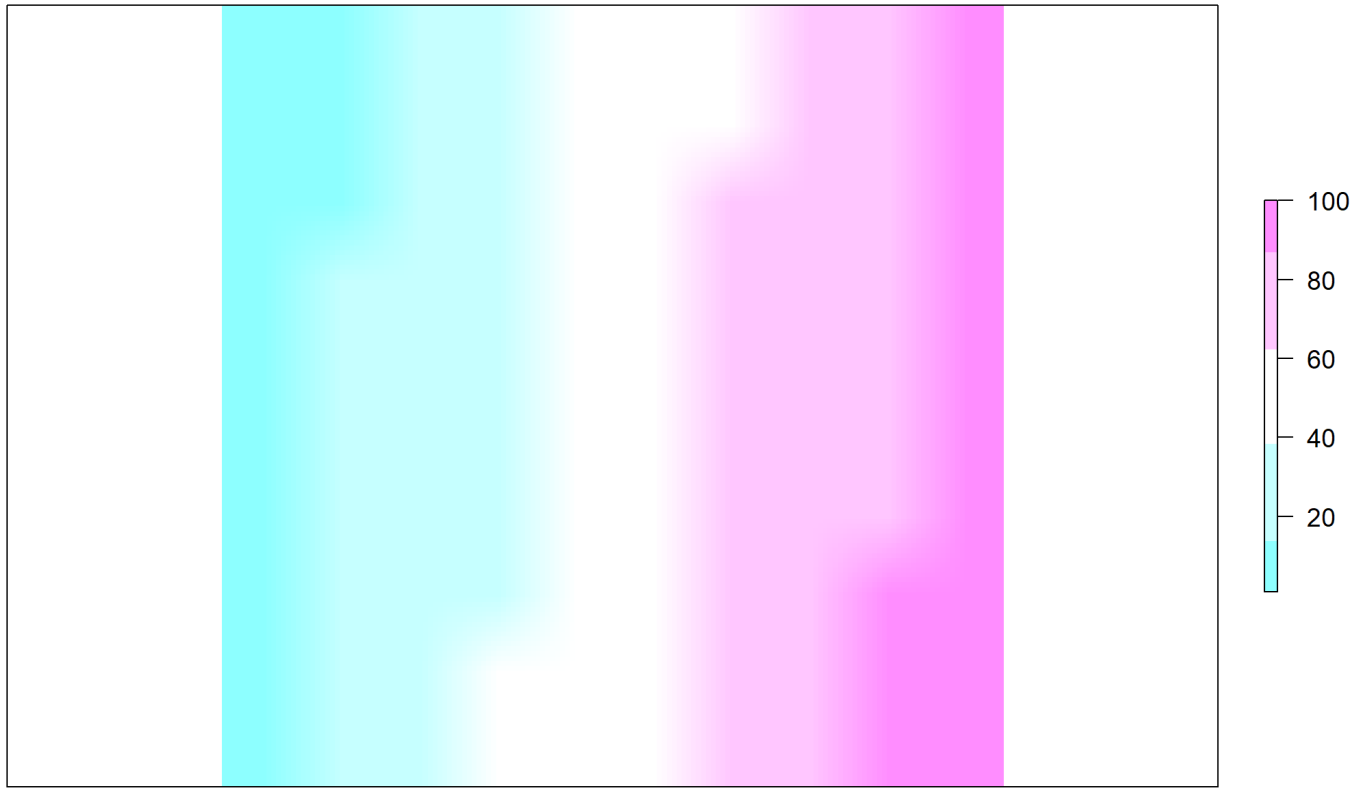
- Ábrázold a “szamraszter2” nevű `RasterLayert`
  - a `cm.colors` paletta öt színével,
  - 90%-os átlátszatlansággal,
  - tengelytüskék és koordináta-feliratok nélkül,
  - a cellahatárokat elkenve.
- Ábrázold az “ortofoto” nevű rasztert színes kompozitképként úgy, hogy
  - a piros színcsatornát a 2.,
  - a zöldet a 3.,
  - a kéket pedig az 1. rétegből veszed.





5. feladat (órai) – megoldás

```
plot(x = szamraszter2, interpolate = TRUE, col =
 cm.colors(5), alpha = 0.9, axes = FALSE)
```



5. feladat (órai) – megoldás

```
plotRGB(x = ortofoto, r = 2, g = 3, b = 1)
```

Raszter és vektor együttes megjelenítése

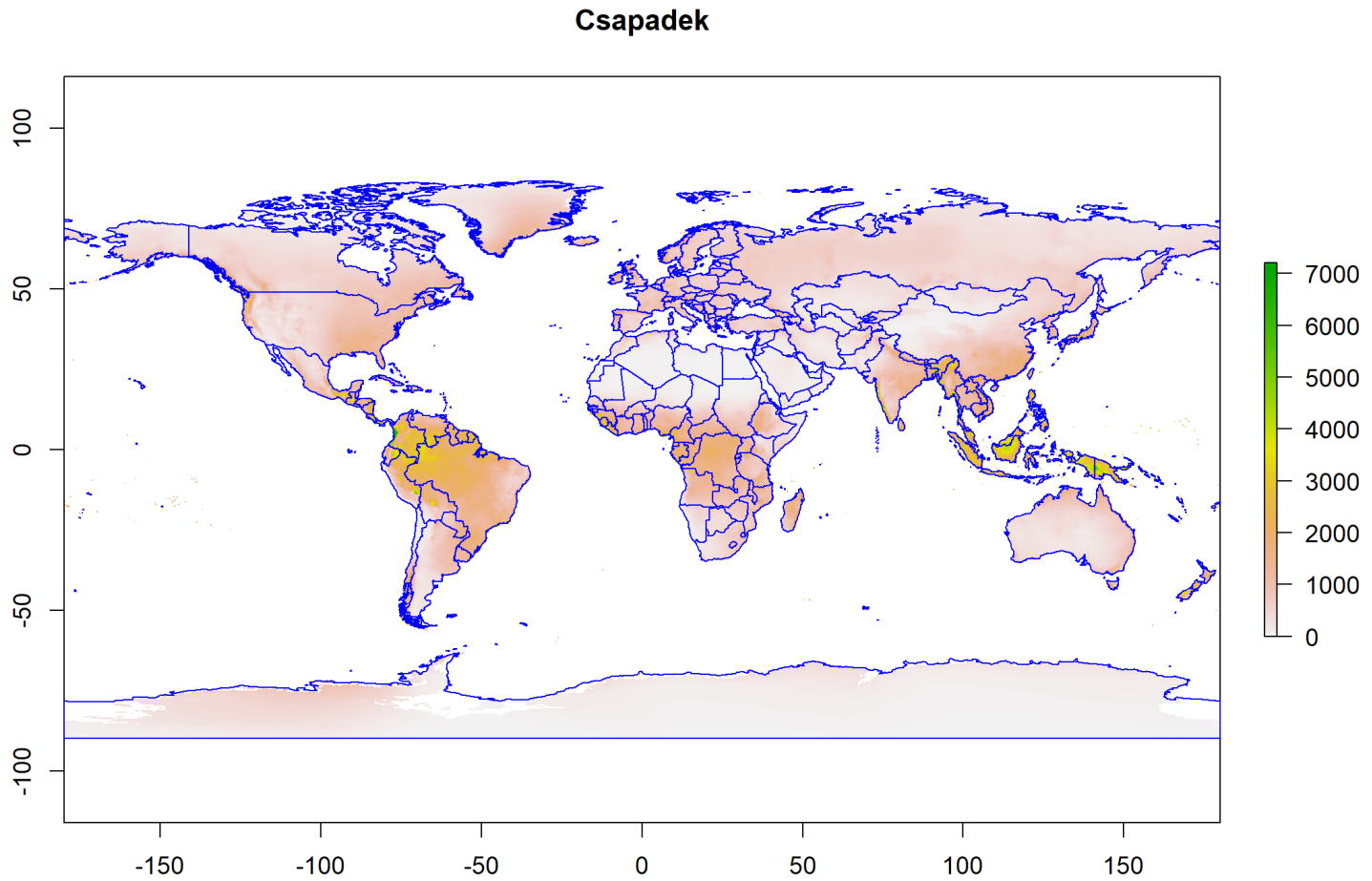
A raszterek és vektorok kényelmesen egymásra vetíthetőek.
A vektorok egymásra vetítésénél megszokott módon:

- az első réteg határozza meg a kivágatot
- sorrend számít (kitakarás miatt)
- azonos vetületben legyenek!
- `add = TRUE` és `reset = FALSE` paraméterek használandóak (utóbbi abban az esetben, ha tulajdonság szerint színezett Simple Features az első réteg)

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

Raszter és vektor együttes megjelenítése

```
plot(x = csapadek, main = "Csapadek")
plot(orszagok_osszes_geometria, border = "blue", add = TRUE)
```



Raszter és vektor együttes megjelenítése

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

Természetesen `plotRGB()`-vel is működik:

```
plotRGB(x = ortofoto)
plot(utak_geometria, col = "red", lwd = 3, add = TRUE)
plot(st_transform(x = st_geometry(folyok), crs = 23700),
 col = "blue", lwd = 5, add = TRUE)
```

Raszter és vektor együttes megjelenítése

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

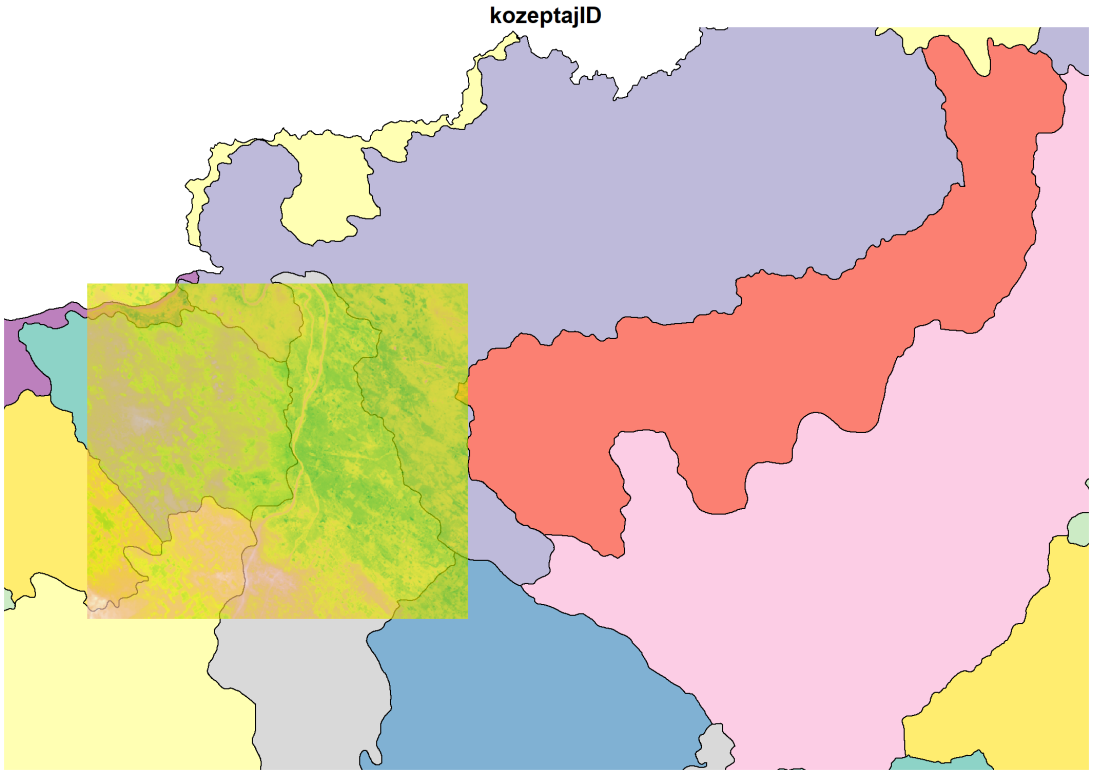
A raszter kerülhet fölülre is, de akkor

- vegyük le a jelmagyarázatot
- érdemes átlátszóbbá tenni, hogy az alatta lévő vektor is látszódjon

```
plot(kozeptajak, xlim = c(600000, 800000), ylim =
 c(200000, 300000), reset = FALSE)
plot(felszinhomerseklet, alpha = 0.7, legend = FALSE, add
 = TRUE)
```

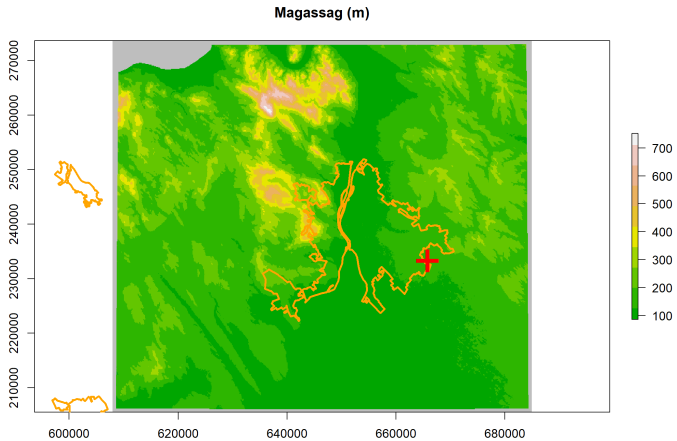
Raszter és vektor együttes megjelenítése

```
plot(kozeptajak, xlim = c(600000, 800000), ylim =
 c(200000, 300000), reset = FALSE)
plot(felszinhomerseklet, alpha = 0.7, legend = FALSE, add
 = TRUE)
```



6. feladat (órai)

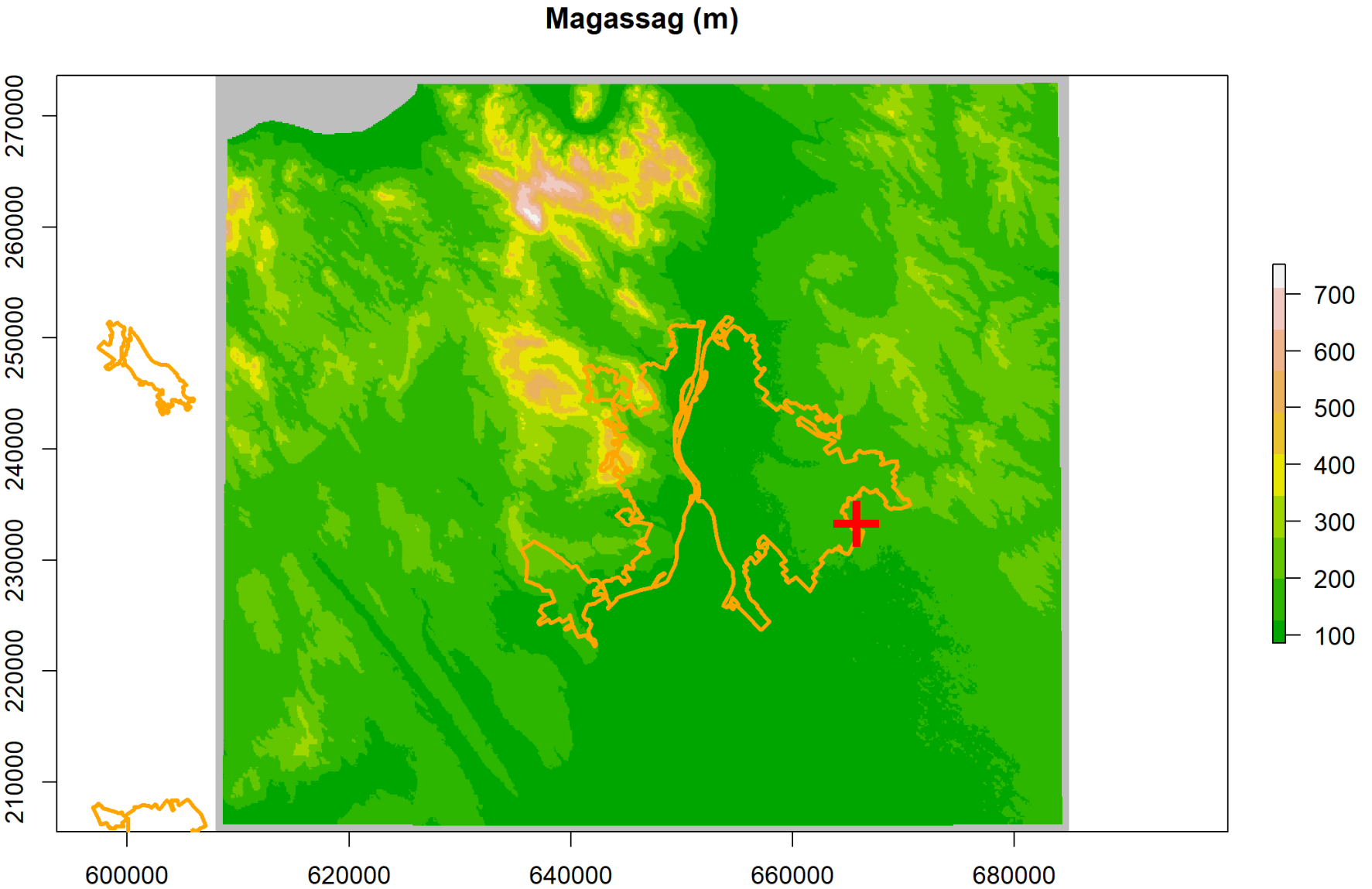
- Olvasd be a domborzatmodell.RData, varosok_geometria.RData és repterek.RData fájlokat.
- Jelenítsd meg a domborzatmodellt
  - a `terrain.colors` paletta 10 színével
  - úgy, hogy az ismeretlen értékek szürke színt kapjanak.
  - Az ábra címe legyen “Magassag (m)”.
- Add hozzá a városok geometriáját
  - narancssárga, háromszoros vastagságú körvonallal,
- és a repterek geometriáját
  - piros, négyszeres méretű +-jellel.



6. feladat (órai) – megoldás

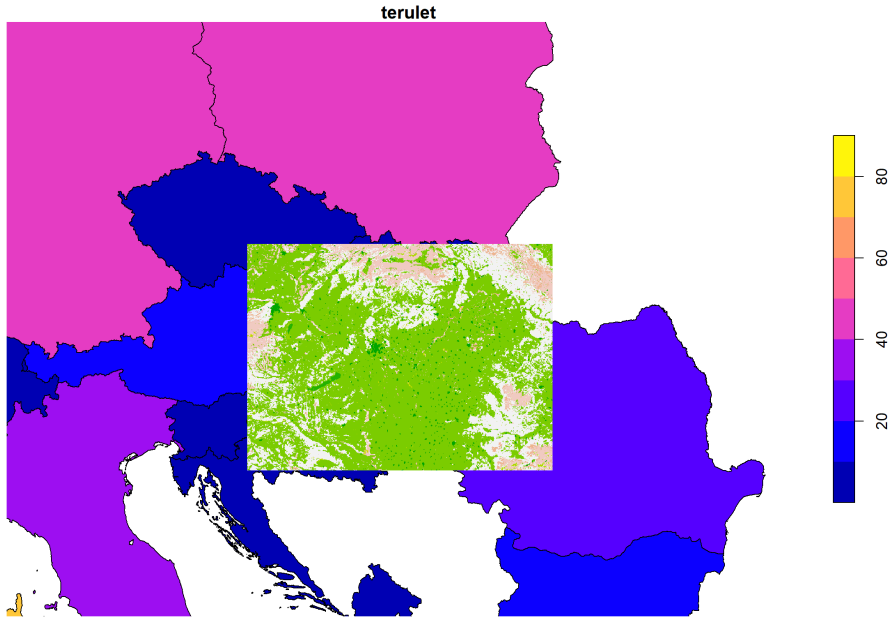
```
load("domborzatmodell.RData")
load("varosok_geometria.RData")
load("repterek.RData")
plot(domborzatmodell, colNA = "gray", main = "Magassag
 (m)", col = terrain.colors(10))
plot(varosok_geometria, border = "orange", lwd = 3, add =
 TRUE)
plot(st_geometry(repterek), col = "red", cex = 4, pch =
 "+" , add = TRUE)
```

6. feladat (órai) – megoldás



7. feladat (házi)

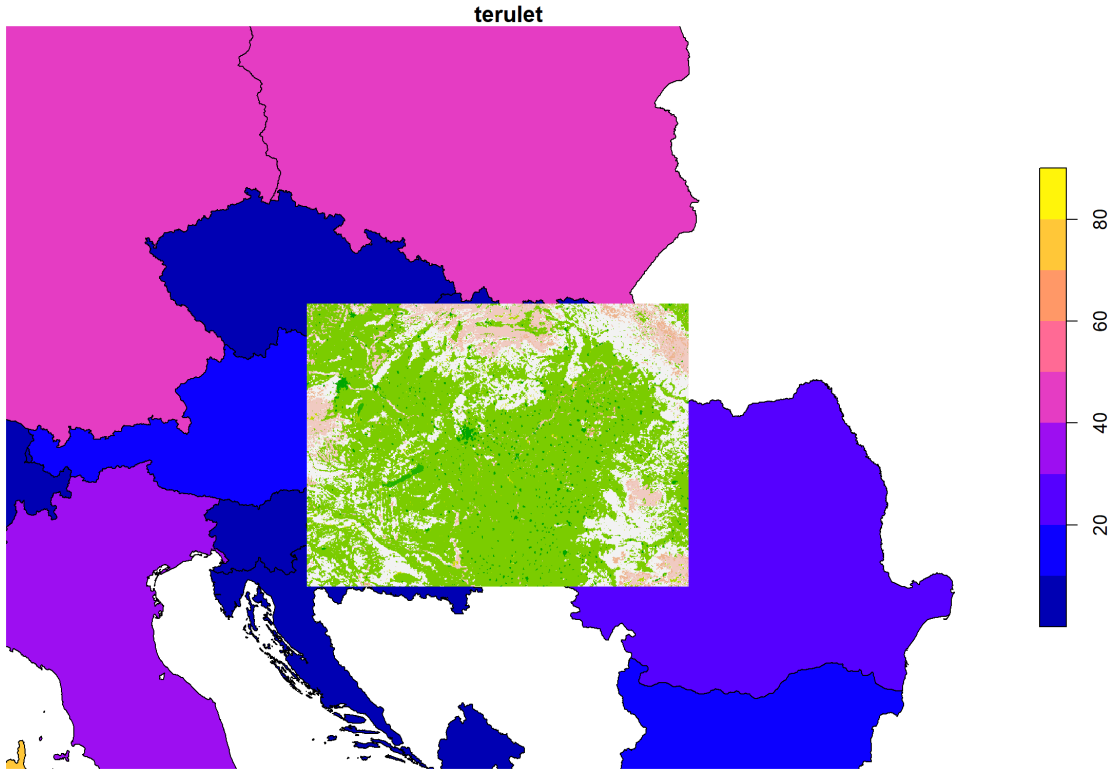
- Olvasd be az orszag_eu.RData fájlt.
- Jelenítsd meg az EU országait
  - a “terulet” oszlop alapján színezve,
  - a 15°–25° keleti hosszúsági és 43°–53° északi szélességi koordináták között.
  - Ne felejtsd el a képvásznat nyitva hagyni, hogy hozzá tudd adni a második réteget.
- Ábrázold rajta a felszínborítást (amit az 1. feladatban töltöttél be), jelmagyarázat nélkül.



7. feladat (házi) – megoldás

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

```
plot(orszag_eu[, "terulet"], xlim = c(15, 25), ylim =
  c(43, 53), reset = FALSE)
plot(felszinboritas, add = TRUE, legend = FALSE)
```



Section 4

Raszter jellemzői

Raszter jellemzői

- sorok számának lekérése: `nrow(x)`
- oszlopok számának lekérése: `ncol(x)`
- cellák számának lekérése: `ncell(x)`
- rétegek számának lekérése: `nlayers(x)`
- rétegnevek lekérése: `names(x)`
- rétegnevek módosítása: `names(x) <- value`
- befoglaló doboz lekérése: `extent(x)`
- vetület lekérése: `projection(x)`

Raszter jellemzői

```
nrow(felszinhomerseklet)
```

```
[1] 541
```

```
ncol(felszinhomerseklet)
```

```
[1] 898
```

```
ncell(felszinhomerseklet)
```

```
[1] 485818
```

Ellenőrizzük, jól számolt-e!

```
ncell(felszinhomerseklet) == nrow(felszinhomerseklet) *
 ncol(felszinhomerseklet)
```

```
[1] TRUE
```

Raszter jellemzői

```
nlayers(felszinhomerseklet)
```

```
[1] 1
```

```
nlayers(ortofoto)
```

```
[1] 3
```

A rétegszám lekérhető `RasterLayer`re is, de ritkán van értelme...

Raszter jellemzői

```
names(felszinhomerseklet)
```

```
[1] "felszinhomerseklet"
```

```
names(dem)
```

```
[1] "magassag"
```

```
names(ortofoto)
```

```
[1] "ortofoto_1" "ortofoto_2" "ortofoto_3"
```

```
names(ortofoto) <- c("piros", "zold", "kek")
names(ortofoto)
```

```
[1] "piros" "zold"  "kek"
```

Raszter jellemzői

```
extent(felszinboritas)
```

```
class       : Extent
xmin        : 15.6116
xmax        : 23.9241
ymin        : 45.25447
ymax        : 49.37054
```

```
projection(felszinboritas)
```

```
[1] "+proj=longlat +datum=WGS84 +no_defs"
```

8. feladat (házi)

- Jelenítsd meg a domborzatmodell legfontosabb jellemzőit a képernyőn a `print()` függvénnyel.
- A kiírt információk közül külön is kérd le az 1., 2. 4., 5. és 7. sorok tartalmát.

```
class       : RasterLayer
dimensions  : 541, 898, 485818  (nrow, ncol, ncell)
resolution  : 85.7, 126  (x, y)
extent  : 607957.6, 684916.2, 205511.8, 273677.8 (xmin,
 xmax, ymin, ymax)
crs         : NA
source      : memory
names       : dem_wgs84_Mo
values      : 86, 752  (min, max)
```

8. feladat (házi) – megoldás

```
print(domborzatmodell)
```

```
class(domborzatmodell)
```

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

```
nrow(domborzatmodell)
```

```
[1] 541
```

```
ncol(domborzatmodell)
```

```
[1] 898
```

```
ncell(domborzatmodell)
```

```
[1] 485818
```

8. feladat (házi) – megoldás

```
extent(domborzatmodell)
```

```
class       : Extent
xmin        : 607957.6
xmax        : 684916.2
ymin        : 205511.8
ymax        : 273677.8
```

```
projection(domborzatmodell)
```

```
[1] NA
```

```
names(domborzatmodell)
```

```
[1] "dem_wgs84_Mo"
```

Section 5

Raszter rétegeinek és celláinak kezelése

Raszterrétegek elérése

Raszterrétegek elérése

- `[[]]`: sorszámmal vagy névvel, akár több réteg is
- `$`: névvel (idézőjelek nélkül is), csak egy réteg
- `subset(x, subset, drop = TRUE)`: sorszámmal vagy névvel, akár több réteg is, `drop` paraméter befolyásolja, hogy egy réteg esetén egyszerűsödjön-e `RasterLayerré`

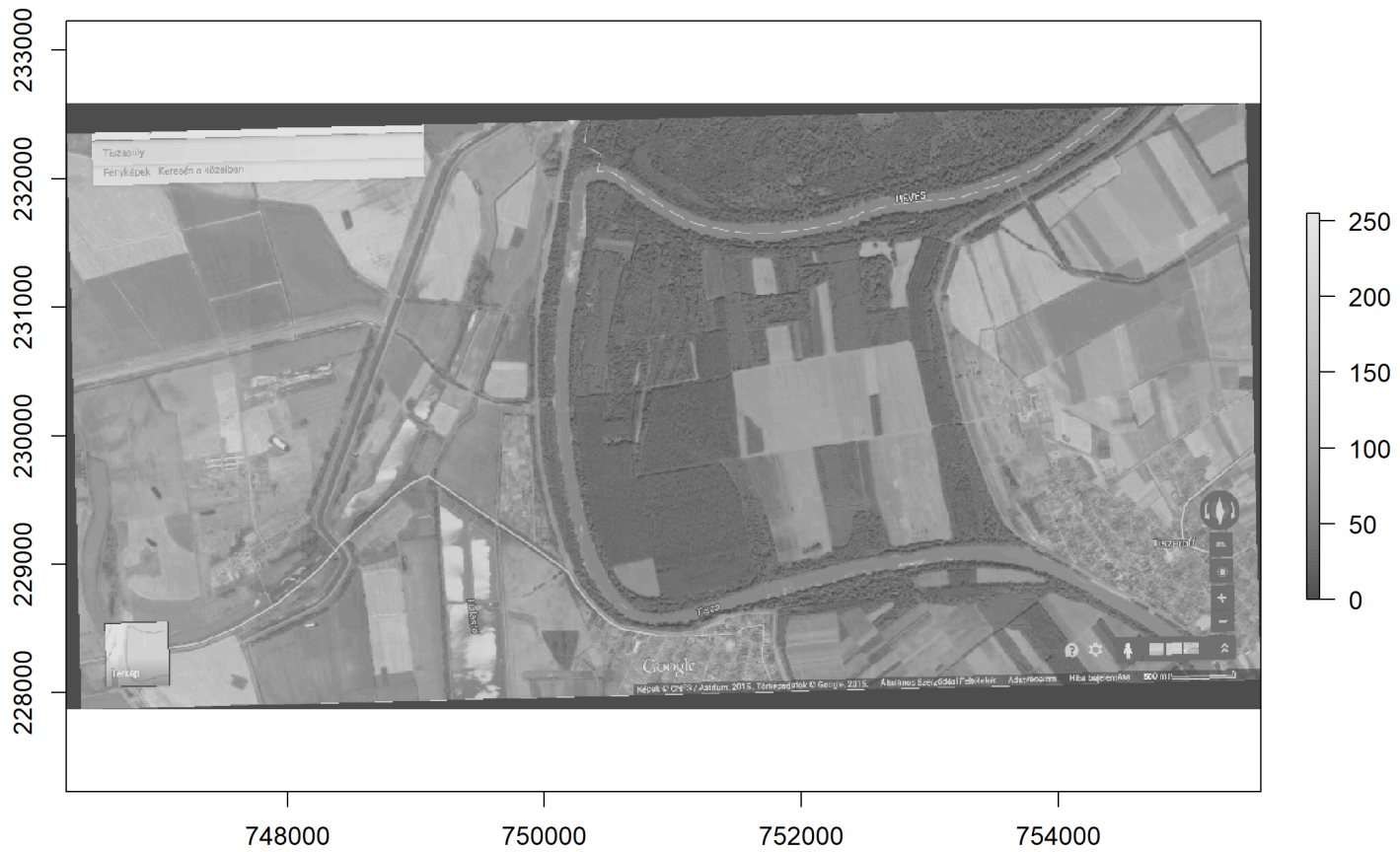
Raszterrétegek elérése

```
piros_szincsatorna <- ortofoto[[1]]
class(piros_szincsatorna)
```

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

Raszterrétegek elérése

```
plot(piros_szincsatorna, col = gray.colors(100))
```



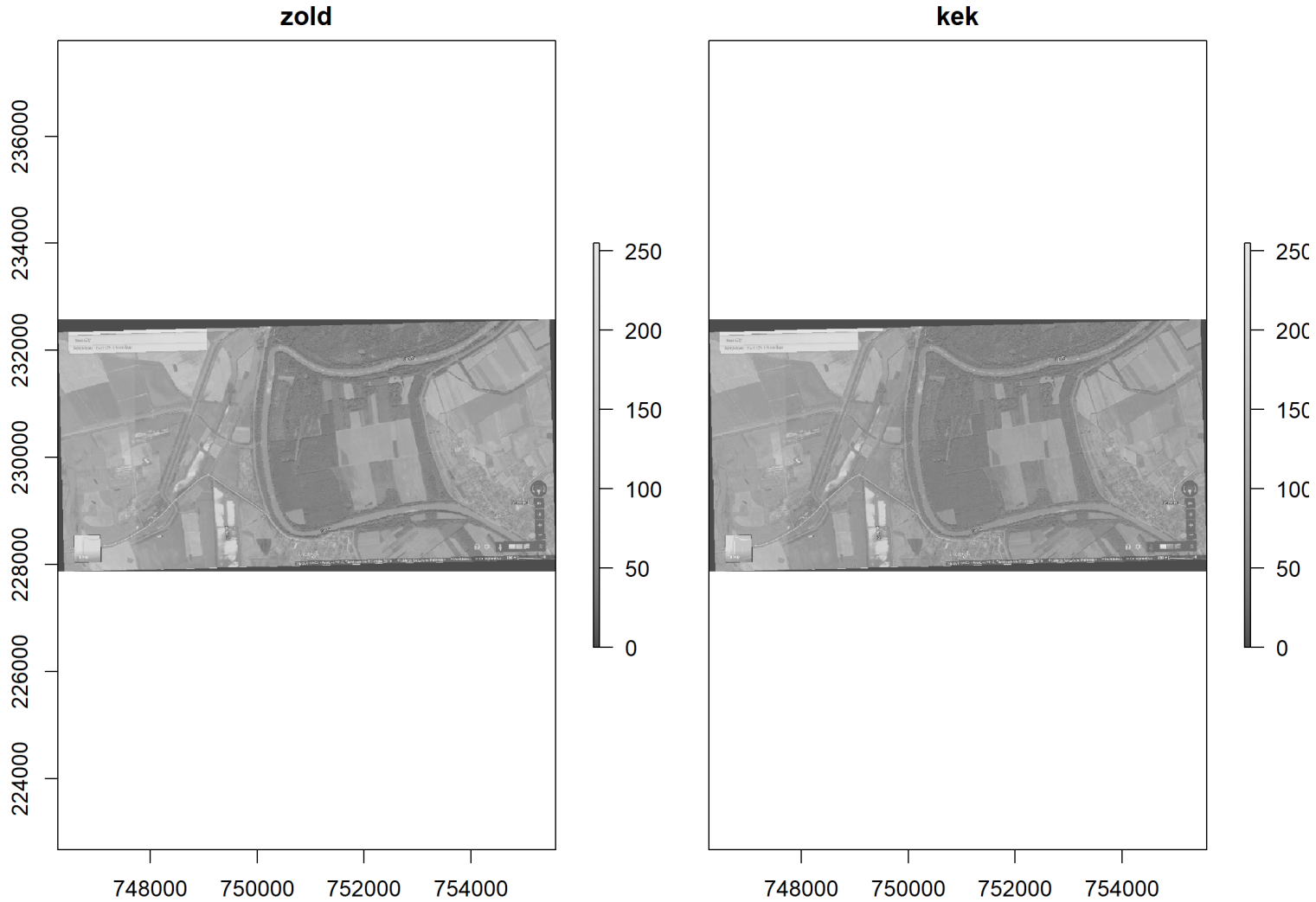
Raszterrétegek elérése

```
ketto_szincsatorna <- ortofoto[[c("zold", "kek")]]
class(ketto_szincsatorna)
```

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

Raszterrétegek elérése

```
plot(ketto_szincsatorna, col = gray.colors(100))
```



Raszterrétegek elérése

```
ketto_szincsatorna <- subset(x = ortofoto, subset =
 c("zold", "kek"))
class(ketto_szincsatorna)
```

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

```
egy_szincsatorna <- subset(x = ortofoto, subset = 3)
class(egy_szincsatorna)
```

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

Raszterrétegek elérése

```
egy_szincsatorna <- subset(x = ortofoto, subset = 3, drop
 = FALSE)
class(egy_szincsatorna)
```

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

Megmaradt RasterStacknek, hiába egyrétegű.

Raszterrétegek elérése

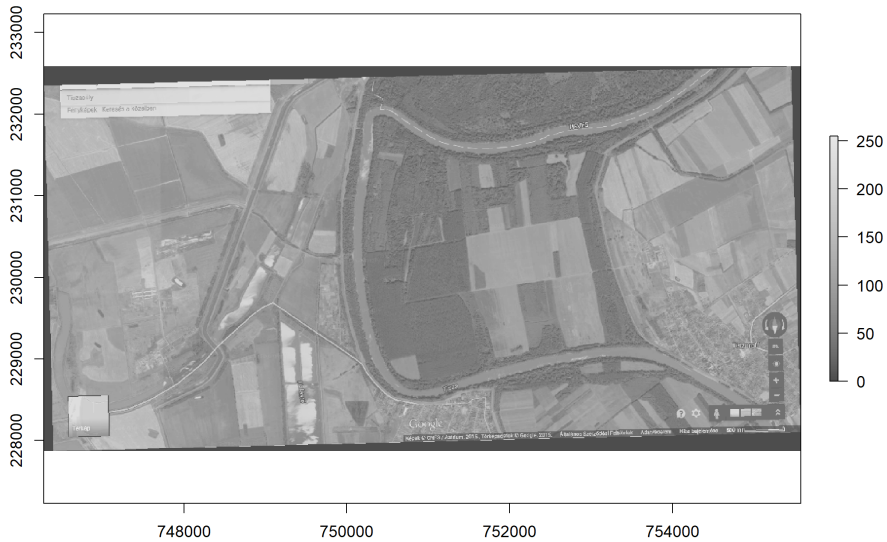
Egy réteg lekérése esetén a legkényelmesebb a $-jeles megoldás:

```
kek_szincsatorna <- ortofoto$kek
```

Raszterrétegek létrehozása/módosítása

Módosítás: a `[[]]` és `$` értékadás-operátorral (<-) történő kombinálása. Ha a réteg még nem létezik, létrehozza, ha létezik, módosítja.
Raszteralgebra: a különböző műveletek (+, -, *, /) és néhány függvény (pl. `min()`) kényelmesen használhatóak.

```
ortofoto$vilagossag <- (ortofoto$piros + ortofoto$zold +
 ortofoto$kek) / 3
plot(ortofoto$vilagossag, col = gray.colors(100))
```



Raszterrétegek létrehozása/módosítása

Használható továbbá az `addLayer(x, ...)` függvény is:

- `x`: mihez adunk réteget
- `...`: hozzáadandó rétegek

Raszterrétegek létrehozása/módosítása

```
ortofoto <- addLayer(x = ortofoto, ortofoto$zold,
 ortofoto$piros)
nlayers(ortofoto)
```

```
[1] 6
```

```
names(ortofoto)
```

```
[1] "piros.1"    "zold.1"     "kek"          "vilagossag"
[5] "zold.2"     "piros.2"
```

Raszterrétegek létrehozása/módosítása

Lehet RasterLayerhez is új rétegeket adni (bármelyik operátorral/függvénnyel), ekkor RasterStacket kapunk eredményként.

```
felszinhomerseklet_es_domborzatmodell <- felszinhomerseklet
class(felszinhomerseklet_es_domborzatmodell)
```

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

```
felszinhomerseklet_es_domborzatmodell$domborzatmodell <-
 domborzatmodell
class(felszinhomerseklet_es_domborzatmodell)
```

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

Raszterrétegek elhagyása

Háromféle módon:

- a `[[]]` operátorral (negatív számot használhatunk a réteg sorszámának megjelölésénél)
- a `subset()` függvénnyel
- vagy a `dropLayer(x, i)` függvénnyel, ahol `i` az elhagyandó réteg sorszáma

Raszterrétegek elhagyása

```
ortofoto <- ortofoto[[-6]]
nlayers(ortofoto)
```

```
[1] 5
```

```
ortofoto <- dropLayer(x = ortofoto, i = 5)
nlayers(ortofoto)
```

```
[1] 4
```

9. feladat (órai)

- Kérd le a “felszinhomerseklet_es_domborzatmodell” nevű raszter első (“felszinhomerseklet” nevű) rétegét (raszterként) olyan sokféle módon, ahányféleképpen csak tudod.
  - *Én 5 megoldást találtam, de biztosan van több is...*
- Adj hozzá egy új, “felszinhomerseklet_F” nevű réteget, amely a felszinhomerseklet rétegben °C-ban tárolt hőmérsékletet Fahrenheit-fokba váltja az alábbi képlettel

```
F = C * 1.8 + 32
```

- Töröld az 1., immáron fölöslegessé vált réteget a `dropLayer()` függvényt használva.

9. feladat (órai) – megoldás

```
felszinhomerseklet_es_domborzatmodell[[1]]
felszinhomerseklet_es_domborzatmodell[["felszinhomerseklet"]]
subset(x = felszinhomerseklet_es_domborzatmodell, subset =
 1)
subset(x = felszinhomerseklet_es_domborzatmodell, subset =
 "felszinhomerseklet")
felszinhomerseklet_es_domborzatmodell$felszinhomerseklet
```

```
felszinhomerseklet_es_domborzatmodell$felszinhomerseklet_F
 <-
 felszinhomerseklet_es_domborzatmodell$felszinhomerseklet
 * 1.8 + 32
felszinhomerseklet_es_domborzatmodell <- dropLayer(x =
 felszinhomerseklet_es_domborzatmodell, i = 1)
```

Rasztercellák lekérdezése

Rasztercellák értékét a `[]` operátorral vagy a `values(x)` függvénnyel érhetjük el.

```
magassagok <- domborzatmodell[]
class(magassagok)
```

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

```
str(magassagok)
```

```
 num [1:485818] NA NA NA NA NA NA NA NA NA NA ...
```

```
magassagok2 <- values(domborzatmodell)
identical(magassagok, magassagok2)
```

```
[1] TRUE
```

Rasztercellák lekérdezése

Ha nem az összes cella értékére van szükségünk, csak egyre vagy néhányra:

```
magassagok <- domborzatmodell[20000:150000]
class(magassagok)
```

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

```
str(magassagok)
```

```
 num [1:130001] 180 189 200 213 229 229 244 260 276 299 ...
```

Sorfolytonosan tárolódnak az értékek, a legfölső (legészakibb) sorral kezdve.

Rasztercellák lekérdezése

`[i, drop = TRUE]`

- `i`: cellatartomány (cellák sorszáma) vagy üres (= összes cella)
- `drop`: számvektorrá egyszerűsítsen-e
- `drop = FALSE` esetén rasztert ad eredményül

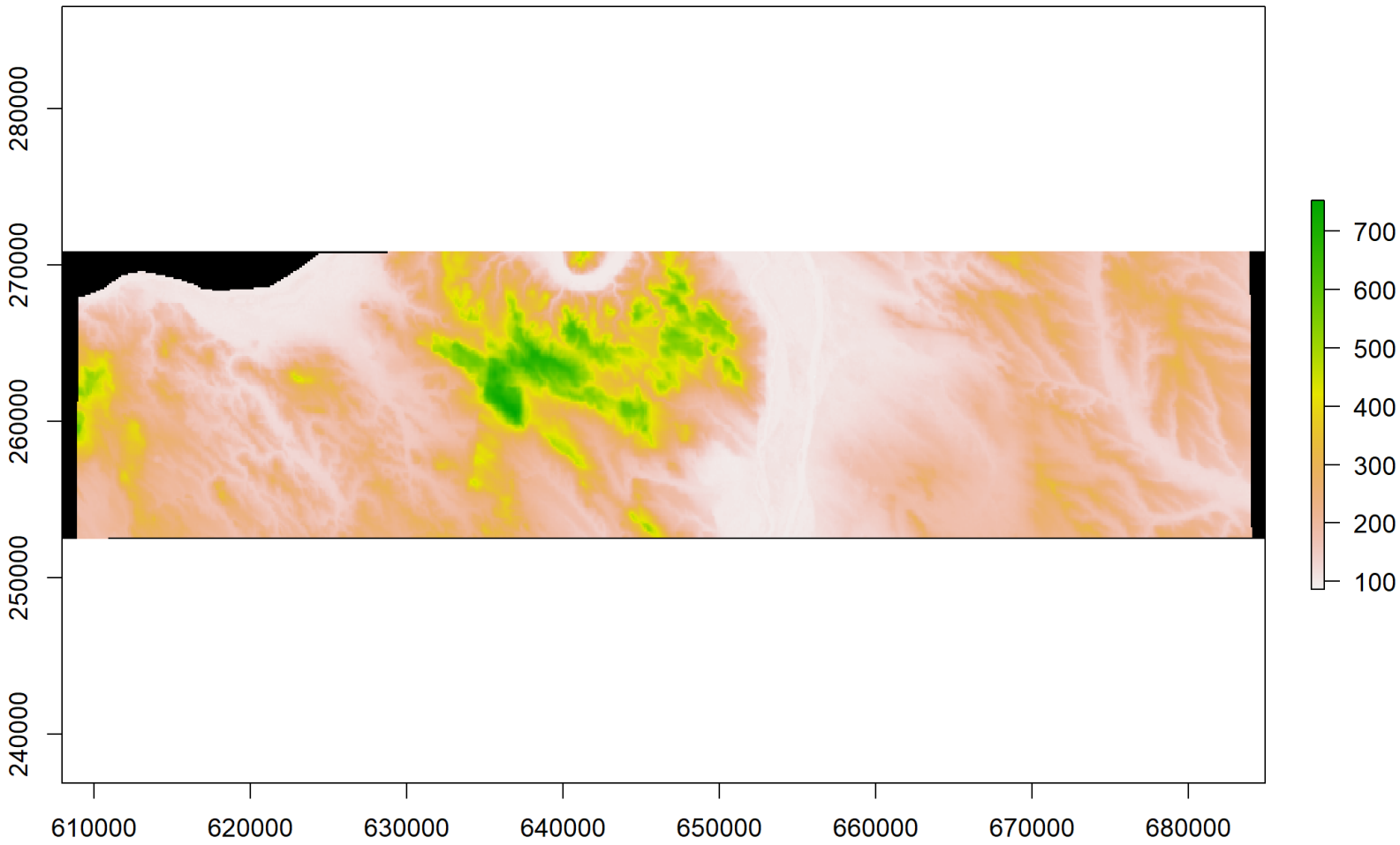
Rasztercellák lekérdezése

```
magassagok <- domborzatmodell[20000:150000, drop = FALSE]
class(magassagok)
```

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

Rasztercellák lekérdezése

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



Rasztercellák sorszámának lekérése

`cellFromXY(object, xy)`

- `object`: a raszter
- `xy`: koordináták (táblázatban), amelyek alá eső cellák sorszámára kíváncsiak vagyunk

Rasztercellák sorszámának lekérése

```
cellFromXY(object = domborzatmodell, xy =
 st_coordinates(repterek))
```

```
[1] 288035     NA     NA     NA     NA     NA     NA
[8]     NA     NA
```

```
ferihegy_cellaszama <- cellFromXY(object =
 domborzatmodell, xy =
 st_coordinates(repterek[repterek$nev == "Ferihegy", ]))
domborzatmodell[ferihegy_cellaszama]
```

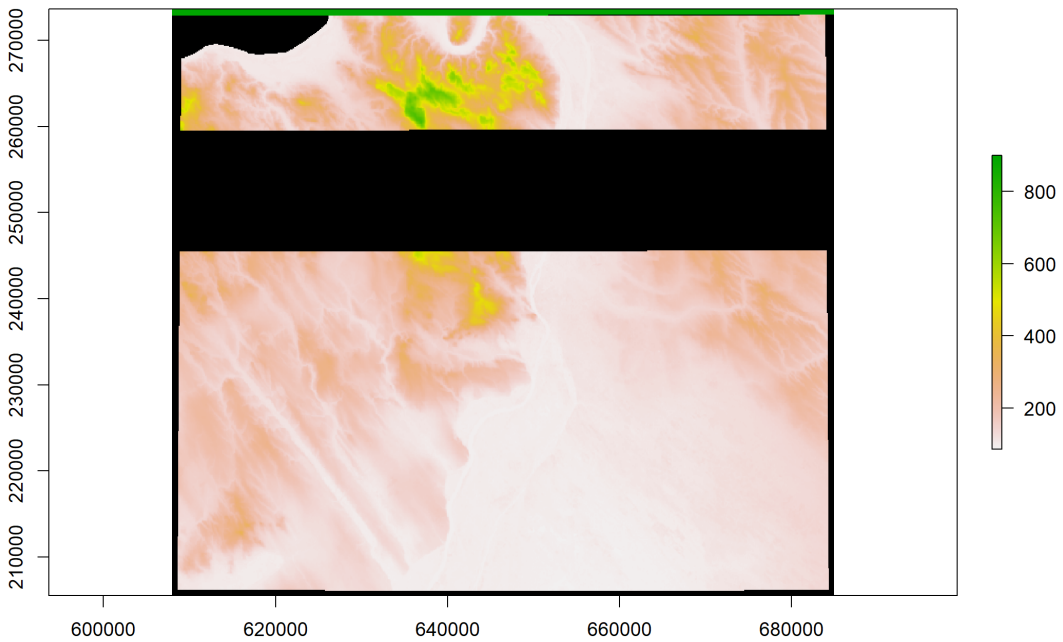
```
[1] 147
```

Erre majd tanulunk később egy kényelmesebb módszert is (`extract()`).

Rasztercellák módosítása

A [] és <- operátorok kombinálásával:

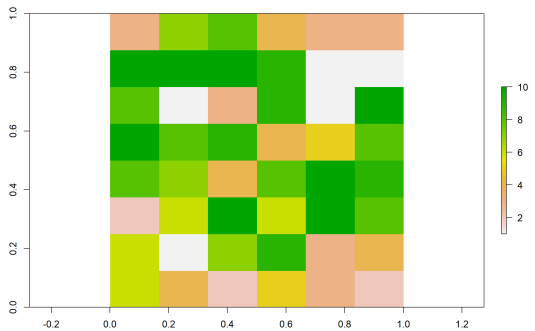
```
domborzatmodell[100000:200000] <- NA
domborzatmodell[1:5000] <- 900
plot(domborzatmodell, colNA = "black")
```



Rasztercellák lekérése sor és oszlop alapján

A `[i, j, drop = TRUE]` operátor használható sor és oszlop alapján történő lekérdezésre és módosításra (<- operátorral kombinálva).

```
plot(szamraszter)
```




```
szamraszter[11]
```

```
[1] 1
```

```
szamraszter[2, 5]
```

```
[1] 1
```

Rasztercellák lekérése sor és oszlop alapján

Akár sor- és oszloptartományokat is megadhatunk:

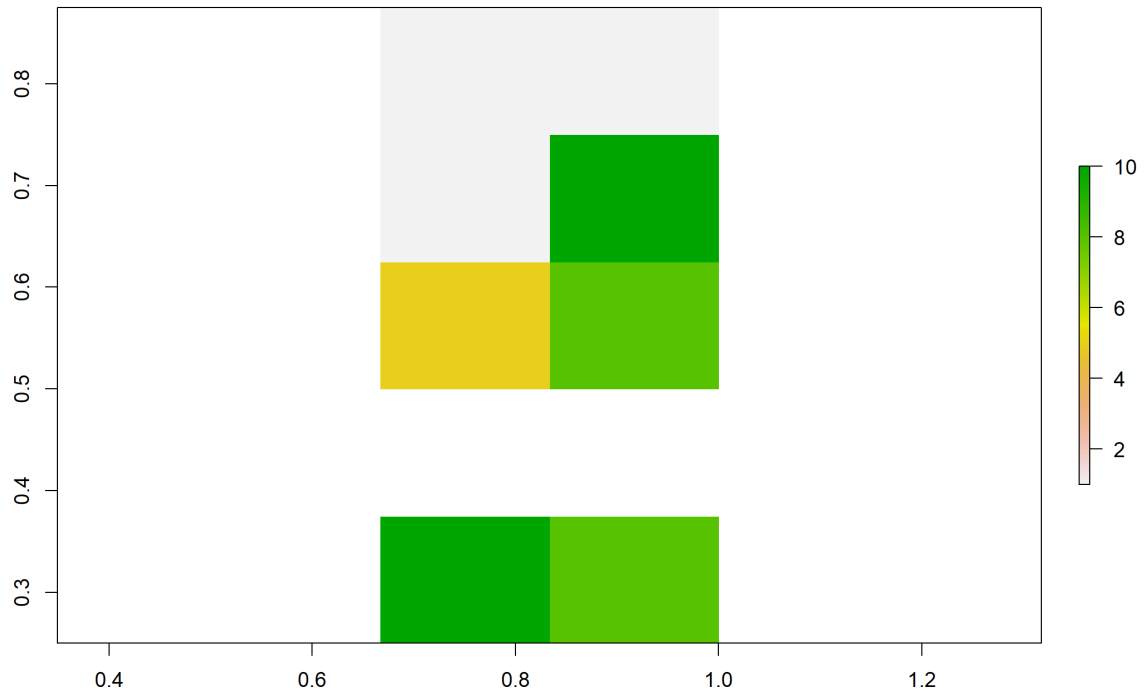
```
szamraszter[c(2:4, 6), 5:6]
```

```
[1]  1  1  1 10  5  8 10  8
```

Rasztercellák lekérése sor és oszlop alapján

`drop = FALSE` esetén raszter marad:

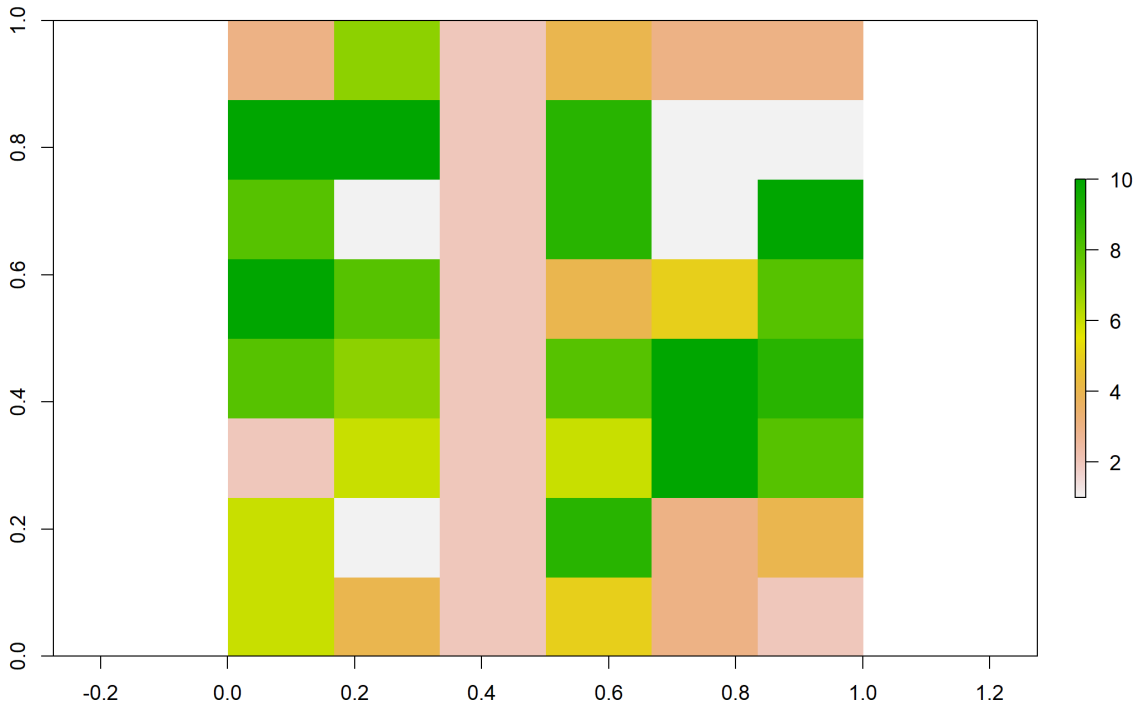
```
plot(szamraszter[c(2:4, 6), 5:6, drop = FALSE])
```



Rasztercellák módosítása sor és oszlop alapján

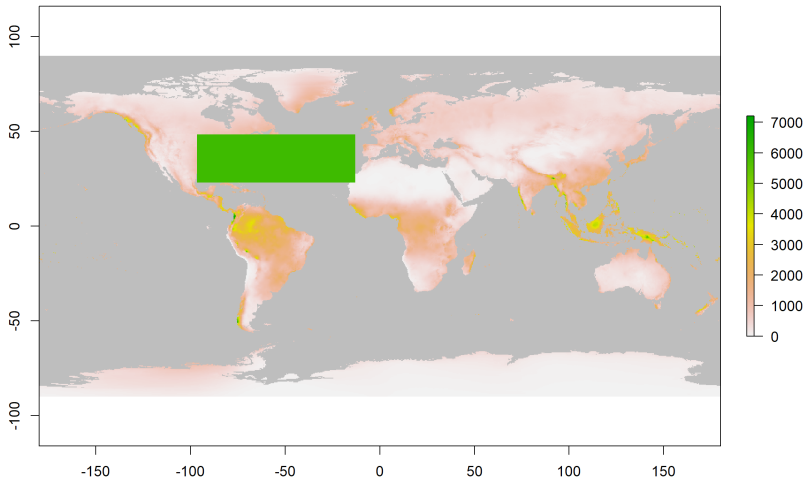
Kombinálva értékadás-operátorral:

```
szamraszter[, 3] <- 2
plot(szamraszter)
```



10. feladat (órai)

- Olvasd be a “varosok.RData” fájlt.
- Válogasd le Debrecent, és vetítsd át WGS-84-be (4326).
- Kérd le, hogy csapadek nevű raszter hanyadik sorszámú cellája fölé esik Debrecen középpontja.
- Mennyi a csapadék ebben a cellában?
- Módosítsd 6000-re a csapadékot a 500–800. sorok és 1000–2000 oszlopok által kijelölt tartományban.
- Jelenítsd meg az eredményt.
- Kérd le kétféle módon az összes csapadékértéket.



10. feladat (órai) – megoldás

```
load("varosok.RData")
debrecen <- st_transform(x = varosok[varosok$nev ==
 "Debrecen", ], crs = 4326)
cella_szama <- cellFromXY(object = csapadek, xy =
 st_coordinates(st_centroid(debrecen)))
```

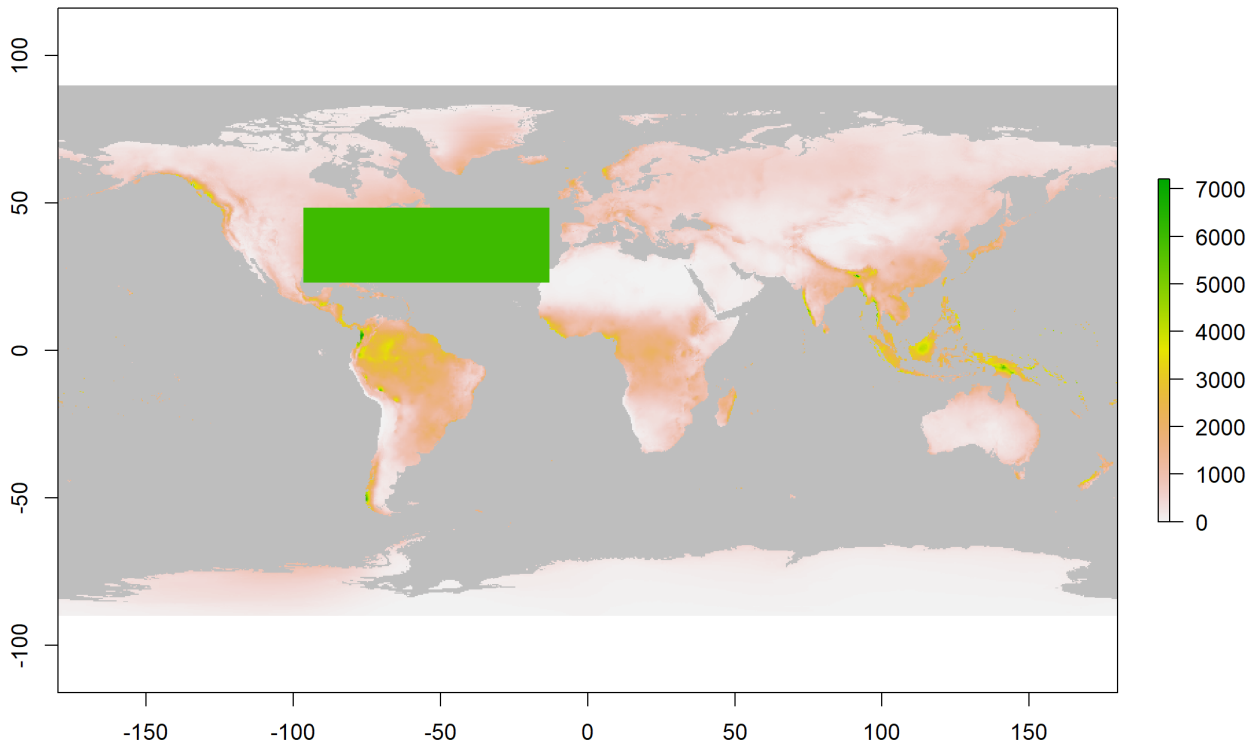
```
csapadek[cella_szama]
```

```
[1] 558
```

```
csapadek[500:800, 1000:2000] <- 6000
```

10. feladat (órai) – megoldás

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




```
csapadekertekek <- values(csapadek)
csapadekertekek <- csapadek[]
```

11. (összefoglaló) feladat (házi)

- Olvasd be az “ortofoto.tif” fájlt `RasterStack`ként.
- Írd a képernyőre a vetületét.
- Nezezd át a rétegeit rendre “r”, “g” és “b”-re.
- Készítsd “piros”, “zold” és “kek” néven új, egyrétegű rasztereket az ortofotó megfelelő rétegeiből úgy, hogy a három réteghez három különböző leválógató operátort/függvényt használsz. Add meg a `subset()` függvény `drop` paraméterét, függetlenül attól, hogy az alapértelmezett értéke jó lenne-e most számunkra.
- Ellenőrizd, hogy mindhárom új raszter `RasterLayer` típusú-e.
- A kék raszter első 100 sorát töröld (állítsd ismeretlenre).
- Fűzd össze a kék és a piros rasztert `RasterBrick`ké, majd add hozzá a zöld rasztert is további rétegként.
- Jelenítsd meg az összefűzött rasztert színhelyesen.
- Helyezd rá az utak geometriáját ötszörös vastagságú fehér vonalként.

11. (összefoglaló) feladat (házi) – megoldás

```
ortofoto <- stack(x = "ortofoto.tif")
```

```
projection(ortofoto)
```

```
[1] "+proj=somerc +lat_0=47.1443937222222
 +lon_0=19.0485717777778 +k_0=0.99993 +x_0=650000
 +y_0=200000 +ellps=GRS67 +units=m +no_defs"
```

```
names(ortofoto) <- c("r", "g", "b")
piros <- ortofoto$r
zold <- subset(x = ortofoto, subset = 2, drop = TRUE)
kek <- ortofoto[[3]]
all(c(class(piros), class(zold), class(kek)) ==
 "RasterLayer")
```

```
[1] TRUE
```

```
kek[1:(100 * ncol(kek))] <- NA
```

11. (összefoglaló) feladat (házi) – megoldás

```
osszefuzott <- addLayer(x = brick(kek, piros), zold)
plotRGB(x = osszefuzott, r = 2, g = 3, b = 1)
plot(utak_geometria, col = "white", lwd = 5, add = TRUE)
```