Static maps

Andrew Ba Tran

Contents

This is from the fifth chapter of learn.r-journalism.com.

In this section we're going to go over the basics of spatial data, shapefiles, and various ways to map Census data.

Spatial data can be difficult to wrap your head around at first.

I'll describe it briefly as best I can before we move on to how journalists use it in their work process. But I hope you'll look up more details later on as you come to appreciate it more.

There are two underlying important pieces of information for spatial data:

- Coordinates of the object
- How the coordinates relate to a physical location on Earth
  - Also known as coordinate reference system or **CRS**

There are two types of **CRS**:

- Geographic
  - Uses three-dimensional model of the earth to define specific locations on the surface of the grid
  - longitude (East/West) and latitude (North/South)
- Projected
  - A translation of the three-dimensional grid onto a two-dimensional plane

There are so many map projections to choose from. The one you've probably been exposed to the most is Mercator (also known as WGS84) on Google Maps.

If you've worked with projections, then you've probably already seen this famous West Wing clip.

Raster versus Vector data

Spatial data with a defined CRS can either be vector or rster data.

- Vector
  - Based on points that can be connected to form lines and polygons
  - Located with in a coordinate reference system
  - Example: Road map
- Raster
  - Are values within a grid system
  - Example: Satellite imagery

{{% notice note %}} This class will focus on vector data and the **sf** package. An older package, **sp**, lets a user handle both vector and raster data. It also takes much more effort to get your system ready for it (*shakes fist at gdal*). The main differences between the **sp** and **sf** packages are how they store CRS information. While **sp** uses spatial subclasses, **sf** stores data in dataframes, allowing it to interact with **dplyr** methods we've learned so far. I encourage you to check out other spatial data analysis and modeling classes if you remain interested in this afterward. {{% /notice %}}

Shape files

R can handle importing different kinds of file formats for spatial data, including KML and geojson. We'll focus on shape files, which was created by ESRI in the '90s.

Though we refer to a shape file in the singular, it's actually a collection of at least three basic files:

- .shp - lists shape and vertices
- .shx - has index with offsets
- .dbf - relationship file between geometry and attributes (data)

All files must be present in the directory and named the same (except for the file extension) to import correctly.

The plan

We'll walk through several methods for dealing with spatial data, each time improving on the style a little bit.

1. Map blank shapefile after downloading
2. Join Census data to blank shapefile and map
3. Use R package **Tigris** to download shape file
4. Use R package **censusapi** to download census data and join to new shape file
5. Use **tidycensus** to download Census data and the shape file all at once

Let's use the **sf** package in conjunction with **ggplot2** to visualize the data.

{{% notice important %}} There are performance issues when creating maps with the **sf** package **if you're using a Mac**. To fix, download and intall XQuartz. Restart and then run these commands: options(device = "X11") and then X11.options(type = "cairo") {{% /notice %}}

Mapping a simple shape file

We'll start by reading in a shapefile of state boundaries from the Census.

```
# If you haven't installed ggplot2 or sf yet, uncomment and run the lines below
#install.packages("ggplot2")
#install.packages("sf")

library(ggplot2)
library(sf)

# If you're using a Mac, uncomment and run the lines below
#options(device = "X11")
#X11.options(type = "cairo")

fifty_location <- "data/cb_2017_us_state_20m/cb_2017_us_state_20m.shp"
fifty_states <- st_read(fifty_location)
```

```
## Reading layer `cb_2017_us_state_20m' from data source `/Users/andrewtran/Projects/r-journalism/learn
## Simple feature collection with 52 features and 9 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -179.1743 ymin: 17.91377 xmax: 179.7739 ymax: 71.35256
## epsg (SRID):    4269
## proj4string:    +proj=longlat +datum=NAD83 +no_defs
```

Filter

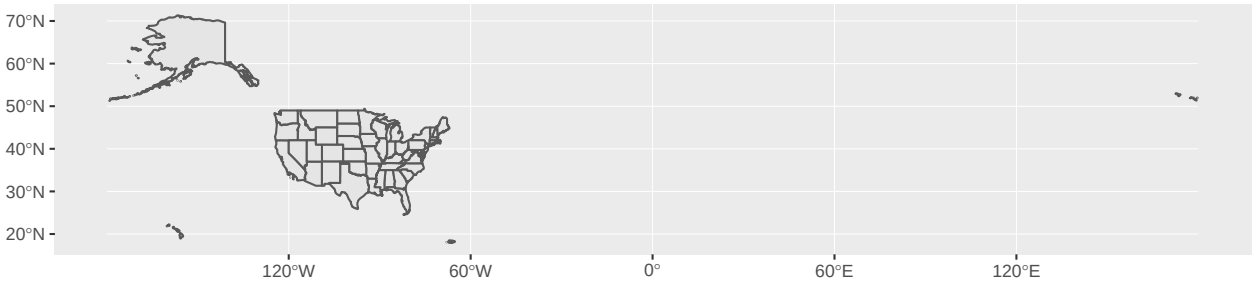
	STATEFP	STATENS	AFFGEOID	GEOID	STUSPS	NAME	LSAD	ALAND	AWATER	g
1	02	01785533	0400000US02	02	AK	Alaska	00	1.478588e+12	277723861311	
2	06	01779778	0400000US06	06	CA	California	00	4.034832e+11	20484637928	
3	08	01779779	0400000US08	08	CO	Colorado	00	2.684260e+11	1178495763	
4	11	01702382	0400000US11	11	DC	District of Columbia	00	1.583516e+08	18675956	
5	16	01779783	0400000US16	16	ID	Idaho	00	2.140482e+11	2393355752	
6	17	01779784	0400000US17	17	IL	Illinois	00	1.437841e+11	6211277447	
7	19	01779785	0400000US19	19	IA	Iowa	00	1.446642e+11	1081293682	
8	21	01779786	0400000US21	21	KY	Kentucky	00	1.022661e+11	2388731561	
9	22	01629543	0400000US22	22	LA	Louisiana	00	1.119048e+11	23746413153	
10	24	01714934	0400000US24	24	MD	Maryland	00	2.515070e+10	6980371026	
11	27	00662849	0400000US27	27	MN	Minnesota	00	2.062292e+11	18944967530	
12	29	01779791	0400000US29	29	MO	Missouri	00	1.780520e+11	2488190402	

Figure 1:

```
View(fifty_states)
```

We pointed to the shapefile and used the `st_read()` function to import it.

```
ggplot(fifty_states) + geom_sf()
```



Well, that's interesting. We have the boundaries of each state, including Hawaii and Alaska.

And **ggplot2** is doing its best to fit everything on one image. Which is taxing on the system.

Also, there are no colors because we don't have any data to fill with.

Let's pull in population data from CensusReporter.org

```
# If you don't have readr installed yet, uncomment and run the line below
#install.packages("readr")

library(readr)
populations <- read_csv("data/acs2016_1yr_B02001_04000US55.csv")
```

```
View(populations)
```

###. Join data to blank shapefile and map

We have a shapefile and a data set of populations. They're both data frames so should be easy to join. State names are where the data sets can join on. The column names for each data frame is different for state names, but we can account for that easily.

	geoid	name	B02001001	B02001001, Error	B02001002	B02001002, Error	B02001003	B02001003, Error
1	01000US	United States	323127515	0	234644039	111971	40893369	64
2	04000US01	Alabama	4863300	0	3316384	6545	1301102	7
3	04000US02	Alaska	741894	0	477895	3267	23753	1
4	04000US04	Arizona	6931071	0	5254944	21891	299674	6
5	04000US05	Arkansas	2988248	0	2290066	5262	464516	4
6	04000US06	California	39250017	0	23420234	62421	2265280	14
7	04000US08	Colorado	5540545	0	4654921	13381	234142	5
8	04000US09	Connecticut	3576452	0	2741892	12294	378932	7
9	04000US10	Delaware	952065	0	659091	5072	209911	4
10	04000US11	District of Columbia	681170	0	277268	3530	320554	2
11	04000US12	Florida	20612439	0	15574165	29122	3310428	22
12	04000US13	Georgia	10310371	0	6054861	18041	3254495	14

Figure 2:

```
ncol(fifty_states)
```

```
## [1] 10
```

```
library(dplyr)

fifty_states <- left_join(fifty_states, populations,
                          by=c("NAME"="name"))
```

```
ncol(fifty_states)
```

```
## [1] 31
```

Excellent. We went from 10 variables in **fifty_states** to 31.

There are a lot of variable names in this data frame. Check them out.

```
colnames(fifty_states)
```

```
##  [1] "STATEFP"          "STATENS"          "AFFGEOID"
##  [4] "GEOID"            "STUSPS"           "NAME"
##  [7] "LSAD"             "ALAND"            "AWATER"
## [10] "geoid"            "B02001001"        "B02001001, Error"
## [13] "B02001002"        "B02001002, Error" "B02001003"
## [16] "B02001003, Error" "B02001004"        "B02001004, Error"
## [19] "B02001005"        "B02001005, Error" "B02001006"
## [22] "B02001006, Error" "B02001007"        "B02001007, Error"
## [25] "B02001008"        "B02001008, Error" "B02001009"
## [28] "B02001009, Error" "B02001010"        "B02001010, Error"
## [31] "geometry"
```

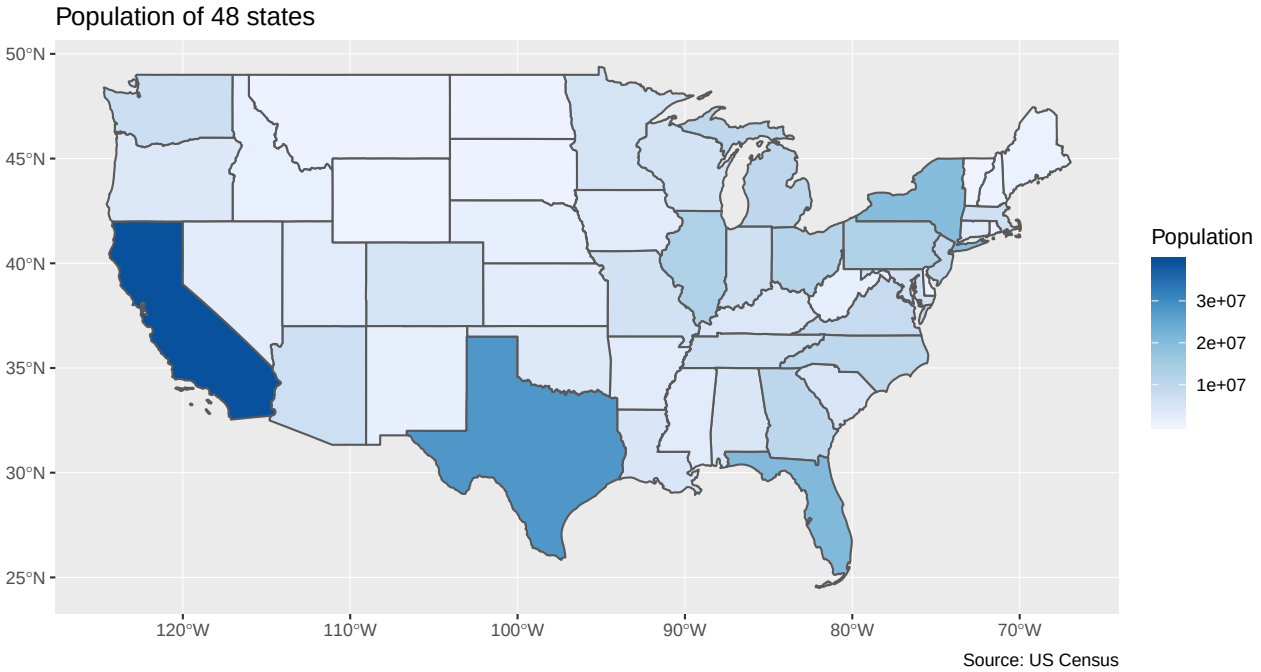
Alright, this is good to go over now.

- **STATEFP** is the state fips code.
  - That stands for the Federal Information Processing Standard. It's a standardized way to identify states, counties, census tracts, etc.
- **GEOID** is also part of the fips code.
  - In this instance it's only two digits wide.

  - The more specific you get into the Census boundaries, the longer the number gets.
- **B02001001**, **B02001002**, etc.
  - This is reference to a Census table of information.
  - For example, **B02001001** is total population for that polygon of data in that row
  - When you export data from the Census, the variables get translated to this sort of format
  - You'll have to remember when you download it or look it up.
- **B02001001, Error**
  - Margin of error included because these are just estimates, after all
- **geometry**
  - This is the CRS data

Let's map it with `geom_sf()` and fill it with the population variable **B02001001**. And we'll filter out Hawaii and Alaska for now because it'll slow things down if we don't. Sorry! We'll bring them back in later, I promise.

```
forty_eight <- fifty_states %>%
  filter(NAME!="Hawaii" & NAME!="Alaska" & NAME!="Puerto Rico")


ggplot(forty_eight) +
  geom_sf(aes(fill=B02001001)) +
  scale_fill_distiller(direction=1, name="Population") +
  labs(title="Population of 48 states", caption="Source: US Census")
```



Not bad. Very basic. Notice that the x and y axis are latitude and longitude.

So we've gone over how to bring in shape files and data locally, join them, and how to map it.

There's a more efficient way of dealing with shape files if you know what you're looking for.

Downloading shape files directly into R

Let's use the **tigris** package, which lets us download Census shapefiles directly into R without having to unzip and point to directories, etc. Here's a pretty thorough introduction from the package creator, Kyle

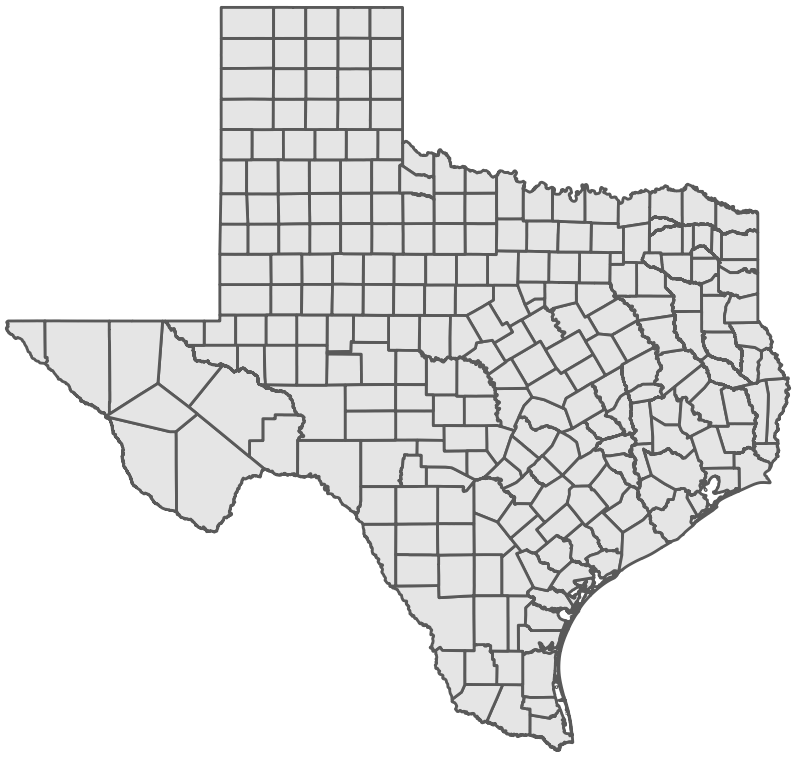
Walker.

Shapefiles can be downloaded simply by referring to them as a function such as

- `tracts()`
- `counties()`
- `school_districts()`
- `roads()`

First, let's make sure the shapefiles download as **sf** files (because it can also handle **sp** versions, as well)

```
# If you don't have tigris installed yet, uncomment the line below and run
#install.packages("tigris")

library(tigris)

# set sf option

options(tigris_class = "sf")

tx <- counties("TX", cb=T)

#If cb is set to TRUE, download a generalized (1:500k) counties file. Defaults to FALSE (the most detai

# Excluding Non-Continguous states (sorry!)

ggplot(tx) +
  geom_sf() +
  theme_void() +
  theme(panel.grid.major = element_line(colour = 'transparent')) +
  labs(title="Texas counties")
```



Great. Notice how we used a couple of new lines to eliminate the axes and the grids and backgrounds?

Looking like a real map. We just need to add some data.

Downloading Census data into R via API

Instead of downloading data from the horrible-to-navigate Census FactFinder or pleasant-to-navigate Census-Reporter.org we can pull the code with the **censusapi** package from Hannah Recht, of Bloomberg.

First, sign up for a census key.

```
# Add key to .Renviron
Sys.setenv(CENSUS_KEY="YOURKEYHERE")
# Reload .Renviron
readRenviron("~/.Renviron")
# Check to see that the expected key is output in your R console
Sys.getenv("CENSUS_KEY")
```

```
# If you don't have censusapi installed yet, uncomment the line below and run
#install.packages("censusapi")

library(censusapi)
```

Check out the dozens of data sets you have access to now.

```
apis <- listCensusApis()
View(apis)
```

We won't get too deep into the usage of **censusapi**, though I recommend the excellent documentation later.

title	name	vintage	url	isTimeseries	temporal	description	modified
1990 Decennial Census of Population and Housing - S...	sf3	1990	https://api.census.gov/data/1990/sf3	NA	1990/1990	The census of population and housing, taken by the ...	2012-06-06
Vintage 2013 Population Estimates: US, State, and PR ...	pep/natstprc	2013	https://api.census.gov/data/2013/pep/natstprc	NA	April 1, 2010 - Current	Annual Population Estimates, Estimated Components ...	2014-03-14
Vintage 2013 Population Estimates: US, State, and PR ...	pep/natstprc18	2013	https://api.census.gov/data/2013/pep/natstprc18	NA	April 1, 2010 - Current	Estimates of the Total Resident Population and Reside...	2014-03-14
Vintage 2013 Population Estimates: Puerto Rico Muni...	pep/prm	2013	https://api.census.gov/data/2013/pep/prm	NA	April 1, 2010 - Current	Annual Resident Population Estimates for Puerto Rico ...	2014-03-28
Vintage 2013 Population Estimates: County Total Pop...	pep/cty	2013	https://api.census.gov/data/2013/pep/cty	NA	April 1, 2010 - Current	Annual Resident Population Estimates, Estimated Com...	2014-04-01
2013 American Community Survey - Data Profiles: 1-Y...	acs1/profile	2013	https://api.census.gov/data/2013/acs1/profile	NA	2013/2013	The American Community Survey (ACS) is a nationwid...	2014-08-11
2013 American Community Survey - Summarized Dat...	acs1	2013	https://api.census.gov/data/2013/acs1	NA	2013/2013	The American Community Survey (ACS) is a nationwid...	2014-08-14
2011-2013 American Community Survey - Summarize...	acs3	2013	https://api.census.gov/data/2013/acs3	NA	2011/2013	The American Community Survey (ACS) is a nationwid...	2014-08-14
Vintage 2013 Population Estimates: Subcounty Popula...	pep/subcty	2013	https://api.census.gov/data/2013/pep/subcty	NA	April 1, 2010 - Current	Subcounty Resident Population Estimates: April 1, 20...	2014-08-20
2011-2013 American Community Survey - Data Profile...	acs3/profile	2013	https://api.census.gov/data/2013/acs3/profile	NA	2011/2013	The American Community Survey (ACS) is a nationwid...	2014-09-24
2012 Public Elementary-Secondary Education Finance:...	pubschlfin	2012	https://api.census.gov/data/2012/pubschlfin	NA	2012/2012	The survey covers all public school systems that provi...	2014-09-30
2011 American Community Survey 1-Year Profiles for ...	acs1/cd113	2011	https://api.census.gov/data/2011/acs1/cd113	NA	2011/2011	The American Community Survey (ACS) is a nationwid...	2014-10-06
Vintage 2013 Population Estimates: County Populatio...	pep/cochar5	2013	https://api.census.gov/data/2013/pep/cochar5	NA	April 1, 2010 - Current	Annual County Resident Population Estimates for 5 Ra...	2014-10-22
Vintage 2013 Population Estimates: County Populatio...	pep/cochar6	2013	https://api.census.gov/data/2013/pep/cochar6	NA	April 1, 2010 - Current	Annual County Resident Population Estimates for 6 Ra...	2014-10-22
Vintage 2013 Population Estimates: National Monthly ...	pep/monthlynatchar5	2013	https://api.census.gov/data/2013/pep/monthlynatch...	NA	April 1, 2010 - Current	Monthly Population Estimates by Universe, Age, Sex, ...	2014-10-22
Vintage 2013 Population Estimates: National Monthly ...	pep/monthlynatchar6	2013	https://api.census.gov/data/2013/pep/monthlynatch...	NA	April 1, 2010 - Current	Monthly Population Estimates by Universe, Age, Sex, ...	2014-10-22
Vintage 2013 Population Estimates: Puerto Rico Com...	pep/prcagesex	2013	https://api.census.gov/data/2013/pep/prcagesex	NA	April 1, 2010 - Current	Annual Estimates of the Resident Population by Single...	2014-10-22
Vintage 2013 Population Estimates: Puerto Rico Muni...	pep/prmagesex	2013	https://api.census.gov/data/2013/pep/prmagesex	NA	April 1, 2010 - Current	Annual Estimates of the Resident Population by Five-Y...	2014-10-22
Vintage 2013 Population Estimates: State Population ...	pep/stchar5	2013	https://api.census.gov/data/2013/pep/stchar5	NA	April 1, 2010 - Current	Annual State Resident Population Estimates for 5 Race...	2014-10-22
Vintage 2013 Population Estimates: State Population ...	pep/stchar6	2013	https://api.census.gov/data/2013/pep/stchar6	NA	April 1, 2010 - Current	Annual State Resident Population Estimates for 6 Race...	2014-10-22
Vintage 2014 Population Estimates: US, State, and PR ...	pep/natstprc	2014	https://api.census.gov/data/2014/pep/natstprc	NA	April 1, 2010 - Current	Annual Population Estimates, Estimated Components ...	2014-12-15
Vintage 2014 Population Estimates: US, State, and PR ...	pep/natstprc18	2014	https://api.census.gov/data/2014/pep/natstprc18	NA	April 1, 2010 - Current	Annual Population Estimates, Estimated Components ...	2014-12-15
Vintage 2014 Population Estimates: County Total Pop...	pep/cty	2014	https://api.census.gov/data/2014/pep/cty	NA	April 1, 2010 - Current	Annual Resident Population Estimates, Estimated Com...	2015-03-12
Vintage 2014 Population Estimates: Puerto Rico Muni...	pep/prm	2014	https://api.census.gov/data/2014/pep/prm	NA	April 1, 2010 - Current	Annual Resident Population Estimates for Puerto Rico ...	2015-03-12
Time Series Longitudinal Employer-Household Dynam...	timeseries/qwi/rh	NA	https://api.census.gov/data/timeseries/qwi/rh	TRUE	Time Series	The Quarterly Workforce Indicators (QWI) are a set of ...	2015-04-16
Time Series Longitudinal Employer-Household Dynam...	timeseries/qwi/sa	NA	https://api.census.gov/data/timeseries/qwi/sa	TRUE	Time Series	The Quarterly Workforce Indicators (QWI) are a set of ...	2015-04-16
Time Series Longitudinal Employer-Household Dynam...	timeseries/qwi/se	NA	https://api.census.gov/data/timeseries/qwi/se	TRUE	Time Series	The Quarterly Workforce Indicators (QWI) are a set of ...	2015-04-16
Vintage 2014 Population Estimates: Subcounty Popula...	pep/subcty	2014	https://api.census.gov/data/2014/pep/subcty	NA	April 1, 2010 - Current	Subcounty Resident Population Estimates // Source: U...	2015-05-11
Vintage 2014 Population Estimates: County Populatio...	pep/cochar5	2014	https://api.census.gov/data/2014/pep/cochar5	NA	April 1, 2010 - Current	Annual County Resident Population Estimates for 5 Ra...	2015-06-09
Vintage 2014 Population Estimates: County Populatio...	pep/cochar6	2014	https://api.census.gov/data/2014/pep/cochar6	NA	April 1, 2010 - Current	Annual County Resident Population Estimates for 6 Ra...	2015-06-09
Vintage 2014 Population Estimates: National Monthly ...	pep/monthlynatchar5	2014	https://api.census.gov/data/2014/pep/monthlynatch...	NA	April 1, 2010 - Current	Monthly Population Estimates by Universe, Age, Sex, ...	2015-06-09
Vintage 2014 Population Estimates: National Monthly ...	pep/monthlynatchar6	2014	https://api.census.gov/data/2014/pep/monthlynatch...	NA	April 1, 2010 - Current	Monthly Population Estimates by Universe, Age, Sex, ...	2015-06-09
Vintage 2014 Population Estimates: Puerto Rico Com...	pep/prcagesex	2014	https://api.census.gov/data/2014/pep/prcagesex	NA	April 1, 2010 - Current	Annual Estimates of the Resident Population by Single...	2015-06-09

Figure 3:

We'll focus on using the `getCensus()` function form the package. It makes an API call and returns a data frame of results.

These are the arguments you'll need to pass it:

- `name` - the name of the Census data set, like "acs5" or "timeseries/bds/firms"
- `vintage` - the year of the data set
- `vars` - one or more variables to access (remember *B02001001* from above?)
- `region` - the geography level of data, like county or tracts or state

Real quick, let's use `listCensusMetadata()` to see what tables might be available from the ACS Census survey.

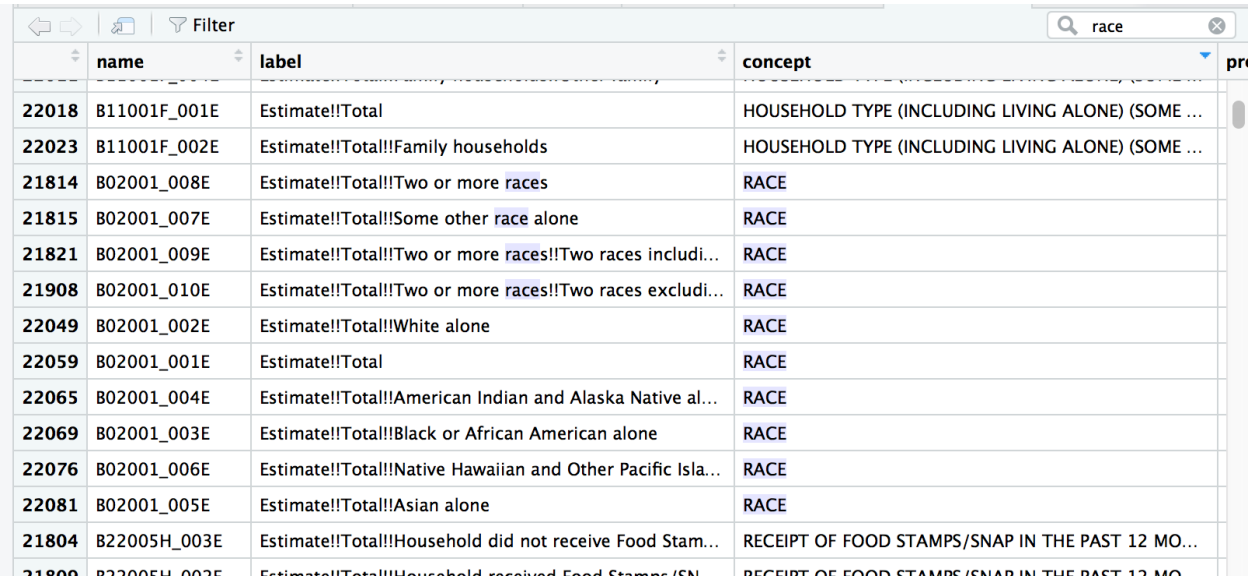
```
acs_vars <- listCensusMetadata(name="acs/acs5", type="variables", vintage=2016)

View(acs_vars)
```

It takes a couples to download the list of this data set (23,000 rows!) but once you get it, you can explore it to see what sort of data you might like to download. You can also refer to the Census for some guidance.

We'll pull median income: *B21004_001E*

```
tx_income <- getCensus(name = "acs/acs5", vintage = 2016,
    vars = c("NAME", "B19013_001E", "B19013_001M"),
    region = "county:*", regionin = "state:48")
head(tx_income)
```

```
##   state county                    NAME B19013_001E B19013_001M
## 1    48    001  Anderson County, Texas       42146        2539
## 2    48    003   Andrews County, Texas       70121        7053
## 3    48    005  Angelina County, Texas       44185        2107
## 4    48    007   Aransas County, Texas       44851        4261
## 5    48    009    Archer County, Texas       62407        5368
```



Filter | race

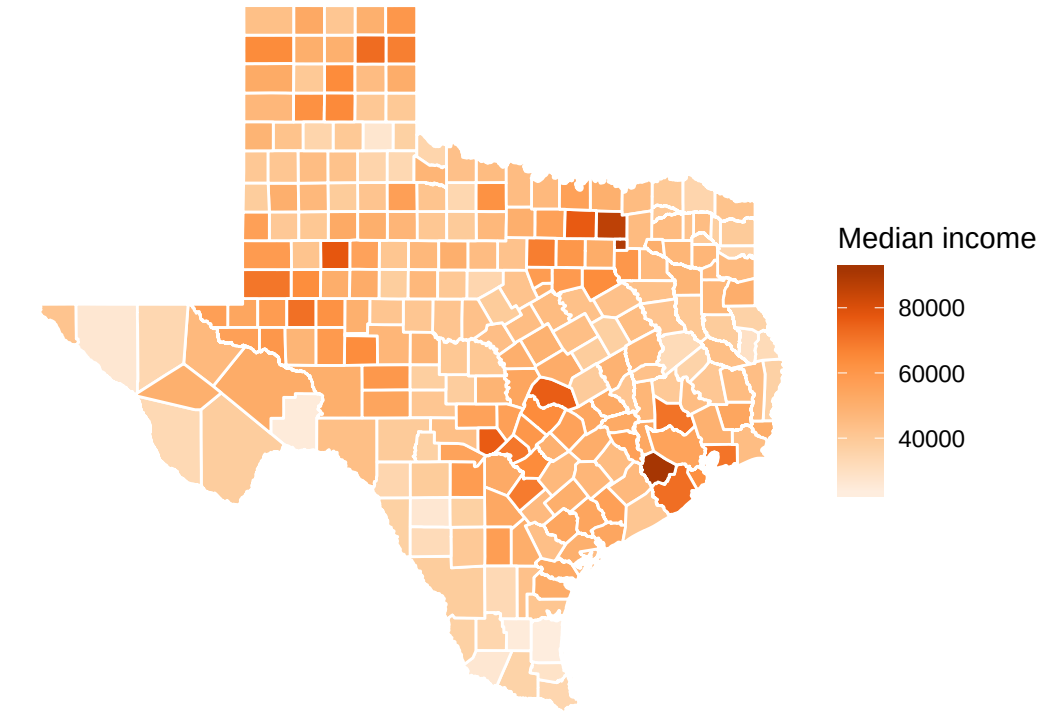
	name	label	concept
22018	B11001F_001E	Estimate!!Total	HOUSEHOLD TYPE (INCLUDING LIVING ALONE) (SOME ...
22023	B11001F_002E	Estimate!!Total!!Family households	HOUSEHOLD TYPE (INCLUDING LIVING ALONE) (SOME ...
21814	B02001_008E	Estimate!!Total!!Two or more races	RACE
21815	B02001_007E	Estimate!!Total!!Some other race alone	RACE
21821	B02001_009E	Estimate!!Total!!Two or more races!!Two races includi...	RACE
21908	B02001_010E	Estimate!!Total!!Two or more races!!Two races excludi...	RACE
22049	B02001_002E	Estimate!!Total!!White alone	RACE
22059	B02001_001E	Estimate!!Total	RACE
22065	B02001_004E	Estimate!!Total!!American Indian and Alaska Native al...	RACE
22069	B02001_003E	Estimate!!Total!!Black or African American alone	RACE
22076	B02001_006E	Estimate!!Total!!Native Hawaiian and Other Pacific Isla...	RACE
22081	B02001_005E	Estimate!!Total!!Asian alone	RACE
21804	B22005H_003E	Estimate!!Total!!Household did not receive Food Stam...	RECEIPT OF FOOD STAMPS/SNAP IN THE PAST 12 MO...

Figure 4:

```
## 6    48    011 Armstrong County, Texas       65000        9415
```

Alright, time to join it to our **tx** spatial data frame and map it.

```
# Can't join by NAME because tx_income data frame has "County, Texas" at the end
# We could gsub out the string but we'll join on where there's already a consistent variable, even thou

tx4ever <- left_join(tx, tx_income, by=c("COUNTYFP"="county"))


ggplot(tx4ever) +
  geom_sf(aes(fill=B19013_001E), color="white") +
  theme_void() +
  theme(panel.grid.major = element_line(colour = 'transparent')) +
  scale_fill_distiller(palette="Oranges", direction=1, name="Median income") +
  labs(title="2016 Median income in Texas counties", caption="Source: US Census/ACS5 2016")
```



Download Census data and shapefiles together

The most recent package dealing with Census data is **tidycensus** and it brings together what we've done above– the data and the geography. It's also created by Kyle Walker.

You can use it to pull data only like with **censusapi** or you can use it to pull shape files only, like with **tigris**.

But with **tidycensus**, you can download the shapefiles with the data you want already attached. No joins necessary.

I won't get into the particulars of looking up geography types and Census variables.

Let's get right into mapping. We'll calculate unemployment percents by Census tract in Jersey City. It'll involve wrangling some data. But querying the data with `get_acs()` will be easy and so will getting the shape file by simply passing it `geometry=T`.

```
# if you don't have tidycensus installed yet, uncomment and run the line below

#install.packages("tidycensus")
library(tidycensus)

# Pass it the census key you set up before
```

```
census_api_key("YOUR API KEY GOES HERE")
```

```
## To install your API key for use in future sessions, run this function with `install = TRUE`.
```

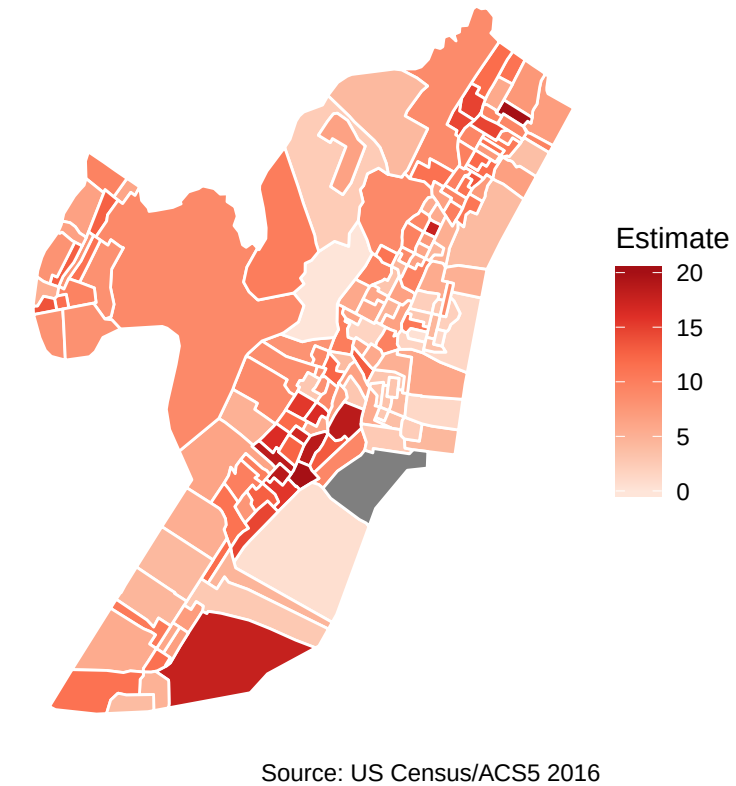
```
jobs <- c(labor_force = "B23025_005E",
              unemployed = "B23025_002E")
```

```
jersey <- get_acs(geography="tract", year=2016, variables= jobs, county = "Hudson", state="NJ", geometry

head(jersey)
```

Time for some math. Can you follow what's happening in the code based on what you've learned in previous chapters?

We can string the **dplyr** wrangling and **ggplot2** code together. Just watch and look out for the transition from %>% to +.

```
library(tidyr)

jersey %>%
  mutate(variable=case_when(
    variable=="B23025_005" ~ "Unemployed",
    variable=="B23025_002" ~ "Workforce")) %>%
  select(-moe) %>%
  spread(variable, estimate) %>%
  mutate(percent_unemployed=round(Unemployed/Workforce*100,2)) %>%
ggplot(aes(fill=percent_unemployed)) +
  geom_sf(color="white") +
  theme_void() +
  theme(panel.grid.major = element_line(colour = 'transparent')) +
  scale_fill_distiller(palette="Reds", direction=1, name="Estimate") +
  labs(title="Percent unemployed in Jersey City", caption="Source: US Census/ACS5 2016") +
  NULL
```



Faceting maps

One more example.

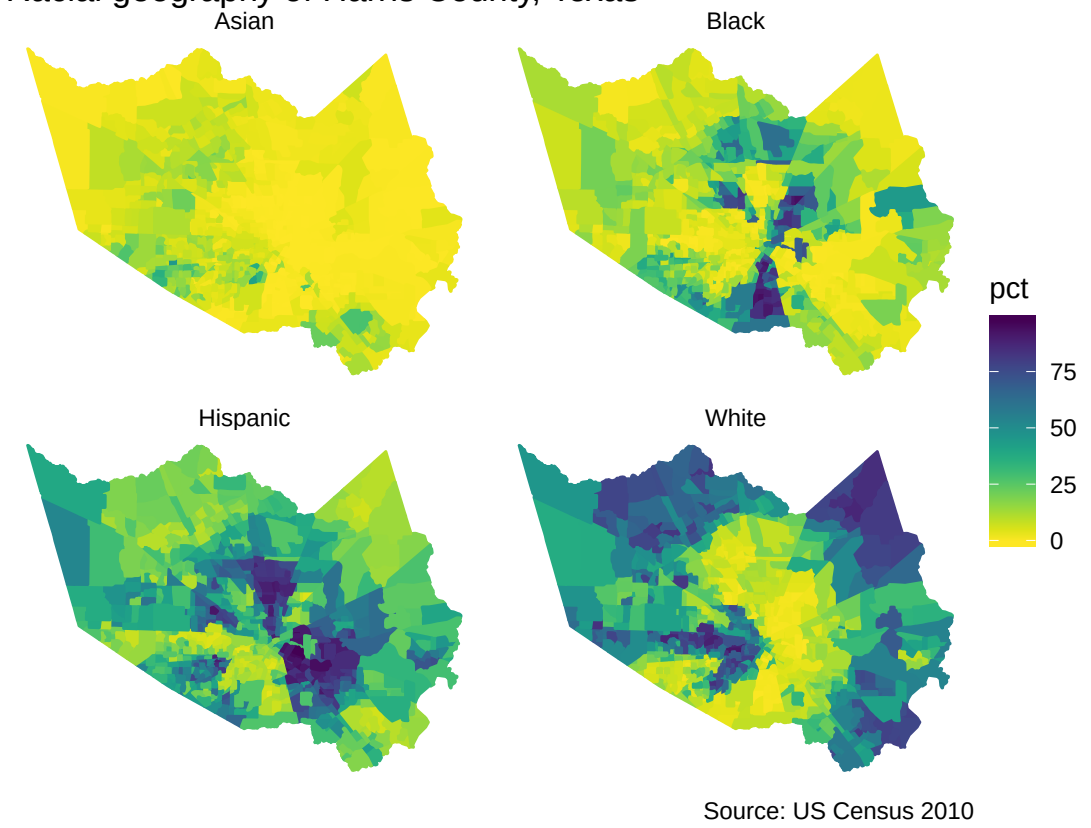
We'll pull the population of non-Hispanic whites, non-Hispanic blacks, non-Hispanic Asians, and Hispanics by Census tract for the 2010 Census. The function is `get_decennial()` and we'll also add the `summary_var` argument to get multi-group denominators.

```
racevars <- c(White = "P0050003",
              Black = "P0050004",
              Asian = "P0050006",
              Hispanic = "P0040003")

harris <- get_decennial(geography = "tract", variables = racevars,
                  state = "TX", county = "Harris County", geometry = TRUE,
                  summary_var = "P0010001")

head(harris)
```

This is a very tidy data frame.

And looks like we've have some grouping material.

```
# If you dont have the viridis package installed yet, uncomment and run the line below
#install.packages("viridis")

library(viridis)

harris %>%
  mutate(pct = 100 * (value / summary_value)) %>%
  ggplot(aes(fill = pct, color = pct)) +
  facet_wrap(~variable) +
  geom_sf() +
  coord_sf(crs = 26915) +
  scale_fill_viridis(direction=-1) +
  scale_color_viridis(direction=-1) +
  theme_void() +
  theme(panel.grid.major = element_line(colour = 'transparent')) +
  labs(title="Racial geography of Harris County, Texas", caption="Source: US Census 2010")
```



Well, we've gone over a lot of mapping techniques that do pretty much the same thing.

But now you've got a grasp of all the options.

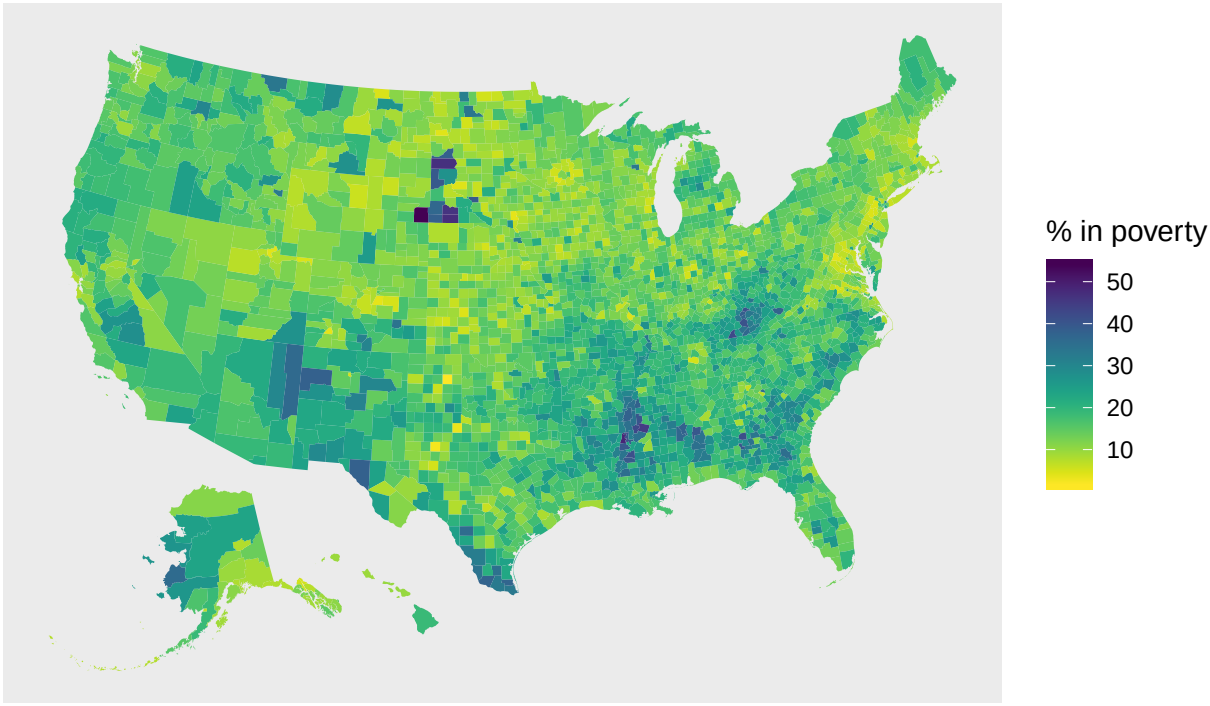
Pick which one works best for your case.

About Alaska and Hawaii

Oh yeah.

If you pass `shift_geo=T` to the `get_acs()` function in **tidycensus** then the states will be repositioned.

```
county_pov <- get_acs(geography = "county",
                      variables = "B17001_002",
                      summary_var = "B17001_001",
                      geometry = TRUE,
                      shift_geo = TRUE) %>%
  mutate(pctpov = 100 * (estimate/summary_est))

ggplot(county_pov) +
  geom_sf(aes(fill = pctpov), color=NA) +
  coord_sf(datum=NA) +
  labs(title = "Percent of population in poverty by county",
       subtitle = "Alaska and Hawaii are shifted and not to scale",
       caption = "Source: ACS 5-year, 2016",
       fill = "% in poverty") +
  scale_fill_viridis(direction=-1)
```



So, why not use **tidycensus** every time instead of **tigris**?

Well, you don't need a Census key api to use **tigris**.