

Startit centar, Savska 5



BelgradeR Meetup

Analiza geoprostornih podataka u R-u



14. decembar 18:00 - 20:00

Sadržaj

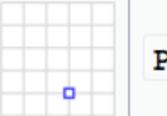
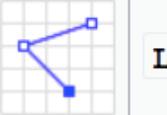
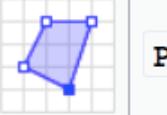
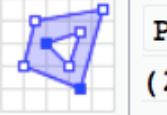
Primer 1 – Vektorski i rasterski tipovi geoprostornih podataka

Primer 2 – Prostоризација загадујућих материја – коришћење R-а као GIS окружења

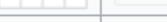
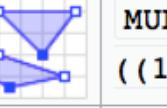
Primer 3 – Просторна интерполяција органског угљеника у земљишту

Primer 1 – Vektorski i rasterski tipovi geoprostornih podataka

Geometry primitives (2D)

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

Multipart geometries (2D)

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

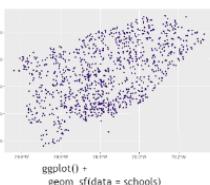
Primer 1 – Vektorski i rasterski tipovi geoprostornih podataka

Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.

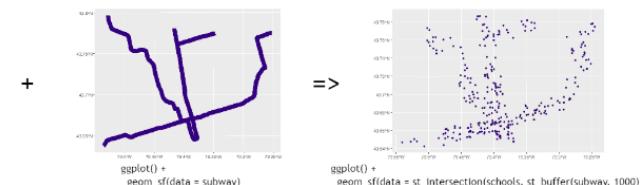
Geometric confirmation

- ▢ st_contains(x, y, ...) Identifies if y is within x (i.e. point within polygon)
- ▢ st_covered_by(x, y, ...) Identifies if x is completely within y (i.e. polygon completely within polygon)
- ▢ st_covers(x, y, ...) Identifies if any point from x is outside of y (i.e. polygon outside polygon)
- ▢ st_crosses(x, y, ...) Identifies if any geometry of x has commonalities with y
- ▢ st_disjoint(x, y, ...) Identifies when geometries from x do not share space with y
- ▢ st_equals(x, y, ...) Identifies if x and y share the same geometry
- ▢ st_intersects(x, y, ...) Identifies if x and y geometry share any space
- ▢ st_overlaps(x, y, ...) Identifies if geometries of x and y share space, are of the same dimension, but are not completely contained by each other
- ▢ st_touches(x, y, ...) Identifies if geometries of x and y share a common point but their interiors do not intersect
- ▢ st_within(x, y, ...) Identifies if x is in a specified distance to y



Geometric operations

- ▢ st_boundary(x) Creates a polygon that encompasses the full extent of the geometry
- ▢ st_buffer(x, dist, nQuadSegs) Creates a polygon covering all points of the geometry within a given distance
- ▢ st_centroid(x, ..., of_largest_polygon) Creates a point at the geometric centre of the geometry
- ▢ st_convex_hull(x) Creates geometry that represents the minimum convex geometry of x
- ▢ st_line_merge(x) Creates linestring geometry from sewing multi linestring geometry together
- ▢ st_node(x) Creates nodes on overlapping geometry where nodes do not exist
- ▢ st_point_on_surface(x) Creates a point that is guaranteed to fall on the surface of the geometry
- ▢ st_polygon(x) Creates polygon geometry from linestring geometry
- ▢ st_segmentize(x, dfMaxLength, ...) Creates linestring geometry from x based on a specified length
- ▢ st_simplify(x, preserveTopology, dTolerance) Creates a simplified version of the geometry based on a specified tolerance



Geometry creation

- ▢ st_triangulate(x, dTolerance, bOnlyEdges) Creates polygon geometry as triangles from point geometry
- ▢ st_voronoi(x, envelope, dTolerance, bOnlyEdges) Creates polygon geometry covering the envelope of x, with x at the centre of the geometry
- ▢ st_point(x, c(numeric vector), dim = "XYZ") Creating point geometry from numeric values
- ▢ st_multipoint(x = matrix(numeric values in rows), dim = "XYZ") Creating multi point geometry from numeric values
- ▢ st_linestring(x = matrix(numeric values in rows), dim = "XYZ") Creating linestring geometry from numeric values
- ▢ st_multilinestring(x = list(numeric matrices in rows), dim = "XYZ") Creating multi linestring geometry from numeric values
- ▢ st_polygon(x = list(numeric matrices in rows), dim = "XYZ") Creating polygon geometry from numeric values
- ▢ st_multipolygon(x = list(numeric matrices in rows), dim = "XYZ") Creating multi polygon geometry from numeric values



Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.

Geometry operations

- ▢ st_contains(x, y, ...) Identifies if y is within x (i.e. point within polygon)
- ▢ st_crop(x, y, ..., xmin, ymin, xmax, ymax) Creates geometry of x that intersects a specified rectangle
- ▢ st_difference(x, y) Creates geometry from x that does not intersect with y
- ▢ st_intersection(x, y) Creates geometry of the shared portion of x and y
- ▢ st_sym_difference(x, y) Creates geometry representing portions of x and y that do not intersect
- ▢ st_snap(x, y, tolerance) Snap nodes from geometry x to geometry y
- ▢ st_union(x, y, ..., by_feature) Creates multiple geometries into a single geometry, consisting of all geometry elements

Geometric measurement

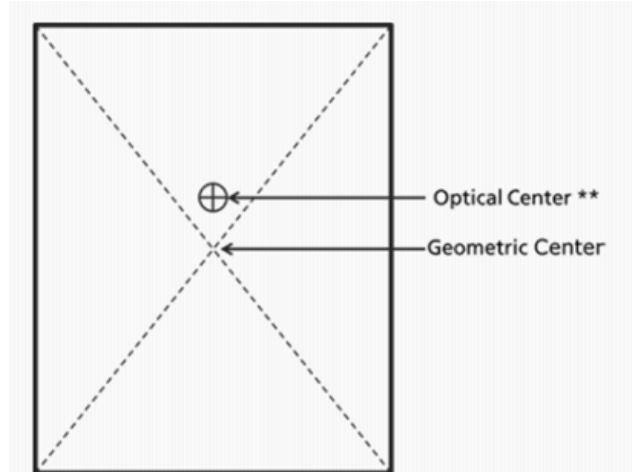
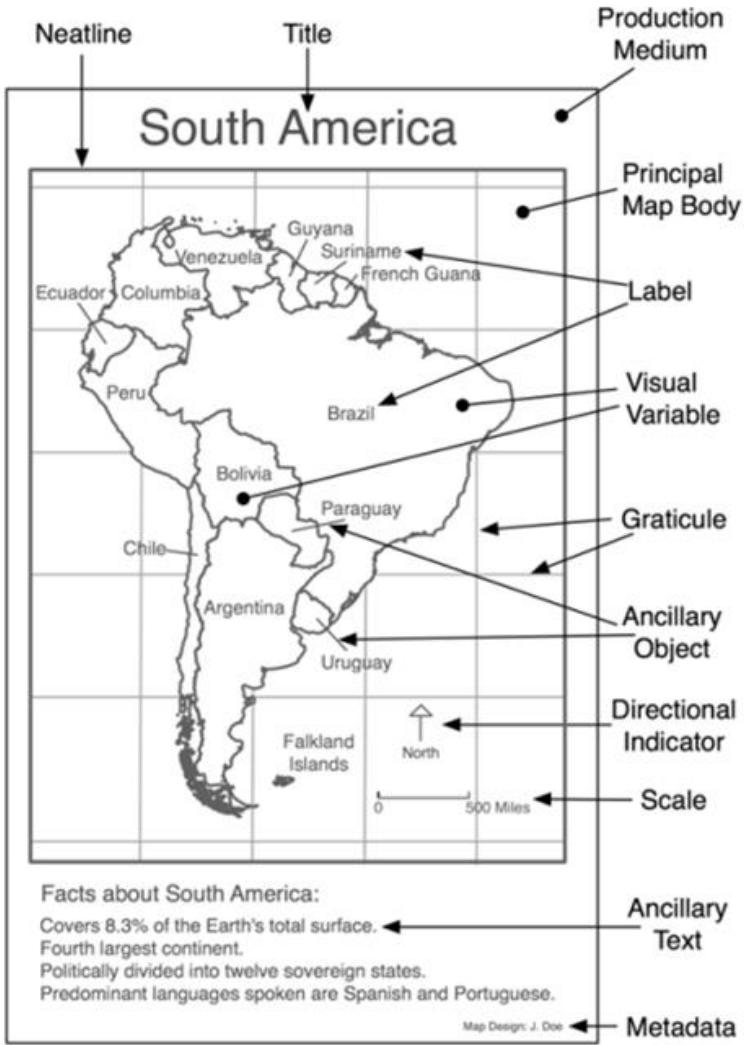
- ▢ st_area(x) Calculate the surface area of a polygon geometry based on the current coordinate reference system
- ▢ st_distance(x, y, ..., dist_fun, by_element, which) Calculates the 2D distance between x and y based on the current coordinate system
- ▢ st_length(x) Calculates the 2D length of a geometry based on the current coordinate system

Misc operations

- ▢ st_as_sf(x, ...) Create a sf object from a non-geospatial tabular data frame
- ▢ st_cast(x, to, ...) Change x geometry to a different geometry type
- ▢ st_coordinates(x, ...) Creates a matrix of coordinate values from x
- ▢ st_crs(x, ...) Identifies the coordinate reference system of x
- ▢ st_join(x, y, join, FUN, suffix, ...) Performs a spatial left or inner join between x and y
- ▢ st_make_grid(x, cellsize, offset, n, crs, what) Creates rectangular grid geometry over the bounding box of x
- ▢ st_nearest_feature(x, y) Creates an index of the closest feature between x and y
- ▢ st_nearest_points(x, y, ...) Returns the closest point between x and y
- ▢ st_read(dsn, layer, ...) Read file or database vector dataset as a sf object
- ▢ st_transform(x, crs, ...) Convert coordinates of x to a different coordinate reference system



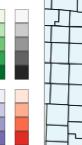
Geovizuelizacija – dizajn karte



Nature of your data: sequential diverging qualitative

Pick a color scheme:

Multi-hue: 

Single hue: 

Only show: colorblind safe print friendly photocopy safe

Context: roads cities borders

Background: solid color terrain

color transparency

Source code and feedback

how to use | **updates** | **downloads** | **credits**

COLORBREWER 2.0
color advice for cartography



EXPORT

HEX

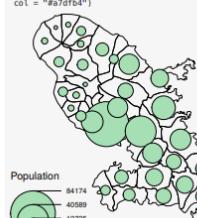
#e5f5f9
#99d8c9
#2ca25f

Thematic maps with cartography :: CHEAT SHEET

Symbology

In most functions the x argument should be an sf object. sp objects are handled through spdf and df arguments.

- Choropleth**
choroplotter(x = mtq, var = "myvar", method = "quantile", nclass = 8)
- Typology**
typewriter(x = mtq, var = "myvar")
- Proportional Symbols**
propSymbolLayer(x = mtq, var = "myvar", inches = 8.1, symbols = "circle")
- Colored Proportional Symbols (relative data)**
propSymbolLayer(x = mtq, var = "myvar", var2 = "myvar2")
- Colored Proportional Symbols (qualitative data)**
propSymbolLayer(x = mtq, var = "myvar", var2 = "myvar2")
- Double Proportional Symbols**
propTriangularLayer(x = mtq, vari = "myvar", var2 = "myvar2")
- OpenStreetMap Basemap** (see osm package)
titles <- getTitles(x = mtq, type = "osm")

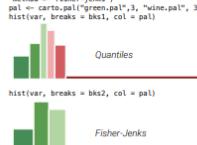


Classification

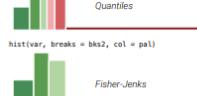
Available methods are: quantile, equal, qf, fisher-jenks, meanstd, sd, geomsplitter, breaks.

```
bk1 <- getBreaks(x = var, nclass = 6, method = "quantile")
bk2 <- getBreaks(x = var, nclass = 6, method = "fisher-jenks")
pal <- carspal("green+3", "wine,pal", 3)
histvar, breaks = bk1, col = pal
```

Quantiles



Fisher-Jenks



Transformations

sf::st_grid <- getGridLayer(x = mtq, cellsize = 3.6e+07, type = "hexagonal", var = "myvar")

Grids layers can be used by choropleth or propSymbolLayer.



Points to Links

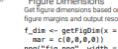
mtq <- stGetLinkLayer(x = mtq, df = link)



Link layer can be used by typewriter()

Polylines to Borders

stq_border <- getBorders(x = mtq)



Borders layers can be used by dscLayer()

Map Layout

North Arrow: northarrows = "topright"

Scale Bar: scalebarSize = 5

Full Layout:

```
layoutLayer(
  title = "Martinique",
  subtitle = NULL,
  frame = TRUE,
  author = "Author",
  sources = "Sources",
  notes = "Notes",
  scale = 5)
```

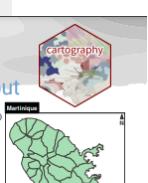


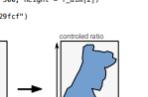
Figure Dimensions
Get figure dimensions based on the dimension ratio of a spatial object, figure margin and resolution.

```
f.dim <- getFdim(x = sf_obj, width = 500,
  mar = c(6, 6, 0, 0))
png("fp.png", width = 500, height = f.dim[2])
par(mar = c(6, 6, 0, 0), col = "#2ca25f")
plot(sf_obj, col = "#2ca25f")
dev.off()
```

Small



Large



controlled ratio



Legends

Legend Choropleth

```
legendChorop() <- topleft(x = "topleft",
  y = "topleft", n = 5, col = "blue,pal", n1 = 5,
  n2 = 3, breaks = c(0, 29, 48, 69, 88, 100),
  col = "orange,pal", n1 = 3, n2 = 1, nodata = TRUE, nodata.txt = "No Data")
```

Legend Proportional

```
legendPropType1() <- topleft(x = "topleft",
  y = "topright", n = 5, col = "red,pal", n1 = 5,
  n2 = 3, breaks = c(0, 29, 48, 69, 88, 100),
  col = "brown,pal", n1 = 3, n2 = 1, nodata = FALSE)
```

Legend Discontinuities

```
legendDiscont() <- topleft(x = "bottomleft",
  y = "topleft", n = 5, col = "purple,pal", n1 = 5,
  n2 = 3, breaks = c(0, 29, 48, 69, 88, 100),
  col = "green,pal", n1 = 3, n2 = 1, nodata = TRUE, nodata.txt = "No Data")
```

Legend Flow

```
legendFlow() <- topleft(x = "bottomleft",
  y = "bottomleft", n = 5, col = "pink,pal", n1 = 5,
  n2 = 3, breaks = c(0, 29, 48, 69, 88, 100),
  col = "wine,pal", n1 = 3, n2 = 1, nodata = FALSE)
```

Legend Dot Density

```
legendDotDensity() <- topleft(x = "bottomright",
  y = "topleft", n = 5, col = "grey,pal", n1 = 5,
  n2 = 3, breaks = c(0, 29, 48, 69, 88, 100),
  col = "turquoise,pal", n1 = 3, n2 = 1, nodata = TRUE, nodata.txt = "No Data")
```

Legend Labels

```
legendLabel() <- topleft(x = "bottomright",
  y = "bottomright", n = 5, col = "sand,pal", n1 = 5,
  n2 = 3, breaks = c(0, 29, 48, 69, 88, 100),
  col = "tulip,pal", n1 = 3, n2 = 1, nodata = FALSE)
```

Color Palettes

carspal.col(n = 8) orange pal
red pal brown pal
green pal purple pal
pink pal wine pal
grey pal turquoise pal
sand pal tulip pal
kaki pal harmo pal
pastel pal multi pal

display.carto.all(n = 8)

carto.pal.col(n = 8) blue pal
red pal brown pal
green pal purple pal
pink pal wine pal
grey pal turquoise pal
sand pal tulip pal
kaki pal harmo pal
pastel pal multi pal

carto.pal.col(n = 5) blue pal
red pal brown pal
green pal purple pal
grey pal

carto.pal.col(n = 3) blue pal
red pal brown pal

carto.pal.col(n = 2) blue pal
red pal

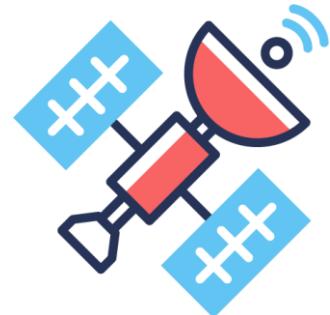
carto.pal.col(n = 1) blue pal

carto.pal.col(n = 0) white

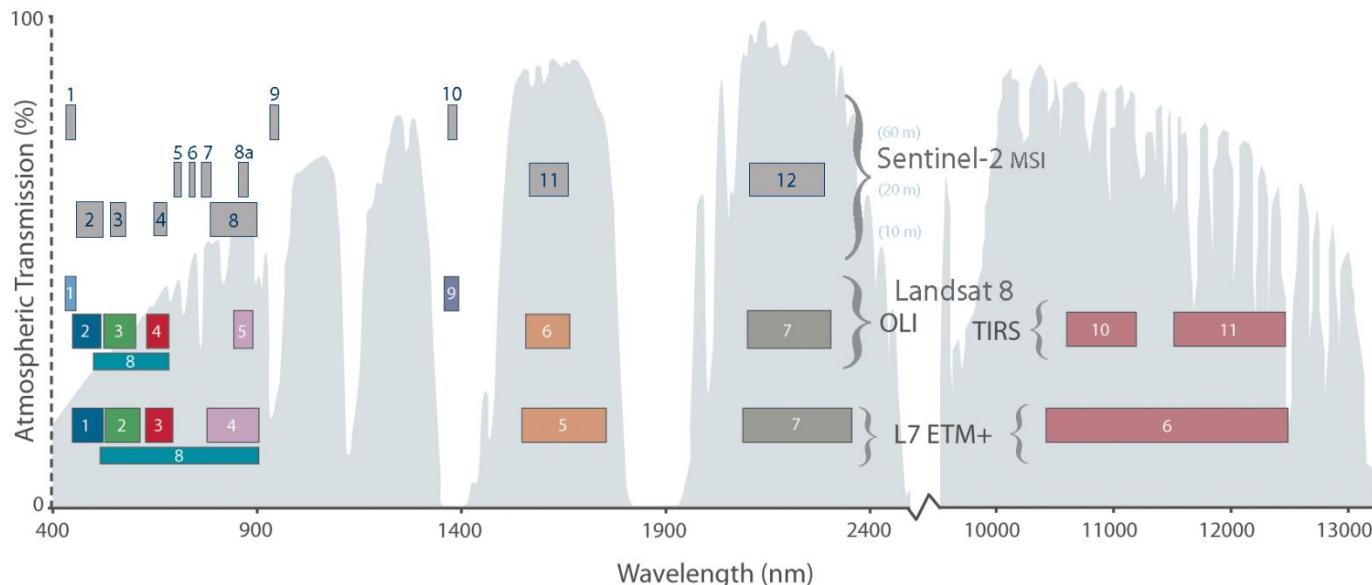
See also: legendSquareGrid(), legendCirclesGrid(), legendPropLines() and legendPropTriangles().

cartography - https://github.com/martinab/cartoography | CC BY SA Timothée Grall - 2019-01

Primer 1 – Vektorski i rasterski tipovi geoprostornih podataka

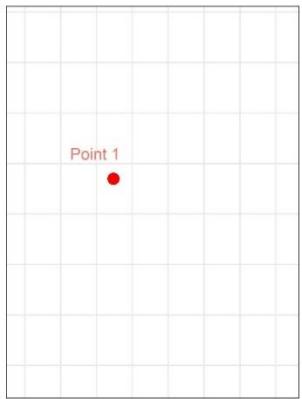


Comparison of Landsat 7 and 8 bands with Sentinel-2



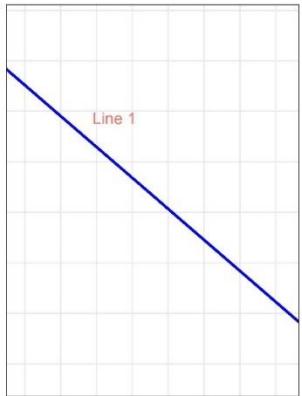
Sentinel-2 Kanali	Talasna dužina (μm)	Rezolucija (m)	Opseg(nm)
Band 1 – Coastal aerosol	0.443	60	20
Band 2 – Blue	0.490	10	65
Band 3 – Green	0.560	10	35
Band 4 – Red	0.665	10	30
Band 5 – Vegetation Red Edge	0.705	20	15
Band 6 – Vegetation Red Edge	0.740	20	15
Band 7 – Vegetation Red Edge	0.783	20	20
Band 8 – NIR	0.842	10	115
Band 8A – Narrow NIR	0.865	20	20
Band 9 – Water vapour	0.945	60	20
Band 10 – SWIR – Cirrus	1.375	60	20
Band 11 – SWIR	1.610	20	90
Band 12 – SWIR	2.190	20	180

Primer 2 – Prostorizacija zagađujućih materija



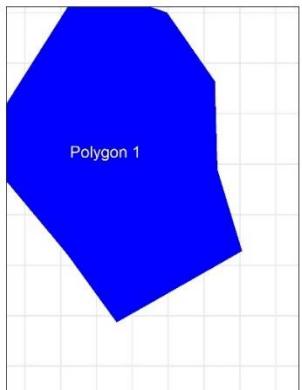
2907	2997	3087
Point 1		
2908	2998	3088

$ECell_{(2908)} = EValue_{(Point1)}$
ECell - Emission value in cell
EValue - Value of pollutant emission of point source



2907	2997	3087
Line 1-1		
2908	2998	3088

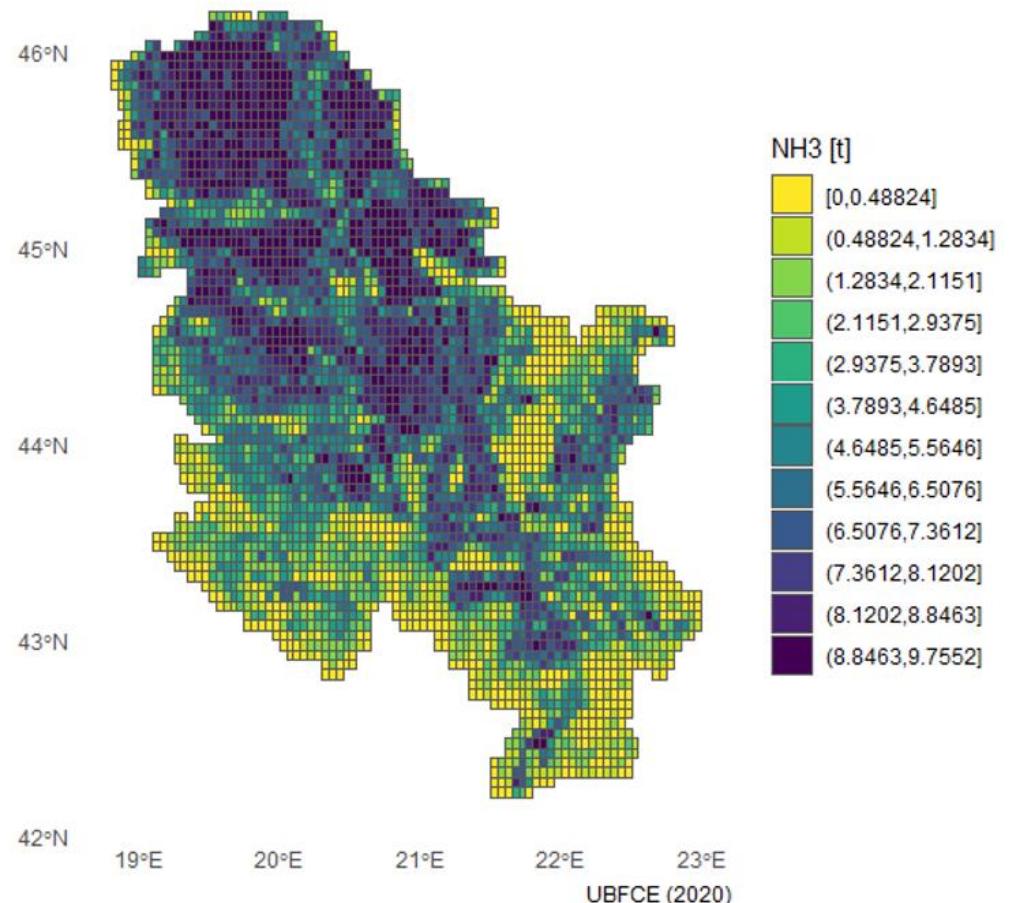
$ECell_{(2908)} = EValue_{(Line1)} * Length_{(Line1-1)} / Length_{(Line1)}$
ECell - Emission value in cell
EValue - Value of pollutant emission of line source



2907	2997	3087
Polygon 1-1		
2908	2998	3088

$ECell_{(2908)} = EValue_{(Polygon1)} * Area_{(Polygon1-1)} / Area_{(Polygon1)}$
ECell - Emission value in cell
EValue - Value of pollutant emission of polygon source

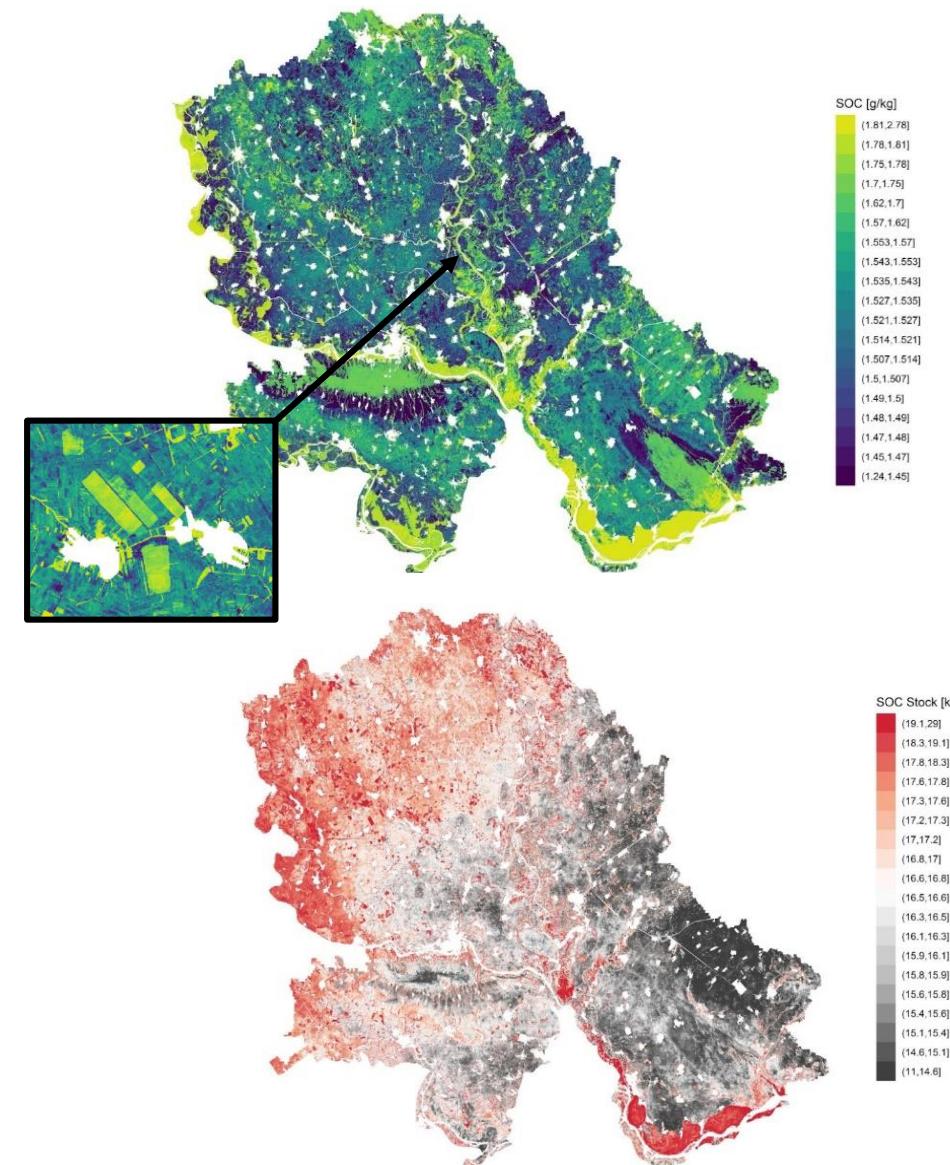
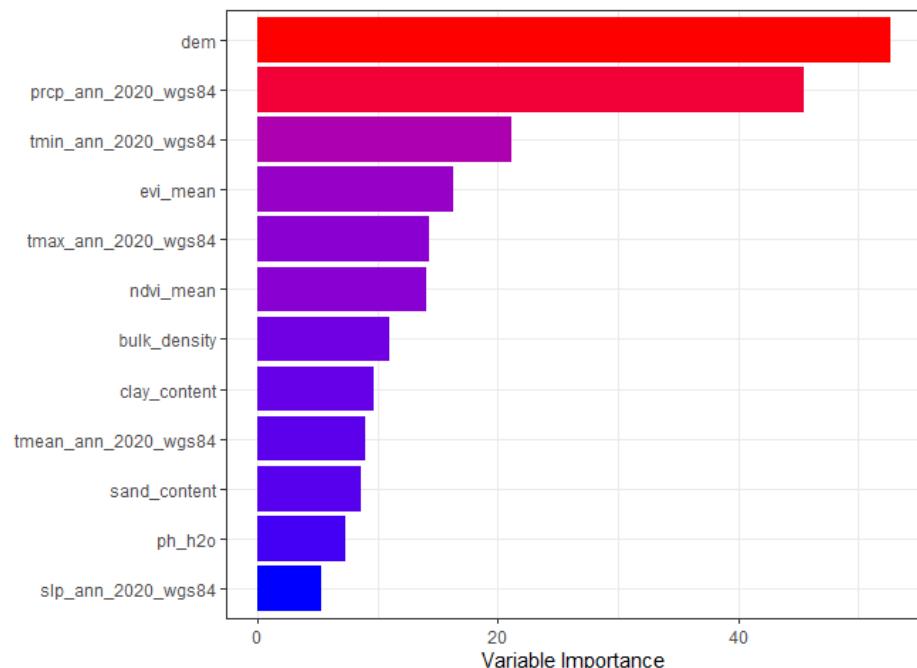
Pollutant inventory spatialization - NH3
Spatial resolution $0.05^\circ \times 0.05^\circ$



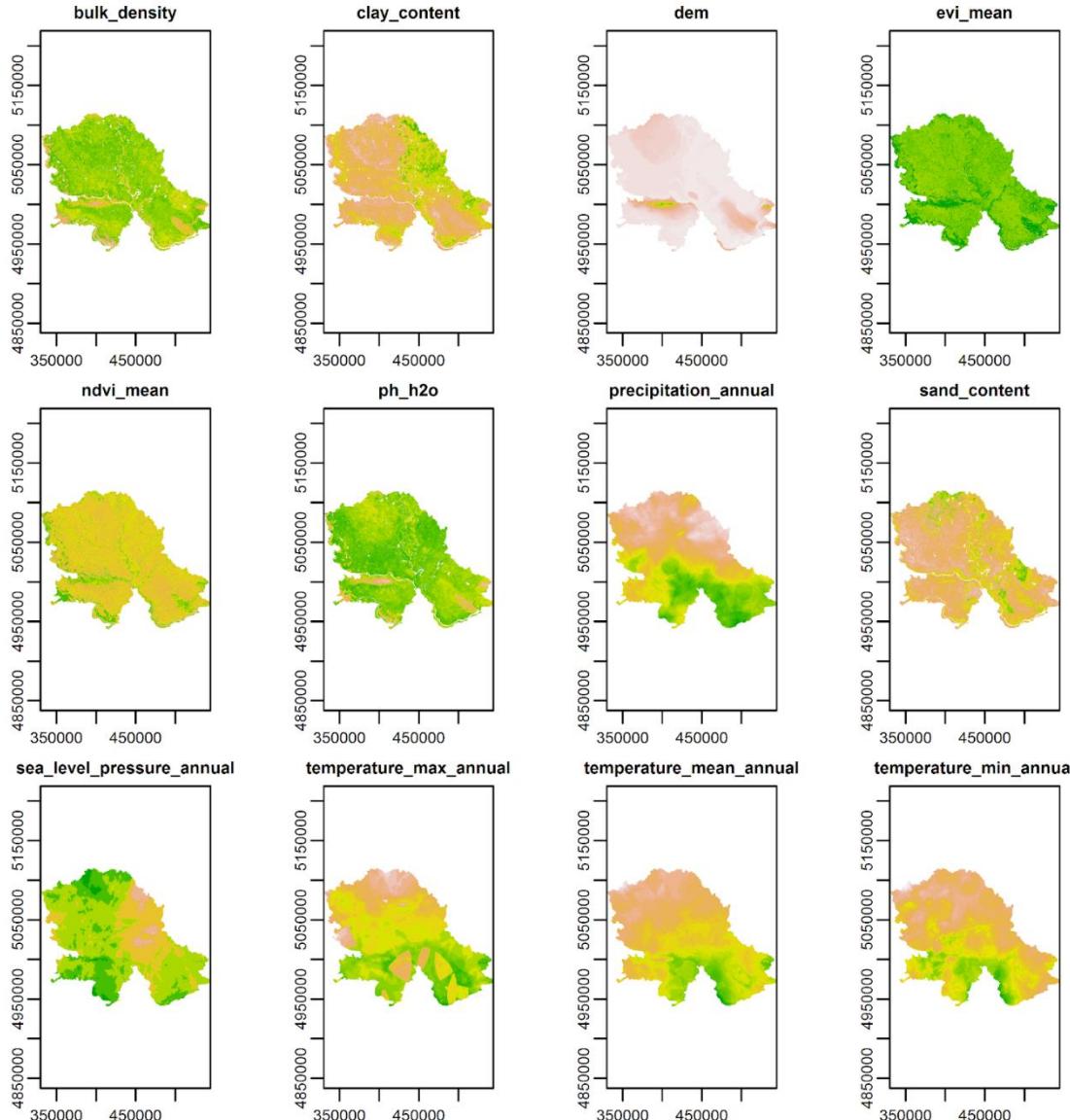
Primer 3 – Prostorna interpolacija organskog ugljenika u zemljištu

Razvoj modela za prostornu interpolaciju organskog ugljenika u zemljištu - na osnovu podataka uzorkovanja zemljišta (1200 profila)

Istraživanje kao deo projekta Fonda sa nauku RS „CERES – Informacije bazirane na satelitskom osmatranju Zemlje za "pametniju" i regenerativnu poljoprivredu“. Istraživanje je zasnovano na upotrebi tehnika mašinskog učenja (npr. Random Forest - RF) u cilju predikcije organskog ugljenika (SOC – Soil Organic Carbon) na osnovu podataka uzrokovana zemljišta i dodatnih prediktora (vegetacionih indeksa dobijenih na osnovu satelistkih snimaka Sentinel 2, digitalnog modela terena, klimatskih prediktora i prediktora kvaliteta zemljišta).



Primer 3 – Prostorna interpolacija organskog ugljenika u zemljištu



Performanse RF modela na na test setu podataka
(trening i test set podataka generisan u odnosu
75% i 25%)

R^2	NRMSE [g/kg]	NMAE [g/kg]
0.66	0.181	0.129

Hvala na pažnji!



https://github.com/pejovic/BelgradeR_2022