Demo Week 3

AUTHOR

Dillon Mahmoudi

Introduction

This section just loads our necessary libraries.

```
library(tidycensus) # We're using Kyle Walker's book.
library(tigris) # This is so we can erase water.
```

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

```
library(sf) # For shapefiles and geometries
```

```
Linking to GEOS 3.11.2, GDAL 3.7.2, PROJ 9.3.0; sf_use_s2() is TRUE
```

```
# census_api_key("YOUR KEY GOES HERE", install = TRUE)
options(tigris_class = "sf")
options(tigris_use_cache = TRUE)
```

Load Sample Data

Look at the different options that are used in the `get_acs` call.

```
baltimore_inc <- get_acs(geography = "tract",
                           variables = c("pop" = "B03002_001", # Total
                                         "pop_nhwhite" = "B03002_003", # NH White
                                         "pop_nhblack" = "B03002_004", # NH Black
                                         "pop_nhamind" = "B03002_005", # NH Am Ind
                                         "pop_nhasian" = "B03002_006", # NH Asian
                                         "pop_nhhwnpi" = "B03002_007", # NH Hawaiian/PI
                                         "pop_nhother" = "B03002_008", # One Other
                                         "pop_nhtwomr" = "B03002_009", # Two+
                                         "pop_hispltx" = "B03002_012", # Hispanic/Latinx
                                         "hu_total"  = "B25001_001", # Housing Units
                                         "hu_totocc" = "B25003_001", # Housing Units - Occ
                                         "hu_totown" = "B25003_002", # Housing Units - Owner Oc
                                         "hu_totrnt" = "B25003_003", # Housing Units - Renter (
                                         "mhhi" = "B19013_001", # Median Household Income
                                         "mhmv" = "B25077_001"), # Median Home Value - Occ


                           year = 2021,
                           year = 2021,
                           county = c(510,5,3,27), # Balt city, Balt county, AA, Howard
                           survey = "acs5",
                           state = c(24), # 24 is the FIPS code for Maryland
                           geometry = TRUE,
                           output = "wide")
```

```
Getting data from the 2017-2021 5-year ACS
```

Combine Race and Ethnicities

With these commands, we problematically, but conveniently, reduce the race+eth into 5 categories:

- NH White: `pop_nhwhite`
- NH Black: `pop_nhblack`
- Hispanic/Latino: `pop_hispltx`
- NH Asian: `pop_nhasianXE` (this is a new column that we're creating below)
- NH Multi/Other: `pop_nhotherXE` (this is a new column that we're creating below)

When creating the new columns, I put a capital "X" in the name to denote that we created it. The ending capital "E" is to remind us that it's an estimate.

```
# Computes the NH Asian Population
baltimore_inc$pop_nhasianXE <-
  baltimore_inc$pop_nhasianE + baltimore_inc$pop_nhhwnpiE

# Computes the NH "Other" Population
baltimore_inc$pop_nhotherXE <-
  baltimore_inc$pop_nhamindE + baltimore_inc$pop_nhotherE + baltimore_inc$pop_nhtwomrE
```

Transform

Let's first see what the "projection" is for the `baltimore_inc` file.

```
st_crs(baltimore_inc)
```

```
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]]
```

At the bottom of the above output (to `st_crs`) we see that it's in `4269` which is an unprojected coordinate system. We'll want to transform it into something like Web Mercator. We are then going to run `st_crs` again, to verify that our transformation (using `st_transform`) had the intended result.

```
baltimore_inc.3857 <- st_transform(baltimore_inc, crs=3857)
st_crs(baltimore_inc.3857)
```

```
Coordinate Reference System:
  User input: EPSG:3857 
  wkt:
PROJCRS["WGS 84 / Pseudo-Mercator",
    BASEGEOGCRS["WGS 84",
        ENSEMBLE["World Geodetic System 1984 ensemble",
            MEMBER["World Geodetic System 1984 (Transit)"],
            MEMBER["World Geodetic System 1984 (G730)"],
            MEMBER["World Geodetic System 1984 (G873)"],
            MEMBER["World Geodetic System 1984 (G1150)"],
            MEMBER["World Geodetic System 1984 (G1674)"],
            MEMBER["World Geodetic System 1984 (G1762)"],
            MEMBER["World Geodetic System 1984 (G2139)"],
            ELLIPSOID["WGS 84",6378137,298.257223563,
                LENGTHUNIT["metre",1]],
            ENSEMBLEACCURACY[2.0]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4326]],
    CONVERSION["Popular Visualisation Pseudo-Mercator",
        METHOD["Popular Visualisation Pseudo Mercator",
            ID["EPSG",1024]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["False easting",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["easting (X)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["northing (Y)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["Web mapping and visualisation."],
        AREA["World between 85.06°S and 85.06°N."],
        BBOX[-85.06,-180,85.06,180]],
    ID["EPSG",3857]]
```

Erase Water

Now we can use one of the `tigris` functions to “erase” (or “clip”) the water from our shapefile.

```
baltimore_inc.3857 <- erase_water(baltimore_inc.3857, area_threshold = 0.9)
```

```
Fetching area water data for your dataset's location...

Erasing water area...
If this is slow, try a larger area threshold value.
```

Save

Here we’re going to save the file into our `data` folder.

Since this qmd file is in the `src` folder:

- we have to go “up” (or “back”) a folder (using `../`)
- then into the `data` folder with `data/` (that ending slash informs the computer that it’s a folder)
- then provide the name of the file.

So the name of the file also include the path of the file which captures all of the above path information: `../data/bmore.geojson`

Also notice that we set `delete_dsn = TRUE` which first deletes the file `bmore.geojson` if it already exists before trying to save it (it’s like overwriting).

```
st_write(baltimore_inc.3857, "../data/bmore.geojson", delete_dsn = TRUE)
```

```
Deleting source `../data/bmore.geojson' using driver `GeoJSON'
Writing layer `bmore' to data source `../data/bmore.geojson' using driver `GeoJSON'
Writing 606 features with 34 fields and geometry type Unknown (any).
```