# Mapping and spatial analysis with ACS data in R

**SSDAN Workshop Series** 

Kyle Walker

**February 15, 2023** 

#### **About me**

- Associate Professor of Geography at TCU
- Spatial data science researcher and consultant
- Package developer: tidycensus, tigris, mapboxapi, crsuggest, idbr (R), pygris (Python)
- Book: Analyzing US Census Data: Methods, Maps and Models in R
  - Print release date February 16 (tomorrow!)
  - To support these workshops: <u>buy on Amazon</u> or <u>direct from CRC Press</u>



### **SSDAN** workshop series

- Last week (February 8): Working with the 2021 American Community Survey with R and tidycensus
- Today: Mapping and spatial analysis with ACS data in R
- Next week (February 22): Working with geographic data and making maps in Python

# Today's agenda

- Hour 1: Working with "spatial" American Community Survey data
- Hour 2: Making maps of ACS data in R
- Hour 3: Applications: segregation, diversity, and location intelligence

# Part 1: Working with "spatial" American Community Survey data

# **The American Community Survey**

- Annual survey of 3.5 million US households
- Covers topics not available in decennial US Census data (e.g. income, education, language, housing characteristics)
- Available as 1-year estimates (for geographies of population 65,000 and greater) and 5-year estimates (for geographies down to the block group)
  - o 2020 1-year data only available as experimental estimates
- Data delivered as estimates characterized by margins of error

### How to get ACS data

- <u>data.census.gov</u> is the main, revamped interactive data portal for browsing and downloading Census datasets, including the ACS
- <u>The US Census Application Programming Interface (API)</u> allows developers to access Census data resources programmatically

#### tidycensus

- R interface to the Decennial Census, American Community Survey, Population Estimates Program, and Public Use Microdata Series APIs
- Key features:
  - Wrangles Census data internally to return tidyverse-ready format (or traditional wide format if requested);
  - Automatically downloads and merges Census geometries to data for mapping (next week's workshop!);
  - Includes tools for handling margins of error in the ACS and working with survey weights in the ACS PUMS;
  - States and counties can be requested by name (no more looking up FIPS codes!)

#### R and RStudio

- R: programming language and software environment for data analysis (and wherever else your imagination can take you!)
- RStudio: integrated development environment (IDE) for R developed by Posit
- Posit Cloud: run RStudio with today's workshop pre-configured at <a href="https://posit.cloud/content/5377428">https://posit.cloud/content/5377428</a>

### **Getting started with tidycensus**

- To get started, install the packages you'll need for today's workshop
- If you are using the Posit Cloud environment, these packages are already installed for you

```
install.packages(c("tidycensus", "tidyverse"))
```

• Optional, to run advanced examples:

### **Optional: your Census API key**

- tidycensus (and the Census API) can be used without an API key, but you will be limited to 500 queries per day
- Power users: visit <a href="https://api.census.gov/data/key\_signup.html">https://api.census.gov/data/key\_signup.html</a> to request a key, then activate the key from the link in your email.
- Once activated, use the census\_api\_key() function to set your key as an environment variable

```
library(tidycensus)
census_api_key("YOUR KEY GOES HERE", install = TRUE)
```

# **Spatial Census data in tidycensus**

# **Spatial Census data: the old way**

Traditionally, getting "spatial" Census data requires:

- Fetching shapefiles from the Census website;
- Downloading a CSV of data, cleaning/formatting it;
- Loading geometries and data into your GIS of choice;
- Aligning key fields in your GIS and joining your data

# Spatial Census data with get\_acs()

- The get\_acs() function is your portal to access ACS data using tidycensus
- The two required arguments are geography and variables. The function defaults to the 2017-2021 5-year ACS
- The argument geometry = TRUE returns pre-joined geometry along with your ACS data!

```
library(tidycensus)

texas_income <- get_acs(
    geography = "county",
    variables = "B19013_001",
    state = "TX",
    year = 2021,
    geometry = TRUE
)

plot(texas_income["estimate"])</pre>
```

# Looking under the hood: simple features in R



- The sf package implements a *simple features data model* for vector spatial data in R
- Vector geometries: points, lines, and polygons stored in a list-column of a data frame

• Spatial data are returned with five data columns: GEOID, NAME, variable, estimate, and moe, along with a geometry column representing the shapes of locations

```
## Simple feature collection with 254 features and 5 fields
## Geometry type: MULTIPOLYGON
## Dimension:
## Bounding box: xmin: -106.6456 ymin: 25.83738 xmax: -93.50829 ymax: 36.5007
## Geodetic CRS: NAD83
## First 10 features:
      GEOID
                                 NAME
                                        variable estimate moe
## 1 48053
                 Burnet County, Texas B19013 001
                                                    65363 4694
## 2 48057
                Calhoun County, Texas B19013 001
                                                    61887 9517
## 3 48341
                 Moore County, Texas B19013 001
                                                    55543 3786
## 4 48185
                Grimes County, Texas B19013 001
                                                    59086 4414
## 5 48035
                 Bosque County, Texas B19013 001
                                                    59328 3702
                 Walker County, Texas B19013 001
## 6 48471
                                                    44104 1820
     48407 San Jacinto County, Texas B19013 001
                                                    46678 6190
               Angelina County, Texas B19013 001
## 8 48005
                                                    52377 3075
## 9 48443
                Terrell County, Texas B19013 001
                                                    47012 5245
                Sherman County, Texas B19013 001
                                                    55667 9007
## 10 48421
                            geometry
## 1 MULTIPOLYGON (((-98.45924 3...
## 2 MULTIPOLYGON (((-96.77168 2...
## 3 MULTIPOLYGON (((-102.163 36...
## 4 MULTIPOLYGON (((-96.18831 3...
## 5 MULTIPOLYGON (((-98.00068 3...
```

texas income

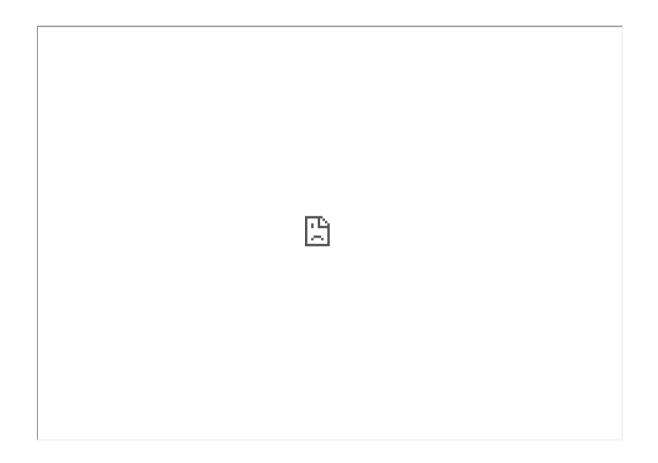
## 6 MULTIPOLYGON (((-95.86227 3...

# Interactive viewing with mapview()



• The **mapview** package allows for interactive viewing of spatial data in R

library(mapview)



# Understanding geography and variables in tidycensus

# **US Census Geography**



Source: <u>US Census Bureau</u>

# **Geography in tidycensus**

• Information on available geographies, and how to specify them, can be found in the tidycensus documentation



### **Searching for variables**

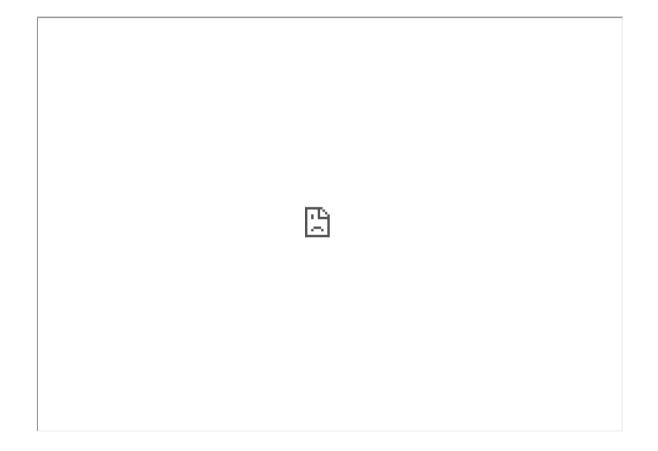
- To search for variables, use the load\_variables() function along with a year and dataset
- For the 2021 5-year ACS, use "acs5" for the Detailed Tables; "acs5/profile" for the Data Profile; "acs5/subject" for the Subject Tables; and "acs5/cprofile" for the Comparison Profile
- The View() function in RStudio allows for interactive browsing and filtering

```
vars <- load_variables(2021, "acs5")
View(vars)</pre>
```

### Small-area spatial demographic data

• Smaller areas like Census tracts or block groups are available with geography = "tract" or geography = "block group"; a county can optionally be specified to focus your query

```
king_income <- get_acs(
  geography = "tract",
  variables = "B19013_001",
  state = "WA",
  county = "King",
  geometry = TRUE
)
mapview(king_income, zcol = "estimate")</pre>
```



# Spatial data structure in tidycensus

# "Tidy" or long-form data

- The default data structure returned by **tidycensus** is "tidy" or long-form data, with variables by geography stacked by row
- For spatial data, this means that geometries will also be stacked, which is helpful for group-wise analysis and visualization

```
orange_race <- get_acs(
  geography = "tract",
  variables = c(
    Hispanic = "DP05_0071P",
    White = "DP05_0077P",
    Black = "DP05_0078P",
    Asian = "DP05_0080P"
  ),
  state = "CA",
  county = "Orange",
  geometry = TRUE
)</pre>
```

```
## Simple feature collection with 2456 features and 5 fields (with 4 geometries empty)
## Geometry type: MULTIPOLYGON
## Dimension:
                  XY
## Bounding box: xmin: -118.1154 ymin: 33.38779 xmax: -117.4133 ymax: 33.94764
## Geodetic CRS: NAD83
## First 10 features:
##
            GEOID
                                                            NAME variable estimate
     06059011402 Census Tract 114.02, Orange County, California Hispanic
                                                                              21.4
## 2 06059011402 Census Tract 114.02, Orange County, California
                                                                              51.7
                                                                    White
## 3 06059011402 Census Tract 114.02, Orange County, California
                                                                    Black
                                                                              1.3
## 4 06059011402 Census Tract 114.02, Orange County, California
                                                                    Asian
                                                                              19.6
## 5 06059087403 Census Tract 874.03, Orange County, California Hispanic
                                                                              87.4
## 6 06059087403 Census Tract 874.03, Orange County, California
                                                                    White
                                                                               8.1
## 7 06059087403 Census Tract 874.03, Orange County, California
                                                                    Black
                                                                               0.3
## 8 06059087403 Census Tract 874.03, Orange County, California
                                                                    Asian
                                                                               3.8
## 9 06059011716 Census Tract 117.16, Orange County, California Hispanic
                                                                              27.7
                                                                              38.6
## 10 06059011716 Census Tract 117.16, Orange County, California
                                                                    White
##
      moe
                                geometry
## 1 5.3 MULTIPOLYGON (((-117.9137 3...
## 2 8.1 MULTIPOLYGON (((-117.9137 3...
## 3 1.9 MULTIPOLYGON (((-117.9137 3...
## 4 9.2 MULTIPOLYGON (((-117.9137 3...
## 5 5.4 MULTIPOLYGON (((-117.9154 3...
## 6 3.6 MULTIPOLYGON (((-117.9154 3...
## 7 0.4 MULTIPOLYGON (((-117.9154 3...
## 8 3.3 MULTIPOLYGON (((-117.9154 3...
```

orange race

#### "Wide" data

- The argument output = "wide" spreads Census variables across the columns, returning one row per geographic unit and one column per variable
- This will be a more familiar data structure for traditional desktop GIS users

```
orange_race_wide <- get_acs(
    geography = "tract",
    variables = c(
        Hispanic = "DP05_0071P",
        White = "DP05_0077P",
        Black = "DP05_0078P",
        Asian = "DP05_0080P"
    ),
    state = "CA",
    county = "Orange",
    geometry = TRUE,
    output = "wide"
)</pre>
```

#### orange race wide ## Simple feature collection with 614 features and 10 fields (with 1 geometry empty) ## Geometry type: MULTIPOLYGON ## Dimension: XY ## Bounding box: xmin: -118.1154 ymin: 33.38779 xmax: -117.4133 ymax: 33.94764 ## Geodetic CRS: NAD83 ## First 10 features: ## GEOID NAME HispanicE 06059011402 Census Tract 114.02, Orange County, California 21.4 ## 2 06059087403 Census Tract 874.03, Orange County, California 87.4 06059011716 Census Tract 117.16, Orange County, California ## 3 27.7 ## 4 06059075504 Census Tract 755.04, Orange County, California 37.6 ## 5 06059063902 Census Tract 639.02, Orange County, California 25.7 ## 6 06059099222 Census Tract 992.22, Orange County, California 15.8 Census Tract 884.03, Orange County, California ## 7 06059088403 57.8 ## 8 06059110108 Census Tract 1101.08, Orange County, California 39.2 06059011403 Census Tract 114.03, Orange County, California 56.6 ## 10 06059087505 Census Tract 875.05, Orange County, California 68.9 ## HispanicM WhiteE WhiteM BlackE BlackM AsianE AsianM ## 1 5.3 51.7 8.1 1.3 1.9 19.6 9.2 ## 2 5.4 8.1 3.6 0.3 0.4 3.3 3.8 0.9 5.2 ## 3 7.2 38.6 9.5 0.5 30.4 ## 4 9.4 46.1 7.1 1.9 2.2 9.5 3.5 ## 5 7.1 56.6 8.7 1.7 1.3 10.4 4.9 ## 6 5.7 18.6 8.9 1.4 58.5 9.6 1.6 16.3 ## 7 9.5 5.9 1.1 0.8 24.2 7.9

3.5

12.4

6.2

## 8

15.7

40.4

11.4

3.4

## 9 6.6 29.8 5.6 0.5 0.6 7.7 2.3

**Working with Census geometry** 

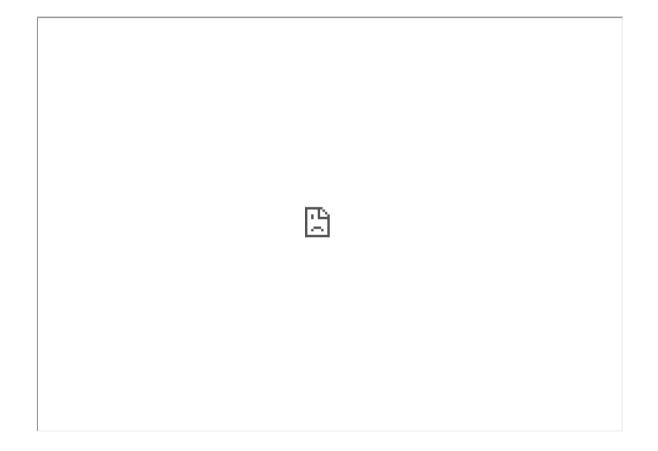
# Census geometry and the tigris R package



• tidycensus uses the **tigris** R package internally to acquire Census shapefiles

#### **Problem: interior water areas**

- Let's re-visit the King County income-by-Census tract map
- Areas like Mercer Island are obscured as Census tracts cover the entirety of Lake Washington
- erase water () in the tigris package offers a solution; it automates the removal of water areas from your shapes



#### Part 1 exercises

- 1. Use the load\_variables() function to find a variable that interests you that we haven't used yet.
- 2. Use get\_acs() to fetch spatial ACS data on that variable for a geography and location of your choice, then use mapview() to display your data interactively.

# Part 2: Mapping ACS data

# **Mapping in R**

- R has a robust set of tools for cartographic visualization that make it a suitable alternative to desktop GIS software in many instances
- Popular packages for cartography include **ggplot2**, **tmap**, and **mapsf**
- Today, we'll be focusing on ggplot2; see Chapter 6 of my book for similar examples using tmap

# ggplot2 and geom\_sf()

- ggplot2: R's most popular visualization package (over 105 million downloads!)
- ggplot2 graphics are defined by an aesthetic mapping and one or more geoms
- geom\_sf() is a special geom that interprets the geometry type of your spatial data and visualizes it accordingly
- As a result, we can make attractive maps using familiar ggplot2 syntax

# Mapping ACS data with ggplot2

# **Continuous choropleth**

- By default, ggplot2 will apply a continuous color palette to create **choropleth** maps
- Choropleth maps: the shading of a polygon / shape is mapped to a data attribute

```
library(tidyverse)

orange_hispanic <- filter(orange_race, variable == "Hispanic")

ggplot(orange_hispanic, aes(fill = estimate)) +
   geom_sf()</pre>
```

# **Continuous choropleth with styling**

- We can apply some styling to customize our choropleth maps
- Used here: a <u>viridis</u> color palette, which is built-in to ggplot2

```
ggplot(orange_hispanic, aes(fill = estimate)) +
  geom_sf() +
  theme_void() +
  scale_fill_viridis_c(option = "rocket") +
  labs(title = "Percent Hispanic by Census tract",
      subtitle = "Orange County, California",
      fill = "ACS estimate",
      caption = "2017-2021 ACS | tidycensus R package")
```

# **Classed choropleth**

• We can also create a binned choropleth; ggplot2 will identify "pretty" breaks, or custom breaks can be supplied

```
ggplot(orange_hispanic, aes(fill = estimate)) +
  geom_sf() +
  theme_void() +
  scale_fill_viridis_b(option = "rocket", n.breaks = 6) +
  labs(title = "Percent Hispanic by Census tract",
      subtitle = "Orange County, California",
      fill = "ACS estimate",
      caption = "2017-2021 ACS | tidycensus R package")
```

#### **Faceted choropleth maps**

• Spatial ACS data in tidy (long) format can be *faceted* by a grouping variable, allowing for comparative mapping

```
ggplot(orange_race, aes(fill = estimate)) +
  geom_sf(color = NA) +
  theme_void() +
  scale_fill_viridis_c(option = "rocket") +
  facet_wrap(~variable) +
  labs(title = "Race / ethnicity by Census tract",
      subtitle = "Orange County, California",
      fill = "ACS estimate (%)",
      caption = "2017-2021 ACS | tidycensus R package")
```

# Mapping count data

## Mapping count data

- At times, you'll want to show variations in *counts* rather than rates on your maps of ACS data
- Choropleth maps are poorly suited for count data
- Let's grab some count data for race / ethnicity and consider some alternatives

```
orange_race_counts <- get_acs(
  geography = "tract",
  variables = c(
    Hispanic = "DP05_0071",
    White = "DP05_0077",
    Black = "DP05_0078",
    Asian = "DP05_0080"
  ),
  state = "CA",
  county = "Orange",
  geometry = TRUE
)</pre>
```

# **Graduated symbol maps**

- Graduated symbol maps show difference on a map with the size of symbols (often circles)
- They are better for count data than choropleth maps as the shapes are directly comparable (unlike differentially-sized polygons)
- We'll need to convert our data to *centroids* to plot graduated symbols in ggplot2

```
library(sf)

orange_black <- filter(
  orange_race_counts,
  variable == "Black"
)

centroids <- st_centroid(orange_black)</pre>
```

#### **Graduated symbol maps**

- We'll first plot a base layer of Census tracts, then a layer of graduated symbols on top
- Use scale size area() to plot proportional symbols

# **Dot-density mapping**

- It can be difficult to show *heterogeneity* or *mixing* of different categories on maps
- Dot-density maps scatter dots proportionally to data size; dots can be colored to show mixing of categories
- Traditionally, dot-density maps are slow to make in R; tidycensus's as\_dot\_density() function addresses this

```
orange_race_dots <- as_dot_density(
  orange_race_counts,
  value = "estimate",
  values_per_dot = 200,
  group = "variable"
)</pre>
```

#### orange race dots ## Simple feature collection with 15199 features and 5 fields ## Geometry type: POINT ## Dimension: XY ## Bounding box: xmin: -118.1149 ymin: 33.39638 xmax: -117.4263 ymax: 33.94577 ## Geodetic CRS: NAD83 ## First 10 features: ## GEOID NAME variable estimate 06059086303 Census Tract 863.03, Orange County, California White 2329 ## 2 06059088301 Census Tract 883.01, Orange County, California 2519 Asian ## 3 06059099601 Census Tract 996.01, Orange County, California Hispanic 2188 ## 4 06059063906 Census Tract 639.06, Orange County, California White 2306 ## 5 06059062657 Census Tract 626.57, Orange County, California White 2802 ## 6 06059001202 Census Tract 12.02, Orange County, California Hispanic 3059 ## 7 06059075404 Census Tract 754.04, Orange County, California White 1793 ## 8 06059074103 Census Tract 741.03, Orange County, California Hispanic 5255 ## 9 06059088302 Census Tract 883.02, Orange County, California Hispanic 1648 ## 10 06059099239 Census Tract 992.39, Orange County, California White 2674 ## moe geometry 580 POINT (-117.8948 33.81615) ## 2 542 POINT (-117.9413 33.78907) 505 POINT (-118.0064 33.74013) ## 4 402 POINT (-117.9109 33.65461) ## 5 695 POINT (-117.8486 33.63091) ## 6 460 POINT (-117.9456 33.92972) 468 POINT (-117.8548 33.76288)

## 8 763 POINT (-117.8821 33.72321)

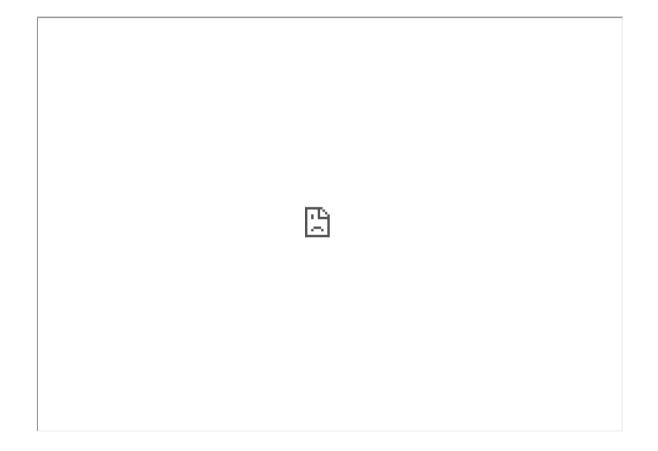
#### **Dot-density mapping**

- Like the graduated symbol map, we plot points over a base layer, but in this case with a much smaller size
- Use override.aes in guide legend() to plot visible colors in the legend

# Interactive mapping and national US maps

# Customizing interactive maps with mapview()

• mapview() accepts custom color palettes and labels, making it a suitable engine for interactive maps for presentations!

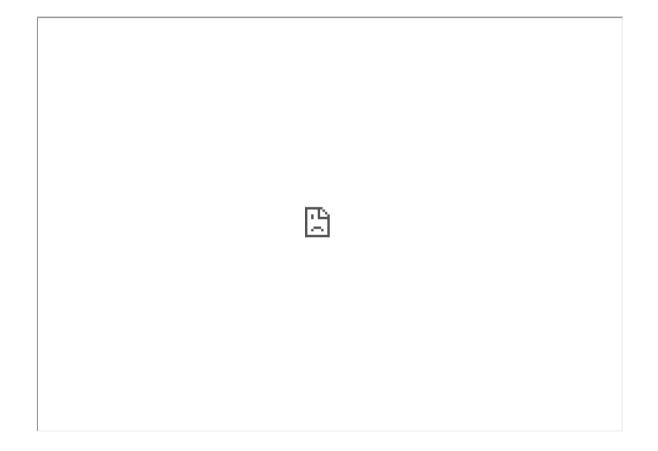


#### Linked interactive maps with mapview()

- In mapview, layers can be stacked with the + operator or swiped between with the | operator
- leafsync takes this one step further by creating side-by-side synced maps

## **Problem: national US maps of the US**

• These approaches to a common use-case -- mapping the entirety of the US -- doesn't work well. Let's take a look:



#### Solution: shifting and rescaling US geometry

- The shift\_geometry() function in the **tigris** package moves and optionally rescales Alaska, Hawaii, and Puerto Rico to help with national mapping
- shift\_geometry() includes options to preserve relative position and/or area if desired

```
library(tigris)
age_shifted <- shift_geometry(state_age)

ggplot(age_shifted, aes(fill = estimate)) +
   geom_sf() +
   scale_fill_viridis_c(option = "plasma") +
   theme_void() +
   labs(fill = "Median age \n2021 ACS")</pre>
```

# Interactivity with ggiraph

- **ggiraph**: Alternative approach for making **ggplot2** graphics interactive
- Includes \*\_interactive() versions of **ggplot2** geoms that can bring chart elements to life
- This includes geom\_sf\_interactive()

# ggiraph example

#### Part 2 exercises

Use load\_variables(2021, "acs5/profile") to find another percentage variable (ends in P) from the ACS Data Profile. Use that variable to try the following:

- Acquire spatial ACS data for your variable and make a customized choropleth map with **ggplot2**
- If time: try converting this map to an interactive map with **ggiraph** or **mapview**

# Part 3: Applications: segregation, diversity, and spatial analysis

# **Analyzing segregation and diversity**

# The dissimilarity index

- The dissimilarity index is one of the most common measures used in the social sciences to calculate segregation
- The statistic reflects the percentage of a group's population that would have to move for *local percentages* to match the overall percentages of the area
- The index *D* is calculated as:

$$D = rac{1}{2} \sum_{i=1}^N \left| rac{a_i}{A} - rac{b_i}{B} 
ight|$$

## Calculating a dissimilarity index

- The **segregation** R package implements the dissimilarity index along with several other commonly-used indices for segregation and diversity
- It works very well with tidycensus and other tidy (long-form) demographic datasets as inputs

```
library(segregation)

orange_race_counts %>%
  filter(variable %in% c("White", "Hispanic")) %>%
  dissimilarity(
    group = "variable",
    unit = "GEOID",
    weight = "estimate"
)
```

## stat est ## 1: D 0.5192184

#### **Group-wise segregation analysis**

- Segregation analysis can also be used within tidyverse-style groupwise data analytic workflows
- Here, we'll grab some data for four counties; generate a county column for grouping; then use group\_modify() to calculate groupwise dissimilarity indices (on the next slide)

```
la race counts %>%
  filter(variable %in% c("White", "Hispanic")) %>%
  group_by(county) %>%
  group modify(
    ~dissimilarity(
      data = .x,
      group = "variable",
      unit = "GEOID",
      weight = "estimate"
## # A tibble: 4 × 3
## # Groups: county [4]
    county
                          stat
                                  est
     <chr>
                          <chr> <dbl>
## 1 Los Angeles County
                                0.613
## 2 Orange County
                                0.519
## 3 Riverside County
                                0.404
## 4 San Bernardino County D
                                0.426
```

## **Analyzing diversity: the entropy index**

• As opposed to the dissimilarity index, the *entropy index* is a measure of evenness between multiple groups. It is computed as:

$$E = \sum_{r=1}^n Q_r ln rac{1}{Q_r}$$

• If the natural log is replaced with  $log_k$  where k is the number of groups, the index will be maximized at 1

### **Calculating the entropy index**

- The entropy index is implemented in the segregation package with **entropy**
- We'll need to use group modify () to group by Census tract, and combine the results into an output dataset

```
orange_entropy <- orange_race_counts %>%
  group_by(GEOID) %>%
  group_modify(~tibble(
    entropy = entropy(
        data = .x,
        group = "variable",
        weight = "estimate",
        base = 4
    )
)))
```

### Mapping the result

• Let's grab Census tracts directly using the **tigris** package to make a map

```
orange_tracts <- tracts("CA", "Orange", year = 2021, cb = TRUE)
orange_diversity_geo <- left_join(orange_tracts, orange_entropy, by = "GEOID")

ggplot(orange_diversity_geo, aes(fill = entropy)) +
    geom_sf() +
    scale_fill_viridis_c(option = "mako") +
    theme_void() +
    labs(fill = "Entropy index")</pre>
```

# **Spatial analysis with ACS data**

## **Spatial analysis with ACS data**

- Spatial analysis: the analysis of data in a way that takes into account, or directly focuses on, the data's spatial properties
- Spatial analysis is a huge topic; read <u>Chapter 7</u> and <u>Chapter 8</u> of my book for a wide variety of examples
- Today's examples: finding neighborhood hot-spots, analyzing demographics within proximity of a site

## **Spatial analysis: identifying hot-spots**

- The main package for analyzing spatial relationships in R is **spdep**
- We will identify *spatial neighbors*; build a *spatial weights matrix* from those neighbors; then calculate a hot-spot statistic, the Getis-Ord Local G

```
library(spdep)

neighbors <- poly2nb(
    orange_diversity_geo,
    queen = TRUE
)

weights <- nb2listw(neighbors)

G <- localG(
    orange_diversity_geo$entropy,
    listw = weights
)</pre>
```

#### **Hot-spot** analysis

• As the local G values are z-scores, we can identify critical thresholds for "hot" or "cold" spots and map our results accordingly

## Problem: how to get data for a custom area?

- Census data are organized based on Census hierarchies: state, county, etc.
- What if you want data for a custom area that doesn't align with these hierarchies?
- Possible solution: filter\_by

Demo: getting data for a custom shape

Demo: getting data around an address

Thank you, hope to see you next week!