Lab 1: Introduction

Statistical methods for spatial data analysis

Joaquin Cavieres

1. My first map

In this lab we will see the basic concepts associated with handling spatial and spatiotemporal data. Here also we will see a set of packages (libraries) that are important for spatial data science in `R`.

```
library(tidyverse)
library(sf)
system.file("gpkg/nc.gpkg", package="sf") |> read_sf() -> nc
            nc.32119 <- st_transform(nc, 'EPSG:32119')
            nc.32119 |>
            select(BIR74) |>
            plot(graticule = TRUE, axes = TRUE)
```

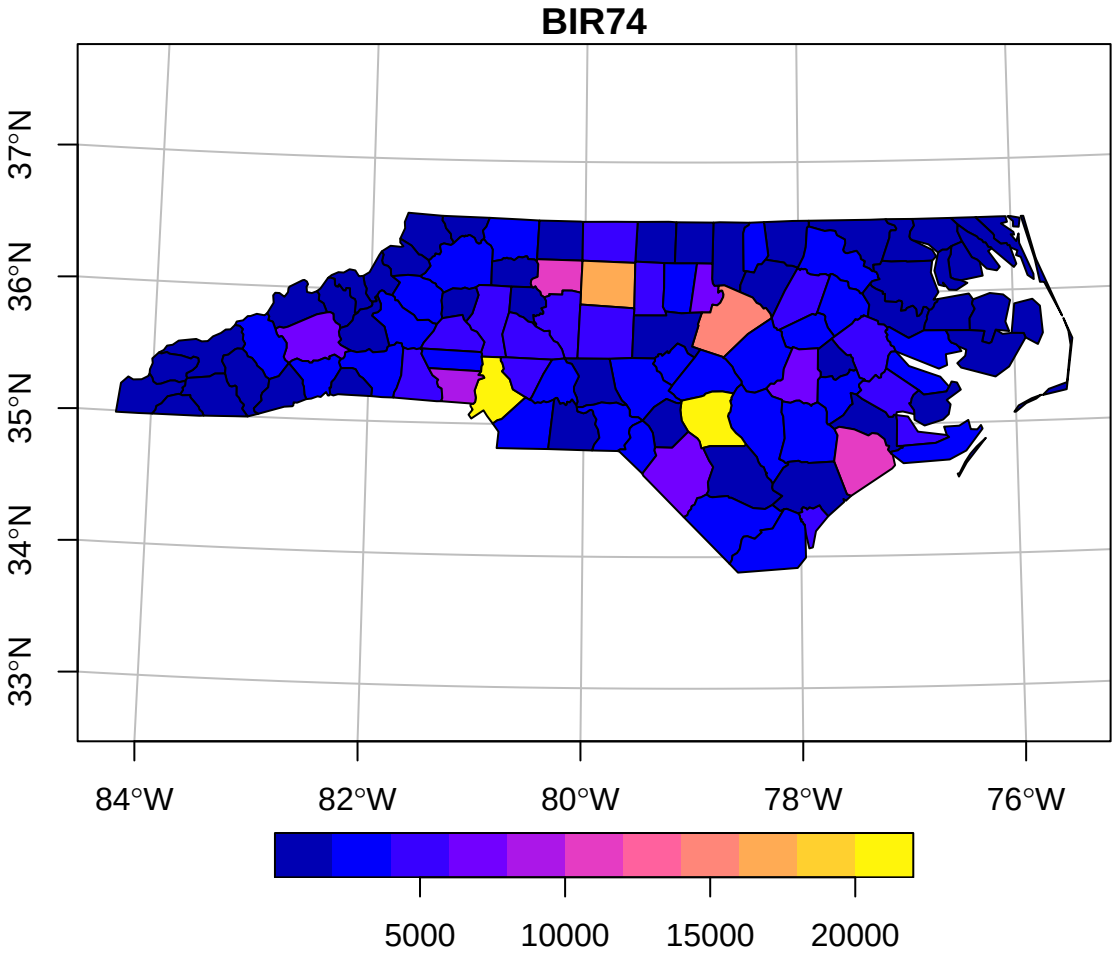


Figura 1: A first map: birth counts 1974-78, North Carolina counties

What are the graphical elements to consider from that graph?

- Polygons are drawn with a black outline and filled with colours chosen according to a variable BIR74, whose name is in the title
- A legend key explains the meaning of the colours, and has a certain colour palette and colour breaks, values at which colour changes
- The background of the map shows curved lines with constant latitude or longitude (graticule)
- The axis ticks show the latitude and longitude values

Remember that:

Polygons are a particular form of geometry (spatial geometries → points, lines, polygons, pixels). In summary, Polygons consist of sequences of points, connected by straight lines. As can be seen from Figure 1.1, lines of equal latitude and longitude do not form straight lines, indicating that some form of projection took place before plotting.

Other important characteristics is the color in the previous map. Those colors are derived from numeric values of a variable, BIR74, which has a single value associated with each geometry or feature. In this case, BIR74 refers to birth counts, meaning counts over the region. This implies that the count does not refer to a value associated with every point inside the polygon, which the continuous colour might suggest, but rather measures an integral (sum) over the polygon.

Ok, but how is structured our data? We can see it using the functions `summary` in `R`

```
nc |> select(AREA, BIR74, SID74) |> print(n = 3)
```

```
## Simple feature collection with 100 features and 3 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## Geodetic CRS:  NAD27
## # A tibble: 100 x 4
##    AREA BIR74 SID74                                                    geom
##   <dbl> <dbl> <dbl>                                          <MULTIPOLYGON [°]>
## 1 0.114  1091     1 (((-81.47276 36.23436, -81.54084 36.27251, -81.56198 36.273~
## 2 0.061   487     0 (((-81.23989 36.36536, -81.24069 36.37942, -81.26284 36.405~
## 3 0.143  3188     5 (((-80.45634 36.24256, -80.47639 36.25473, -80.53688 36.256~
## # ... with 97 more rows
```

where we get the following information:

- The dataset has 100 features (records) and 3 fields (attributes)
- The geometry type is MULTIPOLYGON
- The geometry type has dimension XY, indicating that each point will consist of 2 coordinate values the range of x and y values of the geometry
- The coordinate reference system (CRS) is geodetic, with coordinates in degrees longitude and latitude associated to the NAD27 datum
- The three selected attribute variables are followed by a variable geom of type MULTIPOLYGON with unit degrees that contains the polygon information

From the data set we can create more advance graphs, for example:

```
year_labels <- c("SID74" = "1974 - 1978", "SID79" = "1979 - 1984")
nc.32119 |> select(SID74, SID79) |> pivot_longer(starts_with("SID")) -> nc_longer
ggplot() + geom_sf(data = nc_longer, aes(fill = value), linewidth = 0.4) +
  facet_wrap(~ name, ncol = 1, labeller = labeller(name = year_labels)) +
  scale_y_continuous(breaks = 34:36) +
  scale_fill_gradientn(colors = sf.colors(20)) +
  theme(panel.grid.major = element_line(color = "white"))
```

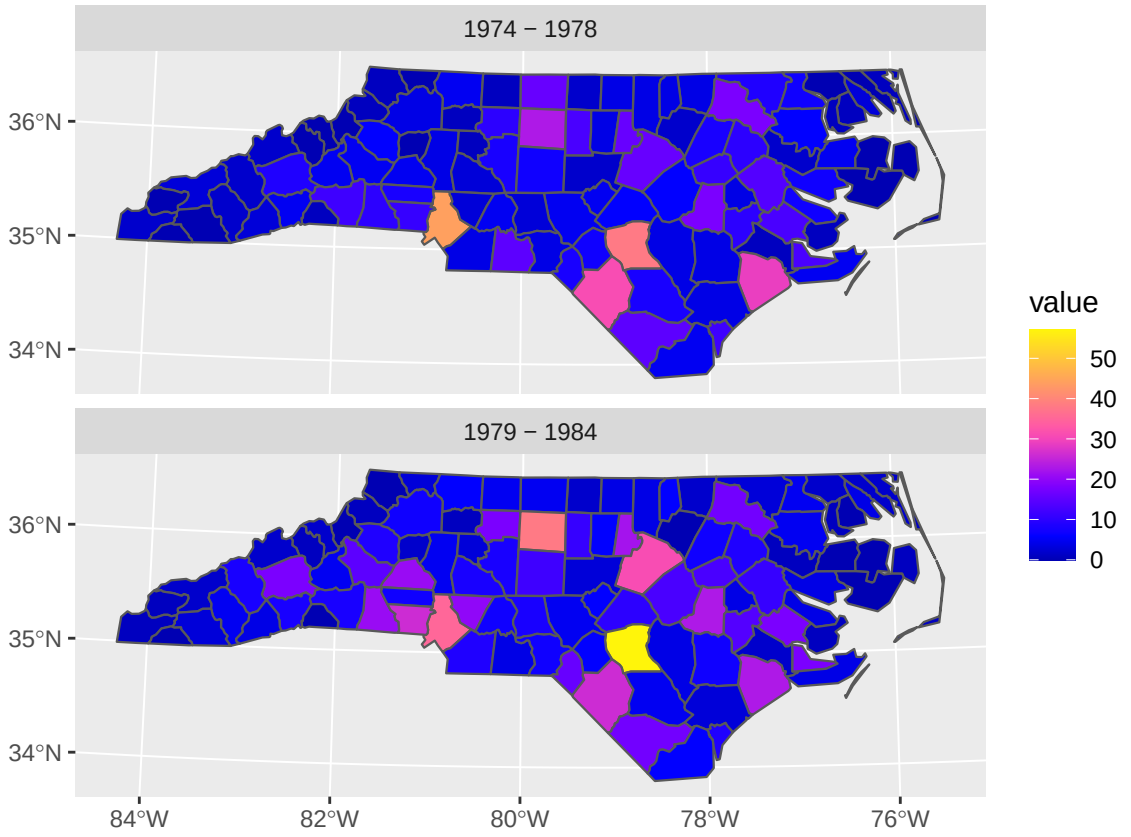


Figura 2: Facet maps of sudden infant death syndrome counts, 1974-78 and 1979-84, North Carolina counties

or an interactive map (only for HTML documents!):

```
library(mapview) |> suppressPackageStartupMessages()
mapviewOptions(fgb = FALSE)
nc.32119 |> mapview(zcol = "BIR74", legend = TRUE, col.regions = sf.colors)
```

2. Coordinate reference systems

Considering the Figure 1, the grey lines denote the *graticule*, that is, a grid with lines along constant latitude or longitude. Also you can see that the grey lines are not straight, which indicates that a projection of the data was used.

The coordinates in the Figure 1 are *ellipsoidal* and they are associated with a particular *datum* (here NAD27). Projections describe a relationship between:

- Ellipsoidal coordinates → are expressed in degrees latitude and longitude, pointing to locations on a shape approximating the Earth's shape (an ellipsoid or spheroid)
- Projected coordinates → are coordinates on a flat, two-dimensional coordinate system, used when plotting maps.

3. Raster and vector data

- Vector data: Point coordinates which describes the "exact" locations that can be anywhere. Polygon, point and line geometries are examples of thys types of data.
- Raster data: Describe data where values are aligned on a raster, meaning on a regularly laid out lattice of usually square pixels.

```
library(stars)
par(mfrow = c(2, 2))
par(mar = rep(1, 4))
tif <- system.file("tif/L7_ETMs.tif", package = "stars")
x <- read_stars(tif)[,,,1]
image(x, main = "(a)")
image(x[,1:10,1:10], text_values = TRUE, border = 'grey', main = "(b)")
image(x, main = "(c)")
set.seed(131)
pts <- st_sample(st_as_sfc(st_bbox(x)), 3)
plot(pts, add = TRUE, pch = 3, col = 'blue')
image(x, main = "(d)")
plot(st_buffer(pts, 500), add = TRUE, pch = 3, border = 'blue', col = NA, lwd = 2)
```

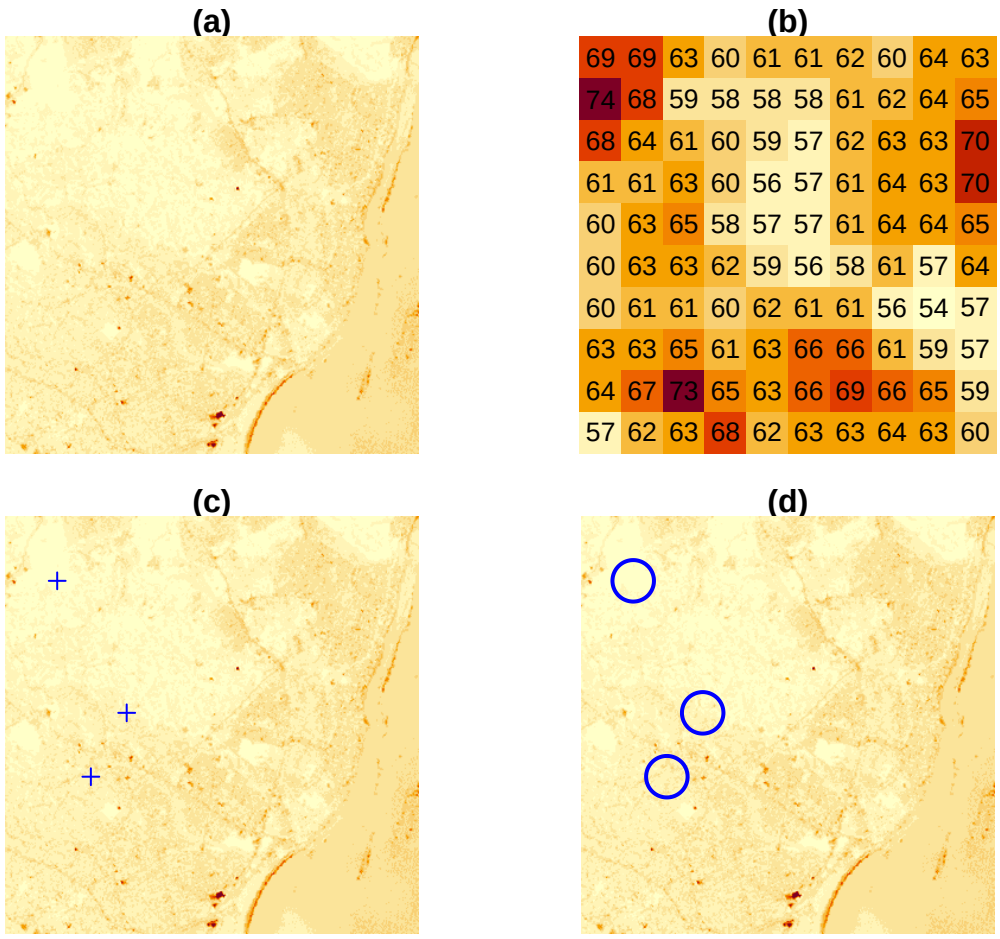


Figura 3: Raster maps (Olinda, Atlantic coast of Brazil): Landsat-7 blue band, with color values derived from data values (a), the top-left 10 X 10 sub-image from (a) with numeric values shown (b), and overlayed by two different types of vector data: three sample points (c), and a 500m radius around the points represented as polygons (d)

we can combine vector and raster data in different ways (plot 4 (c) and 4 (d)).

```
st_extract(x, pts) # query at points
```

```
## Simple feature collection with 3 features and 1 field
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 290019 ymin: 9114499 xmax: 291693.3 ymax: 9119219
## Projected CRS: SIRGAS 2000 / UTM zone 25S
##   L7_ETMs.tif                 geometry
## 1          80 POINT (290829.6 9114499)
## 2          58   POINT (290019 9119219)
## 3          63 POINT (291693.3 9116038)
```

```r
aggregate(x, st_buffer(pts, 500), FUN = mean) |> st_as_sf() # aggregate over circles
```

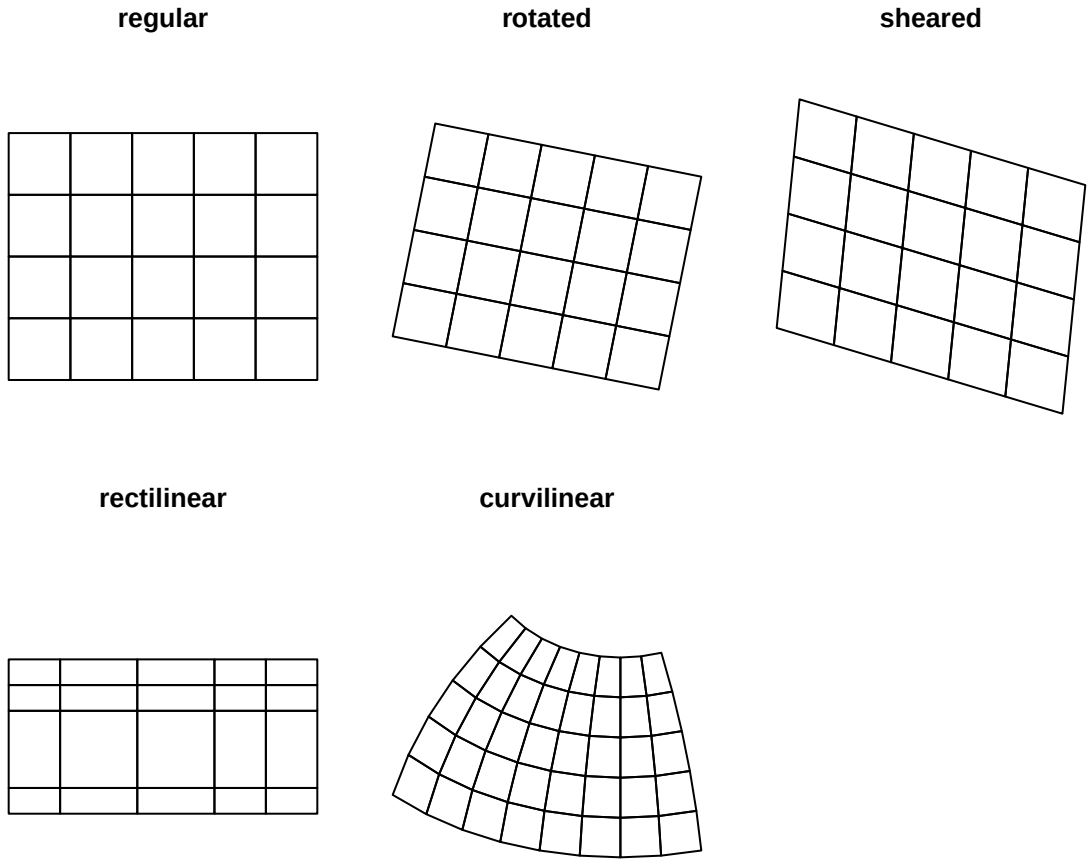
```
## Simple feature collection with 3 features and 1 field
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 289519 ymin: 9113999 xmax: 292193.3 ymax: 9119719
## Projected CRS: SIRGAS 2000 / UTM zone 25S
##         V1                       geometry
## 1 77.18201 POLYGON ((291329.6 9114499,...
## 2 60.12603 POLYGON ((290519 9119219, 2...
## 3 71.57808 POLYGON ((292193.3 9116038,...
```

4. Raster types

There are different dimensions for raster files, and its main characteristic is to relate rows and columns to spatial coordinates.

```r
x <- 1:5
y <- 1:4
d <- st_dimensions(x = x, y = y, .raster = c("x", "y"))
m <- matrix(runif(20),5,4)
r1 <- st_as_stars(r = m, dimensions = d)

r <- attr(d, "raster")
r$affine <- c(0.2, -0.2)
attr(d, "raster") = r
r2 <- st_as_stars(r = m, dimensions = d)

r <- attr(d, "raster")
r$affine <- c(0.1, -0.3)
attr(d, "raster") = r
r3 = st_as_stars(r = m, dimensions = d)

x <- c(1, 2, 3.5, 5, 6)
y <- c(1, 1.5, 3, 3.5)
d <- st_dimensions(x = x, y = y, .raster = c("x", "y"))
r4 <- st_as_stars(r = m, dimensions = d)

grd <- st_make_grid(cellsize = c(10,10), offset = c(-130,10), n = c(8,5),
                    crs = st_crs('OGC:CRS84'))
r5 <- st_transform(grd, "+proj=laea +lon_0=-70 +lat_0=35")

par(mfrow = c(2,3), mar = c(0.1, 1, 1.1, 1))
r1 <- st_make_grid(cellsize = c(1,1), n = c(5,4), offset = c(0,0))
```

```
plot(r1, main = "regular")
plot(st_geometry(st_as_sf(r2)), main = "rotated")
plot(st_geometry(st_as_sf(r3)), main = "sheared")
plot(st_geometry(st_as_sf(r4, as_points = FALSE)), main = "rectilinear")
plot(st_geometry((r5)), main = "curvilinear")
```



5. Spatial data science

This course is completely designed using the R software, however, the libraries (packages) used for spatial data analysis were not developed for the R software specifically.

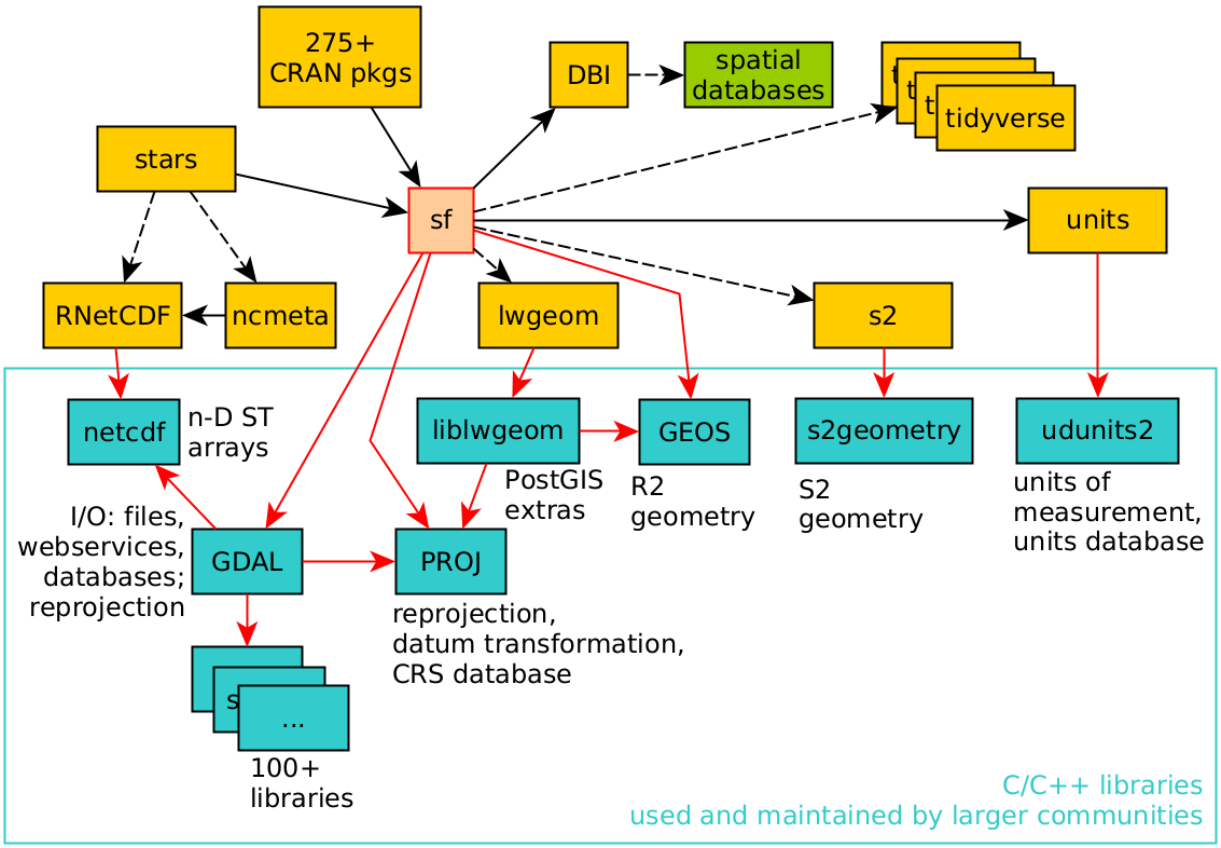


Figura 4: sf and its dependencies; arrows indicate strong dependency, dashed arrows weak dependency

For example, libraries GDAL, GEOS, PROJ, liblwgeom, s2geometry, NetCDF, udunits2, uses C and C++ language, maintained and used by (spatial) data science communities that are large and mostly different from the R community.

GDAL

GDAL is a "library of libraries" – in order to read and write these data, it needs a large number of other libraries. It typically links to over 100 other libraries, each of which provides access to e.g. a particular data file format, a database, a web service or a particular compression codec.

PROJ

PROJ is a library for cartographic projections and datum transformations: it converts spatial coordinates from one coordinate reference system to another.

GEOS and s2geometry

GEOS ("Geometry Engine Open Source") and s2geometry are two libraries for geometric operations. They are used to find measures (length, area, distance), and calculate predicates (do two geometries have any points in common?) or new geometries (which points do these two geometries have in common?).

References

Pebesma, E., & Bivand, R. (2023). Spatial Data Science: With Applications in R. CRC Press.