CSSS 569 Visualizing Data and Models
Lab 6: Visualizing Spatial Data

Ramses Llobet

Department of Political Science, UW

February 11, 2022
Introduction

Income class
1. High income: OECD
2. High income: nonOECD
3. Upper middle income
4. Lower middle income
5. Low income

Metro population (2020)
10 mln 20 mln 30 mln 40 mln

Population growth (%)
Less than 0
0 to 10
10 to 20
20 to 30
30 or more
Introduction

Rent in 2008
- 0 to 500
- 500 to 1,000
- 1,000 to 1,500
- 1,500 to 2,000
- 2,000 to 2,500
- 2,500 to 3,000

Hispanic population in 2008 (%)
- 0 to 10
- 10 to 20
- 20 to 30
- 30 to 40
- 40 to 50
- 50 to 60
- 60 to 70

% of households receiving public assistance in 2000
- 0 to 5
- 5 to 10
- 10 to 15
- 15 to 20
- 20 to 25
Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - .shp (shapefile; the most common)
  - .geojson, .json
  - .gml
  - .csv
  - .tiff...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`
- Many packages to draw maps
  - `tmap` allows easy visualization of static and interactive maps
  - Also employs the "grammar of graphics"
Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - .shp (shapefile; the most common); .geojson, .json; .gml; .csv; .tiff...
Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`, `.gml`, `.csv`, `.tiff`...

- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`
  - Many packages to draw maps
    - `tmap` allows easy visualization of static and interactive maps
    - Also employs the "grammar of graphics"
Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - .shp (shapefile; the most common); .geojson, .json; .gml; .csv; .tiff...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...

- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`
Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`
- Many packages to draw maps
Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - .shp (shapefile; the most common); .geojson, .json, .gml; .csv; .tiff...
- Countless packages to work with spatial data
  - Recent package sf allows geospatial data to be stored in data frames
  - Well integrated with tidyverse
- Many packages to draw maps
  - tmap allows easy visualization of static and interactive maps
Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - .shp (shapefile; the most common); .geojson, .json, .gml; .csv; .tiff...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`
- Many packages to draw maps
  - `tmap` allows easy visualization of static and interactive maps
  - Also employs the “grammar of graphics”
Working with google maps (another alternative)

▶ Set up an account at Google
Working with google maps (another alternative)

- Set up an account at Google
- You need to register with a debit card (however is free, google will not charge for using maps within Rstudio).
Working with google maps (another alternative)

- Set up an account at Google
- You need to register with a debit card (however is free, google will not charge for using maps within Rstudio).
- You need to enable Geocoding API and Maps Static API.
Working with google maps (another alternative)

- Set up an account at Google
- You need to register with a debit card (however is free, google will not charge for using maps within Rstudio).
- You need to enable Geocoding API and Maps Static API.
- Once you have an API key authentication, you can use `get_map()` to grab a map from Google maps and use it in `ggmap()`.
Working with google maps (another alternative)

- Set up an account at Google
- You need to register with a debit card (however is free, google will not charge for using maps within Rstudio).
- You need to enable Geocoding API and Maps Static API.
- Once you have an API key authentication, you can use `get_map()` to grab a map from Google maps and use it in `ggmap()`.
- If you are interested in `ggmap`, you can find a tutorial in [here](#) and a useful cheat sheet in [here](#).
Overview of `tmap` package

<table>
<thead>
<tr>
<th></th>
<th>ggplot2</th>
<th>tmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td><code>ggplot(...) +</code></td>
<td><code>tm_shape(...) +</code></td>
</tr>
<tr>
<td>Layers</td>
<td><code>geom_...(...) +</code></td>
<td><code>tm_...(...) +</code></td>
</tr>
<tr>
<td>Small Multiples</td>
<td><code>facet_grid(...)</code></td>
<td><code>tm_facets(...)</code></td>
</tr>
<tr>
<td>Layout</td>
<td><code>theme(...)</code></td>
<td><code>tm_layout(...)</code></td>
</tr>
</tbody>
</table>
install.packages(c("sf", "tmap"))

# Load packages
library(tidyverse)
library(sf)
library(tmap)

# Load data (from tmap)
data(World, metro)
Basics tmap

```r
World[1,] %>% t() %>% pander::pander()
```

<table>
<thead>
<tr>
<th>iso_a3</th>
<th>AFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Afghanistan</td>
</tr>
<tr>
<td>sovereignt</td>
<td>Afghanistan</td>
</tr>
<tr>
<td>continent</td>
<td>Asia</td>
</tr>
<tr>
<td>area</td>
<td>652860</td>
</tr>
<tr>
<td>pop_est</td>
<td>28400000</td>
</tr>
<tr>
<td>pop_est_dens</td>
<td>43.5009</td>
</tr>
<tr>
<td>economy</td>
<td>7. Least developed region</td>
</tr>
<tr>
<td>income_grp</td>
<td>5. Low income</td>
</tr>
<tr>
<td>gdp_cap_est</td>
<td>784.1549</td>
</tr>
<tr>
<td>life_exp</td>
<td>59.668</td>
</tr>
<tr>
<td>well_being</td>
<td>3.8</td>
</tr>
<tr>
<td>footprint</td>
<td>0.79</td>
</tr>
<tr>
<td>inequality</td>
<td>0.4265574</td>
</tr>
<tr>
<td>HPI</td>
<td>20.22535</td>
</tr>
<tr>
<td>geometry</td>
<td>61.21082, 62.23065, 62.98466, 63.19354, 63.98290, 64.54648, 64.74611, 65.58895, 65.74563, 66.21738, 66.51861, 67.07578, 67.83000, 68.13556, 68.85945, 69.19627, 69.51879, 70.11658, 70.27057, 70.37630, 70.80682</td>
</tr>
</tbody>
</table>
Basics tmap

```r
tm_shape(World) +
  tm_polygons() +
  tm_layout(frame = FALSE)
```
Basics tmap

tm_polygons() is composed of two parts: tm_borders() and tm_fill()

tm_shape(World) +
  tm_borders() +
  tm_layout(frame = FALSE)
Basics tmap

tm_polygons() is composed of two parts: tm_borders() and tm_fill()

```
tm_shape(World) +
  tm_fill() +
  tm_layout(frame = FALSE)
```
Basics tmap

`tm_polygons()` is composed of two parts: `tm_borders()` and `tm_fill()`

```r
tm_shape(World) +
  tm_borders() +
  tm_fill() +
  tm_layout(frame = FALSE)
```
Basics tmap

`tm_polygons()` is composed of two parts: `tm_borders()` and `tm_fill()`

```r
tm_shape(World) +
  tm_borders(lty = 2) +
  tm_fill() +
  tm_layout(frame = FALSE)
```
Basics tmap

tm_shape(World) +
  tm_polygons() +
  tm_layout(frame = FALSE)
Basics tmap

All palettes from RColorBrewer are supported

tm_shape(World) +
  tm_polygons(col = "income.grp", palette = "-Blues") +
  tm_layout(frame = FALSE)
Basics tmap

Use white border to give it a “modern” look

tm_shape(World) +
  tm_polygons(col = "income.grp", palette = "-Blues",
              border.col = "white", border.alpha = 0.5) +
  tm_layout(frame = FALSE)
Basics tmap

Legend title:

```r
tm_shape(World) +
  tm_polygons(col = "income_grp", palette = "-Blues",
              border.col = "white", border.alpha = 0.5,
              title = "Income class") +
  tm_layout(frame = FALSE)
```
Basics tmap

Add country labels as an additional layer:

tm_shape(World) +
  tm_polygons(col = "income_grp", palette = ":-Blues",
              border.col = "white", border.alpha = 0.5,
              title = "Income class") +
  tm_text(text = "iso_a3", size = "AREA", col = "grey25") +
  tm_layout(frame = FALSE)
worldMap <-
  tm_shape(World) +
  tm_polygons(col = "income_grp", palette = "-Blues",
              border.col = "white", border.alpha = 0.5,
              title = "Income class") +
  tm_text(text = "iso_a3", size = "AREA", col = "grey25") +
  tm_layout(frame = FALSE)
Let's add another dimension of information: cities' population

```r
metro <- metro %>%
    mutate(growth = (pop2020 - pop2010) / pop2010 * 100)

metro[1,] %>% t() %>% pander::pander()
```

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>name</td>
<td>Kabul</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>name_long</td>
<td>Kabul</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>iso_a3</td>
<td>AFG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>pop1950</td>
<td>170784</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>pop1960</td>
<td>285352</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>pop1970</td>
<td>471891</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>pop1980</td>
<td>977824</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>pop1990</td>
<td>1549320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>pop2000</td>
<td>2401109</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>pop2010</td>
<td>3722320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>pop2020</td>
<td>5721697</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>pop2030</td>
<td>8279607</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>geometry</td>
<td>69.17246, 34.52889</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>growth</td>
<td>53.71319</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Basics tmap

worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020")
Basics tmap

```r
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020", col = "growth")
```

**Income class**
- 1. High income: OECD
- 2. High income: nonOECD
- 3. Upper middle income
- 4. Lower middle income
- 5. Low income

**pop2020**
- 10 mln
- 20 mln
- 30 mln
- 40 mln

**growth**
- -50 to 0
- 0 to 50
- 50 to 100
- 100 to 150
Basics tmap

```r
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020", col = "growth",
              palette = "-RdYlGn", midpoint = NA)
```
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020", col = "growth",
            palette = "-RdYlGn", midpoint = NA,
            breaks = c(-Inf, 0, 10, 20, 30, Inf))
```r
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020", col = "growth",
              palette = "-RdYlGn", midpoint = NA,
              breaks = c(-Inf, 0, 10, 20, 30, Inf),
              alpha = 0.9,
              border.col = "white",
              border.lwd = 0.1,
              title.size = "Metro population (2020)",
              title.col = "Population growth (%)")
```
Basics tmap

Income class
1. High income: OECD
2. High income: nonOECD
3. Upper middle income
4. Lower middle income
5. Low income

Metro population (2020)
10 mln 20 mln 30 mln 40 mln

Population growth (%)
Less than 0
0 to 10
10 to 20
20 to 30
30 or more
Save the resulting map using `tmap_save()`

```python
tmap_save(world_map, filename = "worldMap.pdf")
```
Useful functions from \texttt{tmap}: New York example

Rent in 2008
- 0 to 500
- 500 to 1,000
- 1,000 to 1,500
- 1,500 to 2,000
- 2,000 to 2,500
- 2,500 to 3,000

Hispanic population in 2008 (%)
- 0 to 10
- 10 to 20
- 20 to 30
- 30 to 40
- 40 to 50
- 50 to 60
- 60 to 70

% of households receiving public assistance in 2000
- 0 to 5
- 5 to 10
- 10 to 15
- 15 to 20
- 20 to 25
Useful functions from tmap: New York example

- Prerequisite
Useful functions from tmap: New York example

▶ Prerequisite
  ▶ Download the .zip data file here
Useful functions from \texttt{tmap}: New York example

- Prerequisite
  - Download the \texttt{.zip} data file [here](#)
  - Unzip it and put it in your working directory
Useful functions from `tmap`: New York example

▶ Load `.shp` file with `sf`

```r
city_bound <- st_read("nyc/nyc.shp")
```

---

```r
## Reading layer ‘nyc’ from data source
## ‘C:\Users\ramse\Google Drive\Phd UW\Courses\Second Year\CSSS 569 - Visualizing Data and Models\{lab 6\}\nyc\nyc.shp’ using driver ‘ESRI Shapefile’
## Simple feature collection with 55 features and 34 fields
## Geometry type: MULTIPOLYGON
## Dimension: XY
## Bounding box: xmin: 913037.2 ymin: 120117 xmax: 1067549 ymax: 272751.4
## Projected CRS: NAD83 / New York Long Island (ftUS)
```
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rent2008</td>
<td>median monthly contract rent in 2008</td>
</tr>
<tr>
<td>forhis08</td>
<td>% of hispanic population in 2008</td>
</tr>
<tr>
<td>pubast00</td>
<td>% of households receiving public assistance in 2000</td>
</tr>
</tbody>
</table>
New York example: exercise 1

- Replicate the following map (or choose any palette you see fit)
New York example: exercise 1

tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008")
New York example: exercise 1

```r
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG")
```
New York example: exercise 1

```
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG",
              border.col = "white", title = "Rent in 2008") +
  tm_layout(frame = FALSE)
```
New York example: interactive mode

- Interactive map visualization

```r
tmap_mode("view")
```
New York example: interactive mode

- Interactive map visualization

```r
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG",
              title = "Rent in 2008",
              alpha = 0.7) +
  tm_basemap(server = "OpenStreetMap", alpha = 0.5)
```
New York example: interactive mode

- Interactive map visualization
New York example: interactive mode

- Switching back to plotting mode

```r
tmap_mode("plot")
```
Create two more maps based on forhis08 and pubast00:
New York example: exercise 2

```r
rentNYC <- tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG",
              border.col = "white", border.alpha = 0.5,
              title = "Rent in 2008") +
  tm_layout(legend.text.size = 0.5,
            legend.width = 0.7,
            frame = FALSE)

hisNYC <- tm_shape(nyc.bound) +
  tm_polygons(col = "forhis08",
              border.col = "white", border.alpha = 0.5,
              title = "Hispanic population in 2008 (%)") +
  tm_layout(legend.text.size = 0.5,
            legend.width = 0.7,
            frame = FALSE)

pubastNYC <- tm_shape(nyc.bound) +
  tm_polygons(col = "pubast00", palette = "Blues",
              border.col = "white", border.alpha = 0.5,
              title = "% of households receiving public assistance in 2000") +
  tm_layout(legend.text.size = 0.5,
            legend.width = 0.7,
            frame = FALSE)
```
tmap_arrange(rentNYC, hisNYC, pubastNYC, nrow = 1)
New York example: small multiples

First, create some cutpoints based on forhis08

```r
cut(nyc.bound$forhis08, breaks = 3)
```

```
## [5] (49.5,69.4] (49.5,69.4] (49.5,69.4] (29.6,49.5] 
## [9] (29.6,49.5] (49.5,69.4] 
## Levels: (9.63,29.6] (29.6,49.5] (49.5,69.4]
```
New York example: small multiples

- Small multiples using `tm_facets()`

```r
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG",
              title = "Rent in 2008") +
  tm_facets(by = "cut.forhis", nrow = 1,
            free.coords = FALSE,
            drop.units = FALSE)
```

![Small multiples of rent in New York City](image)
Concluding remarks

▶ Many more cool functions in tmap
Concluding remarks

▶ Many more cool functions in \texttt{tmap}
  ▶ Animation with maps
Concluding remarks

- Many more cool functions in tmap
  - Animation with maps
- Check out
Concluding remarks

- Many more cool functions in tmap
  - Animation with maps
- Check out
  - tmap vignette
Concluding remarks

- Many more cool functions in tmap
  - Animation with maps
- Check out
  - tmap vignette
  - Basic Mapping: R Notes
Concluding remarks

- Many more cool functions in tmap
  - Animation with maps
- Check out
  - tmap vignette
  - Basic Mapping: R Notes
  - Geocomputation with R: Ch. 8:: Making maps with R
Concluding remarks

- Many more cool functions in \texttt{tmap}
  - Animation with maps
- Check out
  - \texttt{tmap} vignette
  - Basic Mapping: R Notes
  - Geocomputation with R: Ch. 8:: Making maps with R
  - Creating beautiful demographic maps in R with the \texttt{tidycensus} and \texttt{tmap} packages