Making Maps in R with the sf model

Matthew L. Sisk

Last Updated November, 2023

Loading Spatial Data

Loading spatial data in R is relatively simple, but like many things in R you have multiple options. Until recently, most spatial data in R was handled using the *sp* package to represent the different types of spatial data. There is now an additional option: the *sf* package, which is designed to be tidyverse complaint and more directly human readable. There is a corollary document that shows most of these examples in the *sp* package in case you need it as a reference. Recently, it was announced that *sp* will no longer be maintained, but I will keep the document there in case you find old tutorials online.

To start with *sf*. just load the package (after installing)

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.3.1
```

To load a shapefile, just use st_read()

```
districts <- st_read("SampleData\\City_Council_Districts.shp", stringsAsFactors = FALSE)
```

```
## Reading layer 'City_Council_Districts' from data source
##   'C:\Users\msisk1\Documents\GIT\teaching\Data-Viz-2023-Fall\Week03\Demonstrations\SampleData\City_C
##   using driver 'ESRI Shapefile'
## Simple feature collection with 6 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -86.36085 ymin: 41.59745 xmax: -86.19133 ymax: 41.76027
## Geodetic CRS:  WGS 84
```

When we have loaded the shapefile, it gives some basic information about the number of features, the number of fields and the type of data. Remember that the information in the table is separate from the points that make up the feature. This is a polygon layer. Unlike the *sp* model, in sf, the geometry is simply a column in the tibble/data frame.

```
districts[1,]$geometry
```

```
## Geometry set for 1 feature
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -86.36085 ymin: 41.67362 xmax: -86.24622 ymax: 41.76027
## Geodetic CRS:  WGS 84
```

```
## MULTIPOLYGON (((-86.34706 41.72135, -86.34708 4...
```

Loading spatial data from a table

Often, you will not get spatial data as part of an existing shapefile. You may instead get a table with latitude and longitude coordinates (Or street addresses, which we will get to in a different demonstration). Hear, let's load a table that has latitude and longitude coordinates

```
facilities.points <- read.csv("SampleData/Public_Facilities.csv")
```

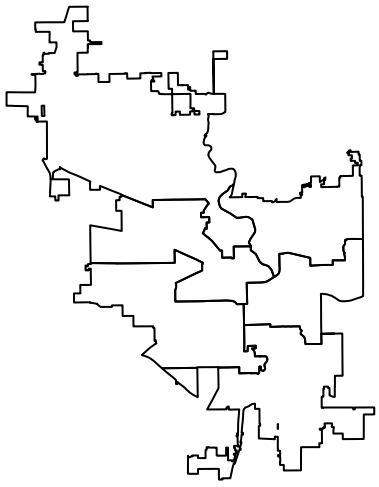
Now, in order to covert this into something we can map, we to define it using sf, we use tidy pipes with commands to convert it to sf and set the coordinate reference system. The number (4326) is a code for the coordinate system in the EPSG, a registry of coordinate systems for geographic data. In this case, 4326 corresponds to a latitude and longitude system using the most common model of the earth's surface.

```
facilities.spatial <- facilities.points %>% #projecting the table as an sf and setting the coordinate s
  st_as_sf(coords = c("Lon","Lat")) %>%
  st_set_crs(value = 4326)
```

Creating Static Maps

Once we have spatial data loaded in R, creating maps is a good next step. We can just use the base R plot to create one, but this is pretty basic. With sf, you want to explicitly call the geometry column.

```
plot(districts$geometry)
```



Alternatively, there are a few packages built for creating maps in R. Probably the most commonly use is ggmap, which is designed to be like ggplot for spatial data.

```
library(ggmap)
```

```
## Warning: package 'ggmap' was built under R version 4.3.1
```

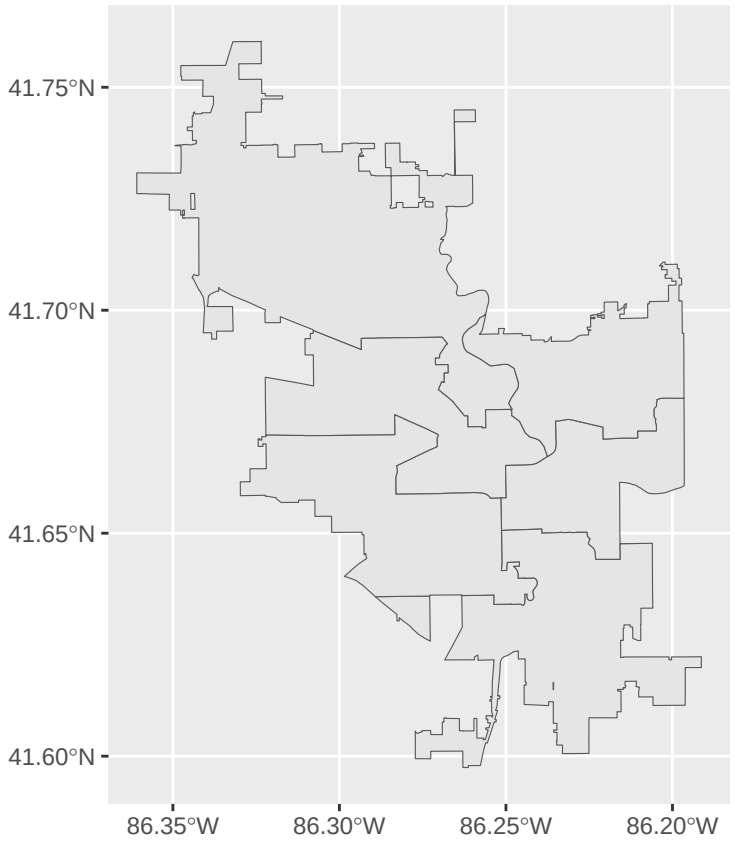
```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.1
```

```
## i Google's Terms of Service: <https://mapsplatform.google.com>
## i Please cite ggmap if you use it! Use `citation("ggmap")` for details.
```

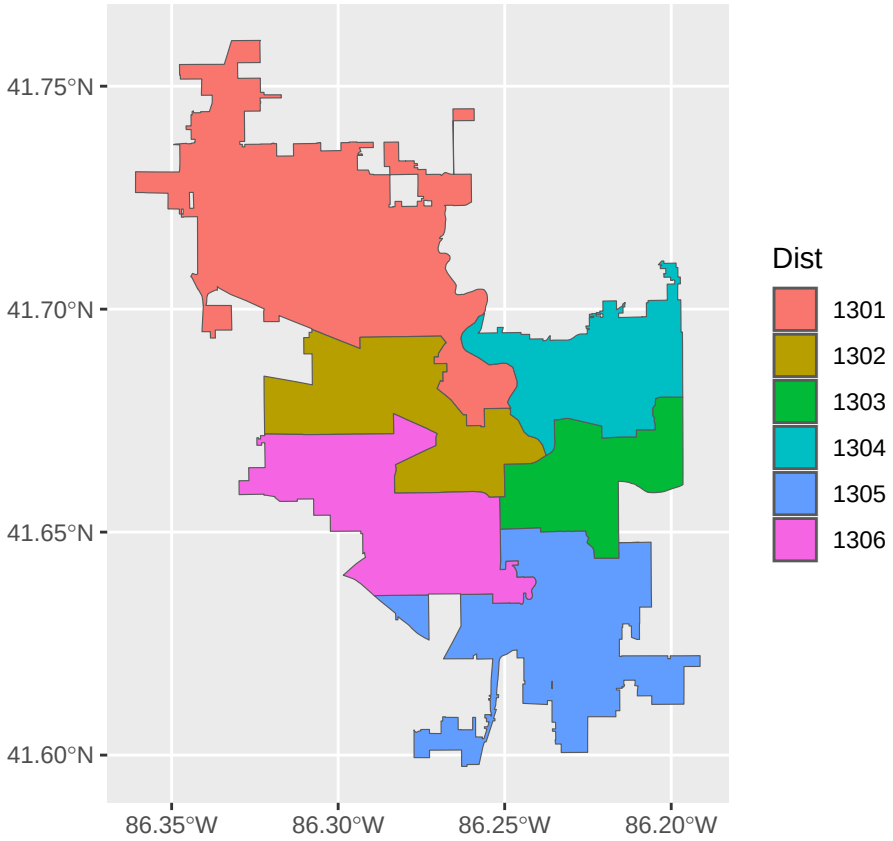
We can now use the same sort of syntax we would with a normal ggplot to create maps. Note that we use *geom_sf()* to load most spatial data. This is a wrapper for other geom functions (like geom_point() or geom_polygon())

```
ggplot() +
  geom_sf(data = districts)
```



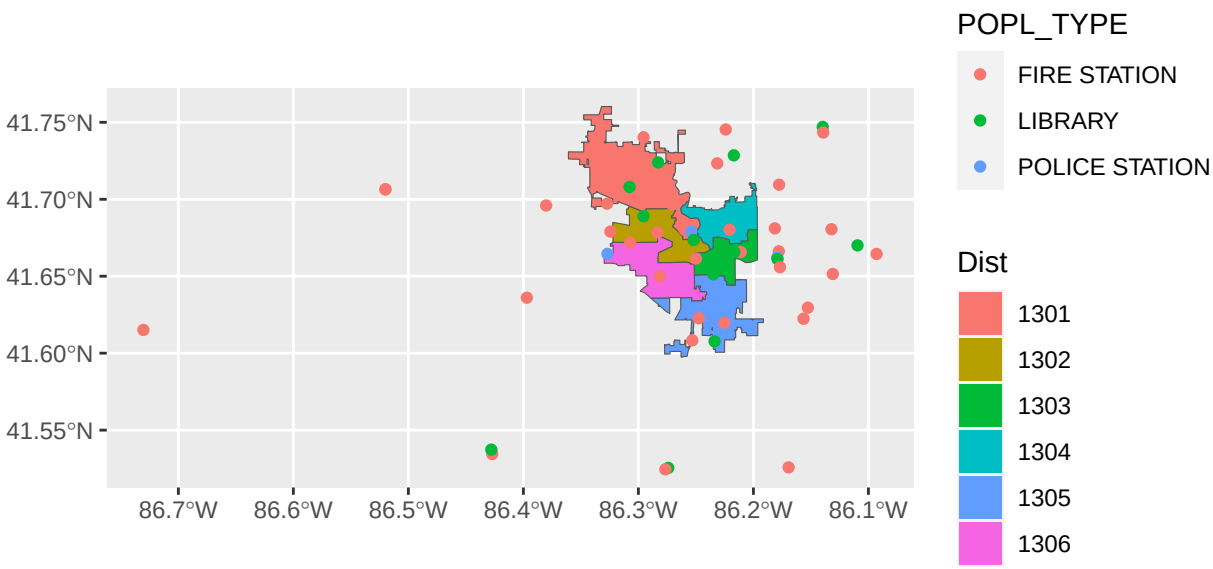
We can now color the map based of off the district.

```
ggplot() +
  geom_sf(data = districts, aes(fill = Dist))
```



We can now add other layers just like we would in normal ggplot. The *show.legend* variable can either be T/F or set what type of geometry to use in the legend. Here, we want these to be points. This does add points to the polygon colors, which we could then fix with manually changes the aes

```
ggplot() +
  geom_sf(data = districts, aes(fill = Dist))+
  geom_sf(data = facilities.spatial, aes(col = POPL_TYPE), show.legend = "point") +
  guides(fill = guide_legend(override.aes = list(colour = NA)))
```

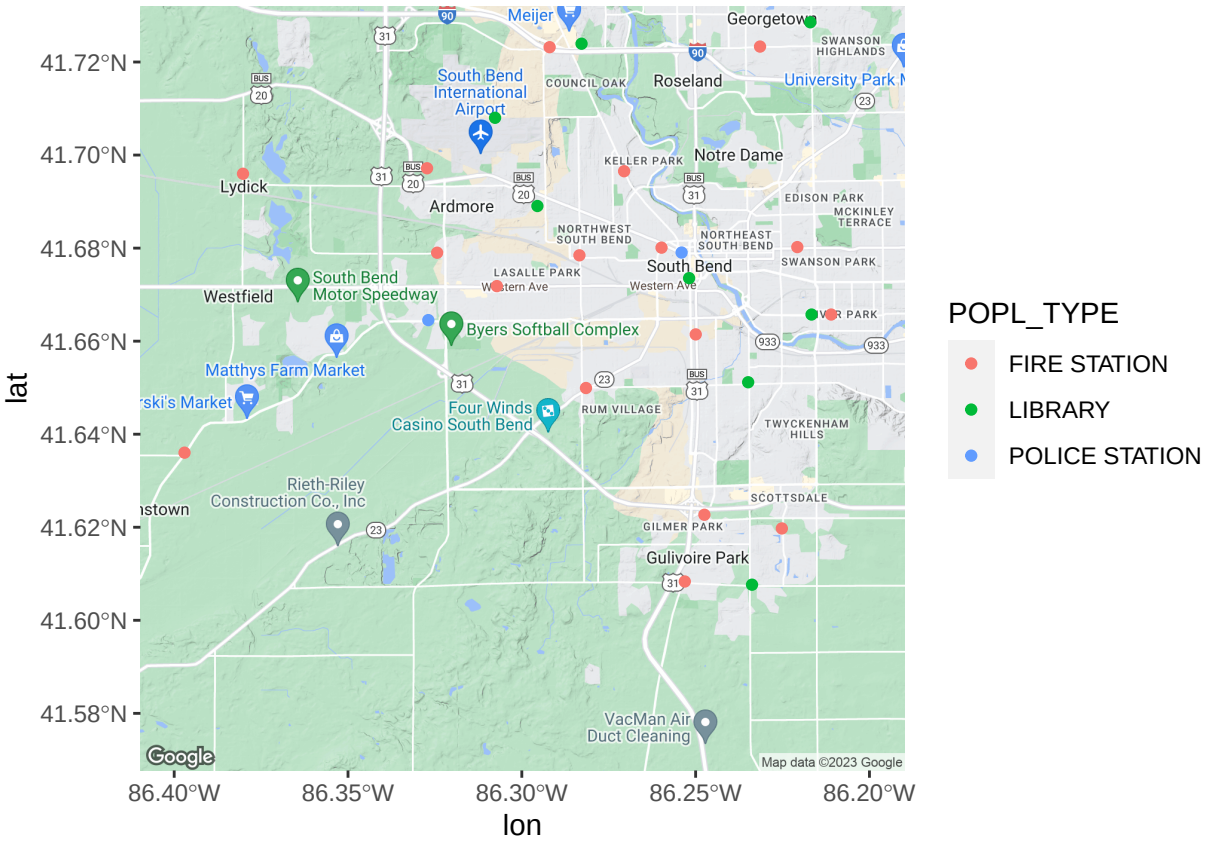


###Downloading Basemaps If it is hard to visualize where there districts or points are, you can also download a static version of one of the common webmaps to use as a basemap. using get_map from the ggmap package will basically cache a single image for you to use. Do note that these basemap layers are usually free to use for academic purposes, but you should look into their terms of service before using them in commercial applications. Make sure your api key (discussed in the Geocoding and Routing Document) is loaded if you want to use the google basemaps.

When doing this with sf objects, you need to add *inherit.aes* = *F* to solve some problems with different coordinate systems.

```
register_google(key = "YOUR KEY HERE")

base.map <- get_map(location = c(-86.3, 41.65), zoom = 12, source = "google")
ggmap(base.map)+
  geom_sf(data = facilities.spatial, aes(col = POPL_TYPE), show.legend = "point", inherit.aes = F)
```


41.72°N
41.70°N
41.68°N
41.66°N
41.64°N
41.62°N
41.60°N
41.58°N
lat
86.40°W
86.35°W
86.30°W
86.25°W
86.20°W
lon
Meijer
Georgetown
SWANSON HIGHLANDS
South Bend International Airport
COUNCIL OAK
Roseland
University Park
KELLER PARK
Notre Dame
Lydick
Ardmore
EDISON PARK
MCKINLEY TERRACE
NORTHWEST SOUTH BEND
NORTHEAST SOUTH BEND
SWANSON PARK
South Bend
Western Ave
LASALLE PARK
Westfield
South Bend Motor Speedway
Byers Softball Complex
Matthys Farm Market
Four Winds Casino South Bend
RUM VILLAGE
TWYCKENHAM HILLS
Rieth-Riley Construction Co., Inc
SCOTTSDALE
GILMER PARK
Gulivoire Park
VacMan Air Duct Cleaning
Google
Map data ©2023 Google
POPL_TYPE
FIRE STATION
LIBRARY
POLICE STATION