Chapter 5 Notes

Table of contents

```{r setup}  
library(tidycensus)  
library(tidyverse)  
```

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.3 ✔ readr 2.1.4  
✔ forcats 1.0.0 ✔ stringr 1.5.0  
✔ ggplot2 3.4.3 ✔ tibble 3.2.1  
✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
✔ purrr 1.0.2   
── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

```{r setup}  
library(scales)  
```

Attaching package: 'scales'  
  
The following object is masked from 'package:purrr':  
  
 discard  
  
The following object is masked from 'package:readr':  
  
 col\_factor

```{r setup}  
library(tigris)  
```

To enable caching of data, set `options(tigris\_use\_cache = TRUE)`  
in your R script or .Rprofile.

```{r setup}  
library(sf)  
```

Linking to GEOS 3.11.2, GDAL 3.6.2, PROJ 9.2.0; sf\_use\_s2() is TRUE

# Chapter 5: Census geographic data and applications in R

* Census and ACS data has associated **geographies**
  + These geographies are represented via the Census Bureau’s *Topologically Integrated Geographic Encoding and Referencing* **(TIGER/Line)** database
  + Most users access datasets as *shapefiles*
  + **TIGER** shapefiles have 33 general data types:
    1. *Legal Entities:* geographies with official legal standing (states, counties, etc).
    2. *Statistical Entities:* geographies defined by Census Bureau for data collection and dissemination (Census tracts, Census block groups, Census blocks)
    3. *Geographic Features:* geographic datasets not linked with demographic data, namely roads, water features, etc.
  + **tigris** package allows for direct importing of TIGER/Line shapefiles without the user needing to wrangle intermediate steps (decompressing from ZIP, etc.)
  + **sf** package is used for representing spatial data as R objects

## 5.1 Basic usage of tigris

* **tigris** functions *download* then *load* requested datasets
  + below we have tigris::states() used to request all states and state equivalent units in US

```{r basic\_usage}  
st <- states()  
```

Retrieving data for the year 2021

* note the Retrieving data alert: **tigris** defaults to most recent year for which a complete set of shapefiles is available
* We can determine the class of our downloaded spatial data now:

```{r class\_check}  
class(st)  
```

[1] "sf" "data.frame"

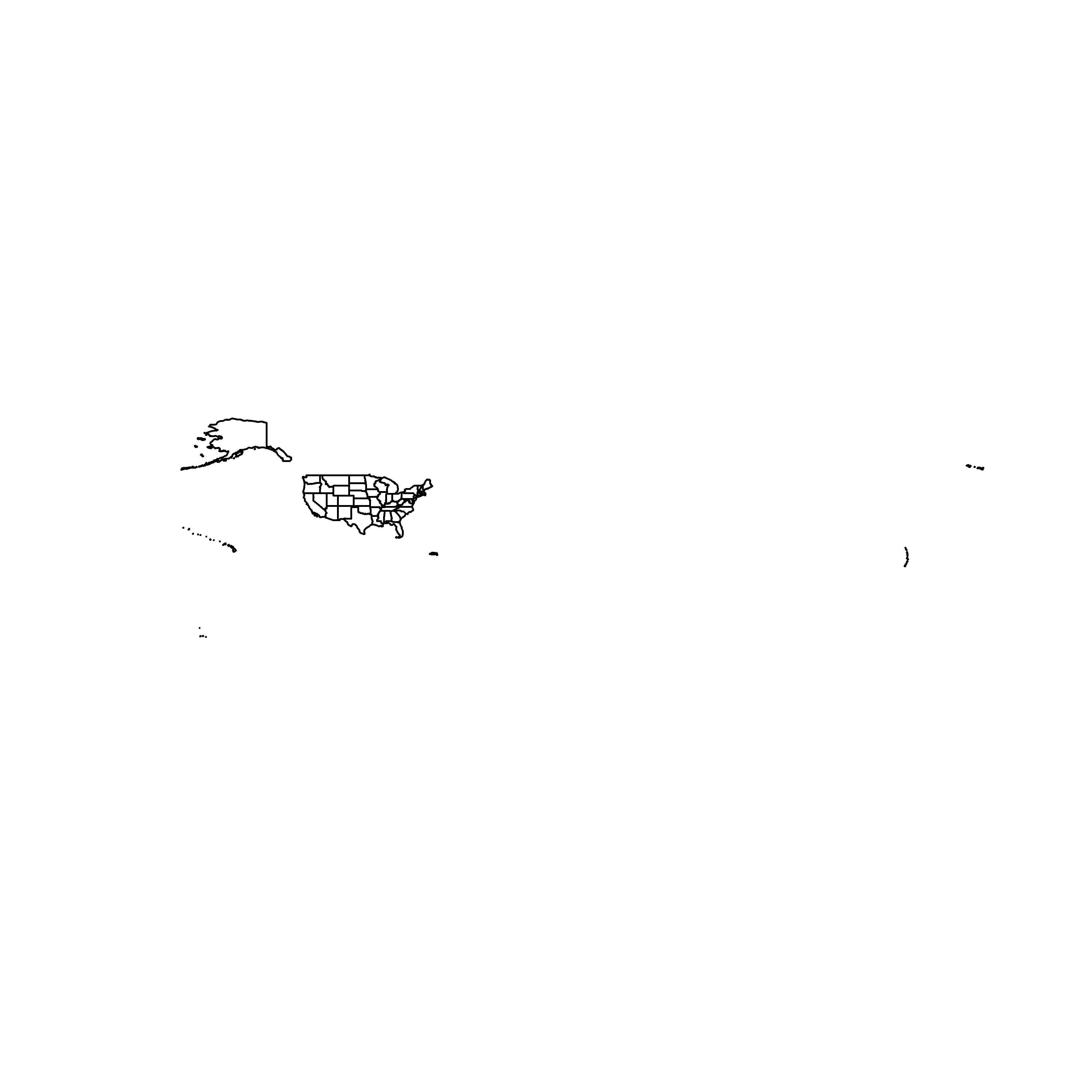
* we have a single object that has two classes - "sf" and "data.frame"
* let’s check the object in more detail:

```{r detail\_return}  
st  
```

Simple feature collection with 56 features and 14 fields  
Geometry type: MULTIPOLYGON  
Dimension: XY  
Bounding box: xmin: -179.2311 ymin: -14.60181 xmax: 179.8597 ymax: 71.43979  
Geodetic CRS: NAD83  
First 10 features:  
 REGION DIVISION STATEFP STATENS GEOID STUSPS NAME LSAD MTFCC  
1 3 5 54 01779805 54 WV West Virginia 00 G4000  
2 3 5 12 00294478 12 FL Florida 00 G4000  
3 2 3 17 01779784 17 IL Illinois 00 G4000  
4 2 4 27 00662849 27 MN Minnesota 00 G4000  
5 3 5 24 01714934 24 MD Maryland 00 G4000  
6 1 1 44 01219835 44 RI Rhode Island 00 G4000  
7 4 8 16 01779783 16 ID Idaho 00 G4000  
8 1 1 33 01779794 33 NH New Hampshire 00 G4000  
9 3 5 37 01027616 37 NC North Carolina 00 G4000  
10 1 1 50 01779802 50 VT Vermont 00 G4000  
 FUNCSTAT ALAND AWATER INTPTLAT INTPTLON  
1 A 62266298634 489204185 +38.6472854 -080.6183274  
2 A 138961722096 45972570361 +28.3989775 -082.5143005  
3 A 143778561906 6216493488 +40.1028754 -089.1526108  
4 A 206232627084 18949394733 +46.3159573 -094.1996043  
5 A 25151992308 6979074857 +38.9466584 -076.6744939  
6 A 2677763359 1323686988 +41.5964850 -071.5264901  
7 A 214049931578 2391569647 +44.3484222 -114.5588538  
8 A 23190115212 1025971768 +43.6726907 -071.5843145  
9 A 125933327733 13456093195 +35.5397100 -079.1308636  
10 A 23872569964 1030754609 +44.0589536 -072.6710173  
 geometry  
1 MULTIPOLYGON (((-80.85847 3...  
2 MULTIPOLYGON (((-83.10874 2...  
3 MULTIPOLYGON (((-89.17208 3...  
4 MULTIPOLYGON (((-92.74568 4...  
5 MULTIPOLYGON (((-75.76659 3...  
6 MULTIPOLYGON (((-71.67881 4...  
7 MULTIPOLYGON (((-111.0455 4...  
8 MULTIPOLYGON (((-71.24548 4...  
9 MULTIPOLYGON (((-76.91598 3...  
10 MULTIPOLYGON (((-72.43462 4...

* we see that we have a fairly typical data frame, except that it has a column we may be unfamiliar with: geometry
  + the geometry column is a column filled with lists, wherein each list contains longitude/latitude coordinate pairs representing each state’s boundary
* Let’s look at the boundaries plotted:

```{r state\_boundaries}  
plot(st$geometry)  
```

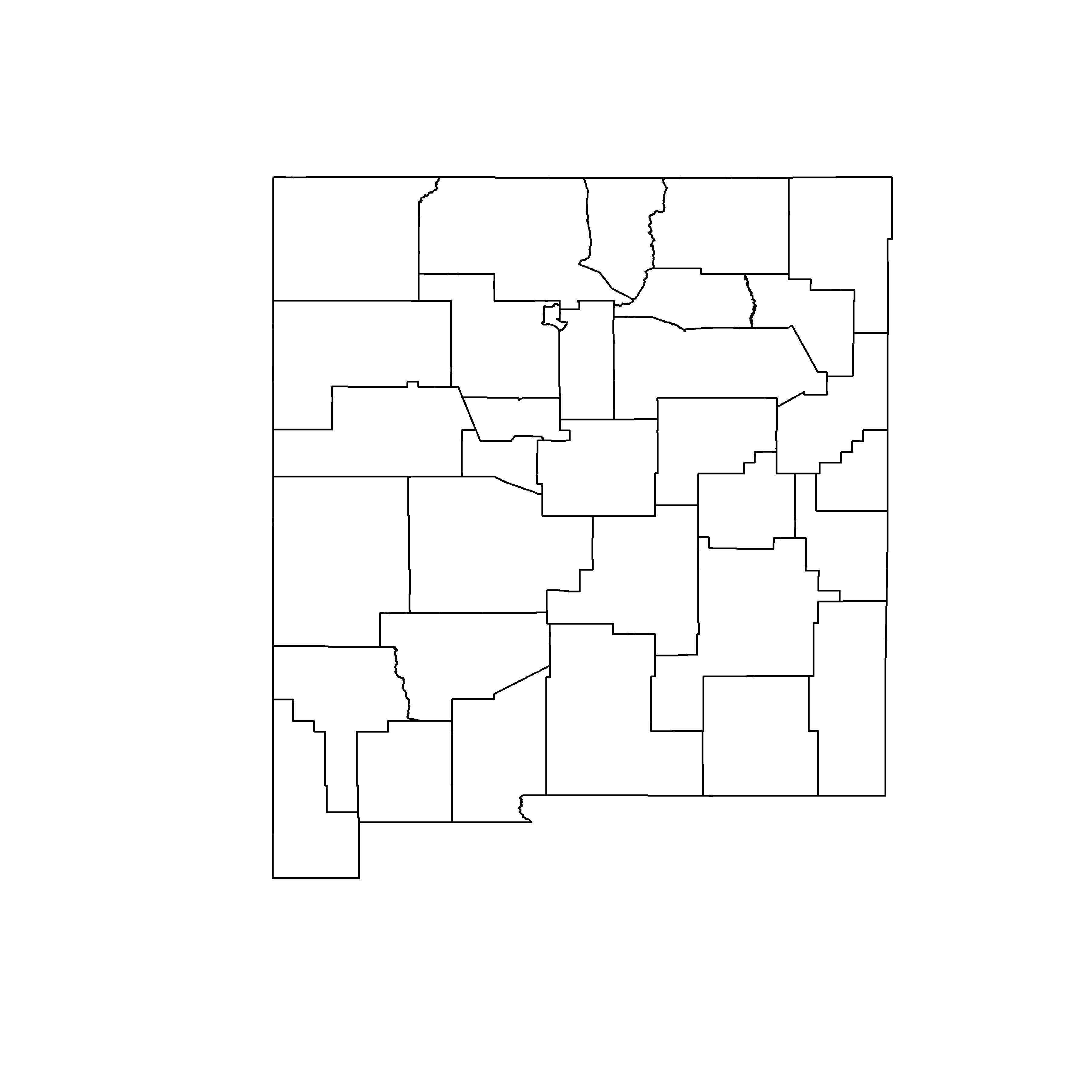


* Census datasets may be available by state or by county
  + depending on use context, we may either, or both, of these units to be designated to pull down the data
    - e.g. counties() pulls down all US counties, but we may use state argument to get those counties belonging to a single State.
      * NOTE: Below, we use “NM”, the State Postal Code abbreviation - we may also use “New Mexico”

```{r nm\_counties}  
nm\_counties <- counties("NM")  
```

Retrieving data for the year 2021

```{r nm\_counties}  
plot(nm\_counties$geometry)  
```

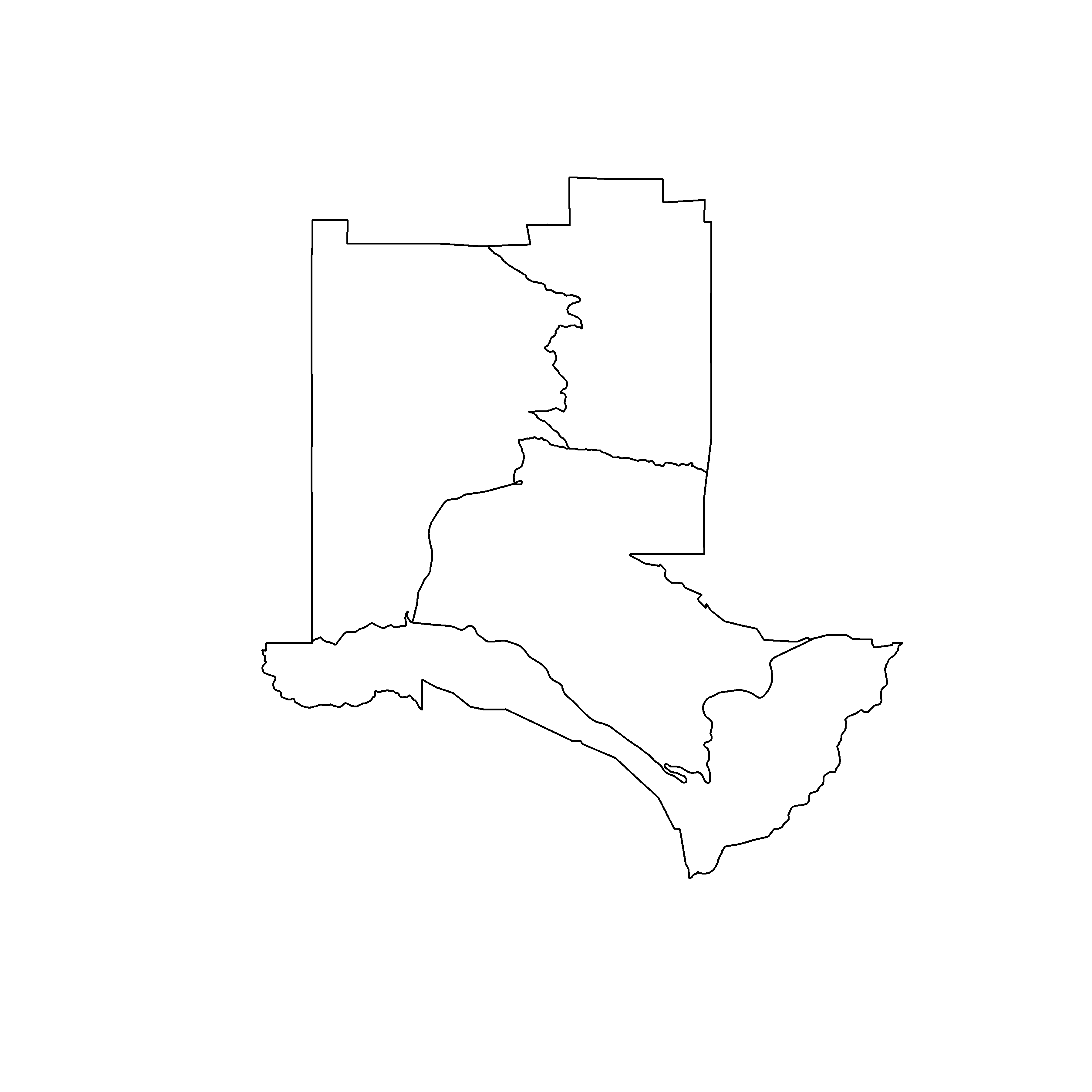


* We have pulled down *Legal Entities* so far, but we can use **tigris** to pull down *Statistical Entities* as well

```{r los\_alamos\_tracts}  
la\_tracts <- tracts("NM", "Los Alamos")  
```

Retrieving data for the year 2021

```{r los\_alamos\_tracts}  
plot(la\_tracts$geometry)  
```

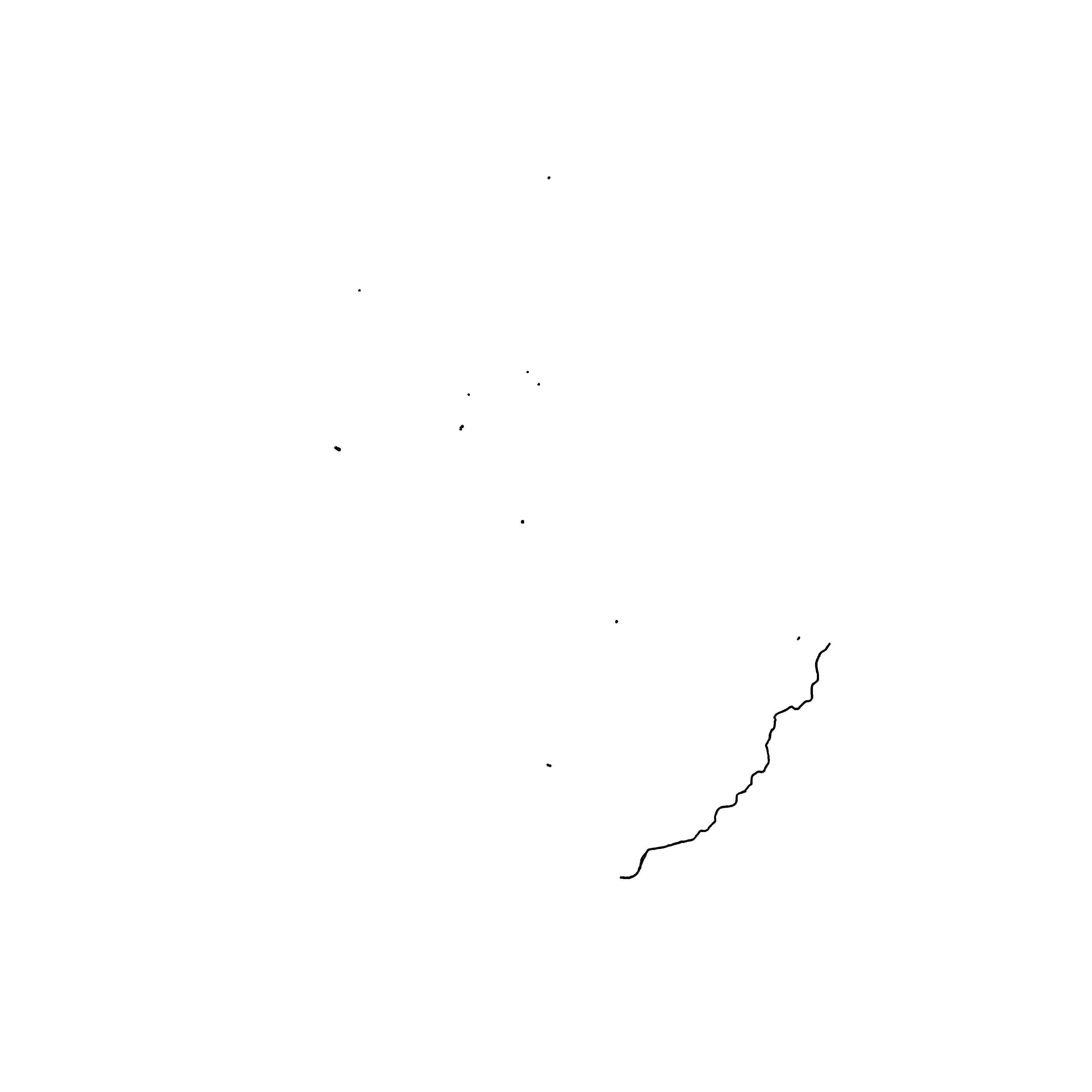


* We can pull down geographic features such as roads and water features; below we pull down Los Alamo County’s water bodies.

```{r los\_alamo\_water}  
la\_water <- area\_water("NM", "Los Alamos")  
```

Retrieving data for the year 2021

```{r los\_alamo\_water}  
plot(la\_water$geometry)  
```



### Understanding tibris and simple features

* **tigris** returns vector data, that is, data represented as points, lines, and polygons.
* in R, we represent vector data with the **sf** package (short for *simple features*, an ISO standard)
  + **sf** is a means to call several C libraries without programming in C; it draws from:
* - \*\*GDAL:\*\* read/write spatial data  
    
  - \*\*GEOS:\*\* model spatial relationships  
    
  - \*\*PROJ:\*\* represent coordinate reference systems
  + **sf** represents vector data as a data frame with a geometry column
    - each entry in the geometry column represents the spatial boundaries/dimensions of the associated feature
    - when an sf object is printed, a data frame is returned with additional geographical context above:
      * geometry type (point, line, polygon, multipoint, multiline, multipolygon, etc)
      * bounding box: the coordinates for the corners of the smallest rectangle that would encompass the entirety of a feature/set of features
      * Coordinate Reference System:

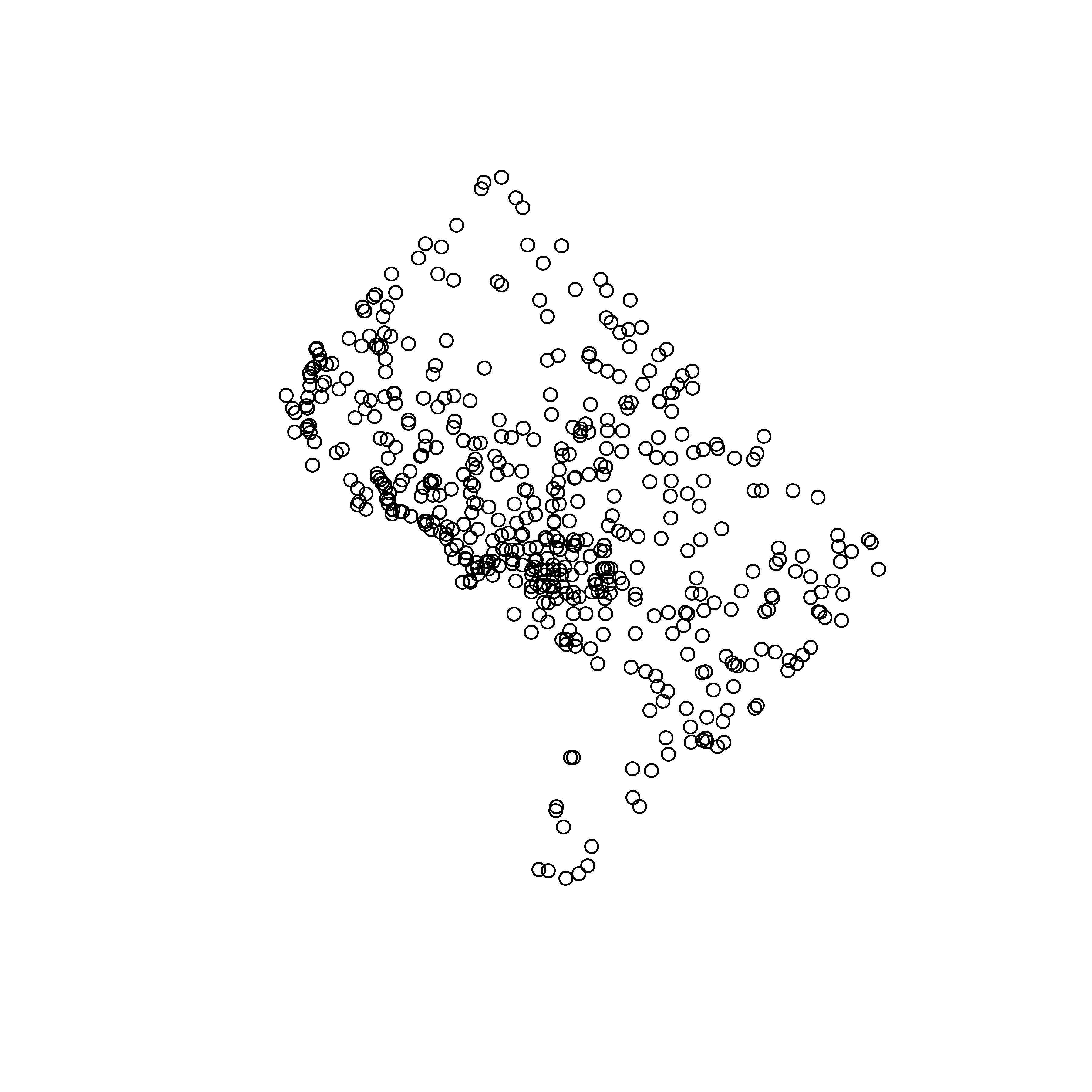
#### Points

* **points** are represented as a single coordinate pair (X,Y)
  + points are 0D
  + sf point objects are geometry type POINT

```{r dc\_landmarks\_points}  
dc\_landmarks <- landmarks("DC", type = "point")  
```

Retrieving data for the year 2021

```{r dc\_landmarks\_points}  
plot(dc\_landmarks$geometry)  
```



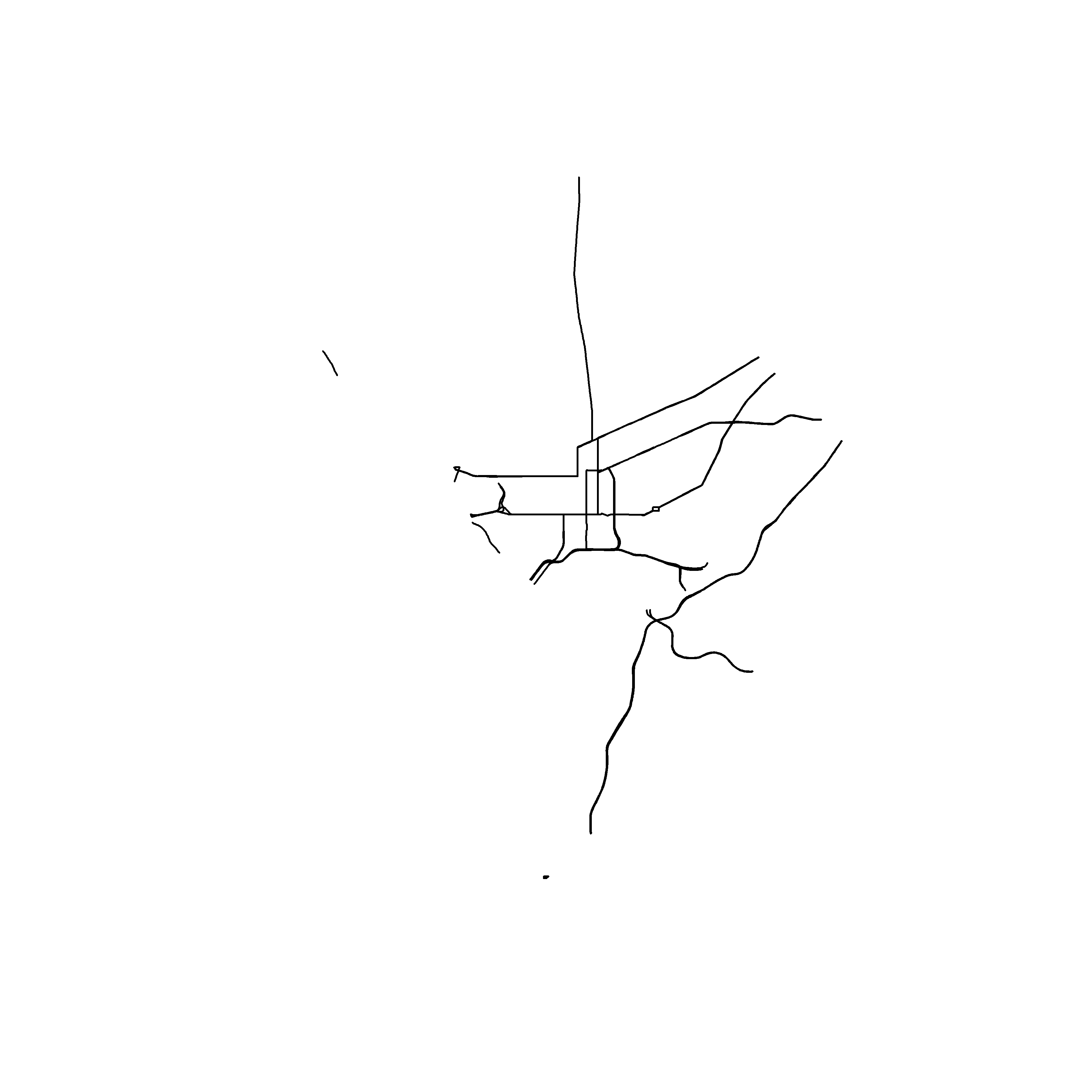
#### Lines

* **Lines** are composed of two or more points (coordinate pairs)
  + Lines are 1D as they only possess Length - the distance between their defining coordinate pairs.
  + Roads, railroads, etc. are represented as lines by TIGER
  + **sf** represents lines with the geometry type LINESTRING

```{r dc\_roads\_lines}  
dc\_roads <- primary\_secondary\_roads("DC")  
```

Retrieving data for the year 2021

```{r dc\_roads\_lines}  
plot(dc\_roads$geometry)  
```



#### Polygons

* **Polygons** are enclosed shapes defined by three or more connected coordinate pairs.
  + TIGER polygons include features such as block groups, tracts, etc.
  + Polygons are 2D.

#### Multis

* The above types each have a corresponding multi counterpart for more complex relationships
  + MULTIPOINT
  + MULTILINESTRING
  + MULTIPOLYGON

### Data availability in tigris

* check the book [here](https://walker-data.com/census-r/census-geographic-data-and-applications-in-r.html#data-availability-in-tigris) for a full listing of data set functions available in tigris

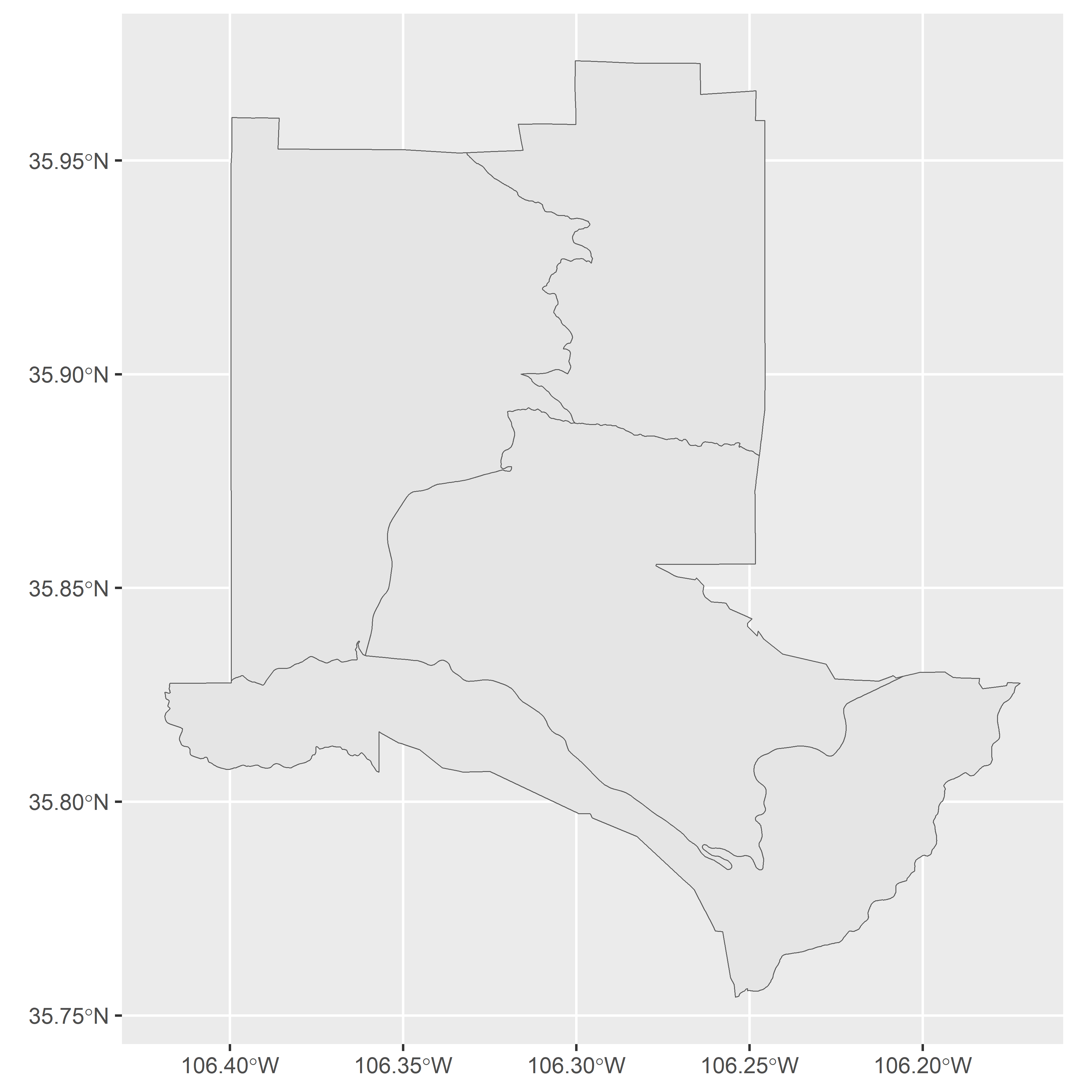
## 5.2 Plotting geographic data

* while base plot() has been extended and is capable of visualizing geometry, more options exist for effective spatial visualization

### ggplot2 and geom\_sf()

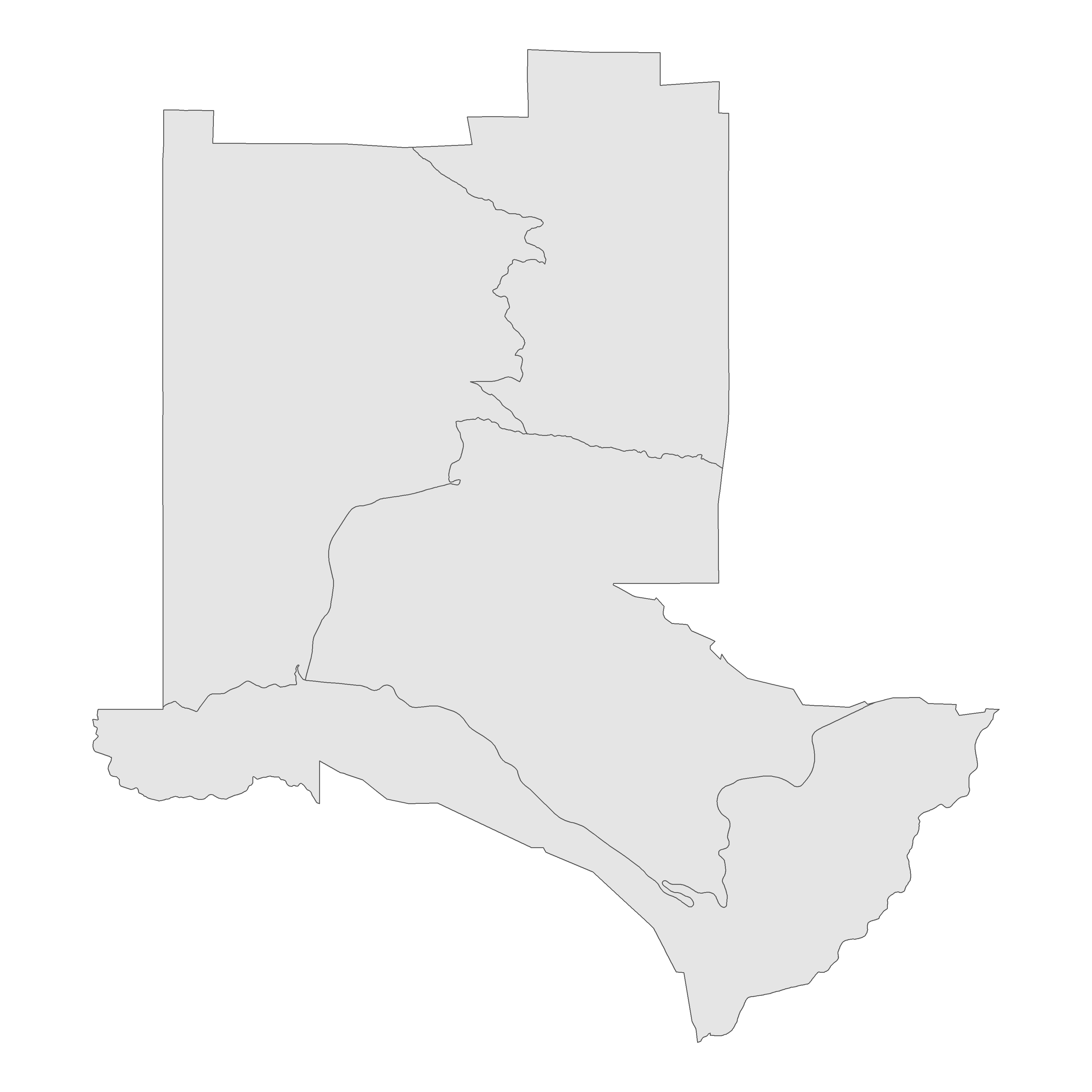
* geom\_sf() introduced in **ggplot2** 3.0, supporting simple feature plotting

```{r los\_alamos\_geom}  
ggplot(la\_tracts) +  
 geom\_sf()  
```



* by default, **ggplot2** uses standard grey grid, with lat/long in Decimal Degrees
* theme\_void() removes the background map elements

```{r los\_alamos\_void}  
ggplot(la\_tracts) +  
 geom\_sf() +  
 theme\_void()  
```

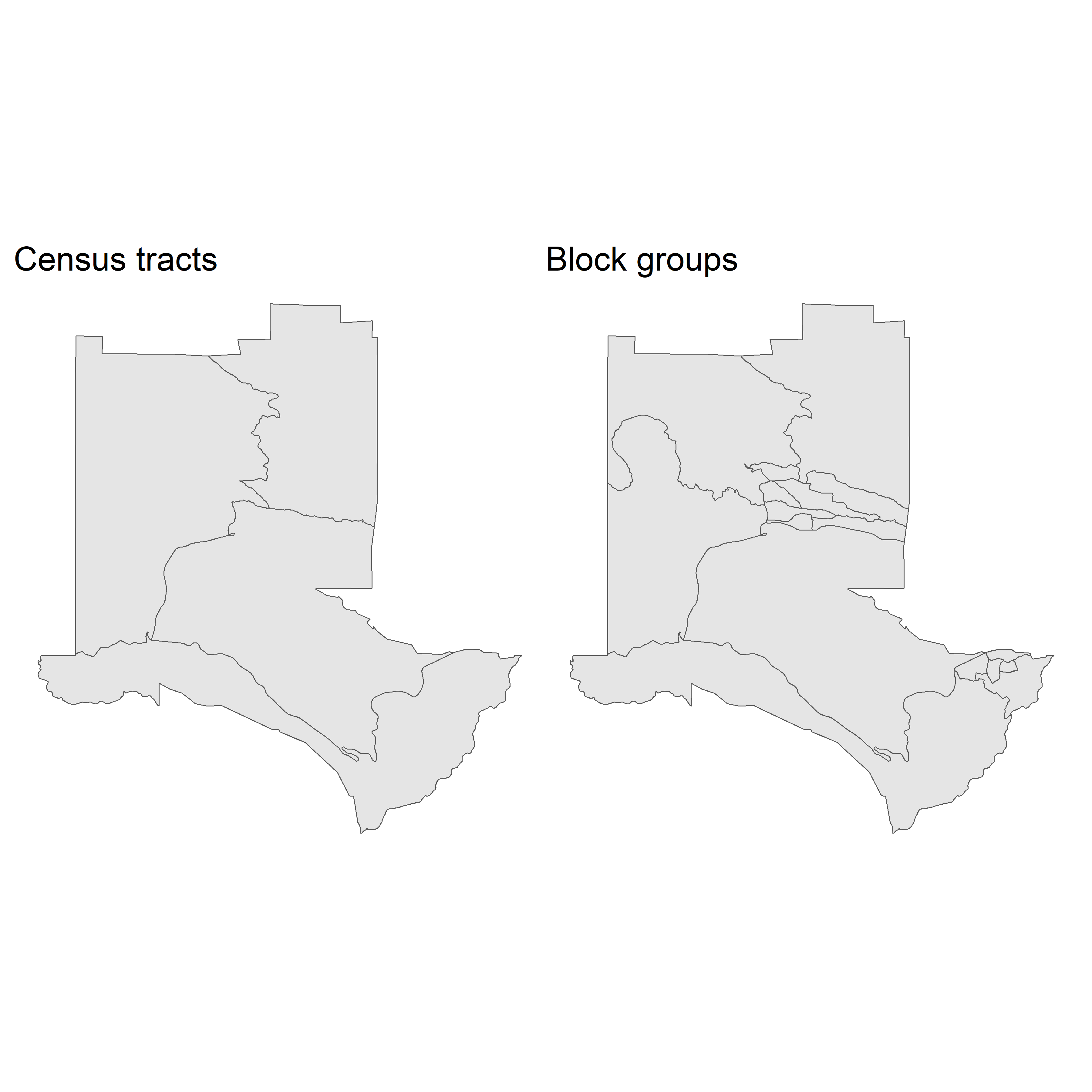


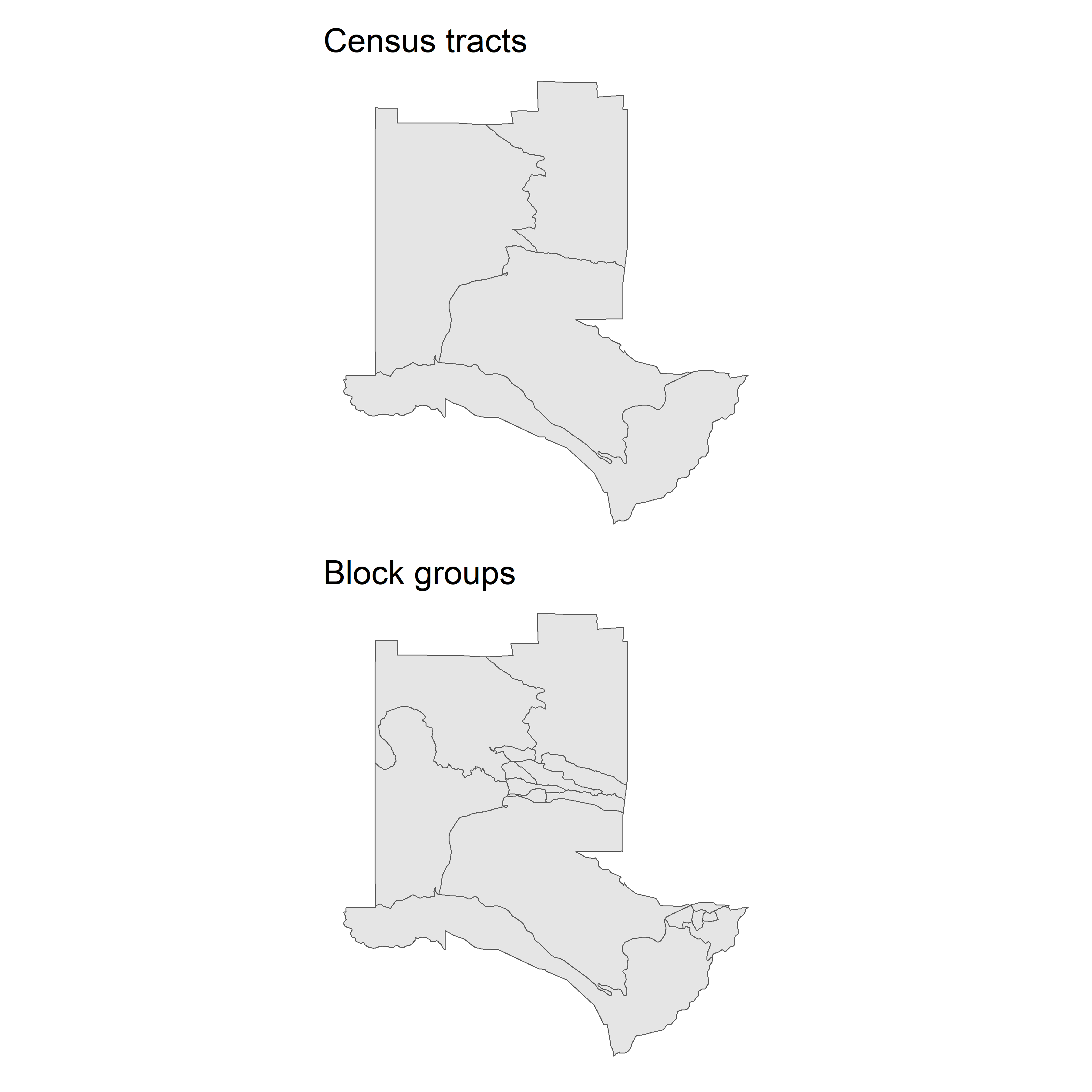
* **patchwork** allows us to arrange multi-plot layouts with **ggplot2** spatial visuals

```{r patchwork\_los\_alamos}  
library(patchwork)  
  
la\_block\_groups <- block\_groups("NM", "Los Alamos")  
```

Retrieving data for the year 2021

```{r patchwork\_los\_alamos}  
gg1 <- ggplot(la\_tracts) +  
 geom\_sf() +  
 theme\_void() +  
 labs(title = "Census tracts")  
  
gg2 <- ggplot(la\_block\_groups) +  
 geom\_sf() +  
 theme\_void() +   
 labs(title = "Block groups")  
  
gg1 + gg2 # horizontal arrangement  
gg1 / gg2 # vertical arrangement  
```





### Interactive viewing with mapview

* For many years, R did not have the capabilities for interactive visual exploration
  + this all changed with the development of htmlwidgets
* We can use mapview() to make interactive, zoomable maps

```{r la\_mapview}  
library(mapview)  
```

The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,  
which was just loaded, will retire in October 2023.  
Please refer to R-spatial evolution reports for details, especially  
https://r-spatial.org/r/2023/05/15/evolution4.html.  
It may be desirable to make the sf package available;  
package maintainers should consider adding sf to Suggests:.  
The sp package is now running under evolution status 2  
 (status 2 uses the sf package in place of rgdal)

```{r la\_mapview}  
mapview(la\_tracts)  
```

## 5.3 tigris workflows

This section covers some of the other functions present in **tigris** to integrate it into broader spatial analyses

### TIGER/Line and cartographic boundary shapefiles

* **tigris** includes functions to download *cartographic boundary shapefiles*
  + derived from TIGER/Line shapefiles
    - generalized interior
    - clipped to shore
    - preferred for Thematic mapping
    - may be pulled down by argument cb = TRUE
  + TIGER/LINE shapefiles (may) represent “official” county areas - including water

**Demo of default and cartographic boundary differences for Michigan**

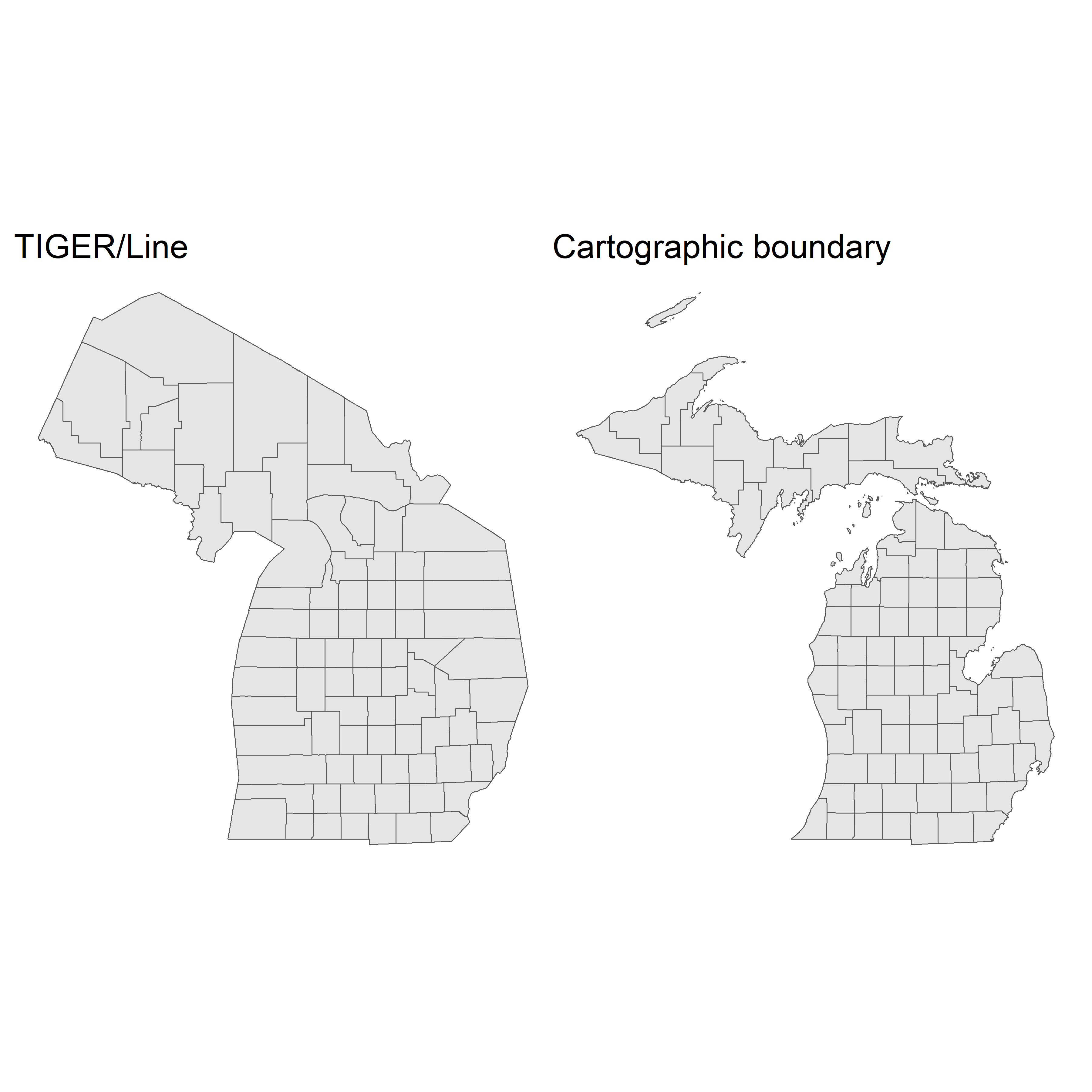
```{r michigan\_cb\_tiger}  
mi\_counties <- counties("MI")  
```

Retrieving data for the year 2021

```{r michigan\_cb\_tiger}  
mi\_counties\_cb <- counties("MI", cb = TRUE)  
```

Retrieving data for the year 2021

```{r michigan\_cb\_tiger}  
mi\_tiger\_gg <- ggplot(mi\_counties) +   
 geom\_sf() +  
 theme\_void() +  
 labs(title = "TIGER/Line")  
  
mi\_cb\_gg <- ggplot(mi\_counties\_cb) +  
 geom\_sf() +  
 theme\_void() +  
 labs(title = "Cartographic boundary")  
  
mi\_tiger\_gg + mi\_cb\_gg  
```



* when using cb = TRUE, we also have an available resolution argument that may be set to one of three options with increasing levels of generalization and decreasing relative file size:
  1. resolution = "500k": default resolution, 1:500,000
  2. resolution = "5m": 1:5,000,000
  3. resolution = "20m" 1:25,000,000

### Caching tigris data

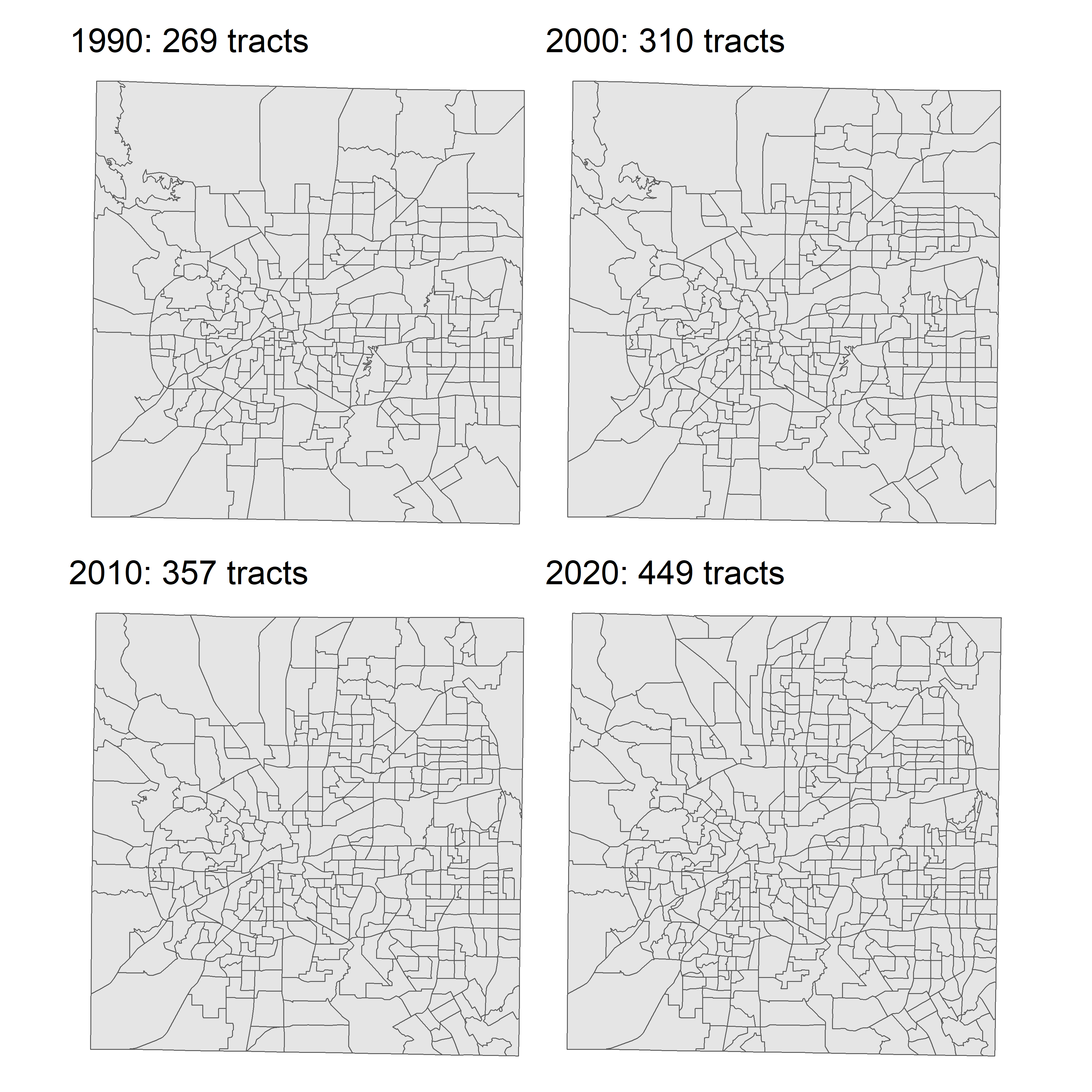
* options can be set to cache tigris data for repeat usage
  + especially useful for larger files or poorer quality internet connections
  + to set the option, use options(tigris\_use\_cache = TRUE)
  + to check the tigris cache directory, use rappdirs::user\_cache\_dir("tigris")
  + Use tigris\_cache\_dir() to set a different location for cache files

### Understanding yearly differences in TIGER/Line files

* The Census Bureau has a time series of TIGER/Line shapefiles for years 1990 thru 2020.
* Some geographies are mostly static across time (State boundaries)
* Others change frequently due to demographic shifts (enumeration units)
* The Census Bureau tries to keep census tract populations around 4000 people
  + fast growing areas are frequently subdivided and redrawn to account for this
* We will demo the changes in Tarrant, TX using ggplot(), purrr::map(), glue(), and **patchwork**

**Tarrant, TX Demo**

```{r tarrant\_tract\_changes}  
library(tidyverse)  
library(patchwork)  
library(glue)  
  
yearly\_plots <- map(seq(1990, 2020, 10), ~{ # for each decade from 1990 to 2020:  
 year\_tracts <- tracts("TX","Tarrant", # year\_tracts gets Tarrant, TX tracts  
 year = .x, # pulled for each decade  
 cb = TRUE) # returning the cartographic boundaries  
   
 ggplot(year\_tracts) + # initialize a plot for each decade  
 geom\_sf() + # plot it with the sf geometry  
 theme\_void() + # remove map elements  
 labs(  
 title = glue("{.x}: {nrow(year\_tracts)} tracts")  
 ) # label each plot the decade and the number of tracts   
 }  
)  
  
# create the patchwork plot  
(yearly\_plots[[1]] + yearly\_plots[[2]]) /  
 (yearly\_plots[[3]] + yearly\_plots[[4]])  
```



* From this, we can see a gain of 180 census tracts in Tarrant County from 1990 to 2020
  + These changes are especially important for time-series analyses at the census tract level; data from 2010 will for different geographies (different census tracts) than for 2020.
    - one means of accounting for this is areal interpolation, covered in chapter 7
* by default, **tigris** uses the most recent year for which cartographic boundary shapefiles are fully available
  + presently, this is 2021
  + options(tigris\_year) can be used to modify the default year

### Combining tigris datasets

* From 2019 onward, the Census releases national small-area cartographic boundary files
  + specify cb = TRUE to get such cartographic boundaries

```{r national\_boundaries}  
us\_bgs\_2020 <- block\_groups(cb = TRUE, year = 2020)  
```

Retrieving Census block groups for the entire United States

```{r national\_boundaries}  
nrow(us\_bgs\_2020)  
```

[1] 242298

* this option is not available for 2018 and earlier
* we can generate them using existing **tidyverse** tools such as purrr::map(), specifically purrr::map\_dfr()

**Demonstration for 2018, all states + DC and Puerto Rico**

```{r}  
state\_codes <- c(state.abb, "DC", "PR")  
  
us\_bgs\_2018 <- map\_dfr(  
 state\_codes,  
 ~block\_groups(  
 state = .x,  
 cb = TRUE,  
 year = 2018  
 )  
)  
  
nrow(us\_bgs\_2018)  
```

[1] 220016

* The initial call is notably long, as the cache hasn’t been developed.
  + once files are cached, processes are much quicker.

## 5.4 Coordinate reference systems

* data is *referenced* to some point on the Earth’s surface using an appropriate Earth model
  + A model the data is referenced to is called its Coordinate Reference System
  + Two types:
    - *Geographic Coordinate System*
    - *Projected Coordinate System*
* Default CRS for tigris data is “NAD83”
* sf::st\_crs() WKT representation of the CRS

```{r fl\_counties\_nad83}  
library(sf)  
fl\_counties <- counties("FL", cb = TRUE)  
```

Retrieving data for the year 2021

```{r fl\_counties\_nad83}  
st\_crs(fl\_counties)  
```

Coordinate Reference System:  
 User input: NAD83   
 wkt:  
GEOGCRS["NAD83",  
 DATUM["North American Datum 1983",  
 ELLIPSOID["GRS 1980",6378137,298.257222101,  
 LENGTHUNIT["metre",1]]],  
 PRIMEM["Greenwich",0,  
 ANGLEUNIT["degree",0.0174532925199433]],  
 CS[ellipsoidal,2],  
 AXIS["latitude",north,  
 ORDER[1],  
 ANGLEUNIT["degree",0.0174532925199433]],  
 AXIS["longitude",east,  
 ORDER[2],  
 ANGLEUNIT["degree",0.0174532925199433]],  
 ID["EPSG",4269]]

* we can see the Datum, the ellipsoid (generalized 3D model of Earth’s shape), Prime Meridian used, and the unique EPSG ID
* **sf** uses **s2** to perform spheroid calculations at global or regional scales
  + when working with smaller areas, *projected CRS* will be preferred
  + **crsuggest helps choose appropriate CRS**

#### Using the crsuggest package

* suggest\_crs() will return a tibble of recommended projections for the provided data
  + it does this by comparing input geometry against built-in dataset extents and finding the pairing with the smallest “Hausdorff distance”
  + Below we run it for Florida and find a recommendation for “Florida GDL Albers”

```{r fl\_counties\_crs}  
library(crsuggest)  
```

Using the EPSG Dataset v10.019, a product of the International Association of Oil & Gas Producers.   
Please view the terms of use at https://epsg.org/terms-of-use.html.

```{r fl\_counties\_crs}  
fl\_crs <- suggest\_crs(fl\_counties)  
  
fl\_crs  
```

# A tibble: 10 × 6  
 crs\_code crs\_name crs\_type crs\_gcs crs\_units crs\_proj4  
 <chr> <chr> <chr> <dbl> <chr> <chr>   
 1 6439 NAD83(2011) / Florida GDL Albe… project… 6318 m +proj=ae…  
 2 3513 NAD83(NSRS2007) / Florida GDL … project… 4759 m +proj=ae…  
 3 3087 NAD83(HARN) / Florida GDL Albe… project… 4152 m +proj=ae…  
 4 3086 NAD83 / Florida GDL Albers project… 4269 m +proj=ae…  
 5 6443 NAD83(2011) / Florida West (ft… project… 6318 us-ft +proj=tm…  
 6 6442 NAD83(2011) / Florida West project… 6318 m +proj=tm…  
 7 3517 NAD83(NSRS2007) / Florida West… project… 4759 us-ft +proj=tm…  
 8 3516 NAD83(NSRS2007) / Florida West project… 4759 m +proj=tm…  
 9 2882 NAD83(HARN) / Florida West (ft… project… 4152 us-ft +proj=tm…  
10 2778 NAD83(HARN) / Florida West project… 4152 m +proj=tm…

* we can then project it to our chosen *projected* CRS as follows:

```{r fl\_projected\_demo}  
fl\_projected <- st\_transform(fl\_counties, crs = 3087)  
head(fl\_projected)  
```

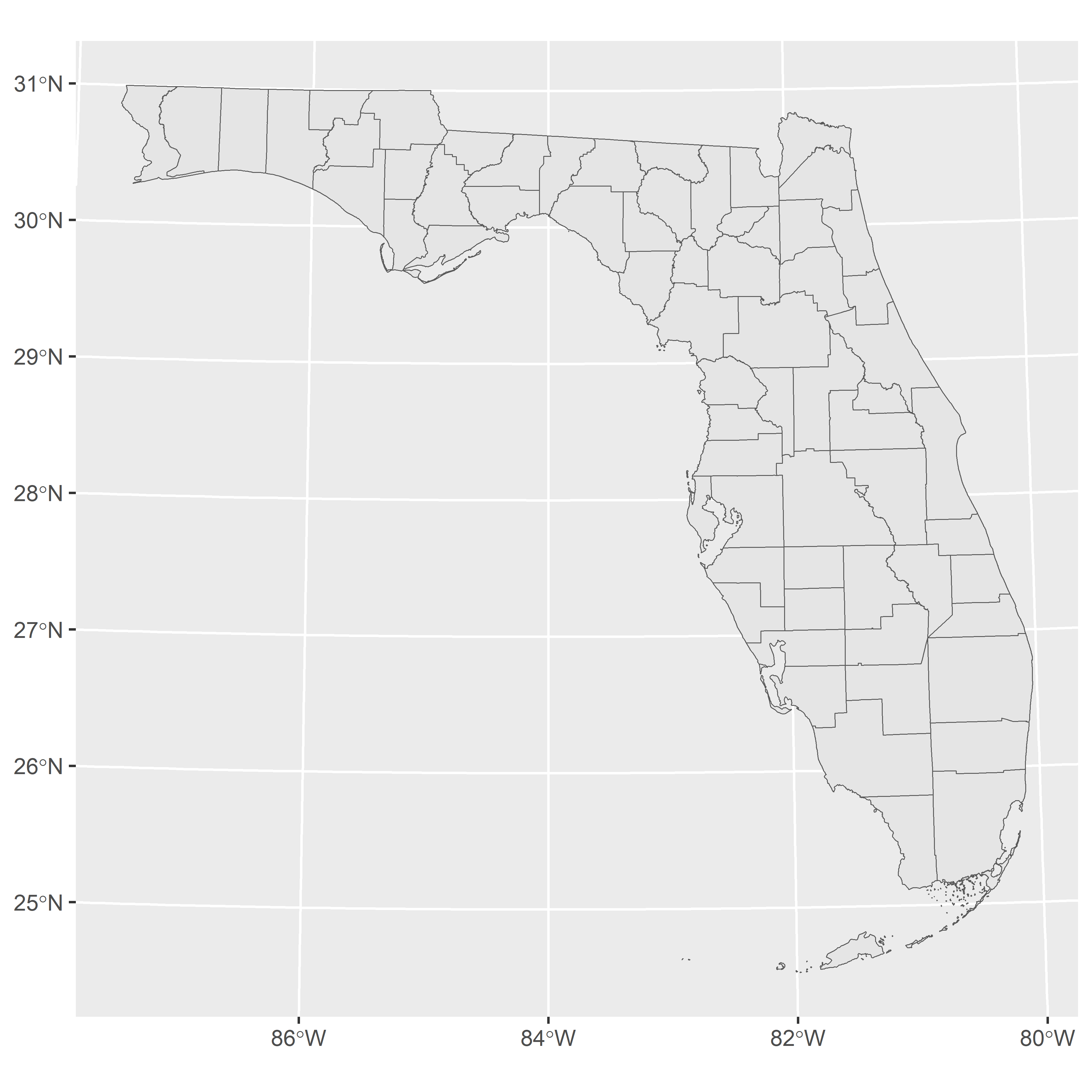
Simple feature collection with 6 features and 12 fields  
Geometry type: MULTIPOLYGON  
Dimension: XY  
Bounding box: xmin: 537215 ymin: 406036.2 xmax: 718307.8 ymax: 683112.8  
Projected CRS: NAD83(HARN) / Florida GDL Albers  
 STATEFP COUNTYFP COUNTYNS AFFGEOID GEOID NAME NAMELSAD  
67 12 095 00295750 0500000US12095 12095 Orange Orange County  
88 12 125 00306913 0500000US12125 12125 Union Union County  
96 12 069 00308551 0500000US12069 12069 Lake Lake County  
126 12 127 00306921 0500000US12127 12127 Volusia Volusia County  
184 12 105 00295747 0500000US12105 12105 Polk Polk County  
215 12 119 00295740 0500000US12119 12119 Sumter Sumter County  
 STUSPS STATE\_NAME LSAD ALAND AWATER geometry  
67 FL Florida 06 2338519047 260364621 MULTIPOLYGON (((628232.9 52...  
88 FL Florida 06 630804541 16049423 MULTIPOLYGON (((537215 6656...  
96 FL Florida 06 2464981507 531018779 MULTIPOLYGON (((599838.7 49...  
126 FL Florida 06 2852382099 857613983 MULTIPOLYGON (((624847.1 59...  
184 FL Florida 06 4656518803 550424838 MULTIPOLYGON (((585432.3 47...  
215 FL Florida 06 1442978468 58762672 MULTIPOLYGON (((564282.7 55...

* the dimensions and coordinates for our spatial object have changed:
  + distance is now measured in meters
  + instead of latitude and longitude, we have Northings and Eastings
  + the dataset now uses a ‘false origin’ different from the 0,0 of the intersection of the Prime Meridian and the Equator
* These changes make planar geometry easier

#### Plotting with coord\_sf()

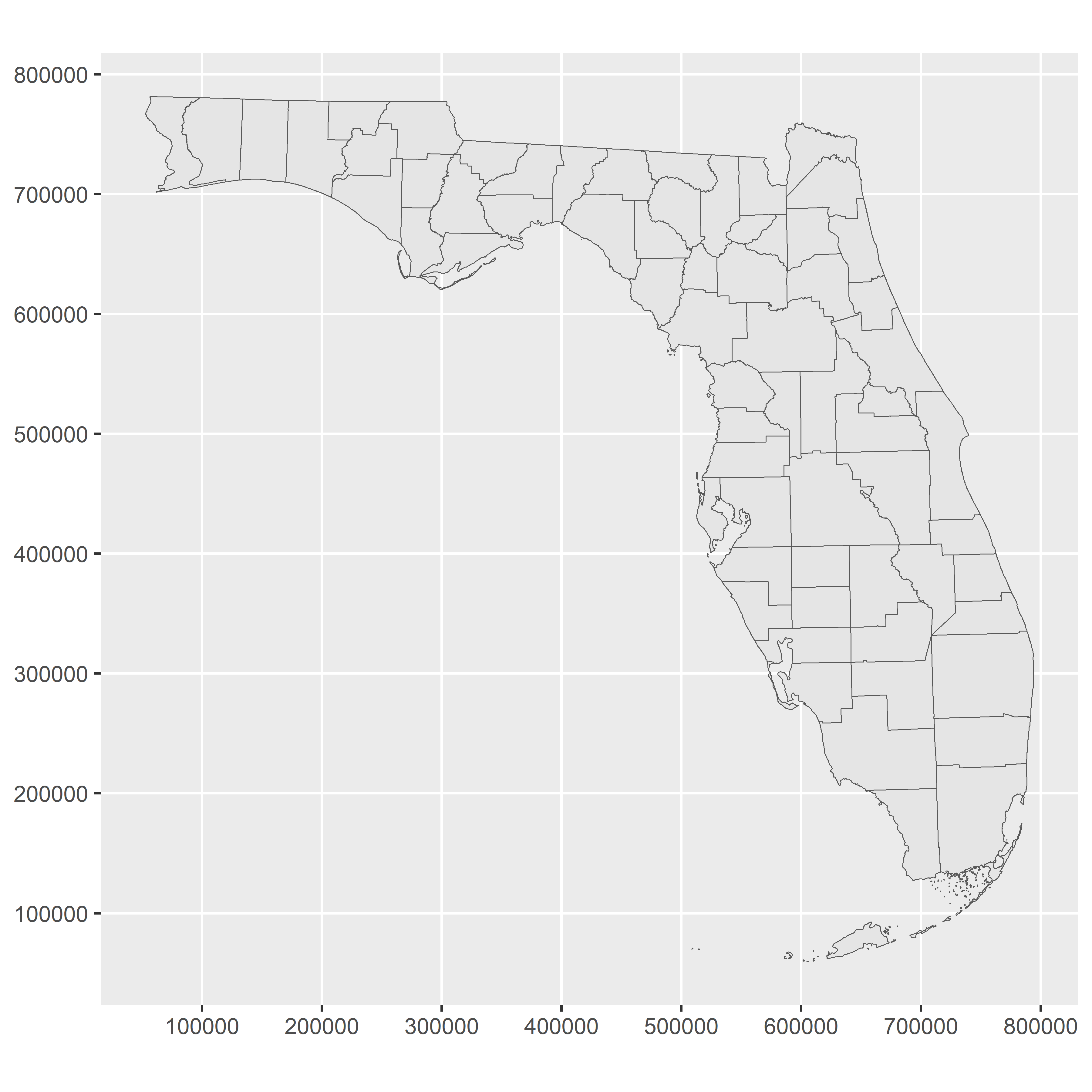
* coord\_sf can be paired with geom\_sf to transform a plot’s crs in situ

```{r}  
options(scipen = 999)  
  
ggplot(fl\_counties) +  
 geom\_sf() +  
 coord\_sf(crs = 3087)  
```



* the chart still defaults to long/lat; to display the datum units, use the datum argument

```{r}  
ggplot(fl\_counties) +  
 geom\_sf() +  
 coord\_sf(crs = 3087, datum = 3087)  
```



## 5.5 Working with geometries

* This section demonstrates several geometric operations that will be useful in constructing spatial visuals moving forward.

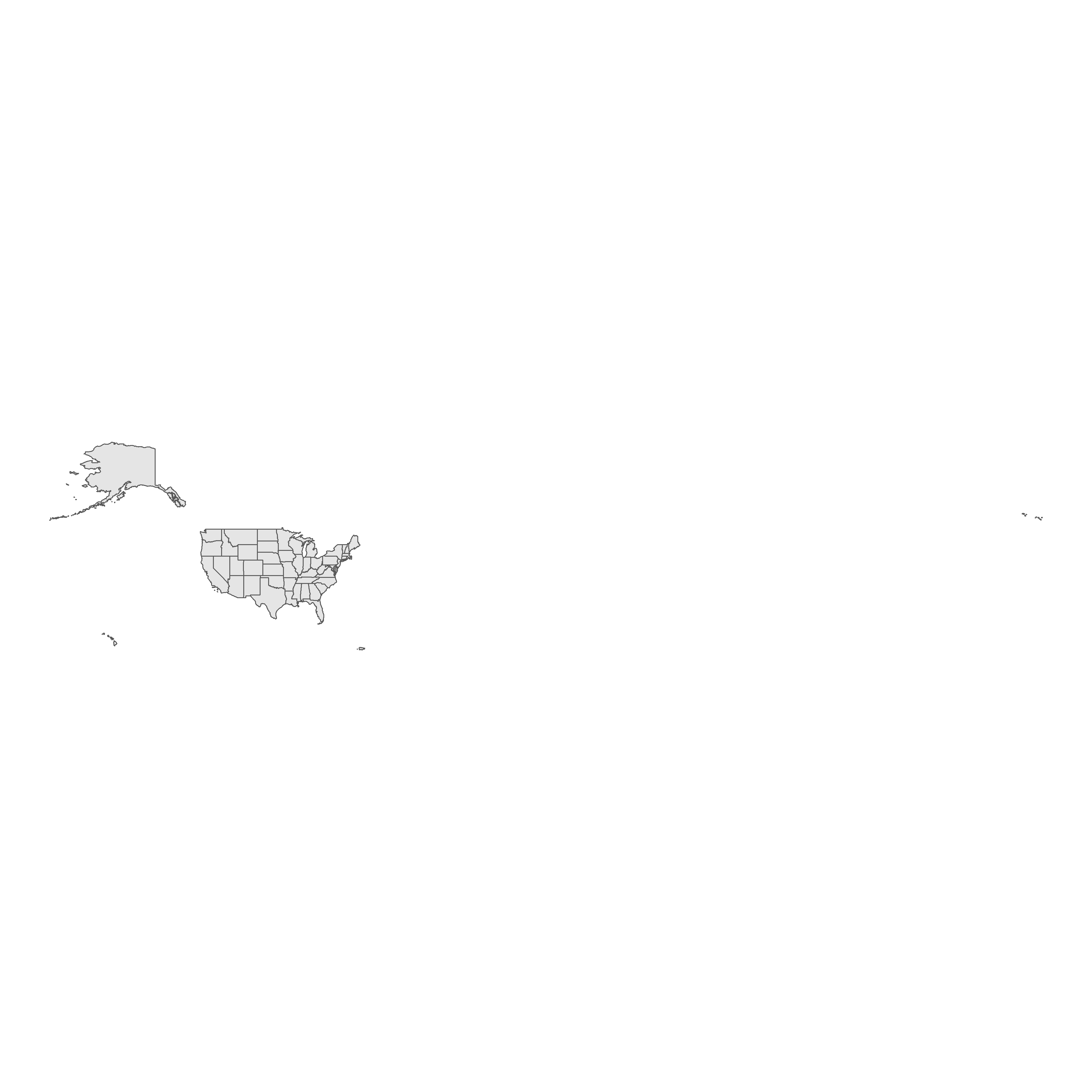
### Shifting and rescaling geometry for national US mapping

* A common issue in US mapping is determining how to handle the non-contiguous States and territories

```{r us\_unshifted}  
us\_states <- states(cb = TRUE, resolution = "20m")  
```

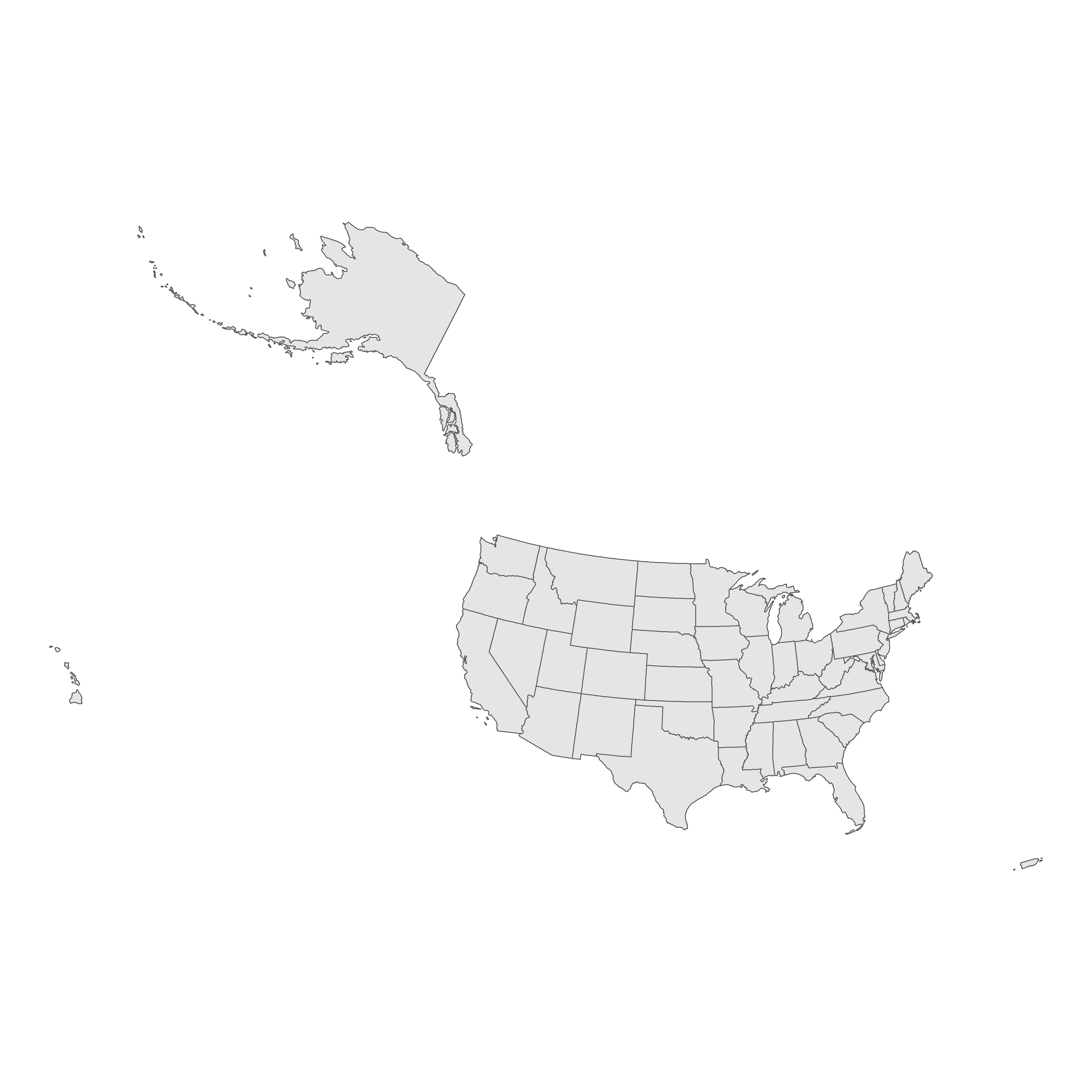
Retrieving data for the year 2021

```{r us\_unshifted}  
ggplot(us\_states) +  
 geom\_sf() +  
 theme\_void()  
```



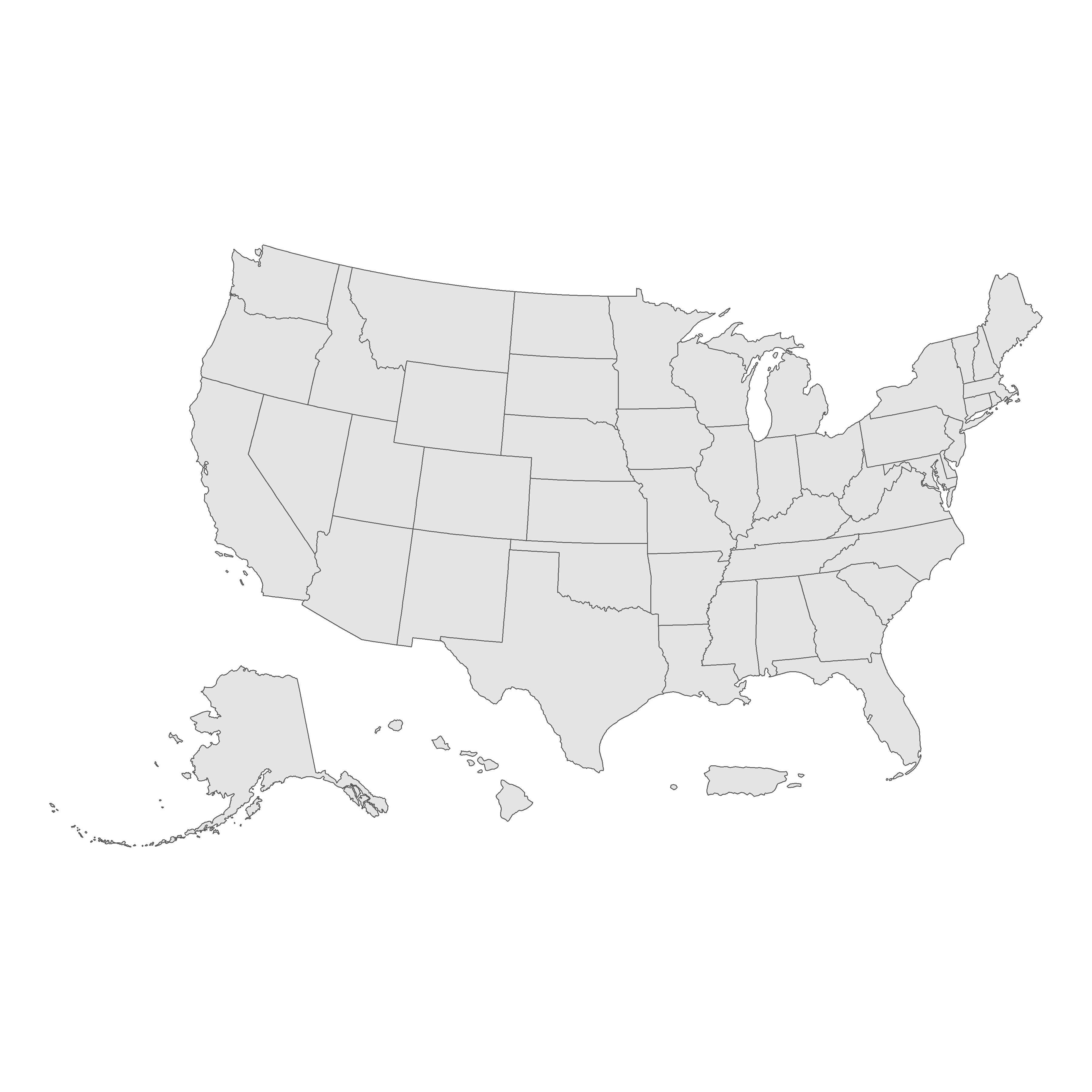
* We can try to reproject the data so that, for example, Alaska doesn’t stretch across 180° Long
  + Note below that Alaska, Hawaii, and Puerto Rico appear distorted by curvature

```{r US\_Albers\_Equal\_Area}  
ggplot(us\_states) +  
 geom\_sf() +  
 coord\_sf(crs = 'ESRI:102003') +  
 theme\_void()  
```



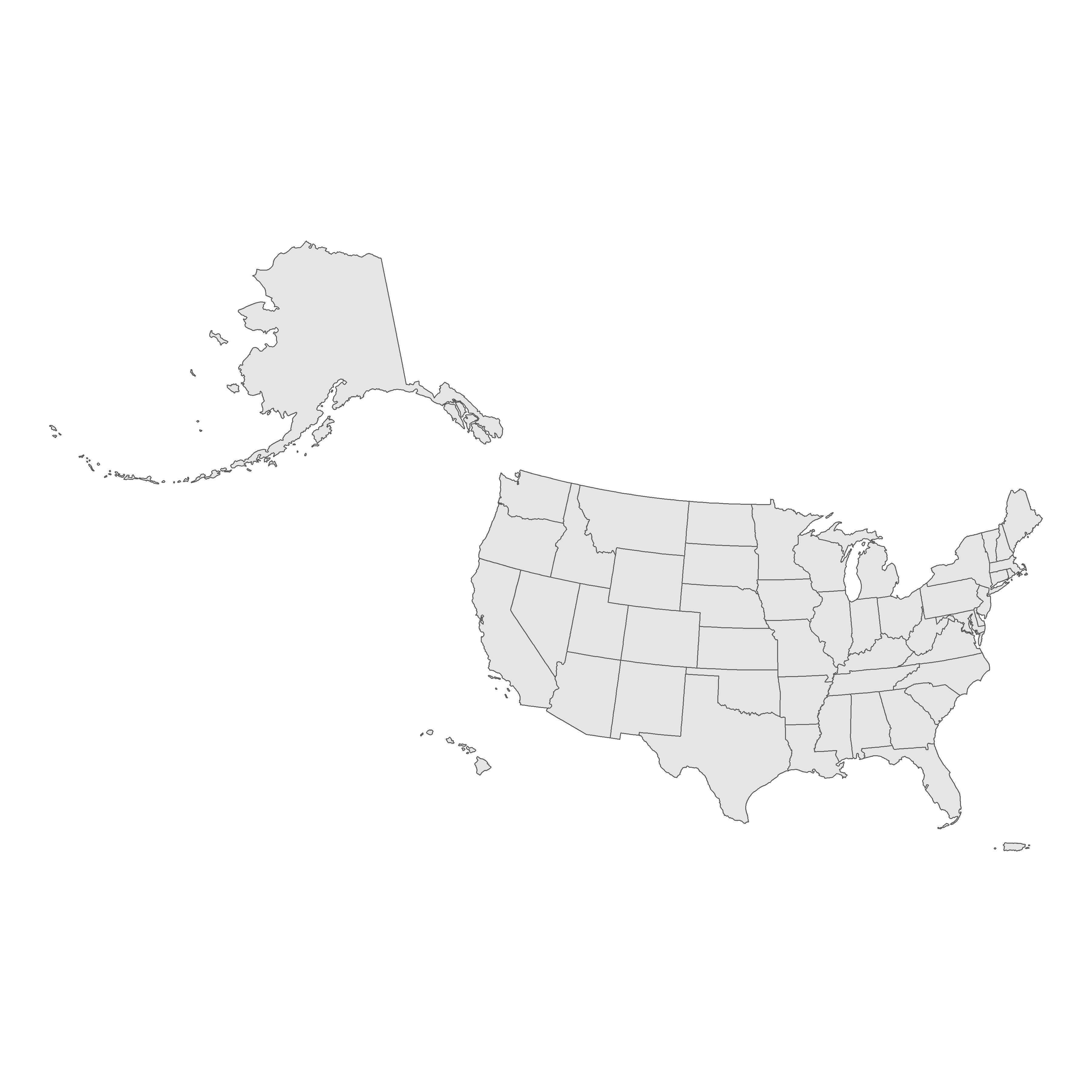
* **tigris** has a utility function called shift\_geometry() which works by re-plotting Alaska, Hawaii, and Puerto Rico (but not other US territories)

```{r}  
us\_states\_shifted <- shift\_geometry(us\_states)  
  
ggplot(us\_states\_shifted) +  
 geom\_sf() +  
 theme\_void()  
```



* It has optional arguments:
  + preserve\_area which, when set to FALSE by default, resizes Hawaii and Alaska relative to the US’s area
  + position which sets the non-contiguous states and territory to below or outside the plot relative to the contiguous States

```{r}  
us\_states\_outside <- shift\_geometry(us\_states,  
 preserve\_area = TRUE,  
 position = "outside")  
  
ggplot(us\_states\_outside) +  
 geom\_sf() +  
 theme\_void()  
```



### Converting polygons to points

* Sometimes we may want to map features that are, by default, polygonal in nature - i.e. the relative location of cities on a Statewide map.
  + One technique we can use is the computation of a centroid for the polygons using st\_centroid()
* As we can see below, the polygons for the Cities add unnecessary clutter for our map, relative to what we are trying to portray.

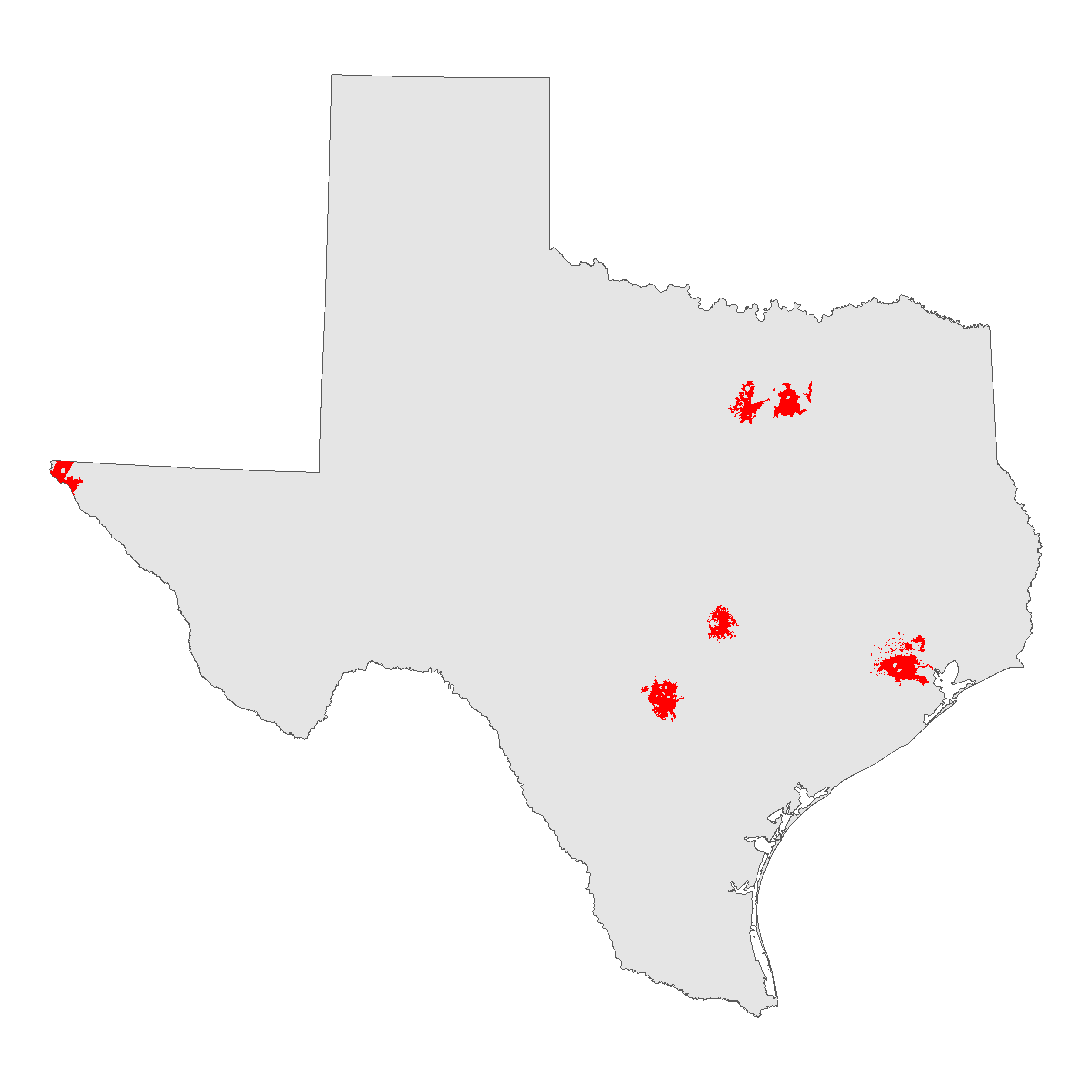
```{r initial\_texas\_cities\_map}  
tx\_places <- places("TX", cb = TRUE) %>%   
 filter(NAME %in% c("Dallas", "Fort Worth", "Houston",  
 "Austin", "San Antonio", "El Paso")) %>%   
 st\_transform(6580)  
```

Retrieving data for the year 2021

```{r initial\_texas\_cities\_map}  
tx\_outline <- states(cb = TRUE) %>%   
 filter(NAME == "Texas") %>%   
 st\_transform(6580)  
```

Retrieving data for the year 2021

```{r initial\_texas\_cities\_map}  
ggplot() +  
 geom\_sf(data = tx\_outline) +  
 geom\_sf(data = tx\_places, fill = "red", color = NA) +  
 theme\_void()  
```

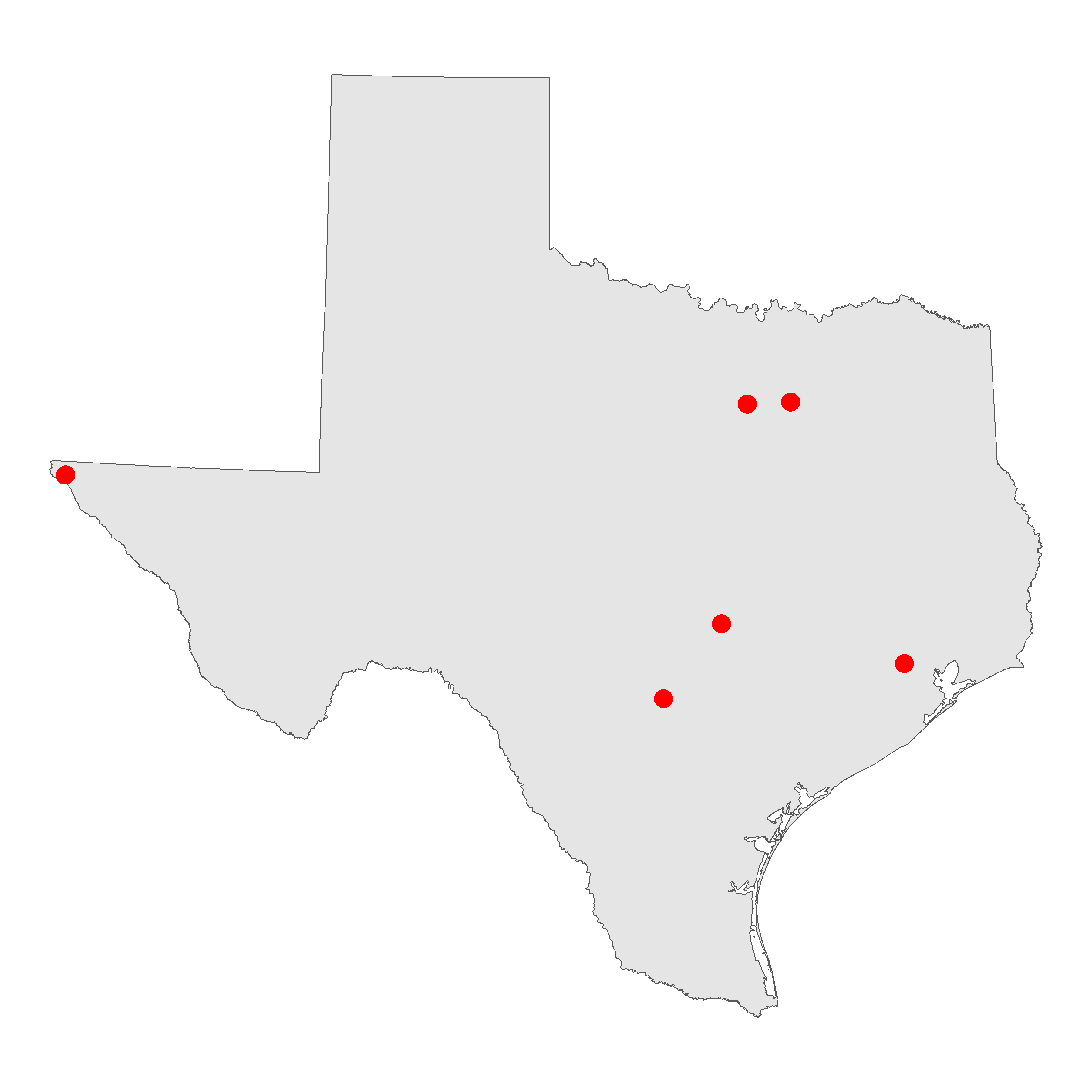


* By computing centroids, we have a much cleaner and more communicative map.

```{r}  
tx\_centroids <- st\_centroid(tx\_places)  
```

Warning: st\_centroid assumes attributes are constant over geometries

```{r}  
ggplot() +   
 geom\_sf(data = tx\_outline) +   
 geom\_sf(data = tx\_centroids, color = "red", size = 3) +   
 theme\_void()  
```



### Exploding multipolygon geometries to single parts

* The census returns areas as multipolygons because often, areas have detached elements, such as islands, that are still in the same census area.

```{r lee\_county\_florida}  
lee <- fl\_projected %>%   
 filter(NAME == "Lee")  
```

```{r}  
mapview(lee)  
```

* the county multipolygon is composed of four entities:
  + a mainland area
  + three disconnected island areas
* We can “explode” the multipolygon to extract specific features using st\_cast()

```{r}  
lee\_singlepart <- st\_cast(lee, "POLYGON")  
```

Warning in st\_cast.sf(lee, "POLYGON"): repeating attributes for all  
sub-geometries for which they may not be constant

```{r}  
lee\_singlepart  
```

Simple feature collection with 4 features and 12 fields  
Geometry type: POLYGON  
Dimension: XY  
Bounding box: xmin: 571477.3 ymin: 258767.6 xmax: 642721.2 ymax: 310583.5  
Projected CRS: NAD83(HARN) / Florida GDL Albers  
 STATEFP COUNTYFP COUNTYNS AFFGEOID GEOID NAME NAMELSAD STUSPS  
1 12 071 00295758 0500000US12071 12071 Lee Lee County FL  
1.1 12 071 00295758 0500000US12071 12071 Lee Lee County FL  
1.2 12 071 00295758 0500000US12071 12071 Lee Lee County FL  
1.3 12 071 00295758 0500000US12071 12071 Lee Lee County FL  
 STATE\_NAME LSAD ALAND AWATER geometry  
1 Florida 06 2022803065 1900583561 POLYGON ((580415.6 300219.1...  
1.1 Florida 06 2022803065 1900583561 POLYGON ((576540.7 289935.2...  
1.2 Florida 06 2022803065 1900583561 POLYGON ((572595.7 298880.5...  
1.3 Florida 06 2022803065 1900583561 POLYGON ((571477.3 310583, ...

* we can now interact with each element as a discrete polygon

```{r}  
sanibel\_island <- lee\_singlepart[2, ]  
mapview(sanibel\_island)  
```

## 5.6 Exercises

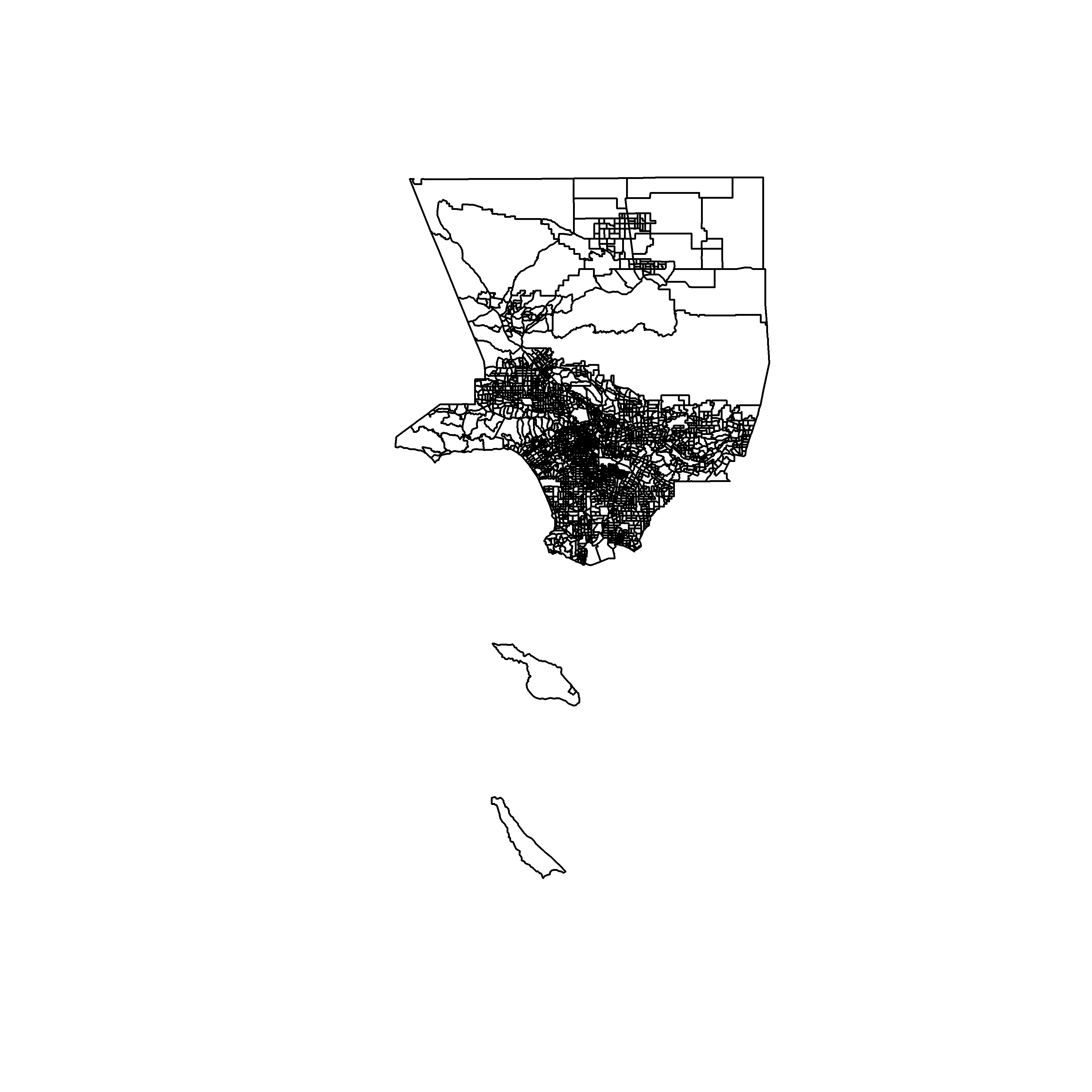
* get a specific state/county geography of your choice

```{r}  
los\_angeles\_county <- tracts("CA", "Los Angeles", cb = TRUE)  
```

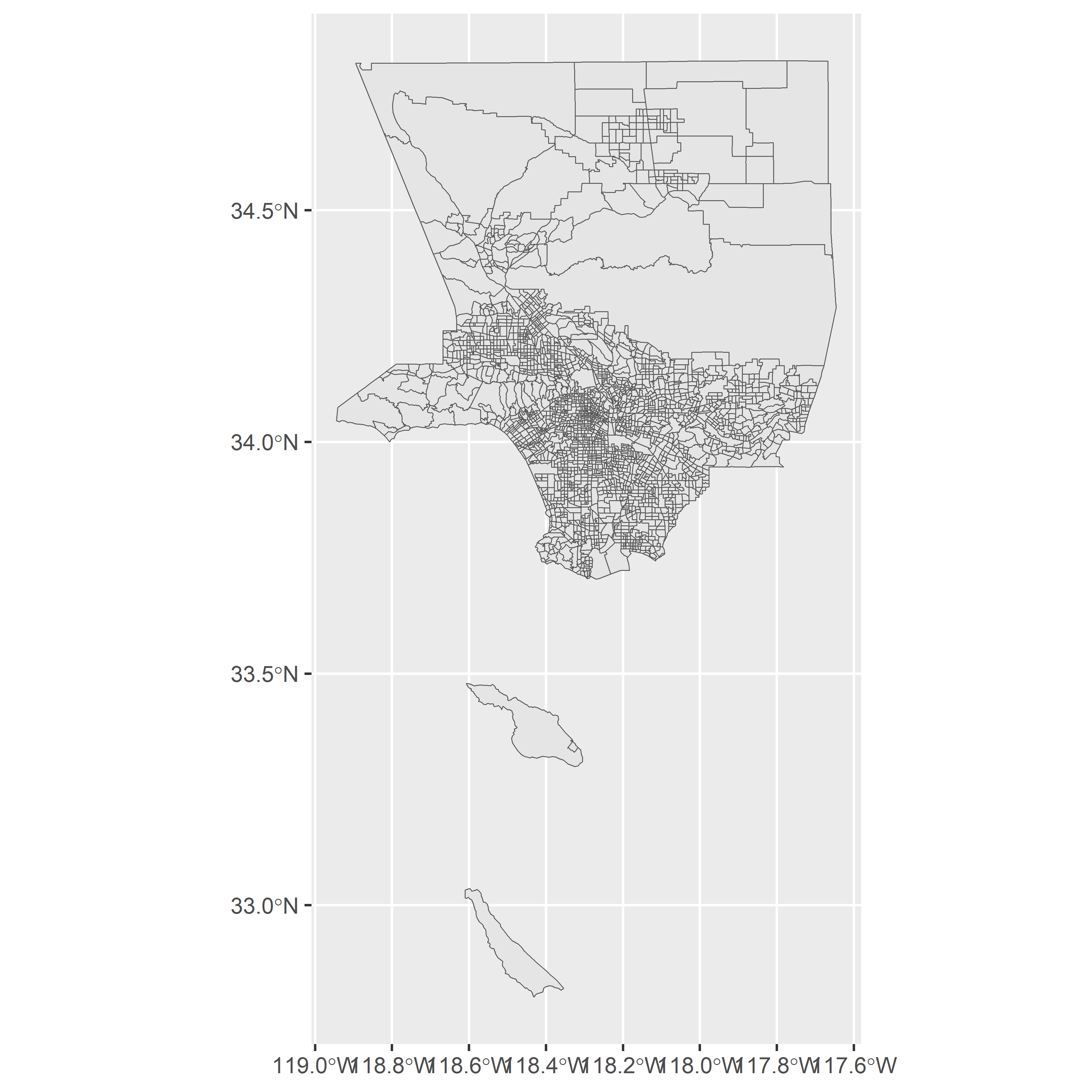
Retrieving data for the year 2021

* plot it using plot(), geom\_sf(), and mapview()

```{r plot}  
plot(los\_angeles\_county$geometry)  
```



```{r geom\_sf}  
ggplot() +   
 geom\_sf(data = los\_angeles\_county)  
```



```{r mapview}  
mapview(los\_angeles\_county)  
```

* use suggest\_crs() to identify an appropriate projected CRS, then transform with st\_transform()

```{r}  
crsuggest::suggest\_crs(los\_angeles\_county) %>% View() # doesn't recommend the LA-specific zone 5  
la\_county\_projected <- los\_angeles\_county %>% st\_transform(26945)  
  
ggplot() +  
 geom\_sf(data = la\_county\_projected) +  
 coord\_sf(crs = 26945, datum = 26945)  
```

