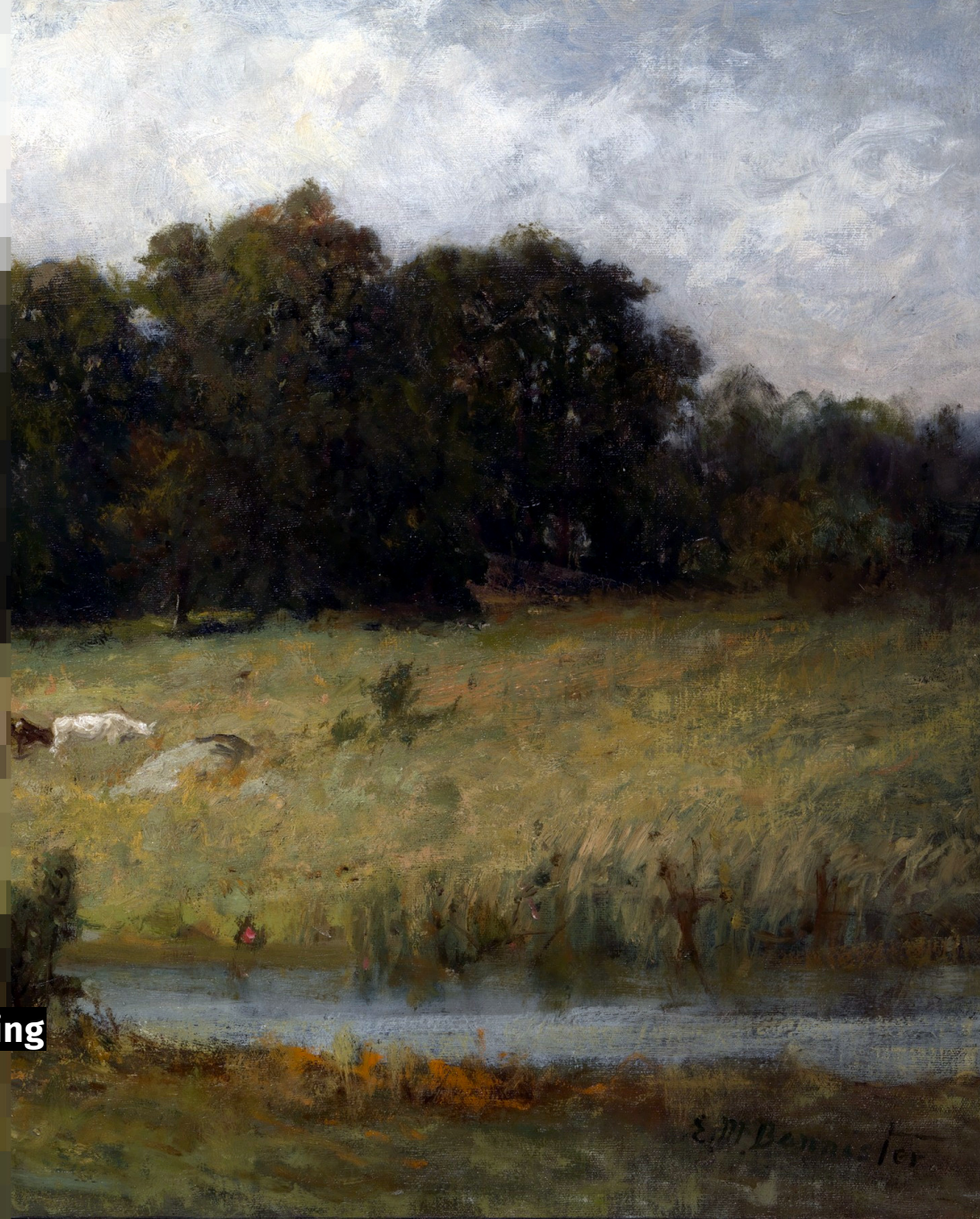
Introduction to landscape ecology with R

Jakub Nowosad and Maximilian H.K. Hesselbarth

2020-05-14, the IALE-North America 2020 Annual Meeting



Prerequisites

Packages:

- You can install the packages used in the workshop as follows:

```
install.packages("landscapemetrics")
install.packages("usethis")
install.packages("remotes")
remotes::install_github("mtennekes/tmap", upgrade = TRUE)
remotes::install_github("rspatial/raster", upgrade = TRUE)
```

Slides:

- **https://r-spatialecology.github.io/ialena-2020/**

Scripts and data:

- The code below can be used to download all the scripts and data for the workshop:

```
usethis::use_course("https://github.com/r-spatialecology/ialena-2020/raw/master/ialena-2020.zip")
```

Workshop agenda

- **Part I** (90 minutes): Introduction to the spatial data analysis in R:
  - Spatial data representations
  - Spatial data reading
  - Making maps in R
  - Vector-raster interactions
  - Reprojecting spatial data
- **Break** (30 minutes)
- **Part II** (90 minutes): *landscapemetrics*: An open-source R tool to calculate landscape metrics:
  - Introdution to landscape metrics
  - How to calculate landscape metrics in R
  - Sample metrics in buffer areas
  - Moving window
  - Building blocks
- **Q and A**:
  - You can ask us any questions related to spatial data analysis in R and the *landscapemetrics* package during the workshop using **https://www.sli.do/**
  - The event code is `IALENA`

Part I

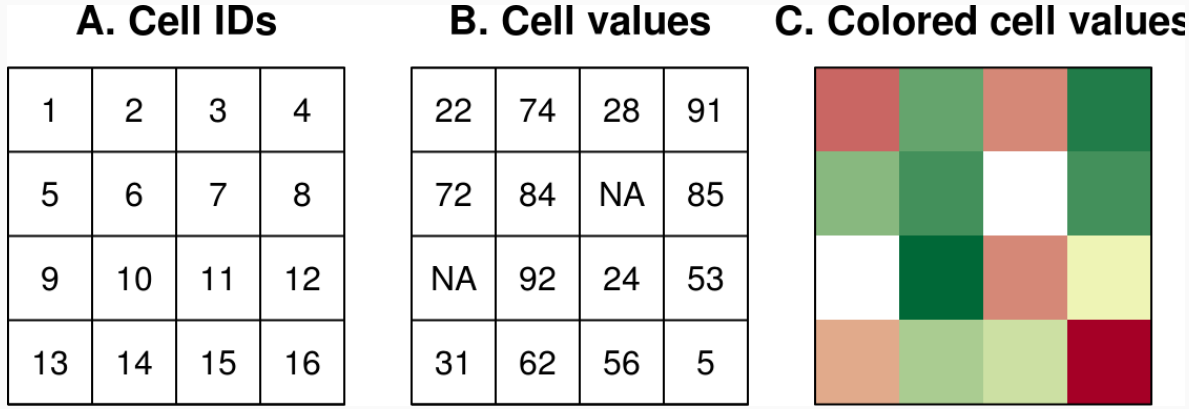
Introduction to the spatial data analysis in R

Jakub Nowosad

Institute of Geoecology and Geoinformation, Adam Mickiewicz University, Poznan, Poland

Spatial data representations: raster data

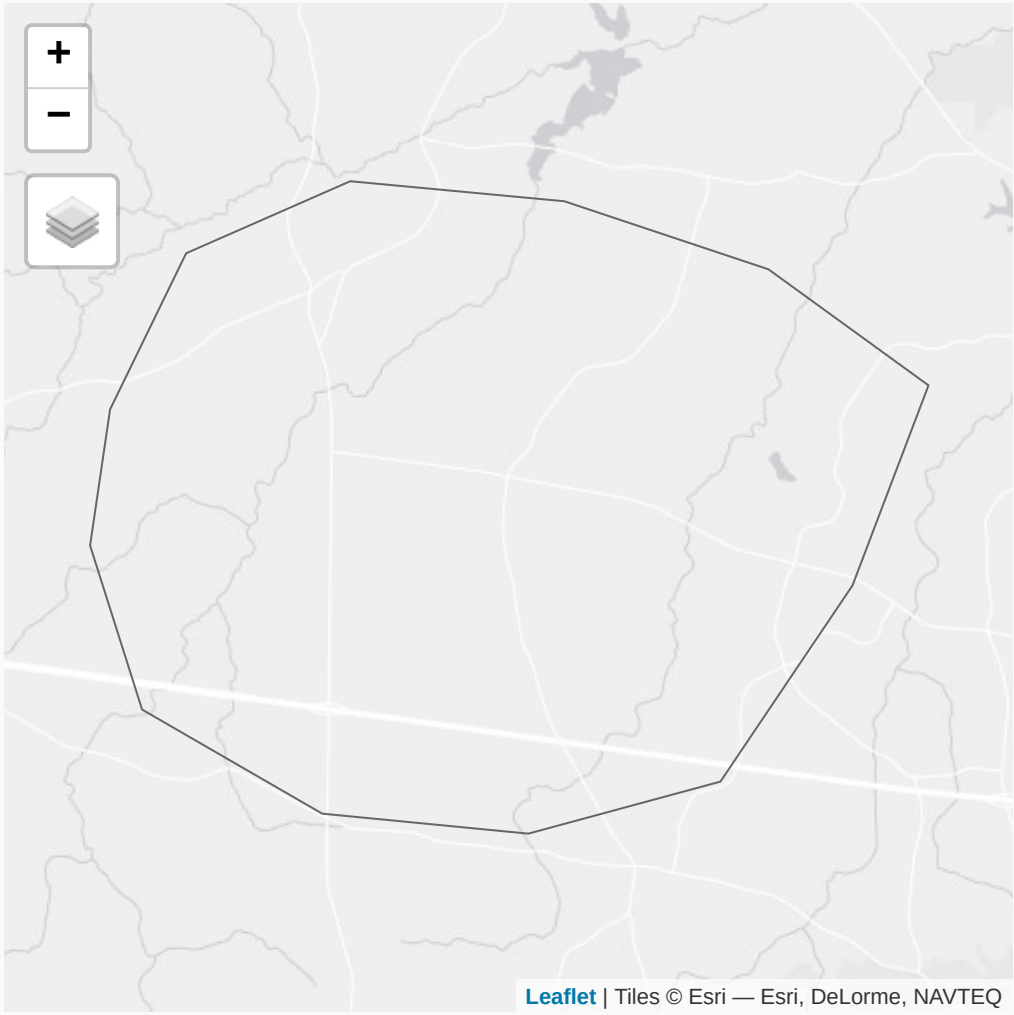
- The **raster** package
- Classes for spatial raster data: `RasterLayer`, and multilayer `RasterStack` and `RasterBrick`
- Raster data input/output
- Raster algebra and raster processing
- ?`raster-package`, **link1**, **link2**, and **link3**




```
library(raster)
elev_data <- raster("data/example_elevation.tif")
elev_data
```

```
## class      : RasterLayer
## dimensions : 550, 753, 414150  (nrow, ncol, ncell)
## resolution : 0.000322, 0.000266  (x, y)
## extent     : -82.41659, -82.17413, 33.45642, 33.60272  (xmin, xmax, ymin,
## crs        : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,
## source     : /home/jn/Science/conferences/2020/ialena-2020/data/example_e
## names      : example_elevation
## values     : 57, 178.9385  (min, max)
```

Spatial data representations: vector data




```
library(sf)
study_area <- read_sf("data/study_area.gpkg")
head(study_area)
```

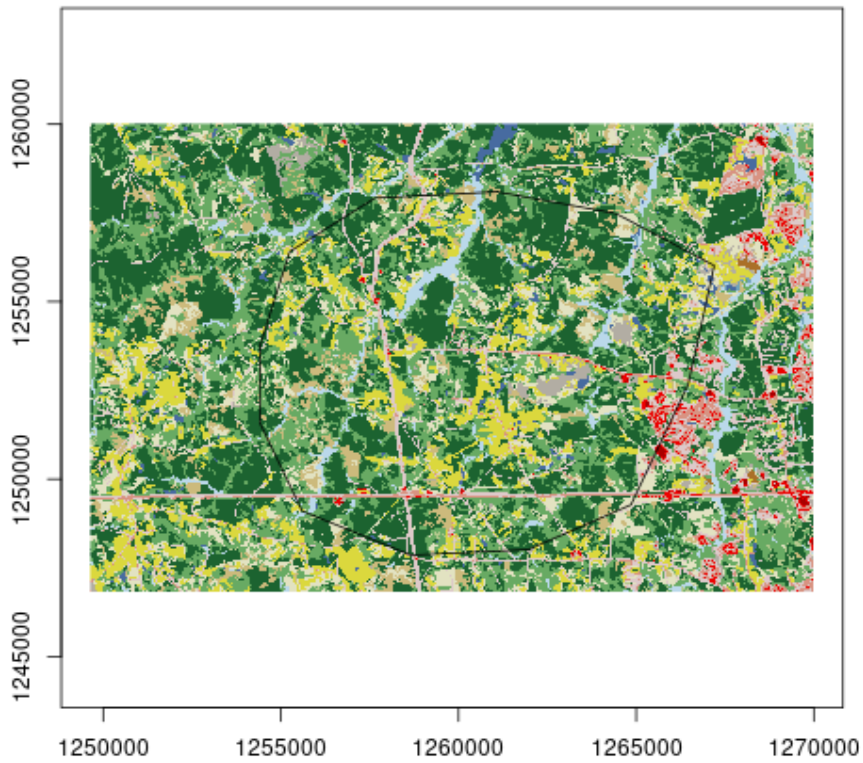
```
## Simple feature collection with 1 feature and 1 fiel
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 1254413 ymin: 1247861 xmax: 1
## projected CRS:  unnamed
## # A tibble: 1 x 2
##      id
##   <dbl>
## 1     1 ((1264853 1249278, 1266468 1252680, 126719
```

- The **sf** package
- A class system for spatial vector data
- Vector data input/output
- Transformations between different coordinate reference systems (CRS)
- Geometric operations
- **link1**, **link2**, **link3**

Spatial data reading

```r
library(sf)
library(raster)
study_area <- read_sf("data/study_area.gpkg")
lc_data <- raster("data/example_landscape.tif")
```

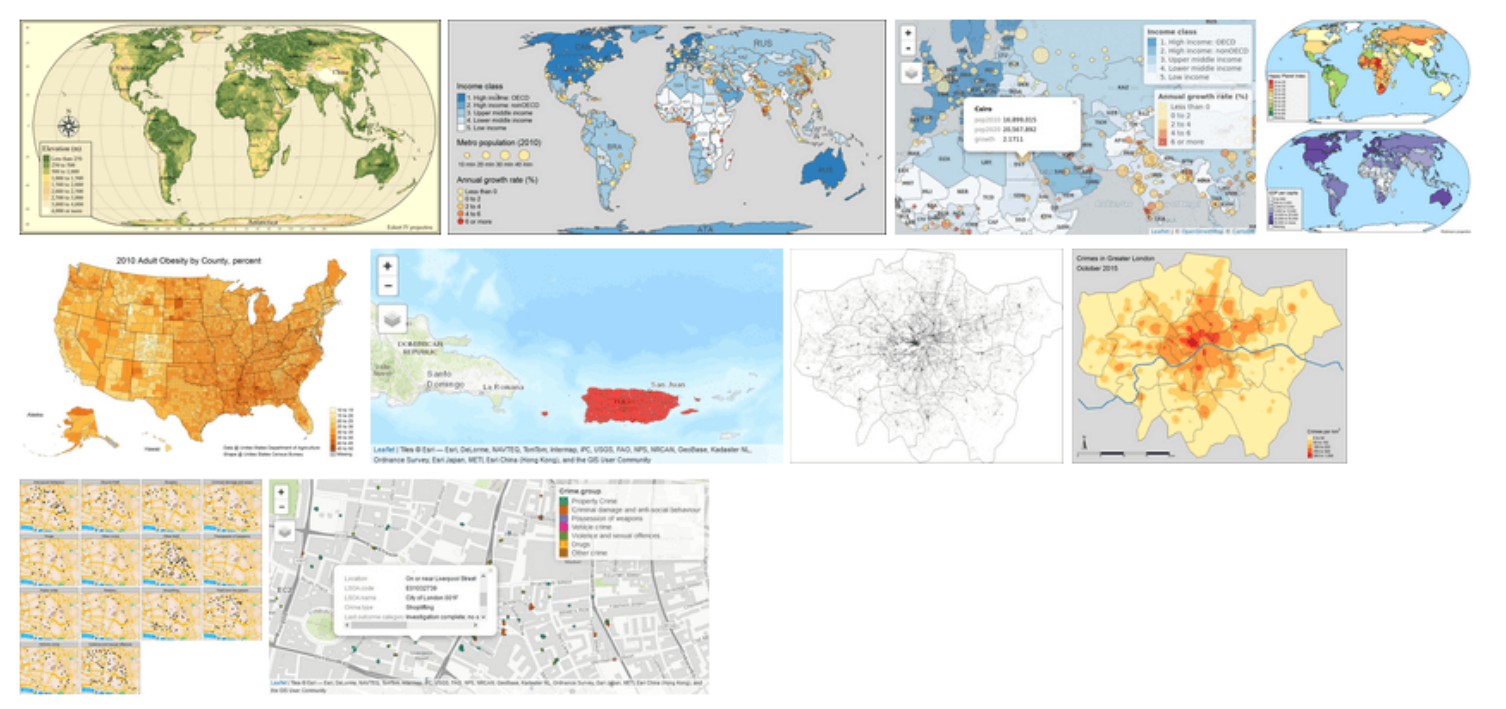
```r
plot(lc_data, axes = TRUE)
plot(st_geometry(study_area), add = TRUE)
```



Making maps in R: tmap

```
library(tmap)
```

- Thematic mapping package
- It allows you to create static maps, animated maps and interactive maps.
- It works by adding subsequent layers to visualize and then modifying them.
- **https://github.com/mtennekes/tmap** - list of additional materials about the **tmap** package



First practical part

Try to work through *Exercise_1_1.R*. The exercises will throw you in at the deep end - **you need to modify the provided code to improve the resulting map**.

During the exercise there are part where you are supposed to type your own solution indicated by the following structure:

```
# /Start Code/ #

print("Hello World") # This would be your code contribution

# /End Code/ #
```

In case you run into problems or have any questions, please contact us at any time via *Zoom* private chat.

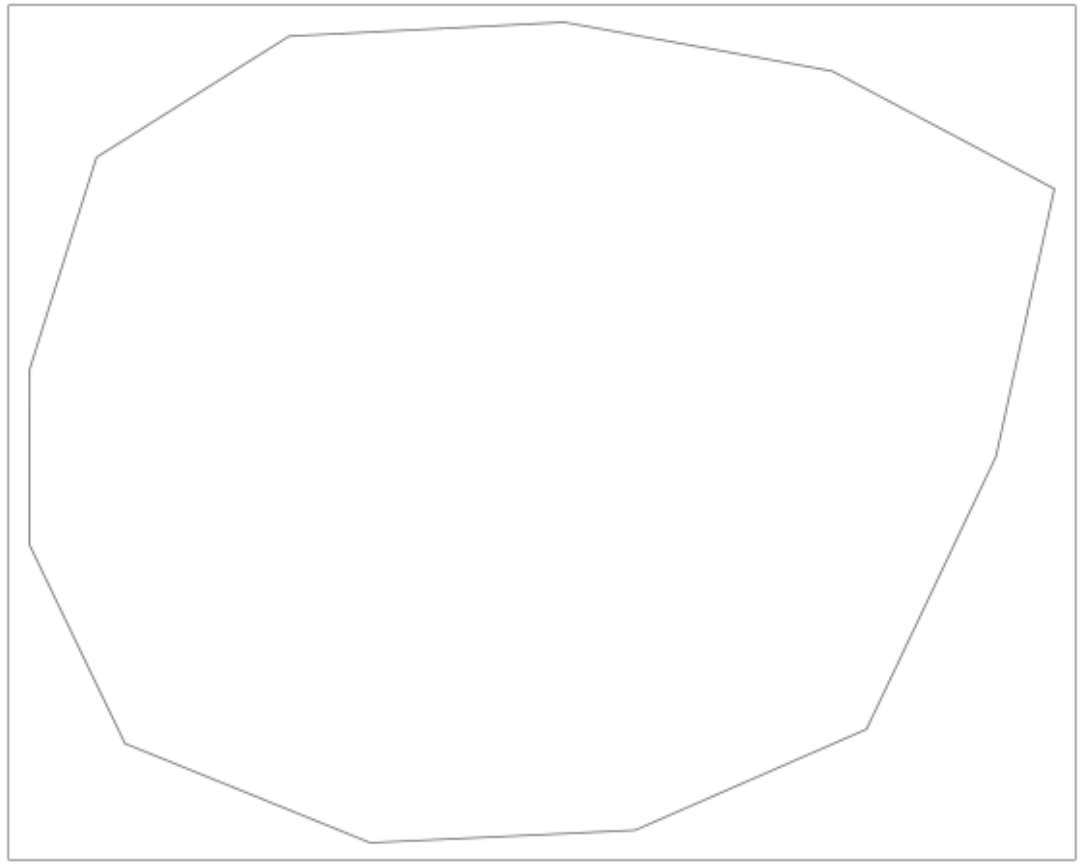
Making maps in R

```
tm_shape(study_area)
```

```
## Error: no layer elements defined after tm_shape
```

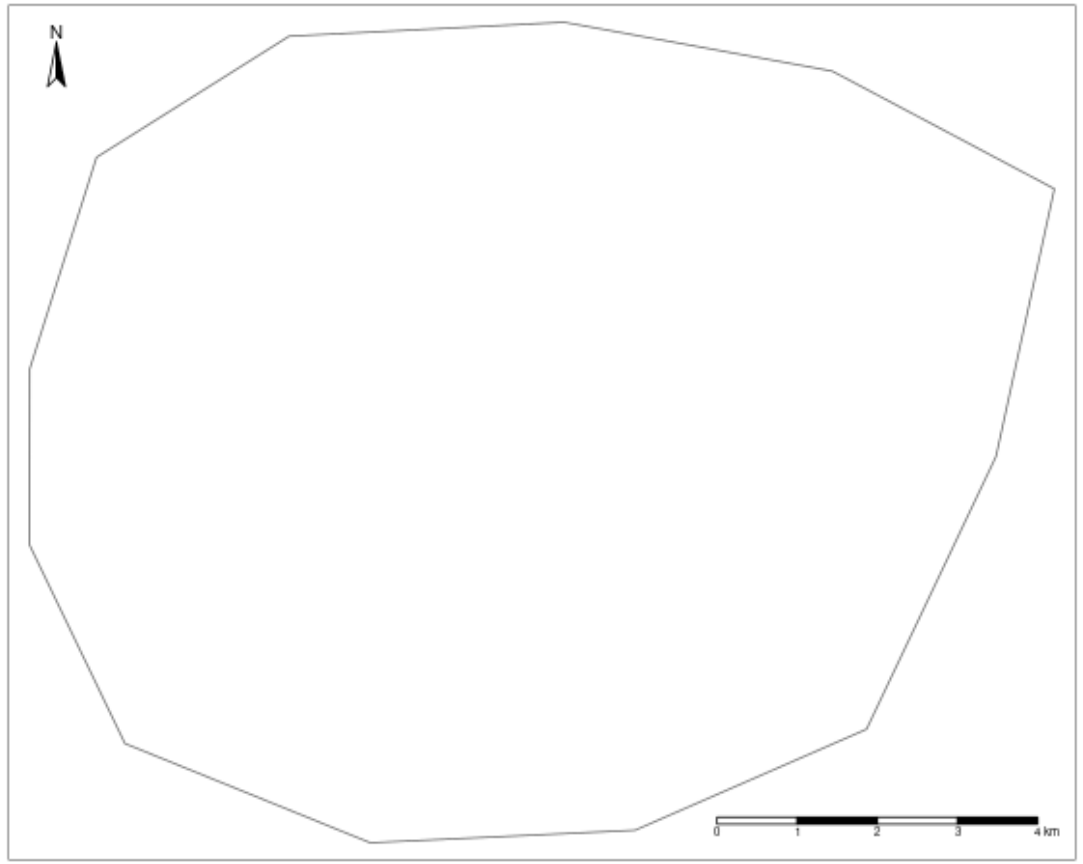
Making maps in R

```
tm_shape(study_area) +
  tm_borders()
```



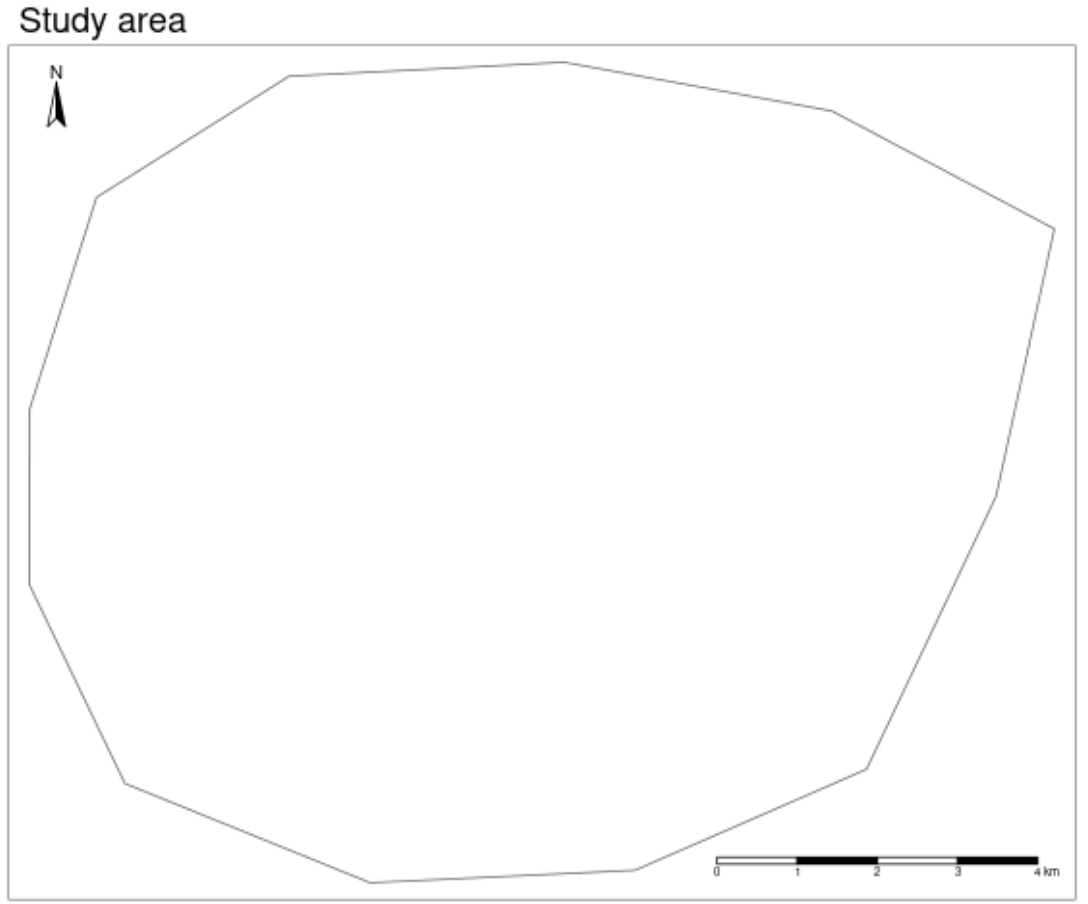
Making maps in R

```
tm_shape(study_area) +
  tm_borders() +
  tm_scale_bar(position = c("right",
                            "bottom")) +
  tm_compass(position = c("left", "top"))
```



Making maps in R

```
tm_shape(study_area) +
  tm_borders() +
  tm_scale_bar(position = c("right",
                            "bottom")) +
  tm_compass(position = c("left", "top")) +
  tm_layout(main.title = "Study area")
```



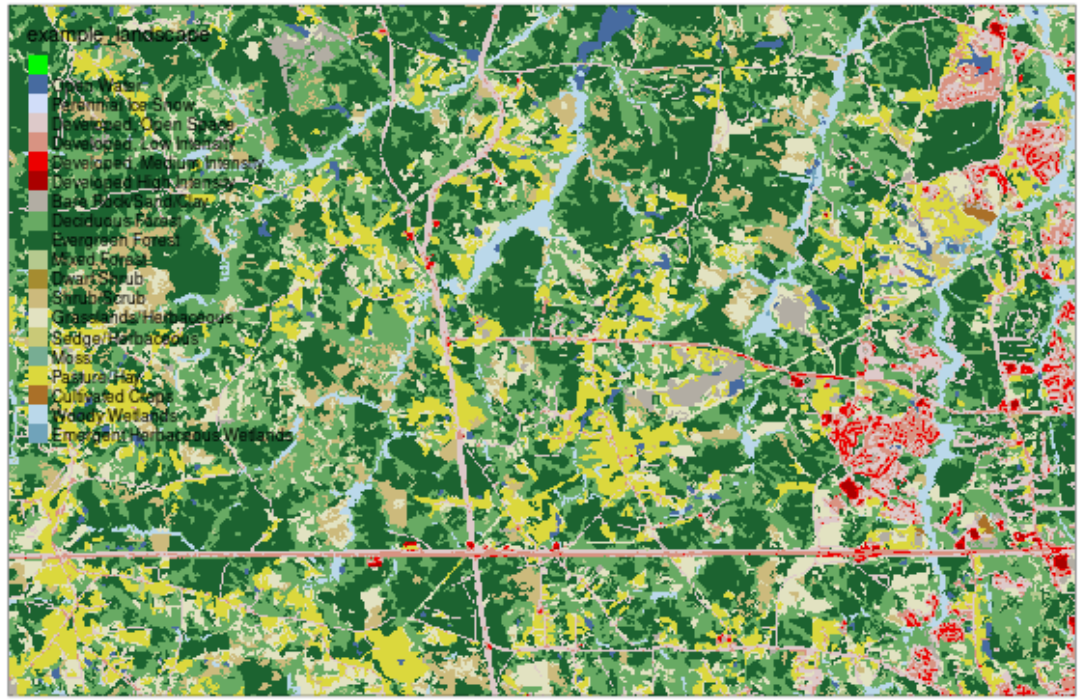
Making maps in R

```
tmap_mode("view")
tm_shape(study_area) +
  tm_borders() +
  tm_layout(main.title = "Study area")
```

```
tmap_mode("plot")
tm_shape(study_area) +
  tm_borders() +
  tm_layout(main.title = "Study area")
```

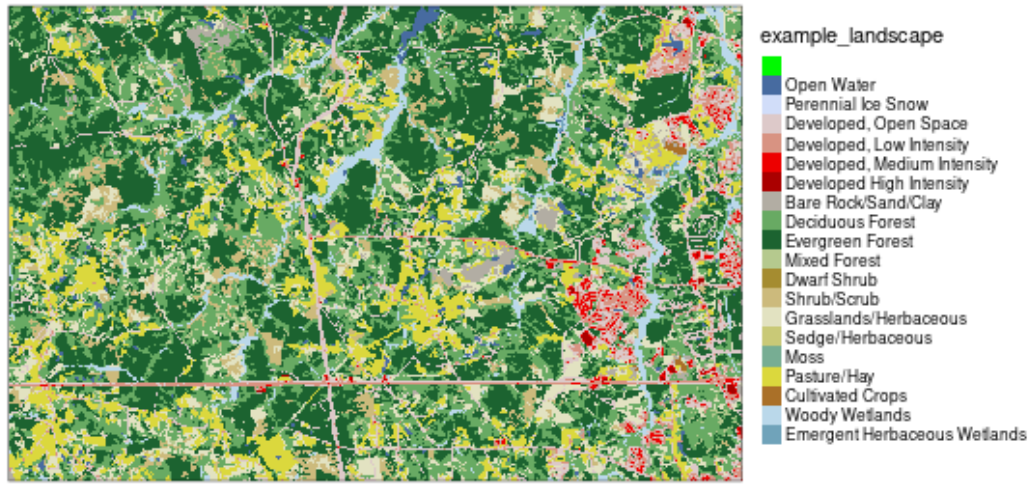
Making maps in R

```
tm_shape(lc_data) +
  tm_raster()
```



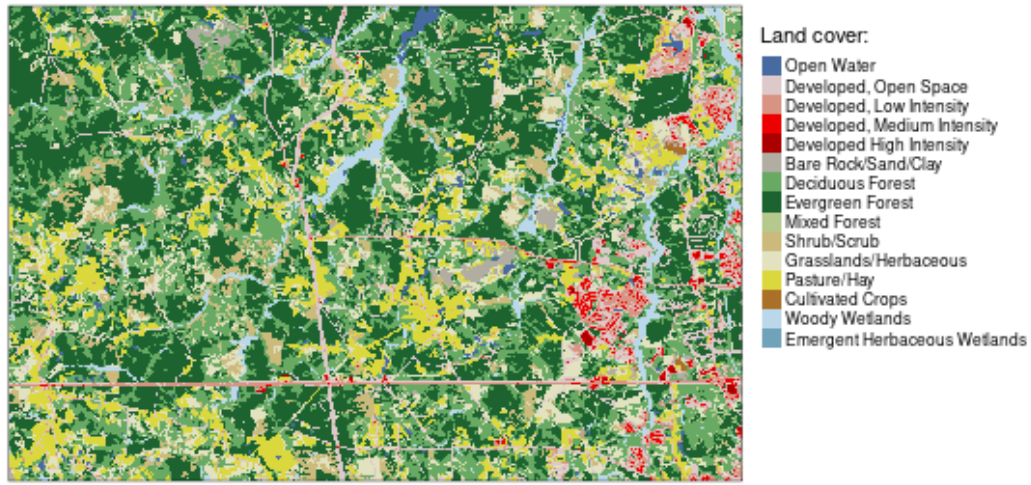
Making maps in R

```
tm_shape(lc_data) +
  tm_raster() +
  tm_layout(legend.outside = TRUE)
```



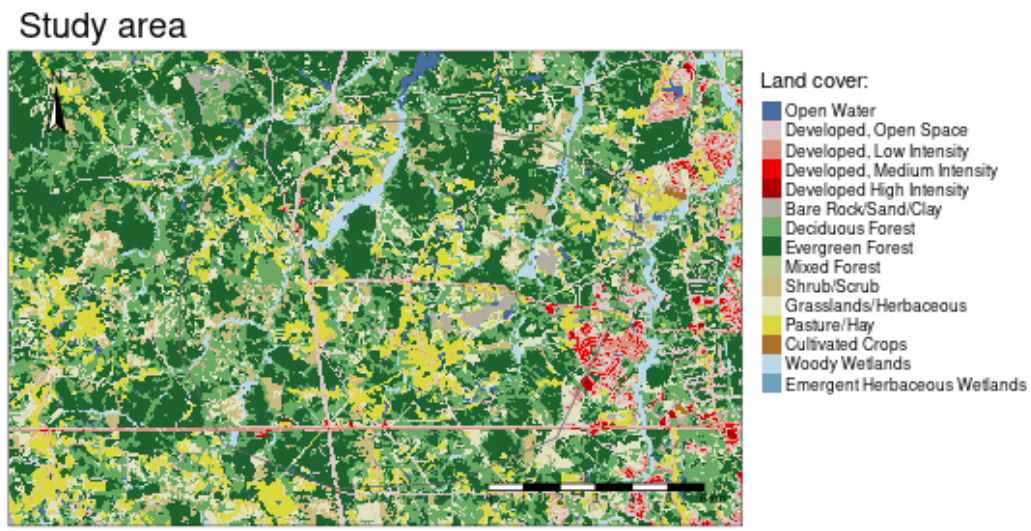
Making maps in R

```
tm_shape(lc_data) +
  tm_raster(drop.levels = TRUE,
            title = "Land cover:") +
  tm_layout(legend.outside = TRUE)
```



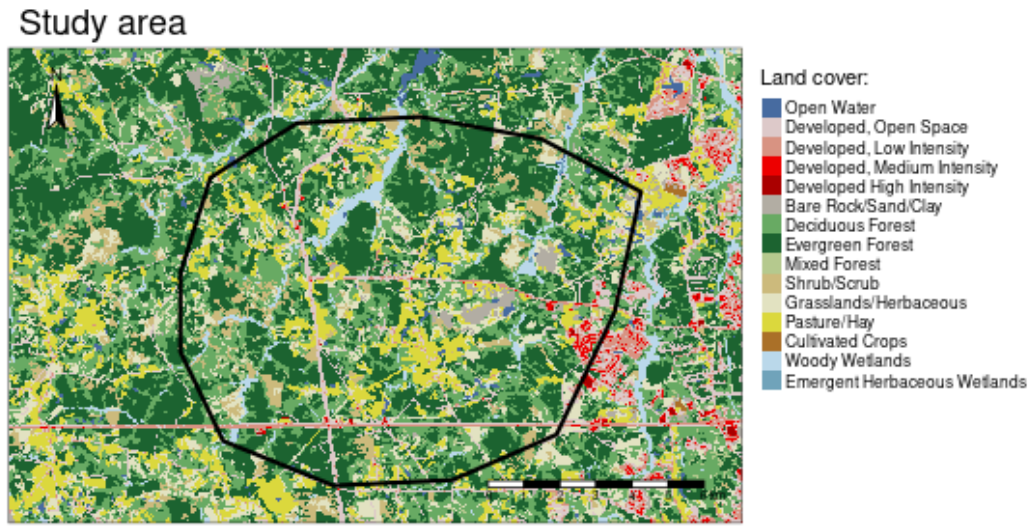
Making maps in R

```
tm_shape(lc_data) +
  tm_raster(drop.levels = TRUE,
            title = "Land cover:") +
  tm_shape(study_area) +
  tm_borders() +
  tm_scale_bar(position = c("right",
                            "bottom")) +
  tm_compass(position = c("left", "top")) +
  tm_layout(main.title = "Study area",
            legend.outside = TRUE)
```



Making maps in R

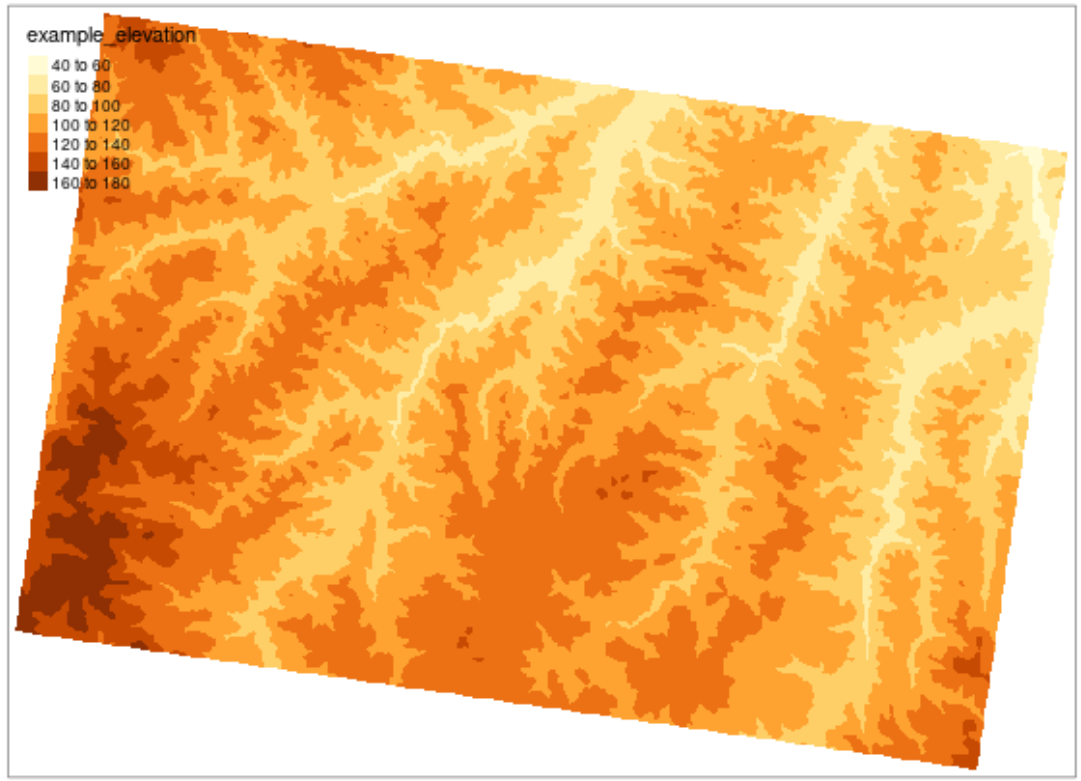
```
tm_shape(lc_data) +
  tm_raster(drop.levels = TRUE,
            title = "Land cover:") +
  tm_shape(study_area) +
  tm_borders(lwd = 3, col = "black") +
  tm_scale_bar(position = c("right",
                            "bottom")) +
  tm_compass(position = c("left", "top")) +
  tm_layout(main.title = "Study area",
            legend.outside = TRUE)
```



Making maps in R

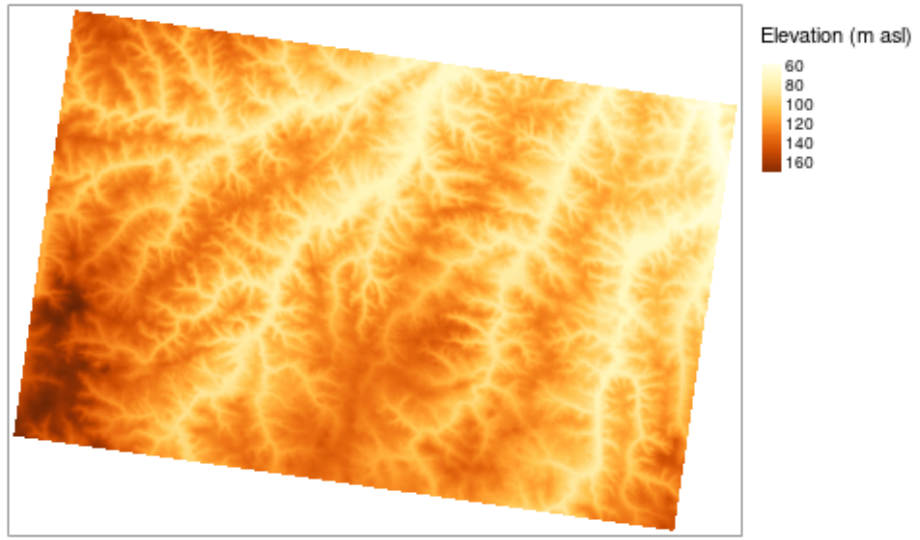
```
elev_data <- raster("data/example_elevation.tif")
```

```
tm_shape(elev_data) +
  tm_raster()
```



Making maps in R

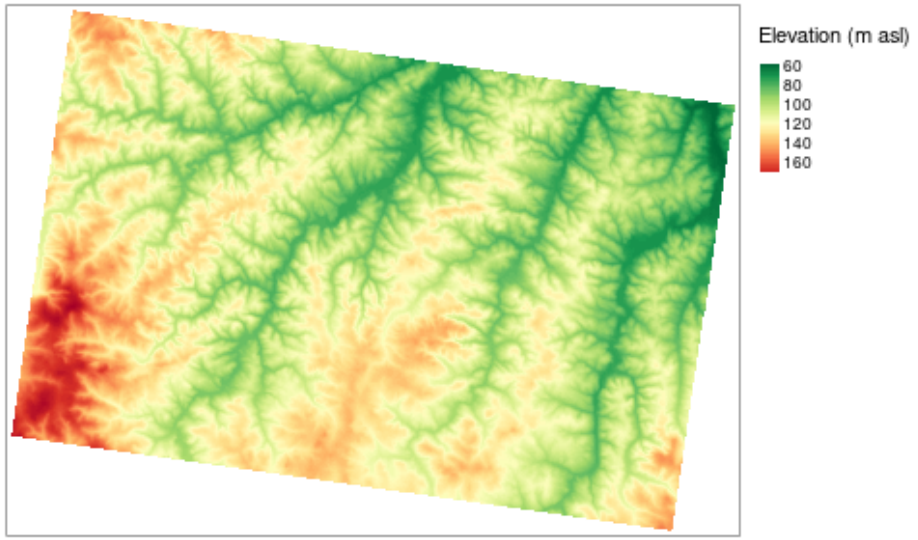
```
tm_shape(elev_data) +
  tm_raster(style = "cont",
            title = "Elevation (m asl)") +
  tm_layout(legend.outside = TRUE)
```



Making maps in R

```
tm_shape(elev_data) +
  tm_raster(style = "cont",
            title = "Elevation (m asl)",
            palette = "-RdYlGn") +
  tm_layout(legend.outside = TRUE)
```

```
# tmaptools::palette_explorer()
```

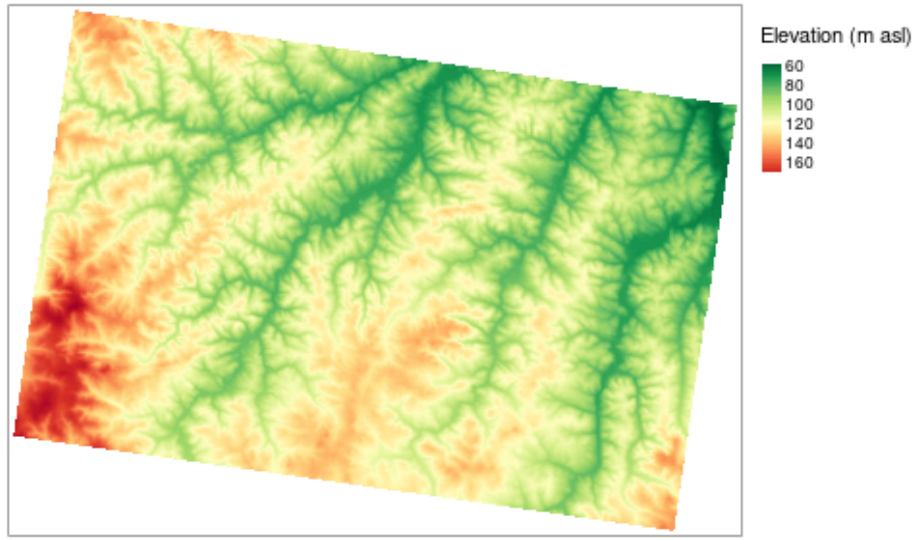


Making maps in R

```
map1 <- tm_shape(elev_data) +
  tm_raster(style = "cont",
            title = "Elevation (m asl)",
            palette = "-RdYlGn") +
  tm_layout(legend.outside = TRUE)
```

```
tmap_save(map1, "my_first_map.png")
```

```
tmap_save(map1, "my_first_map.html")
```



R spatial processing power

Attribute data operations:

- Vector attribute subsetting, aggregation and joins
- Creating new vector attributes
- Raster subsetting
- Summarizing raster objects

Spatial data operations:

- Spatial subsetting
- Topological relations
- Spatial joining
- Aggregation
- Map algebra
- Local, focal, and zonal raster operations

Geometry operations:

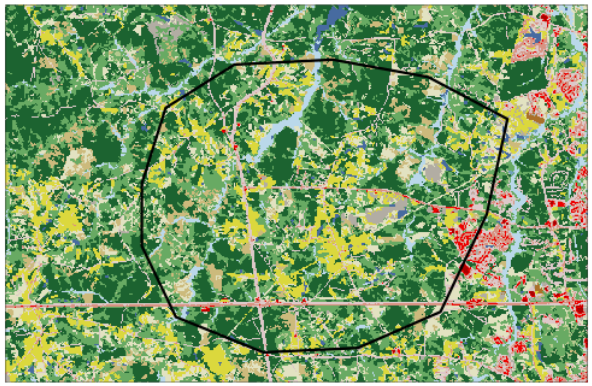
- Geometric operations on vector data
- Geometric operations on raster data
- **Vector-raster interactions**

Coordinate reference systems:

- Understanding map projections
- **Reprojecting spatial data**
- Modifying map projections

Vector-raster interactions: crop

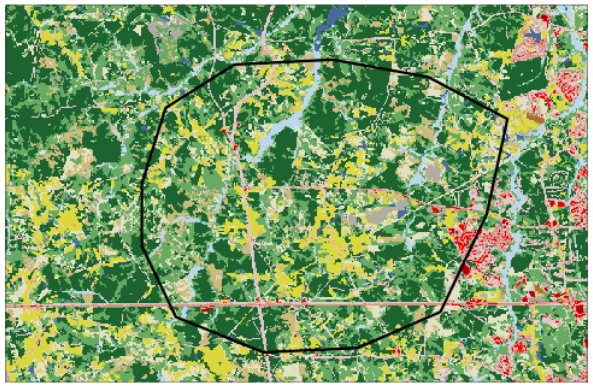
```
lc_data
```



```
lc_data_cropped <- crop(lc_data, study_area)
```

Vector-raster interactions: mask

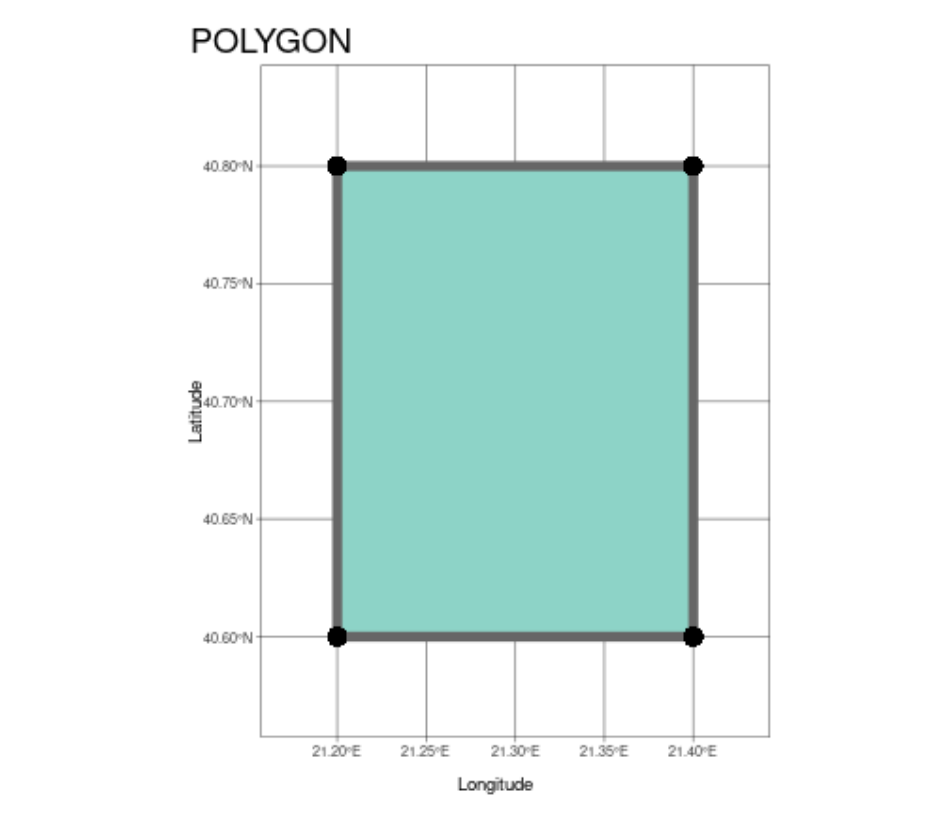
```
lc_data
```

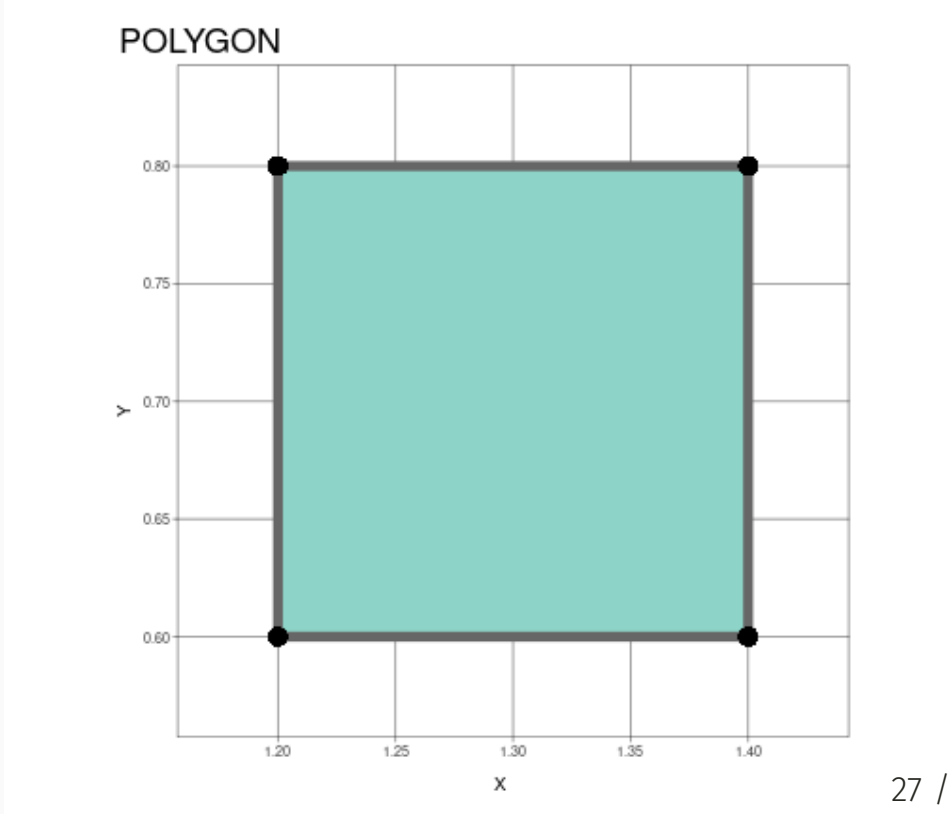


```
# lc_data_masked <- mask(crop(lc_data, study_area), study_area)
lc_data_masked <- mask(lc_data_cropped, study_area)
```

Spatial data coordinates

- **Geographic coordinates**: angles (degrees; longitude and latitude), pointing out locations on a spherical or ellipsoidal surface
- **Projected coordinates**: measured on a two-dimensional flat space (e.g. in metres; x and y), related to an ellipsoid by projection





Coordinate reference systems

```
st_crs(study_area)
```

```
## Coordinate Reference System:
##   User input: unnamed
##   wkt:
## BOUNDCRS[
##     SOURCECRS[
##         PROJCRS["unnamed",
##             BASEGEOGCRS["GRS 1980(IUGG, 1980)",
##                 DATUM["unknown",
##                     ELLIPSOID["GRS80",6378137,298.257222101,
##                         LENGTHUNIT["metre",1,
##                             ID["EPSG",9001]]]],
##                 PRIMEM["Greenwich",0,
##                     ANGLEUNIT["degree",0.0174532925199433]]],
##             CONVERSION["unnamed",
##                 METHOD["Albers Equal Area",
##                     ID["EPSG",9822]],
##                 PARAMETER["Latitude of 1st standard parallel",29.5,
##                     ANGLEUNIT["degree",0.0174532925199433],
##                     ID["EPSG",8823]],
##                 PARAMETER["Latitude of 2nd standard parallel",45.5,
```

```
crs(lc_data)
```

```
## CRS arguments:
##  +proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.5 +lat_2=
## +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_de
```

Reprojecting spatial data

- **A vast topic**
- **Geographic Coordinate Reference Systems:** data integration, web mapping. Usually WGS84 (EPSG: 4326)
- **Projected Coordinate Reference Systems:** data analysis, local maps. Often some CRS is already used in the project/by the institution
- You can find different CRSs and their representations at **https://epsg.io/**

Vector data:

1. `study_area2 <- st_transform(study_area, st_crs(another_sf_obj))`
2. `study_area2 <- st_transform(study_area, crs(lc_data))`
3. `study_area2 <- st_transform(study_area,4326)`
4. `study_area2 <- st_transform(study_area, "ESRI:54030")`

Raster data:

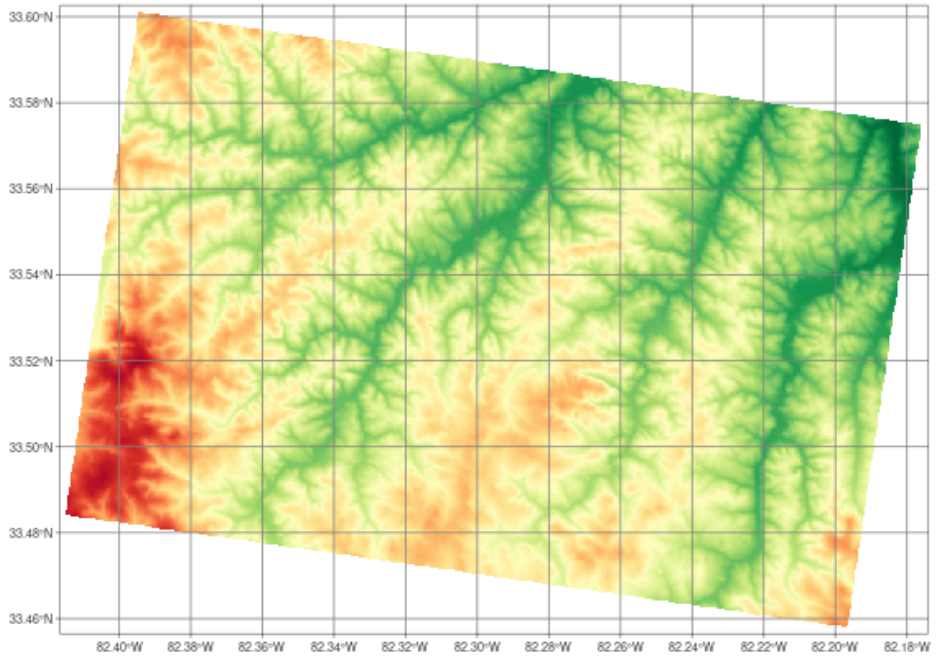
1. `elev_data2 <- projectRaster(elev_data, crs = "+proj=longlat +datum=WGS84 +no_defs")`
2. `elev_data2 <- projectRaster(elev_data, crs = lc_data)`
3. `elev_data2 <- projectRaster(elev_data, crs = st_crs(study_area)$proj4string)`

Method is important here:

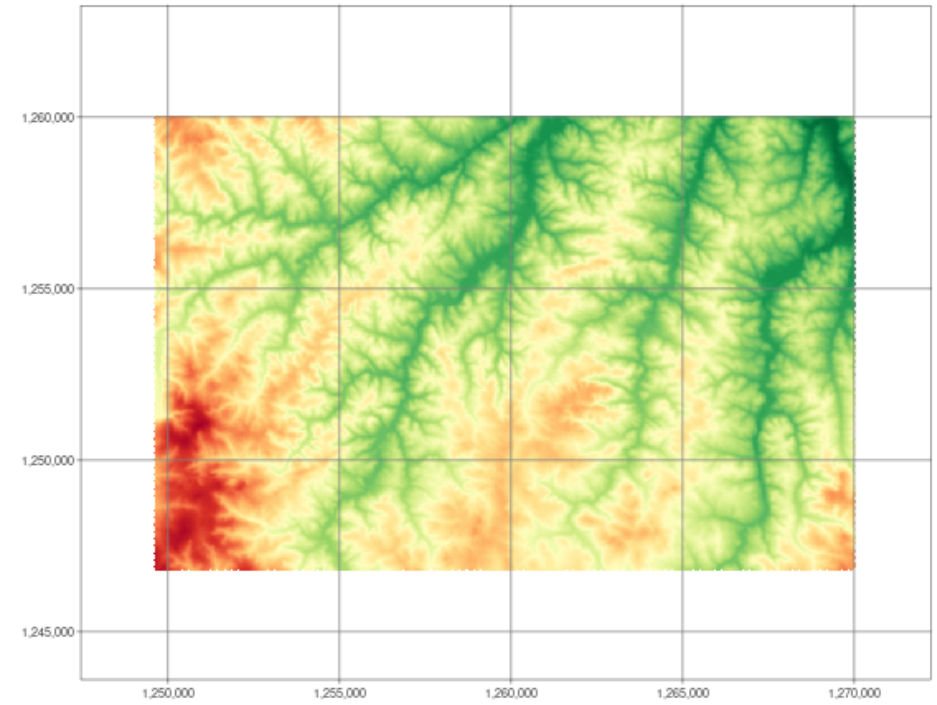
1. `elev_data2 <- projectRaster(elev_data, crs = "+proj=longlat +datum=WGS84 +no_defs", method = "bilinear")`
2. `lc_data2 <- projectRaster(lc_data, crs = "+proj=longlat +datum=WGS84 +no_defs", method = "ngb")`

Reprojecting spatial data

```
elev_data
```




```
elev_data2 <- projectRaster(elev_data,
                            crs = lc_data,
                            method = "bilinear")
```



Second practical part

Try to work through *Exercise_1_2.R*. The exercises will focus on understanding spatial data, interactions between raster and vector representation, and reprojecting spatial objects.

During the exercise there are part where you are supposed to type your own solution indicated by the following structure:

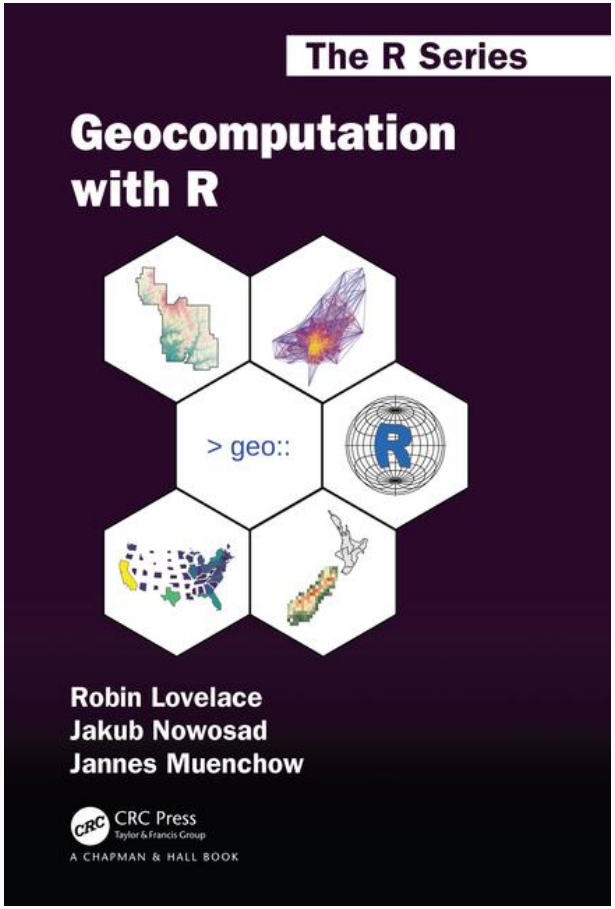
```
# /Start Code/ #

print("Hello World") # This would be your code contribution

# /End Code/ #
```

In case you run into problems or have any questions, please contact us at any time via *Zoom* private chat.

Summary



Recent talks:

- "The landscape of spatial data analysis in R": **video** and **slides**
- "Recent changes in R spatial and how to be ready for them": **video** and **slides**

Resources:

- **The Geocomputation with R website**: open source book, blog posts, workshop materials, and more
- **The Spatial Data Science with R website (focus on raster)**: materials focused on the **raster** package
- **The r-spatial website**: blog posts and a book-in-progress
- **The GIS Stackexchange website**: place to ask questions related to spatial data analysis in R
- `#rspatial` and `#geocompr` on Twitter

Break (30 minutes)

Break (30 minutes)

Part II

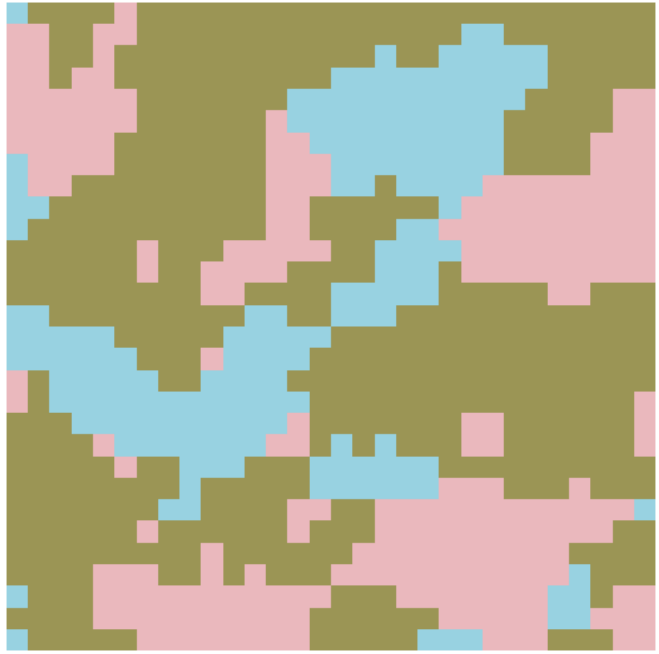
landscapemetrics: An open-source R tool to calculate landscape metrics

Maximilian H.K. Hesselbarth

Department of Ecosystem Modelling, University of Goettingen, Germany

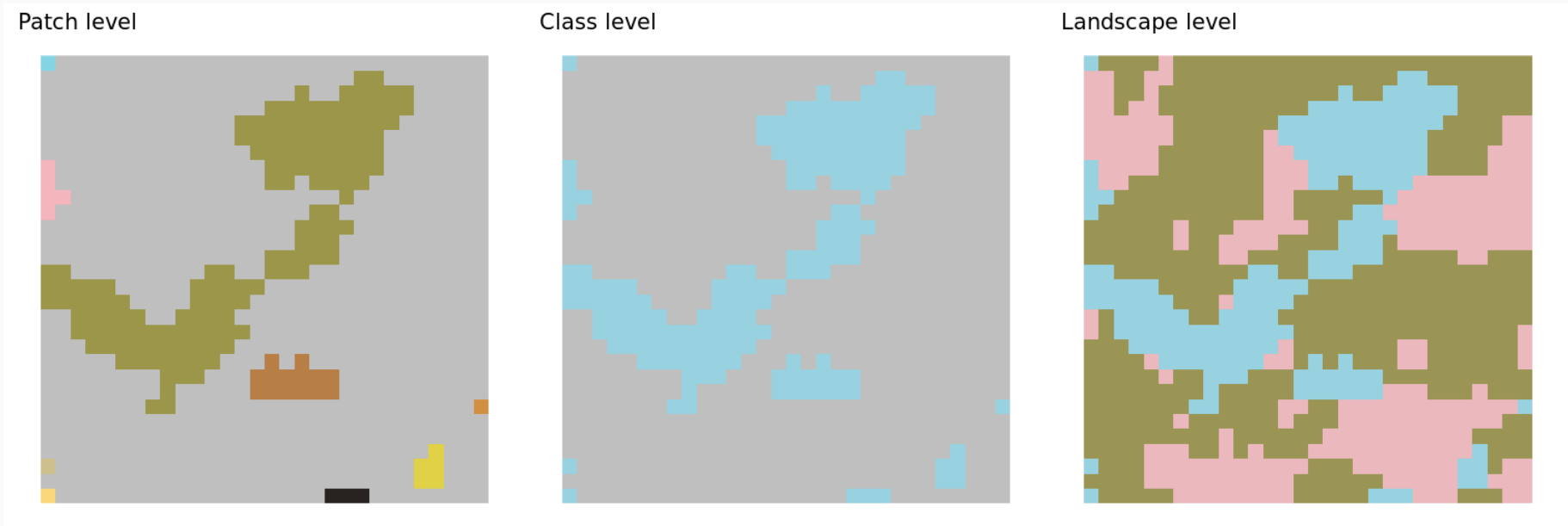
Introdution to landscape metrics

- Tools to quantify composition and configuration of landscapes
    - Composition: number and abundance of land-cover classes
    - Configuration: spatial arrangement of land-cover classes
- Designed for discrete patches of different land-cover classes



The R package *landscapemetrics*

- Based on *raster* package (Hijmans 2017)
- Allows to calculate landscape metrics in transparent and reproducible workflow
- Tidy (sensu Wickham 2014) and type-stable output of all metric functions
- Includes many utility functions for e.g. plotting or sampling



Getting ready to use *landscapemetrics*

Import and check data

```
# install.packages("landscapemetrics")

lc_data <- raster("data/example_lndscp.tif")
check_landscape(lc_data)
```

```
##   layer crs units   class n_classes OK
## 1     1  NA    NA integer         3  ?
```

List available metrics

```
list_lsm()
```

```
## # A tibble: 132 x 5
##   metric name                          type                  level function_name
##   <chr>  <chr>                         <chr>                 <chr> <chr>
## 1 area   patch area                    area and edge metric  patch lsm_p_area
## 2 cai    core area index               core area metric      patch lsm_p_cai
## 3 circle related circumscribing circle shape metric          patch lsm_p_circle
## 4 contig contiguity index              shape metric          patch lsm_p_contig
## 5 core   core area                     core area metric      patch lsm_p_core
## # … with 127 more rows
```

How to calculate landscape metrics

Calculate single metric

```
lsm_p_area(lc_data)
```

```
## # A tibble: 27 x 6
##   layer level class    id metric  value
##   <int> <chr> <int> <int> <chr>   <dbl>
## 1     1 patch     1     1 area   0.0001
## 2     1 patch     1     2 area   0.0005
## 3     1 patch     1     3 area   0.0148
## 4     1 patch     1     4 area   0.0001
## 5     1 patch     1     5 area   0.0001
## # … with 22 more rows
```

Combine several metrics to one *tibble*

```
class_mean <- lsm_c_area_mn(lc_data)
lsm_total <- lsm_l_ta(lc_data)

bind_rows(class_mean, lsm_total)
```

```
## # A tibble: 4 x 6
##   layer level     class    id metric    value
##   <int> <chr>     <int> <int> <chr>     <dbl>
## 1     1 class         1    NA area_mn 0.00199
## 2     1 class         2    NA area_mn 0.00173
## 3     1 class         3    NA area_mn 0.0120
## 4     1 landscape    NA    NA ta      0.09
```

Easy integration into larger workflows

Integrate easily into larger workflows

```
set.seed(42)

sample(x = c(1, 2, 3), prob = c(0.5, 0.25, 0.25), size = 2500, replace = TRUE) %>%
  matrix(nrow = 50, ncol = 50) %>%
  raster() %>%
  lsm_p_enn() %>%
  filter(value <= quantile(value, probs = 0.25) | value >= quantile(value, probs = 0.75)) %>%
  group_by(class) %>%
  summarise(n = n()) %>%
  arrange(-n)
```

```
## # A tibble: 3 x 2
##   class     n
##   <int> <int>
## 1     2   150
## 2     3   130
## 3     1    21
```

Calculation of multiple metrics

Easily calculate multiple metrics based on e.g. level and type

```
calculate_lsm(landscape = lc_data,
              level = "landscape", type = "diversity metric",
              classes_max = 5)
```

```
## # A tibble: 9 x 6
##   layer level     class    id metric    value
##   <int> <chr>     <int> <int> <chr>     <dbl>
## 1     1 landscape    NA    NA msidi     0.929
## 2     1 landscape    NA    NA msiei     0.845
## 3     1 landscape    NA    NA pr        3
## 4     1 landscape    NA    NA prd    3333.
## 5     1 landscape    NA    NA rpr      60
## # … with 4 more rows
```

First practical part

Try to work through *Exercise_2_1.R*. The exercise will teach you how to calculate single and multiple landscape metrics. Furthermore, you will learn how to integrate *landscapemetrics* into larger workflows using e.g. the *tidyverse* (Wickham et al. 2019) .

During the exercise there are part where you are supposed to type your own solution indicated by the following structure:

```
# /Start Code/ #

print("Hello World") # This would be your code contribution

# /End Code/ #
```

In case you run into problems or have any questions, please contact us at any time via *Zoom* private chat.

For a general overview of *landscapemetrics*, please see **https://r-spatialecology.github.io/landscapemetrics**.

Visualize patches

Easily visualize patches of several classes or the whole landscape

```
show_patches(augusta_nlcd, labels = FALSE, class = 42)
```

Further visualization

Visualize core areas of patches

```
show_cores(augusta_nlcd,
           labels = FALSE, class = 42)
```

Visualize landscape metrics on patch level

```
show_lsm(augusta_nlcd, labels = FALSE,
         class = 42, what = "lsm_p_area")
```

Sample metrics in buffer areas

- Samples metrics in a buffer area (sample plot) around sample points
- Also able to use sample lines or *sp*-SpatialPolygons (Pebesma & Bivand, 2005) .
- Metrics can be specified identical to `list_lsm()` and `calculate_lsm()`

Sample metrics in buffer areas

Slightly different output *tibble*

```
sample_points <- matrix(c(1253709, 1249455,
                          1261857, 1256055,
                          1252009, 1257055), ncol = 2, byrow = TRUE)

sample_lsm(landscape = augusta_nlcd, y = sample_points,
           size = 1000, what = "lsm_l_np")
```

```
## # A tibble: 3 x 8
##   layer level     class    id metric value plot_id percentage_inside
##   <int> <chr>     <int> <int> <chr>  <dbl>   <int>              <dbl>
## 1     1 landscape    NA    NA np       103       1               99.5
## 2     1 landscape    NA    NA np       116       2               99.5
## 3     1 landscape    NA    NA np        80       3               99.5
```

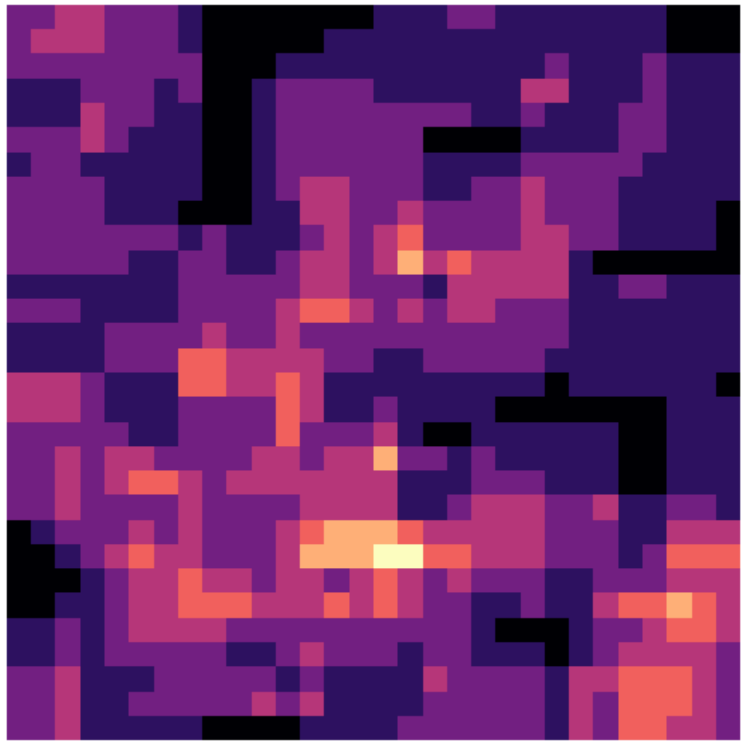
Moving window

- Show gradients and variability in the landscape
- Use matrix to specify moving window and focal cell

Moving window

Results in list with *raster* objects

```
window <- matrix(1, nrow = 5, ncol = 5)
window_result <- window_lsm(lc_data, window = window, what = "lsm_l_np")
```



Building blocks

Fast building blocks to create your own metrics

raster package (Hijmans 2017)

```
adj_raster <- function(x) {
  adjacencies <- adjacent(x,cells = 1:ncell(x))
  table(x[adjacencies[, 1]],x[adjacencies[, 2]])
}
adj_raster(lc_data)
```

```
##
##        1    2    3
##   1  520   43  137
##   2   43  704  184
##   3  137  184 1528
```

```
# A tibble: 2 x 3
expression                    mean        mem_alloc
<chr>                         <bch:tm>    <bch:byt>
1 adj_raster(landscape)       8900 µs     1470 KB
2 get_adjacencies(landscape)  562 µs      6 KB
```

landscapemetrics package

```
get_adjacencies(lc_data)
```

```
## [[1]]
##     1   2    3
## 1 520  43  137
## 2  43 704  184
## 3 137 184 1528
```

Second practical part

Try to work through *Exercise_2_2.R*. The exercise will teach you how to use all utility functions of the *landscapemetrics* package.

During the exercise there are parts where you are supposed to code your own solution indicated by the following structure:

```
# /Start Code/ #

print("Hello World") # This would be your code contribution

# /End Code/ #
```

In case you run into problems or have any questions, please contact us at any time via *Zoom* private chat.

For a general overview of *landscapemetrics*, please see **https://r-spatialecology.github.io/landscapemetrics**.

Citation

- Questions, feature request bugs at **https://github.com/r-spatialecology/landscapemetrics/issues**

```
citation("landscapemetrics")
```

```
##
## To cite landscapemetrics or acknowledge its use, please cite this
## Software note, substituting the version of the application that you
## used for 'version 0'.
##
##   Hesselbarth, M.H.K., Sciaini, M., With, K.A., Wiegand, K., Nowosad,
##   J. 2019. landscapemetrics: an open-source R tool to calculate
##   landscape metrics. - Ecography 42:1648-1657(ver. 0).
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {landscapemetrics: an open-source R tool to calculate landscape metrics},
##     author = {Maximilian H.K. Hesselbarth and Marco Sciaini and Kimberly A. With and Kerstin Wiegand and Ja
##     journal = {Ecography},
##     volume = {42},
##     year = {2019},
##     pages = {1648-1657},
##   }
```

Summary:

- R gives a **"console-based ability to seamlessly switch between geographic and non-geographic data processing, modeling and visualization tasks"**
- **landscapemetrics** allows for calculating landscape metrics for categorical landscape patterns in a tidy workflow
- We encourage bug reports, new metrics or functions suggestions, and code contributions. For more information see **CONTRIBUTING**
- This workshop is just a tip of an iceberg (of possibilities), *but you do not need to know it all*
- Questions?

Contact:

jakub_nowosad

MHKHesselbarth

Resources:

https://nowosad.github.io

https://mhesselbarth.rbind.io/

References

Hijmans, R.J., 2019. raster: Geographic data analysis and modeling. R package version 2.9-5. **https://cran.r-project.org/package=raster**.

McGarigal, K., Cushman, S.A., Ene, E., 2012. FRAGSTATS v4: Spatial pattern analysis program for categorical and continuous maps. Computer software program produced by the authors at the University of Massachusetts, Amherst. **http://www.umass.edu/landeco/research/fragstats/fragstats.html**. University of Massachusetts, Amherst.

Pebesma, E.J., Bivand, R.S., 2005. Classes and methods for spatial data in R. R News 5(2).

Wickham, H., 2014. Tidy Data. J. Stat. Softw. 59, 1–23.