GEOG 5680
Introduction to R
12: Spatial data in R

Simon Brewer

Geography Department
University of Utah
Salt Lake City, Utah 84112

simon.brewer@geog.utah.edu

May 20, 2021

Spatial data analysis

- Characterized by attention to location, spatial interaction, spatial structure and spatial processes. Location may be:
    - Individual site observations
    - Micro-units, such as households, store sites, settlements
    - Aggregate spatial units, such as electoral districts, counties, states or even countries
- Examples:
    - Are disease incidents clustered? Are the clusters related to factors (e.g. poverty or pollution)?
    - Given air quality measures, where are people most at risk of exposure to particulates?
    - Do governments compare policies to their neighbors?

Spatial data analysis

Location, location, location

- Location has crucial role in analysis:
  - Absolute (coordinates)
  - Relative to other observations
- Two classes of spatial effects (Anselin, 1990):
  - Spatial autocorrelation
  - Spatial heterogenetiy

Spatial autocorrelation

- Tobler's law: 'Everything is related to everything else, but near things are more related than distant things' - i.e. similar values cluster
  - High crime areas or climatically similar regions

Spatial autocorrelation

- Tobler's law: 'Everything is related to everything else, but near things are more related than distant things' - i.e. similar values cluster
  - High crime areas or climatically similar regions
- Implies that most geographical data will no longer satisfy the usual statistical assumption of independence of observations
- Results in larger variance and lower significance compared to independent data
- Can be remedied by larger sample sizes, better sampling strategies or by use of specialized analysis

Spatial heterogeneity

- Spatial or regional differentiation resulting from the intrinsic uniqueness of each location

Spatial heterogeneity

- Spatial or regional differentiation resulting from the intrinsic uniqueness of each location
- Different spatial subsets may have different means, variances or other parameter values
- Implies that assumption of stationarity does not hold across study region (same statistical distribution)

Spatial heterogeneity

- Spatial or regional differentiation resulting from the intrinsic uniqueness of each location
- Different spatial subsets may have different means, variances or other parameter values
- Implies that assumption of stationarity does not hold across study region (same statistical distribution)
- Spatial *regimes* — discrete changes across landscape (e.g. difference in mean and variance of income between inner city and suburb)
- Spatial *drift* — continuous variation in parameter (e.g. changing variance of precipitation with distance to monsoon region)

Spatial data types

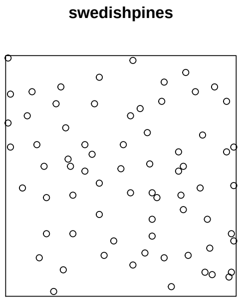
From Cressie (1991):

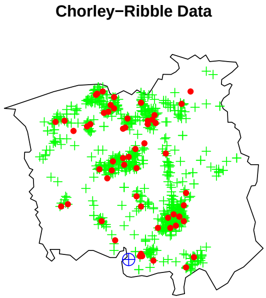
- Point processes (location of objects in space)
- Areal or lattice — discrete variation of values aggregated across regular or irregular regions
- Geostatistical — continuous variation of values

Expressed as geometric features:

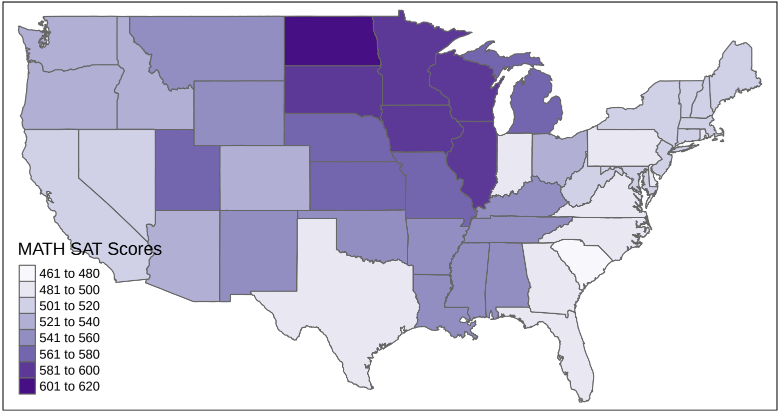
- Points/Lines/Areal units (polygons)/Regular grids
- In a plane, or, less frequently, on a surface

Point process data

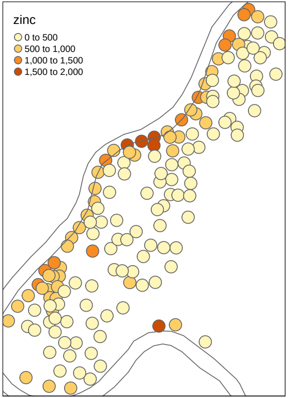


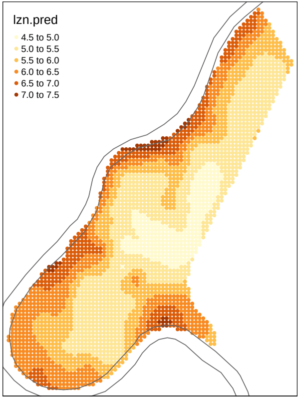


Areal data


MATH SAT Scores
461 to 480
481 to 500
501 to 520
521 to 540
541 to 560
561 to 580
581 to 600
601 to 620

Geostatistical data





Spatial data in R

Key spatial packages:

- **sp** provided original specification of spatial data classes (S4 objects)
- **sf** newer package based on *simple features*, ISO 19125
- Spatial analytical packages build on these
  - **spatstat** — analysis of spatial point processes
  - **spatdep** and **spatialreg** — spatial regression models
  - **gstat** — geostatistical analysis

Working with vector data

Vector data

- **sf** package (uses GDAL/OGR library)
    - Allows import and export of shapefiles and other standard vector formats ($\approx$ 93 different formats)
    - `st_read()` and `st_write()`
- **rgdal** package (uses GDAL/OGR library)
- **rgeos** for extra topology/geometry functions

Working with raster data

Raster data

- **raster** package (uses GDAL library, $\approx$ 150 formats)
- Allows import and export of most raster formats (ArcInfo/GeoTIFF/ESRI/Most RS images/NetCDF)
- Will work with large files and stacks/bricks (multi-band raster data)
- See also `readAsciiGrid()` in **maptools** (reads ESRI grids), **ncdf4** package, **landsat** package for remote sensing image correction
- **stars** package for working with space-time data in dense arrays

Data import/export

Raster data

- **raster** package (uses GDAL library, $\approx$ 150 formats)
- Allows import and export of most raster formats (ArcInfo/GeoTIFF/ESRI/Most RS images/NetCDF)
- Will work with large files and stacks/bricks (multi-band raster data)
- See also `readAsciiGrid()` in **maptools** (reads ESRI grids), **ncdf4** package, **landsat** package for remote sensing image correction
- **stars** package for working with space-time data in dense arrays

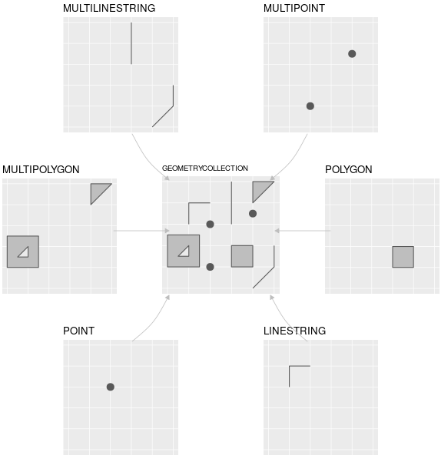
Spatial data classes in **sf**

Each `sf` object consists of four things

- Geometry
- Attributes
- Coordinate Reference System
- Bounding box

Geometries in **sf**

- Defines the type of spatial object
- Standard GIS types (single and multi)



Attributes in **sf**

- Hold data associated with spatial objects
- Standard R `data.frame` with geometry

```
states

## Simple feature collection with 49 features and 4 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -124.6813 ymin: 25.12993 xmax: -67.00742 ymax
## geographic CRS: WGS 84
## First 10 features:
##    nm_bbrv verbal math tkrs_pc                       geometry
## 1      ala    561  555       9 MULTIPOLYGON (((-87.46201 3...
## 2     ariz    524  525      34 MULTIPOLYGON (((-114.6374 3...
## 3      ark    563  556       6 MULTIPOLYGON (((-94.05103 3...
## 4    calif    497  514      49 MULTIPOLYGON (((-120.006 42...
## 5     colo    536  540      32 MULTIPOLYGON (((-102.0552 4...
## 6     conn    510  509      80 MULTIPOLYGON (((-73.49902 4...
## 7     dela    503  497      67 MULTIPOLYGON (((-75.80231 3...
## 8     d.c.    494  478      77 MULTIPOLYGON (((-77.13731 3...
## 9      fla    499  498      53 MULTIPOLYGON (((-85.01548 3...
## 10      ga    487  482      63 MULTIPOLYGON (((-80.89018 3...
```

CRS in **sf**

- Coordinate reference system (CRS)
- Uses WKT format
- Can be easily defined using EPSG codes (e.g. WGS 84 = 4326)

```
st_crs(states)

## Coordinate Reference System:
##   User input: EPSG:4326
##   wkt:
## GEOGCRS["WGS 84",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["geodetic latitude (Lat)",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["geodetic longitude (Lon)",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     USAGE[
##         SCOPE["unknown"],
##         AREA["World"],
##         BBOX[-90,-180,90,180]],
##     ID["EPSG",4326]]
```

Re-projections using CRS

- The **sf** library has a function (`st_transform()`) to convert between projections

```
oregon = st_read("./oregon/orotl.shp")

## Reading layer `orotl' from data source `/Users/u0784726/Dropbox/Data/devtools/geog5680/12 Spatial data in R/oregon/orotl.shp' using drive
## Simple feature collection with 36 features and 1 field
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -124.5584 ymin: 41.98779 xmax: -116.4694 ymax: 46.23626
## CRS:            NA

st_crs(oregon) <- 4326
format(st_crs(oregon))

## [1] "WGS 84"
```

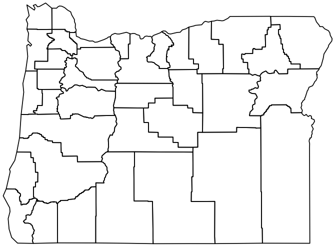
Map projections

- Use the `st_transform()` function to reproject
- Reproject to Lambert (EPSG 2992)

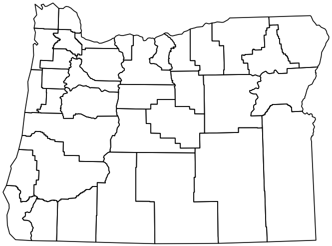
```
oregon.proj <- st_transform(oregon, crs = 2992)
format(st_crs(oregon.proj))

## [1] "NAD83 / Oregon GIC Lambert (ft)"
```

Map projections

WGS 84



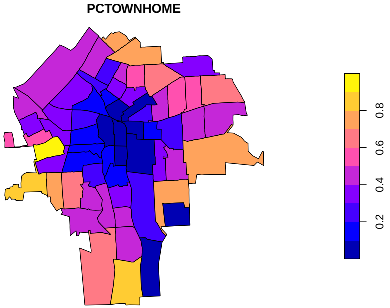
Oregon GIC Lambert



Visualizing spatial data

- **sf** package provides basic function `plot()`, which works in a similar way to R's base plot function

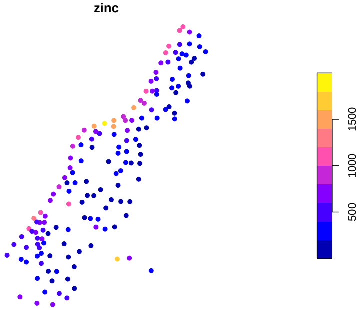
```
plot(Syracuse["PCTOWNHOME"])
```



Visualizing spatial data

- **sf** package provides basic function `plot()`, which works in a similar way to R's base plot function

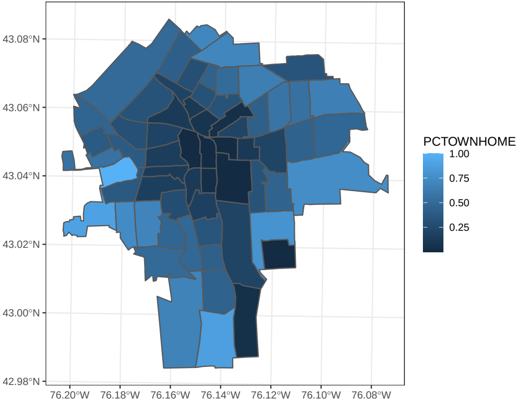
```
plot(meuse["zinc"], pch = 16)
```



Visualizing spatial data

- Extend basic visualization using **ggplot2**

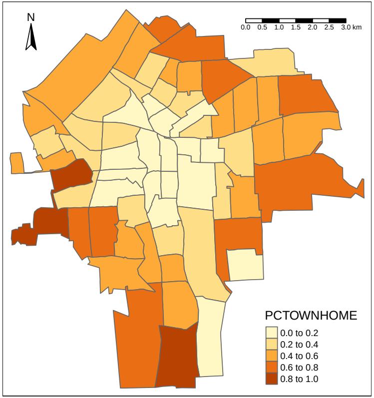
```
library(ggplot2)
ggplot() +
  geom_sf(data = Syracuse, aes(fill = PCTOWNHOME)) +
  theme_bw()
```



Visualizing spatial data

- Extend basic visualization using **tmap**

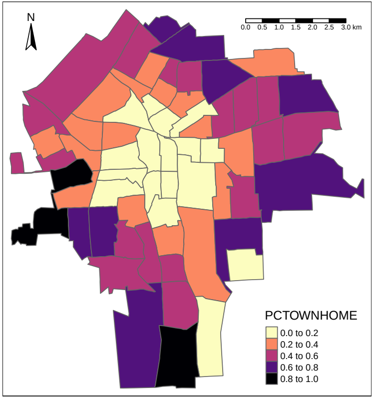
```
library(tmap)
tm_shape(Syracuse) +
  tm_fill("PCTOWNHOME") +
  tm_borders() +
  tm_compass(position = c("left", "top")) +
  tm_scale_bar(position = c("right", "top"))
```



Visualizing spatial data

- Extend basic visualization using **tmap**
- Additional color palettes (viridis and ColorBrewer)

```
library(tmap)
tm_shape(Syracuse) +
  tm_fill("PCTOWNHOME", palette = "-magma") +
  tm_borders() +
  tm_compass(position = c("left", "top")) +
  tm_scale_bar(position = c("right", "top"))
```



Visualizing spatial data

- Extend basic visualization using **tmap**
- Easily add extra layers

```
tm_shape(meuse) +
  tm_symbols(col = "zinc", palette = "Greens") +
  tm_shape(riv) +
  tm_borders()
```