Chapter 2 Notes

Table of contents

# Chapter 2: An introduction to tidycensus

**tidycensus** package has two goals:

1. make ***tidy*** census data
2. streamline the spatial data wrangling by including *geometry* requests

**tidycensus** focuses on subset of Census data, via core functions:

* get\_decennial() returns data from 2000, 2010, and 2020 Decennial Census APIs
* get\_acs() returns data from American Community Survey
  + 1-year back to 2005
  + 5-year back to 2005-2009
* get\_estimates() access population estimates apis for yearly estimates of population characteristics
  + geometries include:
    - state
    - county
    - metropolitan area
  + estimates:
    - births
    - deaths
    - migration rates
* get\_pums() access Public Use Microdata Samples
* get\_flows() Migration flows API access, including info

## 2.1 Getting started with tidycensus

```{r tidycensus-setup}  
# install and/or load tidycensus package and set API key  
#install.packages("tidycensus")  
library(tidycensus)  
library(tidyverse)  
```

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.2 ✔ readr 2.1.4  
✔ forcats 1.0.0 ✔ stringr 1.5.0  
✔ ggplot2 3.4.2 ✔ 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 tidycensus-setup}  
# census\_api\_key("insert api key here", install = TRUE)  
```

### Decennial Census

get\_decennial() function calls

```{r get\_decennial}  
total\_pop\_10 <- get\_decennial(  
 geography = "state", # aggregate by state  
 variables = "P001001",  
 year = 2010 # year we want data for  
)   
```

Getting data from the 2010 decennial Census

Using Census Summary File 1

* pulls from Summary File 1 by default, using default argument sumfile = "sf1"
* Summary File 1 exists for 2000 and 2010 Censuses, and contains core demo info for Census Geographies
* Summary File 2 contains info on population range and housing units
* Summary File 3 and 4 contain the Long Form data for 2000; these data sets are in American Community Survey sets from 2005 onward

### ACS

get\_acs() gets data from American Community Survey.

ACS collects a lot more variables not included in the short form questionnaire of the decennial Census.

```{r ACS}  
born\_in\_mexico <- get\_acs(  
 geography = "state",  
 variables = "B05006\_150",  
 year = 2020  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r ACS}  
born\_in\_mexico  
```

# A tibble: 52 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 01 Alabama B05006\_150 46927 1846  
 2 02 Alaska B05006\_150 4181 709  
 3 04 Arizona B05006\_150 510639 8028  
 4 05 Arkansas B05006\_150 60236 2182  
 5 06 California B05006\_150 3962910 25353  
 6 08 Colorado B05006\_150 215778 4888  
 7 09 Connecticut B05006\_150 28086 2144  
 8 10 Delaware B05006\_150 14616 1065  
 9 11 District of Columbia B05006\_150 4026 761  
10 12 Florida B05006\_150 257933 6418  
# ℹ 42 more rows

If the year isn’t specified, get\_acs() defaults to most recent five-year ACS sample

defaults to 5-year ACS, but can be changed to 1 year using survey = "acs1"

```{r mexican-born-1yr}  
born\_in\_mexico\_1yr <- get\_acs(  
 geography = "state",  
 variables = "B05006\_150",  
 survey = "acs1",  
 year = 2019  
)  
```

Getting data from the 2019 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

```{r mexican-born-1yr}  
born\_in\_mexico\_1yr  
```

# A tibble: 52 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 01 Alabama B05006\_150 NA NA  
 2 02 Alaska B05006\_150 NA NA  
 3 04 Arizona B05006\_150 516618 15863  
 4 05 Arkansas B05006\_150 NA NA  
 5 06 California B05006\_150 3951224 40506  
 6 08 Colorado B05006\_150 209408 12214  
 7 09 Connecticut B05006\_150 26371 4816  
 8 10 Delaware B05006\_150 NA NA  
 9 11 District of Columbia B05006\_150 NA NA  
10 12 Florida B05006\_150 261614 17571  
# ℹ 42 more rows

Variables from ACS detailed tables, data profiles, summary tables, comparison profile, and supplemental estimates are available through the function.

* auto-detects where to look based on variable name
* alternatively, supply table name to get data for all variables in table

***Example: Table B01001 ( Sex by Age)***

```{r B01001}  
age\_table <- get\_acs(  
 geography = "state",  
 table = "B01001",  
 year = 2020  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r B01001}  
age\_table  
```

# A tibble: 2,548 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 01 Alabama B01001\_001 4893186 NA  
 2 01 Alabama B01001\_002 2365734 1090  
 3 01 Alabama B01001\_003 149579 672  
 4 01 Alabama B01001\_004 150937 2202  
 5 01 Alabama B01001\_005 160287 2159  
 6 01 Alabama B01001\_006 96832 565  
 7 01 Alabama B01001\_007 65459 961  
 8 01 Alabama B01001\_008 36705 1467  
 9 01 Alabama B01001\_009 33089 1547  
10 01 Alabama B01001\_010 93871 2045  
# ℹ 2,538 more rows

## 2.2 Geography and variables in tidycensus

geography parameter in get\_acs() and `get\_decennial() requests data aggregated to common Census units

* Census blocks are only available in get\_decennial()
* Only geographies available in 2000 are "state", "county", "county subdivision", "tract", "block group", and "place"
* the geographies have to be typed exactly per the table in the documentation
  + core-based statistical areas and zip code tabulation areas have "cbsa" and "zcta" as aliases

```{r cbsa-demo}  
cbsa\_population <- get\_acs(  
 geography = "cbsa",  
 variables = "B01003\_001",  
 year = 2020  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r cbsa-demo}  
cbsa\_population  
```

# A tibble: 939 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 10100 Aberdeen, SD Micro Area B01003\_001 42864 NA  
 2 10140 Aberdeen, WA Micro Area B01003\_001 73769 NA  
 3 10180 Abilene, TX Metro Area B01003\_001 171354 NA  
 4 10220 Ada, OK Micro Area B01003\_001 38385 NA  
 5 10300 Adrian, MI Micro Area B01003\_001 98310 NA  
 6 10380 Aguadilla-Isabela, PR Metro Area B01003\_001 295172 NA  
 7 10420 Akron, OH Metro Area B01003\_001 703286 NA  
 8 10460 Alamogordo, NM Micro Area B01003\_001 66804 NA  
 9 10500 Albany, GA Metro Area B01003\_001 147431 NA  
10 10540 Albany-Lebanon, OR Metro Area B01003\_001 127216 NA  
# ℹ 929 more rows

### Geographic subsets

**tidycensus** supports geographic subsetting if supported by the API

***Example: Wisconsin income analysis***

```{r wi-income}  
wi\_income <- get\_acs(  
 geography = "county",  
 variables = "B19013\_001",  
 state = "WI",  
 year = 2020  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r wi-income}  
wi\_income  
```

# A tibble: 72 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 55001 Adams County, Wisconsin B19013\_001 48906 2387  
 2 55003 Ashland County, Wisconsin B19013\_001 47869 3190  
 3 55005 Barron County, Wisconsin B19013\_001 52346 2092  
 4 55007 Bayfield County, Wisconsin B19013\_001 57257 2496  
 5 55009 Brown County, Wisconsin B19013\_001 64728 1419  
 6 55011 Buffalo County, Wisconsin B19013\_001 58364 1871  
 7 55013 Burnett County, Wisconsin B19013\_001 53555 2513  
 8 55015 Calumet County, Wisconsin B19013\_001 76065 2314  
 9 55017 Chippewa County, Wisconsin B19013\_001 61215 2064  
10 55019 Clark County, Wisconsin B19013\_001 54463 1089  
# ℹ 62 more rows

Smaller geographies (census tracts, block groups, etc) can be subset by county.

***Example: Dane County, WI***

```{r dane-income}  
dane\_income <- get\_acs(  
 geography = "tract",  
 variables = "B19013\_001",  
 state = "WI",  
 county = "Dane",  
 year = 2020  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r dane-income}  
dane\_income  
```

# A tibble: 125 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 55025000100 Census Tract 1, Dane County, Wisconsin B19013\_… 74054 15662  
 2 55025000201 Census Tract 2.01, Dane County, Wisconsin B19013\_… 92460 27067  
 3 55025000202 Census Tract 2.02, Dane County, Wisconsin B19013\_… 88092 5189  
 4 55025000204 Census Tract 2.04, Dane County, Wisconsin B19013\_… 82717 12175  
 5 55025000205 Census Tract 2.05, Dane County, Wisconsin B19013\_… 100000 17506  
 6 55025000301 Census Tract 3.01, Dane County, Wisconsin B19013\_… 37016 11524  
 7 55025000302 Census Tract 3.02, Dane County, Wisconsin B19013\_… 117321 28723  
 8 55025000401 Census Tract 4.01, Dane County, Wisconsin B19013\_… 100434 12108  
 9 55025000402 Census Tract 4.02, Dane County, Wisconsin B19013\_… 105850 12205  
10 55025000406 Census Tract 4.06, Dane County, Wisconsin B19013\_… 74009 2811  
# ℹ 115 more rows

**NOTE:** 5-year covers down to block group; 1-year only down to geographies with ≥ 65,000 pop

## 2.3 Searching for variables in tidycensus

Variable IDs are a PITA

**tidycensus** has load\_variables() to help with searching

* two required arguments:
  1. year: the year or end-year for census dataset or ACS sample
  2. dataset: the dataset name
     1. "sf1" = Summary File 1 for 2000 or 2010
     2. "sf2" = Summary File 2 for 2000 or 2010
     3. "sf3" = Summary File 3 for 2000
     4. "sf4" = Summary File 4 for 2000
     5. "pl" = PL-94171 Redistricting dataset for 2020
     6. "acs1" 1-year ACS
     7. "acs5" 5-year ACS
        + ACS Data Profile, Summary Tables, and Comparison Profile requests require the suffixes /profile, /summary, /cprofile respectively
     8. "acsse" = ACS Supplemental Estimates variables
  3. Users can specify argument cache = TRUE to store data in user cache

```{r examples}  
v16 <- load\_variables(2016, "acs5", cache = TRUE)  
```

## 2.4 Data structure in tidycensus

**tidycensus** is built to function in the tidyverse; Wickham’s definition of “tidy” data is as follows:

1. Each observation forms a row
2. Each variable forms a column
3. Each observational unit forms a table

**tidycensus** returns tibbles of decennial Census data tidied as follows:

* GEOID - the Census ID code giving unique ID to the geographic unit
* NAME representing a descriptive name of the unit
* variable contain info on Census variable name for the row
* value the data value for each unit-variable combination.
  + in ACS data, value is replaced with estimate and moe which represents the margin of error around the estimate

Census or ACS “variables” are instead *characteristics* of enumeration units in **tidycensus**

* default setting is output = "tidy"
* rows represent data for unique spatial unit-variable combination

***Example: hhinc demo***

```{r hhcin-demo}  
hhinc <- get\_acs(  
 geography = "state",  
 table = "B19001",  
 survey = "acs1",  
 year = 2016  
)  
```

Getting data from the 2016 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Loading ACS1 variables for 2016 from table B19001. To cache this dataset for faster access to ACS tables in the future, run this function with `cache\_table = TRUE`. You only need to do this once per ACS dataset.

```{r hhcin-demo}  
hhinc  
```

# A tibble: 884 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 01 Alabama B19001\_001 1852518 12189  
 2 01 Alabama B19001\_002 176641 6328  
 3 01 Alabama B19001\_003 120590 5347  
 4 01 Alabama B19001\_004 117332 5956  
 5 01 Alabama B19001\_005 108912 5308  
 6 01 Alabama B19001\_006 102080 4740  
 7 01 Alabama B19001\_007 103366 5246  
 8 01 Alabama B19001\_008 91011 4699  
 9 01 Alabama B19001\_009 86996 4418  
10 01 Alabama B19001\_010 74864 4210  
# ℹ 874 more rows

Row 1 demonstrates the estimate for the state-characteristic pair Alabama-B19001\_001, with its associated moe

For a ‘wide’ table, use output = "wide"

```{r hhinc\_wide}  
hhinc\_wide <- get\_acs(  
 geography = "state",  
 table = "B19001",  
 survey = "acs1",  
 year = 2016,  
 output = "wide"  
)  
```

Getting data from the 2016 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Loading ACS1 variables for 2016 from table B19001. To cache this dataset for faster access to ACS tables in the future, run this function with `cache\_table = TRUE`. You only need to do this once per ACS dataset.

```{r hhinc\_wide}  
hhinc\_wide  
```

# A tibble: 52 × 36  
 GEOID NAME B19001\_001E B19001\_001M B19001\_002E B19001\_002M B19001\_003E  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 28 Mississippi 1091245 8803 113124 4835 87136  
 2 29 Missouri 2372190 10844 160615 6705 122649  
 3 30 Montana 416125 4426 26734 2183 24786  
 4 31 Nebraska 747562 4452 45794 3116 33266  
 5 32 Nevada 1055158 6433 68507 4886 42720  
 6 33 New Hampsh… 520643 5191 20890 2566 15933  
 7 34 New Jersey 3194519 10274 170029 6836 118862  
 8 35 New Mexico 758364 6296 66983 4439 48930  
 9 36 New York 7209054 17665 543763 12132 352029  
10 37 North Caro… 3882423 16063 282491 7816 228088  
# ℹ 42 more rows  
# ℹ 29 more variables: B19001\_003M <dbl>, B19001\_004E <dbl>, B19001\_004M <dbl>,  
# B19001\_005E <dbl>, B19001\_005M <dbl>, B19001\_006E <dbl>, B19001\_006M <dbl>,  
# B19001\_007E <dbl>, B19001\_007M <dbl>, B19001\_008E <dbl>, B19001\_008M <dbl>,  
# B19001\_009E <dbl>, B19001\_009M <dbl>, B19001\_010E <dbl>, B19001\_010M <dbl>,  
# B19001\_011E <dbl>, B19001\_011M <dbl>, B19001\_012E <dbl>, B19001\_012M <dbl>,  
# B19001\_013E <dbl>, B19001\_013M <dbl>, B19001\_014E <dbl>, …

### Understanding GEOIDs

* For core Census hierarchy (Census block -> block group -> census tract -> county -> state -> division -> region -> nation) GEOID identifies specific unit AND unit’s parent geographies

***Example: Cimarron County, Oklahoma GEOIDs***

```{r}  
cimarron\_blocks <- get\_decennial(  
 geography = "block",  
 variables = "H1\_001N",  
 state = "OK",  
 county = "Cimarron",  
 year = 2020,  
 sumfile = "pl"  
)  
```

Getting data from the 2020 decennial Census

Using the PL 94-171 Redistricting Data Summary File

Note: 2020 decennial Census data use differential privacy, a technique that  
introduces errors into data to preserve respondent confidentiality.  
ℹ Small counts should be interpreted with caution.  
ℹ See https://www.census.gov/library/fact-sheets/2021/protecting-the-confidentiality-of-the-2020-census-redistricting-data.html for additional guidance.  
This message is displayed once per session.

```{r}  
cimarron\_blocks  
```

# A tibble: 1,269 × 4  
 GEOID NAME variable value  
 <chr> <chr> <chr> <dbl>  
 1 400259501001984 Block 1984, Block Group 1, Census Tract 9501,… H1\_001N 0  
 2 400259503001035 Block 1035, Block Group 1, Census Tract 9503,… H1\_001N 0  
 3 400259503001068 Block 1068, Block Group 1, Census Tract 9503,… H1\_001N 5  
 4 400259503001146 Block 1146, Block Group 1, Census Tract 9503,… H1\_001N 0  
 5 400259503001197 Block 1197, Block Group 1, Census Tract 9503,… H1\_001N 7  
 6 400259503001218 Block 1218, Block Group 1, Census Tract 9503,… H1\_001N 2  
 7 400259501001067 Block 1067, Block Group 1, Census Tract 9501,… H1\_001N 0  
 8 400259501001118 Block 1118, Block Group 1, Census Tract 9501,… H1\_001N 0  
 9 400259501001141 Block 1141, Block Group 1, Census Tract 9501,… H1\_001N 0  
10 400259501001223 Block 1223, Block Group 1, Census Tract 9501,… H1\_001N 0  
# ℹ 1,259 more rows

For the GEOID, 40.025.950100.1.501 (periods added for clarity), it can be read as follows

* first two digits (**40)** are the State FIPS code for Oklahoma
* digits three through 5 (**025)** correspond to Cimarron County within Oklahoma
  + To look at counties across the whole US, the 5 digits together uniquely identify a county - so Oklahoma’s Cimarron County can be read as **40025**
* next six digits represent the Census tract (**950100)** - in this case, Census Tract 9501, with zeroes padding the right.
* Twelfth digit **(1)** represents the parent block group of the Census block
  + A census tract will have up to, but no more than, 9 census block groups
* The final three digits (**501)** represent the individual census block.
  + The block’s name is a combination of this ID and the parent block group digit (**1501)**

For geographies outside the core hierarchy, will only contain IDs of parent geographies they fully nest within.

cbsa’s and zcta’s will have their own unique GEOIDs independent of other aggregation levels.

### Renaming variable IDs

passing a named vector to variables parameter will feturn desired names rather than Census variable IDs

```{r georgia-demo}  
ga <- get\_acs(  
 geography = "county",  
 state = "Georgia",  
 variables = c(medinc = "B19013\_001",  
 medage = "B01002\_001"),  
 year = 2020  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r georgia-demo}  
ga  
  
ga\_wide <- get\_acs(  
 geography = "county",  
 state = "Georgia",  
 variables = c(medinc = "B19013\_001",  
 medage = "B01002\_001"),  
 year = 2020,  
 output = "wide"  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r georgia-demo}  
ga\_wide  
```

# A tibble: 318 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 13001 Appling County, Georgia medage 39.9 1.7  
 2 13001 Appling County, Georgia medinc 37924 4761   
 3 13003 Atkinson County, Georgia medage 35.9 1.5  
 4 13003 Atkinson County, Georgia medinc 35703 5493   
 5 13005 Bacon County, Georgia medage 36.5 1   
 6 13005 Bacon County, Georgia medinc 36692 3774   
 7 13007 Baker County, Georgia medage 52.2 4.8  
 8 13007 Baker County, Georgia medinc 34034 9879   
 9 13009 Baldwin County, Georgia medage 35.8 0.5  
10 13009 Baldwin County, Georgia medinc 46250 4707   
# ℹ 308 more rows  
# A tibble: 159 × 6  
 GEOID NAME medincE medincM medageE medageM  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
 1 13001 Appling County, Georgia 37924 4761 39.9 1.7  
 2 13003 Atkinson County, Georgia 35703 5493 35.9 1.5  
 3 13005 Bacon County, Georgia 36692 3774 36.5 1   
 4 13007 Baker County, Georgia 34034 9879 52.2 4.8  
 5 13011 Banks County, Georgia 50912 4278 41.5 1.1  
 6 13013 Barrow County, Georgia 62990 2562 36 0.3  
 7 13017 Ben Hill County, Georgia 32077 4008 39.5 1.4  
 8 13021 Bibb County, Georgia 41317 1220 36.3 0.3  
 9 13023 Bleckley County, Georgia 46992 6279 36 1.5  
10 13027 Brooks County, Georgia 37516 4438 43.6 0.9  
# ℹ 149 more rows

## 2.5 Other Census Bureau datasets in tidycensus

* **tidycensus** does not grant access to all datasets from Census API
  + **censusapi** provides more general api access
* Two other datasets with custom functions are get\_estimates() and get\_flows() to access Population Estimates and ACS Migration Flows, respectively.

### Using get\_estimates()

**Population Estimates Program (PEP)** provides yearly estimates based on projections drawn from birth, death, and migration rates from the most recent Census.

It does provide access to indicators used for projection

* indicators can be specified using the product argument, either individually or in bulk.
  + population change estimates are pulled with "components"

***Example: Queens County population estimate components***

```{r pep\_components}  
queens\_components <- get\_estimates(  
 geography = "county",  
 product = "components",  
 state = "NY",  
 county = "Queens",  
 year = 2019  
 )  
queens\_components  
```

# A tibble: 12 × 4  
 NAME GEOID variable value  
 <chr> <chr> <chr> <dbl>  
 1 Queens County, New York 36081 BIRTHS 27453   
 2 Queens County, New York 36081 DEATHS 16380   
 3 Queens County, New York 36081 DOMESTICMIG -41789   
 4 Queens County, New York 36081 INTERNATIONALMIG 9883   
 5 Queens County, New York 36081 NATURALINC 11073   
 6 Queens County, New York 36081 NETMIG -31906   
 7 Queens County, New York 36081 RBIRTH 12.1   
 8 Queens County, New York 36081 RDEATH 7.23  
 9 Queens County, New York 36081 RDOMESTICMIG -18.5   
10 Queens County, New York 36081 RINTERNATIONALMIG 4.36  
11 Queens County, New York 36081 RNATURALINC 4.89  
12 Queens County, New York 36081 RNETMIG -14.1

The variables returned above include:

* BIRTHS: raw count of births from July 1 2018 to July 1 2019
* DEATHS: raw count of births from July 1 2018 to July 1 2019
* DOMESTICMIG: Net domestic migration in period
* INTERNATIONALMIG: Net international migration in period
* NATURALINC: Natural increase in period (Birth - Death)
* NETMIG: Net migration in period
* RBIRTH: Crude birth rate in period (per 1000)
* RDEATH: Crude death rate in period (per 1000)
* RDOMESTICMIG: Crude Domestic Migration Rate
* RINTERNATIONALMIG: Crude International Migration Rate
* RNATURALINC: Crude Natural Increase Rate
* RNETMIG: Crude Net Migration Rate

The "characteristics" product lets users get unique demographic info, esp. when using breakdown argument with the arguments "AGEGROUP", "RACE", "SEX", or "HISP"

The breakdown\_labels = TRUE returns informative labels

```{r}  
louisiana\_sex\_hisp <- get\_estimates(  
 geography = "state",  
 product = "characteristics",  
 breakdown = c("SEX", "HISP"),  
 breakdown\_labels = TRUE,  
 state = "LA",  
 year = 2019  
)  
louisiana\_sex\_hisp  
```

# A tibble: 9 × 5  
 GEOID NAME value SEX HISP   
 <chr> <chr> <dbl> <chr> <chr>   
1 22 Louisiana 4648794 Both sexes Both Hispanic Origins  
2 22 Louisiana 4401822 Both sexes Non-Hispanic   
3 22 Louisiana 246972 Both sexes Hispanic   
4 22 Louisiana 2267050 Male Both Hispanic Origins  
5 22 Louisiana 2135979 Male Non-Hispanic   
6 22 Louisiana 131071 Male Hispanic   
7 22 Louisiana 2381744 Female Both Hispanic Origins  
8 22 Louisiana 2265843 Female Non-Hispanic   
9 22 Louisiana 115901 Female Hispanic

### Using get\_flows()

get\_flows() pulls down info on in- and out-migration for states, counties, and metropolitan areas.

* pulls for a given geography, using data from a given 5-year ACS sample.

***Example: Honolulu County, Hawaii***

```{r example-flow}  
honolulu\_migration <- get\_flows(  
 geography = "county",  
 state = "HI",  
 county = "Honolulu",  
 year = 2019  
)  
honolulu\_migration  
```

# A tibble: 3,156 × 7  
 GEOID1 GEOID2 FULL1\_NAME FULL2\_NAME variable estimate moe  
 <chr> <chr> <chr> <chr> <chr> <dbl> <dbl>  
 1 15003 <NA> Honolulu County, Hawaii Africa MOVEDIN 152 156  
 2 15003 <NA> Honolulu County, Hawaii Africa MOVEDOUT NA NA  
 3 15003 <NA> Honolulu County, Hawaii Africa MOVEDNET NA NA  
 4 15003 <NA> Honolulu County, Hawaii Asia MOVEDIN 7680 884  
 5 15003 <NA> Honolulu County, Hawaii Asia MOVEDOUT NA NA  
 6 15003 <NA> Honolulu County, Hawaii Asia MOVEDNET NA NA  
 7 15003 <NA> Honolulu County, Hawaii Central America MOVEDIN 192 100  
 8 15003 <NA> Honolulu County, Hawaii Central America MOVEDOUT NA NA  
 9 15003 <NA> Honolulu County, Hawaii Central America MOVEDNET NA NA  
10 15003 <NA> Honolulu County, Hawaii Caribbean MOVEDIN 97 78  
# ℹ 3,146 more rows

get\_flows() also includes migration flow mapping functionality, discussed later

## 2.6 Debugging tidycensus errors

**tidycensus** carries through census API errors where possible, and tries to translate common errors.

***Example: mis-typed variable error***

```{r}  
state\_pop <- get\_decennial(  
 geography = "state",  
 variables = "P01001",  
 year = 2010  
)  
```

Getting data from the 2010 decennial Census

Using Census Summary File 1

Error in UseMethod("gather"): no applicable method for 'gather' applied to an object of class "character"

Error : Your API call has errors. The API message returned is error: error: unknown variable 'P01001'.

Note in console, error: Error : Your API call has errors. The API message returned is error: error: unknown variable 'P01001'. Using Census Summary File 1

For an example where a passed geography is not present in the given dataset

***Example: Unavailable Geography***

```{r}  
cbsa\_ohio <- get\_acs(  
 geography = "cbsa",  
 variables = "DP02\_0068P",  
 state = "OH",  
 year = 2019  
)  
```

Getting data from the 2015-2019 5-year ACS

Using the ACS Data Profile

Error: Your API call has errors. The API message returned is error: unknown/unsupported geography heirarchy.

**tidycensus** includes the show\_call parameter that prints out the API call made

```{r}  
cbsa\_bachelors <- get\_acs(  
 geography = "cbsa",  
 variables = "DP02\_0068P",  
 year = 2019,  
 show\_call = TRUE  
)  
```

Getting data from the 2015-2019 5-year ACS

Using the ACS Data Profile

Census API call: https://api.census.gov/data/2019/acs/acs5/profile?get=DP02\_0068PE%2CDP02\_0068PM%2CNAME&for=metropolitan%20statistical%20area%2Fmicropolitan%20statistical%20area%3A%2A

## Exercises

1. Review the available geographies in tidycensus from the geography table in this chapter. Acquire data on median age (variable B01002\_001) for a geography we have not used.

* ```{r}  
  california <- get\_acs(  
   geography = "zcta",  
   variables = "B01002\_001",  
   show\_call = FALSE)  
  ```
* Getting data from the 2017-2021 5-year ACS
* ```{r}  
  california  
  ```
* # A tibble: 33,774 × 5  
   GEOID NAME variable estimate moe  
   <chr> <chr> <chr> <dbl> <dbl>  
   1 00601 ZCTA5 00601 B01002\_001 43.7 1   
   2 00602 ZCTA5 00602 B01002\_001 44.4 0.4  
   3 00603 ZCTA5 00603 B01002\_001 44.1 0.7  
   4 00606 ZCTA5 00606 B01002\_001 44.9 1.3  
   5 00610 ZCTA5 00610 B01002\_001 43.5 0.7  
   6 00611 ZCTA5 00611 B01002\_001 47.2 10.5  
   7 00612 ZCTA5 00612 B01002\_001 43.5 1   
   8 00616 ZCTA5 00616 B01002\_001 45.3 3.9  
   9 00617 ZCTA5 00617 B01002\_001 42.4 0.8  
  10 00622 ZCTA5 00622 B01002\_001 48.3 6.2  
  # ℹ 33,764 more rows

1. Use the load\_variables() function to find a variable that interests you that we haven’t used yet. Use get\_acs() to fetch data from the 2016-2020 ACS for counties in the state where you live, where you have visited, or where you would like to visit.

* ```{r}  
  #summary\_files(2020)  
  load\_variables(2020, "dp")  
    
  los\_angeles\_u\_5 <- get\_acs(  
   geography = "tract",  
   state = "CA",  
   county = "Los Angeles",  
   variables = c(male\_u\_5 = "B01001\_003",   
   female\_u\_5 = "B01001\_027",  
   total\_pop\_est = "B01001\_001"),  
   summary\_var = "B01001\_001"  
  )  
  ```
* Getting data from the 2017-2021 5-year ACS
* ```{r}  
  los\_angeles\_u\_5 <- get\_decennial(  
   geography = "tract",  
   state = "CA",  
   county = "Los Angeles",  
   variables = c(total\_u\_5 = "DP1\_0002C"),  
   sumfile = "dp"  
  )  
  ```
* Getting data from the 2020 decennial Census
* Using the Demographic Profile
* # A tibble: 320 × 3  
   name label concept   
   <chr> <chr> <chr>   
   1 DP1\_0001C Count!!SEX AND AGE!!Total population PROFILE OF …  
   2 DP1\_0001P Percent!!SEX AND AGE!!Total population PROFILE OF …  
   3 DP1\_0002C Count!!SEX AND AGE!!Total population!!Under 5 years PROFILE OF …  
   4 DP1\_0002P Percent!!SEX AND AGE!!Total population!!Under 5 years PROFILE OF …  
   5 DP1\_0003C Count!!SEX AND AGE!!Total population!!5 to 9 years PROFILE OF …  
   6 DP1\_0003P Percent!!SEX AND AGE!!Total population!!5 to 9 years PROFILE OF …  
   7 DP1\_0004C Count!!SEX AND AGE!!Total population!!10 to 14 years PROFILE OF …  
   8 DP1\_0004P Percent!!SEX AND AGE!!Total population!!10 to 14 years PROFILE OF …  
   9 DP1\_0005C Count!!SEX AND AGE!!Total population!!15 to 19 years PROFILE OF …  
  10 DP1\_0005P Percent!!SEX AND AGE!!Total population!!15 to 19 years PROFILE OF …  
  # ℹ 310 more rows
* ACS5 data for age under 5
  + B01001\_003 = Male Estimated Total Age Under 5
  + B01001\_027 = Female Estimated Total Age under 5
  + B01001\_001 = Estimated Total, regardless of age or gender
* Decennial census 2020 data for age under 5
  + DP1\_0002C = Count of total population under age of 5