2 Getting geographic boundaries

> The essence of mapping is the geometry of the places to be shown. This chapter identifies the formats in which geometry is encoded and shows you how to work with them.

2.1 Two types of data required

For mapping, two types of data are needed. The first, geographic boundary data, usually called "geometry," consists of a series of points, representing longitude (running north-south) and latitude (running east-west) intersections. When joined, they form a *polygon*. A complex geometry may include several polygons, forming a *multipolygon*. *geometry* is the term that will be used to distinguish geographic boundaries from ordinary data. The second is other data related to the geographic units in the geometry. *data* will be used for this type, and *spatial* will be used for geometry combined with data.

2.2 Encoding geometry

Geographic geometries are stored in several open source and proprietary formats.

- *Shapefiles* are a proprietary format developed by Esri. It claims that its related geographic information system software is used by over 680,000 customer organizations, including 90% of Fortune 100 companies, most national governments, 30,000 cities and local governments, all 50 US states, and 12,000 universities. The U.S. Census geometry files use this format, among others.
- *Well known text* (WKT) is an *International Standards Organization/Open Geospatial Consortium* (ISO/OGC) standard for representing geometric objects. *WKT* uses human-readable text like `POINT(1 2)` or `POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))`, while *well known binary* (WKB) provides the same information in compact binary form.
- *GeoJSON* stores geometries as *Javascript Object Notation* (JSON) files. It is a popular choice for web applications.

- *GeoPackage* is a structured query language (*SQL*) database format using *SQLite* that can store multiple geometry types, raster data, and attributes in a single file.
- *GeoDataBase* is Esri's proprietary, integrated system for managing spatial data.
- *Database Storage PostGIS* extends *PostgreSQL*, a database, with spatial types, storing geometries in optimized binary formats with spatial indexing. Other spatial databases like Oracle Spatial, SQL Server Spatial, and SpatiaLite use similar approaches.
- *Keyhole markup language* (KML) is *eXtended markup language*-based (XML) and designed for Google Earth, though it is now an OGC standard.
- *TopoJSON* extends GeoJSON by encoding *topology*, significantly reducing file sizes for adjacent polygons by eliminating duplicate boundaries.
- *FlatGeobuf* provides a binary format optimized for streaming and random access, particularly useful for large datasets.

Which to use will depend on what the available geographic coverage for your area of interest uses. The examples in this book primarily rely on shapefiles.

2.3 Shapefiles

Important The U.S. Census Bureau's TIGER (Topologically Integrated Geographic Encoding and Referencing) shapefiles files are a gold standard, and they are subject to rigorous quality assurance. *However,* errors in downloads are possible. If you experience difficulty in opening a shapefile, make sure that all of the associated file extensions, in addition to `.shp`, are in the same folder or download a fresh copy.

The following script demonstrates the basic workflow for loading *geometry* data in Julia.

```julia
] activate .
using GeoDataFrames
gdf = GeoDataFrames.read("data/2024_shp/cb_2024_us_state_500k.shp")
```

- A DataFrame is a structure containing rows and columns, much like a spreadsheet. Rows represent observations, and columns represent variables. This is referred to as a "tall" format. While it is possible to use the opposite convention ("wide"), it is not recommended.

There are several other files in `data` with the prefix `data/2024_shp/cb_2024_us_state_500k` that are necessary. They are read in automatically, but be sure to leave them. `gdf` is a conventional name for a *GeoDataFrame*, which differs from a regular DataFrame primarily in having a special type for the `geometry` column.

```
56×10 DataFrame                                                    Output
 Row │ geometry                  STATEFP  STATENS  GEOIDFQ      GEOID   STUSPS  NAME       LSAD    ALAND         AWATER
     │ IGeometry                 String   String   String       String  String  String     String  Int64         Int64
─────┼──────────────────────────────────────────────────────────────────────────────────────────────────────────────────
   1 │ Geometry: wkbPolygon      35       00897535 0400000US35  35      NM      New Mexico 00      314198519809  726531289
   2 │ Geometry: wkbPolygon      46       01785534 0400000US46  46      SD      South Dakota 00    196341670967  3387563375
   3 │ Geometry: wkbMultiPolygon 06       01779778 0400000US06  06      CA      California 00      403673433805  20291632828
   # … omitted
```

For now we are uninterested in four of the columns, so we can trim them. The last two are the land and water areas in square meters. As demonstrated in the following snippet, we use the `select` function to remove unwanted columns.

```
# same session                                                     Julia
select!(gdf, Not(:STATEFP,:STATENS,:GEOIDFQ,:LSAD))
```

The exclamation point (!) operator used in `select` modifies the `gdf` object in place. The colon (:) signifies a variable name.

`GEOID` a unique code to identify the state and `STUSPS` is the two-letter state postal abbreviation. These are unique for each row and can be used to connect with DataFrame having one or more of the same keys.

Read this as "with `gdf` select the rows in which the `STUSPS` variable is not in the values entries of the `VALID_STATE_CODES` dictionary," part of the `JuliaMapping` package.

Usually, we are uninterested in the overseas territories and possessions. They are excluded with

```julia
# same session
include("src/constants.jl")
gdf = subset(gdf, :STUSPS => ByRow(x -> x in values(VALID_STATE_CODES)))
```

If we wanted to further trim the map to the contiguous United States, we could additionally use this.

◇ Interpret this the same way, except instead of using the `VALID_STATE_CODES` dictionary, look in the list that contains the abbreviations for the two states.

```julia
gdf = subset(gdf, :STUSPS => ByRow(x -> !(x in ["AK","HI"])))
```

2.4 GeoJSON files

`GeoDataFrames` can also handle this type of geometry encoding.

```julia
] activate .
using GeoDataFrames
gdf = GeoDataFrames.read("data/geojson_example.json")
select!(gdf, [:NAME,:geometry])
```

2.5 GeoPackage files

■ This format typically has file sizes far in excess of other encoding methods. This file expands to more than 440MB, and you may want to delete it after running the example.

Download the U.S. Census Bureau map of legislative districts from https://www2.census.gov/geo/tiger/TGRGPKG24/tlgpkg_2024_a_us_legislative.gpkg.zip and unzip it to your `data` folder.

□ The resulting `GeoDataFrame` is similar to what we get with the `shapefile` and `GeoJSON` formats.

```julia
] activate .
using GeoDataFrames
gdf = GeoDataFrames.read("data/tlgpkg_2024_a_us_legislative.gpkg")
```