Lecture 2. Handling Spatial Data in R

Spatial Big Data Analysis with GIS

Korean Statistical Society, Winter School, February 24, 2023

Spatial Data and R Packages for Mapping

- Examples of spatial data: areal, geostatistical and point patterns
- The data storage format called shapefile to store geospatial data
- R packages to create static and interactive maps including **ggplot2**, **ggmap**, **leaflet**, **mapview**, **tmap**, and so on

Types of Spatial Data

A spatial process in $d = 2$ dimensions is denoted as

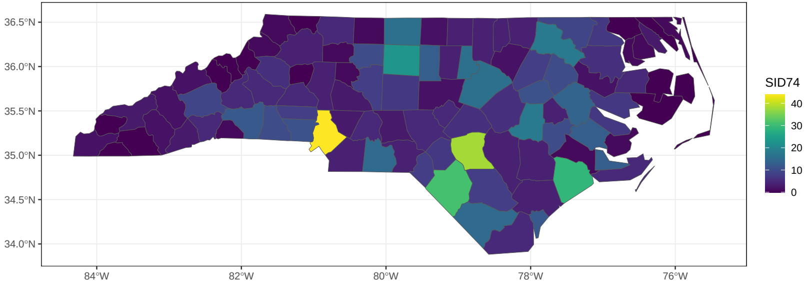
$$\{Z(\mathbf{s}) : \mathbf{s} \in D \subset \mathbb{R}^d\}.$$

Here, Z denotes the attribute we observe, e.g., the number of sudden infant deaths or the level of rainfall, and $\mathbf{s}$ refers to the location of the observation.

Areal Data

- The domain D is fixed (of regular or irregular shape) and partitioned into a finite number of areal units with well-defined boundaries.
- Example: The number of sudden infant deaths in each of the counties of North Carolina, USA, in 1974

```
#library(sf)
#library(ggplot2)
#library(viridis)
nc <- st_read(system.file("shape/nc.shp", package = "sf"),
  quiet = TRUE
)
ggplot(data = nc, aes(fill = SID74)) + geom_sf() +
  scale_fill_viridis() + theme_bw()
```

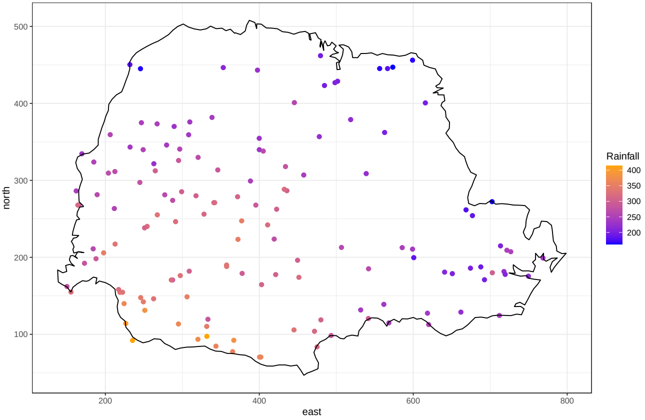


Note: The region of interest (North Carolina) has been partitioned into a finite number of subregions (counties) at which outcomes have been aggregated.

Geostatistical Data

- The domain D is a continuous fixed set.
- $\mathbf{s}$ varies continuously over D and $Z(\mathbf{s})$ can be observed everywhere within D.
- Example: The average rainfall for the period May-June (dry season) over different years collected at 143 recording stations throughout Paraná state, Brazil.

```r
#library(geoR)
ggplot(data.frame(cbind(parana$coords, Rainfall = parana$data)))+
  geom_point(aes(east, north, color = Rainfall), size = 2) +
  coord_fixed(ratio = 1) +
  scale_color_gradient(low = "blue", high = "orange") +
  geom_path(data = data.frame(parana$border), aes(east, north)) +
  theme_bw()
```



Note: These data represent rainfall measurements obtained at specific stations and using model-based geostatistics we could predict rainfall at unsampled sites.

Point Pattern

- The domain D is random. Its index set gives the locations of random events that are the spatial point pattern.
- $Z(\mathbf{s})$ may be equal to 1 $\forall \mathbf{s} \in D$, indicating occurrence of event giving some additional information.
- Example: The locations of deaths of the 1854 London cholera outbreak represent a point pattern.

```r
#library(cholera)
rng <- mapRange()
plot(fatalities[, c("x", "y")],
  pch = 15, col = "black",
  cex = 0.5, xlim = rng$x, ylim = rng$y, asp = 1,
  frame.plot = FALSE, axes = FALSE, xlab = "", ylab = ""
)
addRoads()
```

Note: We can analyze these data using point process methods to understand the spatial distribution deaths, and assess whether there is an excess of risk close to the pump in Broad street.

Shapefiles

A shapefile stores the location, shape, and attributes of geographic features such as points, lines, and polygons.

It is not a unique file, but consists of a collection of related files that have different extensions and a common name and are stored in the same directory.

- Three mandatory files with extensions `.shp`, `.shx`, and `dbf`:
  - `.shp`: contains the geometry data
  - `.shx`: is a positional index of the geometry data that allows to seek forwards and backwards the `.shp` file
  - `.dbf`: stores the attributes for each shape

- Other files that can form a shapefile:
  - `.prj`: plain text file describing the projection
  - `.sbn` and `.sbx`: spatial index of the geometry data
  - `.shp.xml`: geospatial metadata in XML format

Note: When working with shapefiles, it is not enough to obtain the `.shp` file that contains the geometry data, all the other supporting files are also required.

Reading the shapefile in R

- `readOGR()` from `rgdal` package

```
# name of the shapefile of North Carolina of the sf package
nameshp <- system.file("shape/nc.shp", package = "sf")
# read shapefile with readOGR()
#library(rgdal)
map <- readOGR(nameshp, verbose = FALSE)
class(map)
```

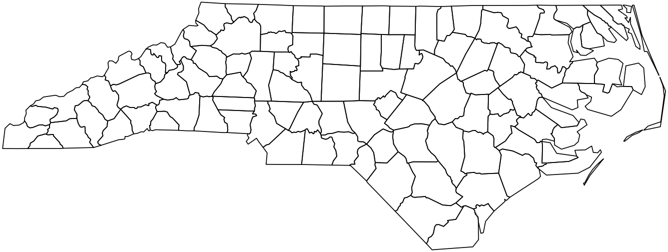
```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
#head(map@data)
#plot(map) # a map of North Carolina
```

```r
head(map@data)
```

```
##    AREA PERIMETER CNTY_ CNTY_ID        NAME  FIPS FIPSNO CRESS_ID BIR74 SID7
## 0 0.114     1.442  1825    1825        Ashe 37009  37009        5  1091
## 1 0.061     1.231  1827    1827   Alleghany 37005  37005        3   487
## 2 0.143     1.630  1828    1828       Surry 37171  37171       86  3188
## 3 0.070     2.968  1831    1831   Currituck 37053  37053       27   508
## 4 0.153     2.206  1832    1832 Northampton 37131  37131       66  1421
## 5 0.097     1.670  1833    1833    Hertford 37091  37091       46  1452
##   NWBIR74 BIR79 SID79 NWBIR79
## 0      10  1364     0      19
## 1      10   542     3      12
## 2     208  3616     6     260
## 3     123   830     2     145
## 4    1066  1606     3    1197
## 5     954  1838     5    1237
```

```r
plot(map) # a map of North Carolina
```



- `st_read()` from sf package

```
# read shapefile with st_read()
#library(sf)
map <- st_read(nameshp, quiet = TRUE)
class(map)
```

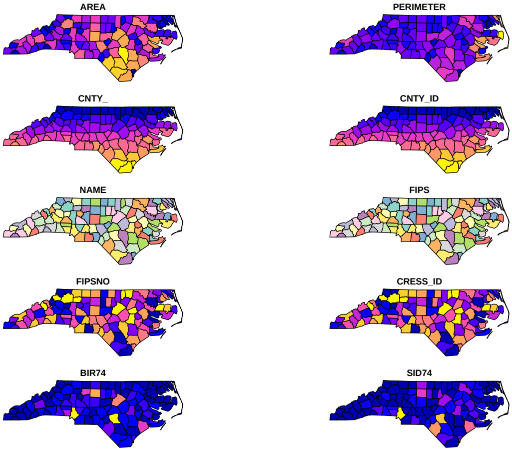
```
## [1] "sf"         "data.frame"
```

```
#head(map)
#plot(map)
```

```r
head(map)
```

```
## Simple feature collection with 6 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -81.74107 ymin: 36.07282 xmax: -75.77316 ymax: 36.58965
## Geodetic CRS:  NAD27
##    AREA PERIMETER CNTY_ CNTY_ID        NAME  FIPS FIPSNO CRESS_ID BIR74 SID7
## 1 0.114     1.442  1825    1825        Ashe 37009  37009        5  1091
## 2 0.061     1.231  1827    1827   Alleghany 37005  37005        3   487
## 3 0.143     1.630  1828    1828       Surry 37171  37171       86  3188
## 4 0.070     2.968  1831    1831   Currituck 37053  37053       27   508
## 5 0.153     2.206  1832    1832 Northampton 37131  37131       66  1421
## 6 0.097     1.670  1833    1833    Hertford 37091  37091       46  1452
##   NWBIR74 BIR79 SID79 NWBIR79                       geometry
## 1      10  1364     0      19 MULTIPOLYGON (((-81.47276 3...
## 2      10   542     3      12 MULTIPOLYGON (((-81.23989 3...
## 3     208  3616     6     260 MULTIPOLYGON (((-80.45634 3...
## 4     123   830     2     145 MULTIPOLYGON (((-76.00897 3...
## 5    1066  1606     3    1197 MULTIPOLYGON (((-77.21767 3...
## 6     954  1838     5    1237 MULTIPOLYGON (((-76.74506 3...
```

```r
plot(map, max.plot = 10)
```

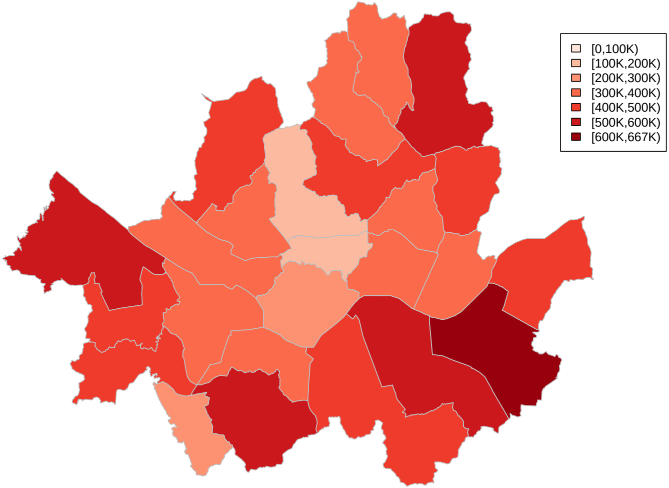


Example: Population of Seoul, South Korea in 2018

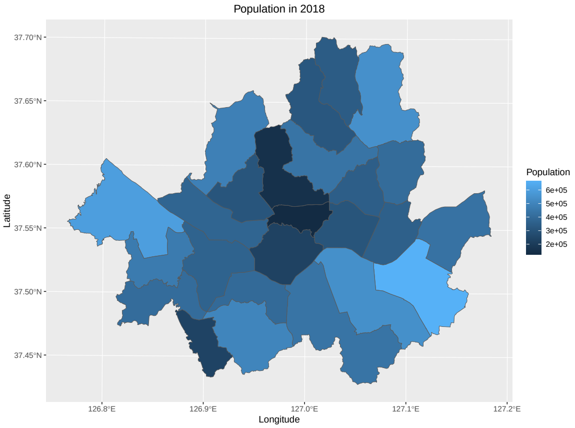
```r
#library(RColorBrewer)
#library(rgdal)
shp.seoul <- readOGR(dsn="../Dataset/Lec2_tmp/LARD_ADM_SECT_SGG_11.shp")
pop <- read.csv("../Dataset/Lec2_tmp/centroid_pop_2018.csv",encoding="UTF-8")

brks <- cut(pop$pop,
          breaks=c(0,100000,200000,300000,400000,500000,600000,max(pop$pop)),
          include.lowest = TRUE, right=FALSE)
col1 <- c()
cols <- brewer.pal(7, "Reds")
for (k in 1:25) {col1[k] <- cols[brks[k]]}
par(mar=c(0,0,3,0))
plot(shp.seoul, col=col1,axes=F, border="gray")
title("Population in 2018")
legend(legend=c("[0,100K)", "[100K,200K)", "[200K,300K)", "[300K,400K)",
                "[400K,500K)","[500K,600K)","[600K,667K)"),fill=cols,
       cex=0.9, x=970000, y=1965000)
```

Note: You can download some `shp` files for Korea from Spatial Data Infrastructure Portal (http://data.nsdi.go.kr/dataset/15144).

Population in 2018



```r
map <- st_as_sf(shp.seoul)
ggplot(map) + geom_sf(aes(fill=pop[,5])) +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(fill="Population") + xlab("Longitude") + ylab("Latitude") +
  ggtitle("Population in 2018")
```



Making Maps with R

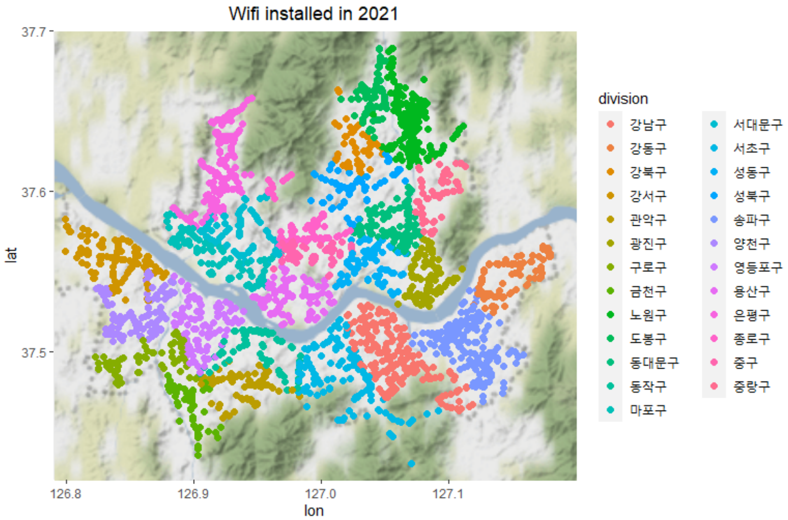
Maps are very useful to convey geospatial information.

Here, we present simple examples that demonstrate the use of some of the packages that are commonly used for mapping in R

ggmap

- Wifi installment from some selective suppliers in Seoul in 2021
- `get_stamenmap`: accesses a tile server for Stamen Maps and downloads/stitches map tiles/formats a map image.

```
wifi <- read.csv("../Dataset/Lec2_tmp/wifi.csv",encoding="UTF-8")
attach(wifi)

Seoul <- get_stamenmap(bbox = c(left = 126.79, bottom = 37.42,
                                right = 127.2, top = 37.70),
                       zoom = 10,
                       maptype = "terrain-background")

ggmap(Seoul) +
  geom_point(data=wifi,aes(x=longitude,y=latitude,colour=division),size=2) +
  labs(title="Wifi installed in 2021") +
  theme(plot.title = element_text(hjust = 0.5))
```


Wifi installed in 2021
37.7
37.6
37.5
lat
126.8
126.9
127.0
127.1
lon
division
강남구
강동구
강북구
강서구
관악구
광진구
구로구
금천구
노원구
도봉구
동대문구
동작구
마포구
서대문구
서초구
성동구
성북구
송파구
양천구
영등포구
용산구
은평구
종로구
중구
중랑구

More Complex Maps to Visualize the Results

Note: For other R packages, refer to https://www.paulamoraga.com/book-geospatial/sec-spatialdataandCRS.html.

- `ggplot2` (https://ggplot2.tidyverse.org/)
- `leaflet` (https://rstudio.github.io/leaflet/): open-source JavaScript library for interactive maps
- `mapview` (https://r-spatial.github.io/mapview/): allows to very quickly create interactive visualizations to investigate both the spatial geometries and the variables in the data
- `tmap`: generates thematic maps with great flexibility

Reference

- Moraga, P. Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny. CRC Press. Chapter 2.
- Cressie, N. Statistics for Spatial Data. Wiley. Chapter 1.