# Package 'tidycensus'

January 31, 2025

**Type** Package

**Title** Load US Census Boundary and Attribute Data as 'tidyverse' and 'sf'-Ready Data Frames

**Version** 1.7.1

**Date** 2025-01-28

**URL** https://walker-data.com/tidycensus/

**BugReports** https://github.com/walkerke/tidycensus/issues

**Description** An integrated R interface to several United States Census Bureau APIs (<https://www.census.gov/data/developers/data-sets.html>) and the US Census Bureau's geographic boundary files. Allows R users to return Census and ACS data as tidyverse-ready data frames, and optionally returns a list-column with feature geometry for mapping and spatial analysis.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.3.0)

**Imports** httr, sf, dplyr (>= 1.0.0), tigris, stringr, jsonlite (>= 1.5.0), purrr, rvest, tidyr (>= 1.0.0), rappdirs, readr, xml2, units, utils, rlang, crayon, tidyselect

**Suggests** ggplot2, survey, srvyr, terra

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Kyle Walker [aut, cre], Matt Herman [aut], Kris Eberwein [ctb]

**Maintainer** Kyle Walker <kyle@walker-data.com>

**Repository** CRAN

**Date/Publication** 2025-01-31 15:30:02 UTC

# Contents

acs5_geography . . . 2
as_dot_density . . . 3
census_api_key . . . 5
check_ddhca_groups . . . 6
county_laea . . . 7
fips_codes . . . 7
get_acs . . . 8
get_decennial . . . 10
get_estimates . . . 13
get_flows . . . 15
get_pop_groups . . . 18
get_pums . . . 18
interpolate_pw . . . 20
load_variables . . . 23
mig_recodes . . . 24
moe_product . . . 25
moe_prop . . . 25
moe_ratio . . . 26
moe_sum . . . 26
pums_variables . . . 27
significance . . . 28
state_laea . . . 28
summary_files . . . 29
to_survey . . . 29

**Index** **31**

---

`acs5_geography` *Dataset used to identify geography availability in the 5-year ACS Detailed Tables*

---

## Description

Built-in dataset for use by `load_variables()` to identify the smallest geography at which 5-year ACS data are available

- `table`: The ACS Table ID
- `geography`: The smallest geography at which a given table is available for a given year
- `year`: The endyear of the 5-year ACS dataset

## Usage

```
data(acs5_geography)
```

### Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 13355 rows and 3 columns.

### Details

Dataset used to identify geography availability in the 5-year ACS Detailed Tables

Built-in dataset that includes information on the smallest geography at which 5-year ACS Detailed Tables data are available, by table, since 2011. This dataset is used internally by `load_variables()` to add a `geography` column when variables are retrieved for a 5-year ACS Detailed Tables dataset.

---

## `as_dot_density` *Convert polygon geometry to dots for dot-density mapping*

---

### Description

Dot-density maps are a compelling alternative to choropleth maps for cartographic visualization of demographic data as they allow for representation of the internal heterogeneity of geographic units. This function helps users generate dots from an input polygon dataset intended for dot-density mapping. Dots are placed randomly within polygons according to a given data:dots ratio; for example, a ratio of 100:1 for an input population value column will place approximately 1 dot in the polygon for every 100 people in the geographic unit. Users can then map the dots using tools like `ggplot2::geom_sf()` or `tmap::tm_dots()`.

### Usage

```
as_dot_density(
  input_data,
  value,
  values_per_dot,
  group = NULL,
  erase_water = FALSE,
  area_threshold = NULL,
  water_year = 2020
)
```

### Arguments

| | |
|---|---|
| `input_data` | An input sf object of geometry type `POLYGON` or `MULTIPOLYGON` that includes some information that can be converted to dots. While the function is designed for use with data acquired with the tidycensus package, it will work for arbitrary polygon datasets. |
| `value` | The value column to be used to determine the number of dots to generate. For tidycensus users, this will typically be the `"value"` column for decennial Census data or the `"estimate"` column for American Community Survey estimates. |
| `values_per_dot` | The number of values per dot; used to determine the output data:dots ratio. A value of 100 means that each dot will represent approximately 100 values in the value column. |

| | |
|---|---|
| group | A column in the dataset that identifies salient groups within which dots should be generated. For a long-form tidycensus dataset, this will typically be the "variable" column or some derivative of it. The output dataset will be randomly shuffled to prevent "stacking" of groups in downstream dot-density maps. |
| erase_water | If TRUE, calls tigris::erase_water() to remove water areas from the polygons prior to generating dots, allowing for dasymetric dot placement. This option is recommended if your location includes significant water area. If using this option, it is recommended that you first transform your data to a projected coordinate reference system using sf::st_transform() to improve performance. This argument only works for data in the United States. |
| area_threshold | The area percentile threshold to be used when erasing water; ranges from 0 (all water area included) to 1 (no water area included) |
| water_year | The year of the TIGER/Line water area shapefiles to use if erasing water. Defaults to 2020; ignore if not using the erase_water feature. |

## Details

as_dot_density() uses terra::dots() internally for fast creation of dots. As terra is not a hard dependency of the tidycensus package, users must first install terra before using this function.

The erase_water parameter will internally call tigris::erase_water() to fetch water area for a given location in the United States and remove that water area from the polygons before placing dots in polygons. This will slow down performance of the function, but can improve cartographic accuracy in locations with significant water area. It is recommended that users transform their data into a projected coordinate reference system with sf::st_transform() prior to using this option in order to improve performance.

## Value

The original dataset but of geometry type POINT, with the number of point features corresponding to the given value:dot ratio for a given group.

## Examples

```
## Not run:

library(tidycensus)
library(ggplot2)

# Identify variables for mapping
race_vars <- c(
  Hispanic = "P2_002N",
  White = "P2_005N",
  Black = "P2_006N",
  Asian = "P2_008N"
)

# Get data from tidycensus
baltimore_race <- get_decennial(
  geography = "tract",
```

```
  variables = race_vars,
  state = "MD",
  county = "Baltimore city",
  geometry = TRUE,
  year = 2020
)

# Convert data to dots
baltimore_dots <- as_dot_density(
  baltimore_race,
  value = "value",
  values_per_dot = 100,
  group = "variable"
)

# Use one set of polygon geometries as a base layer
baltimore_base <- baltimore_race[baltimore_race$variable == "Hispanic", ]

# Map with ggplot2
ggplot() +
  geom_sf(data = baltimore_base,
          fill = "white",
          color = "grey") +
  geom_sf(data = baltimore_dots,
          aes(color = variable),
          size = 0.01) +
  theme_void()


## End(Not run)
```

---

`census_api_key` *Install a CENSUS API Key in Your* `.Renviron` *File for Repeated Use*

---

## Description

This function will add your CENSUS API key to your `.Renviron` file so it can be called securely without being stored in your code. After you have installed your key, it can be called any time by typing `Sys.getenv("CENSUS_API_KEY")` and can be used in package functions by simply typing CENSUS_API_KEY If you do not have an `.Renviron` file, the function will create on for you. If you already have an `.Renviron` file, the function will append the key to your existing file, while making a backup of your original file for disaster recovery purposes.

## Usage

```
census_api_key(key, overwrite = FALSE, install = FALSE)
```

### Arguments

| | |
|---|---|
| key | The API key provided to you from the Census formated in quotes. A key can be acquired at http://api.census.gov/data/key_signup.html |
| overwrite | If this is set to TRUE, it will overwrite an existing CENSUS_API_KEY that you already have in your .Renviron file. |
| install | if TRUE, will install the key in your .Renviron file for use in future sessions. Defaults to FALSE. |

### Examples

```
## Not run:
census_api_key("111111abc", install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("CENSUS_API_KEY")

## End(Not run)

## Not run:
# If you need to overwrite an existing key:
census_api_key("111111abc", overwrite = TRUE, install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("CENSUS_API_KEY")

## End(Not run)
```

---

## check_ddhca_groups *Check to see if a given geography / population group combination is available in the Detailed DHC-A file.*

---

### Description

Check to see if a given geography / population group combination is available in the Detailed DHC-A file.

### Usage

```
check_ddhca_groups(geography, pop_group, state = NULL, county = NULL)
```

### Arguments

| | |
|---|---|
| geography | The requested geography. |
| pop_group | The code representing the population group you'd like to check. |
| state | The state (optional) |
| county | The county (optional) |

## county_laea *County geometry with Alaska and Hawaii shifted and re-scaled*

### Description

Built-in dataset for use with `shift_geo = TRUE`

Dataset of US counties with Alaska and Hawaii shifted and re-scaled

### Usage

```
data(county_laea)

data(county_laea)
```

### Format

An object of class `sf` (inherits from `data.frame`) with 3143 rows and 2 columns.

### Details

Dataset with county geometry for use when shifting Alaska and Hawaii

Built-in dataset for use with the `shift_geo` parameter, with the continental United States in a Lambert azimuthal equal area projection and Alaska and Hawaii counties and Census areas shifted and re-scaled. The data were originally obtained from the albersusa R package (https://github.com/hrbrmstr/albersusa).

## fips_codes *Dataset with FIPS codes for US states and counties*

### Description

Built-in dataset for smart state and county lookup. To access the data directly, issue the command `data(fips_codes)`.

- `county`: County name, title-case
- `county_code`: County code. (3-digit, 0-padded, character)
- `state`: Upper-case abbreviation of state
- `state_code`: State FIPS code (2-digit, 0-padded, character)
- `state_name`: Title-case name of state

### Usage

```
data(fips_codes)
```

## Format

An object of class `data.frame` with 3256 rows and 5 columns.

## Details

Dataset with FIPS codes for US states and counties

Built-in dataset for use with the `lookup_code` function. To access the data directly, issue the command `data(fips_codes)`.

Note: this dataset includes FIPS codes for all counties that have appeared in the decennial Census or American Community Survey from 2010 to the present. This means that counties that have been renamed or absorbed into other geographic entities since 2010 remain in this dataset along with newly added or renamed counties.

If you need the FIPS codes and names for counties for a particular Census year, you can use the counties function from the tigris package and set the year parameter as required.

---

# get_acs *Obtain data and feature geometry for the American Community Survey*

---

## Description

Obtain data and feature geometry for the American Community Survey

## Usage

```
get_acs(
  geography,
  variables = NULL,
  table = NULL,
  cache_table = FALSE,
  year = 2023,
  output = "tidy",
  state = NULL,
  county = NULL,
  zcta = NULL,
  geometry = FALSE,
  keep_geo_vars = FALSE,
  shift_geo = FALSE,
  summary_var = NULL,
  key = NULL,
  moe_level = 90,
  survey = "acs5",
  show_call = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| geography | The geography of your data. |
| variables | Character string or vector of character strings of variable IDs. tidycensus automatically returns the estimate and the margin of error associated with the variable. |
| table | The ACS table for which you would like to request all variables. Uses lookup tables to identify the variables; performs faster when variable table already exists through `load_variables(cache = TRUE)`. Only one table may be requested per call. |
| cache_table | Whether or not to cache table names for faster future access. Defaults to FALSE; if TRUE, only needs to be called once per dataset. If variables dataset is already cached via the `load_variables` function, this can be bypassed. |
| year | The year, or endyear, of the ACS sample. 5-year ACS data is available from 2009 through 2023; 1-year ACS data is available from 2005 through 2023, with the exception of 2020. Defaults to 2023. |
| output | One of "tidy" (the default) in which each row represents an enumeration unit-variable combination, or "wide" in which each row represents an enumeration unit and the variables are in the columns. |
| state | An optional vector of states for which you are requesting data. State names, postal codes, and FIPS codes are accepted. Defaults to NULL. |
| county | The county for which you are requesting data. County names and FIPS codes are accepted. Must be combined with a value supplied to 'state'. Defaults to NULL. |
| zcta | The zip code tabulation area(s) for which you are requesting data. Specify a single value or a vector of values to get data for more than one ZCTA. Numeric or character ZCTA GEOIDs are accepted. When specifying ZCTAs, geography must be set to '"zcta"' and 'state' must be specified with 'county' left as 'NULL'. Defaults to NULL. |
| geometry | if FALSE (the default), return a regular tibble of ACS data. if TRUE, uses the tigris package to return an sf tibble with simple feature geometry in the 'geometry' column. |
| keep_geo_vars | if TRUE, keeps all the variables from the Census shapefile obtained by tigris. Defaults to FALSE. |
| shift_geo | (deprecated) if TRUE, returns geometry with Alaska and Hawaii shifted for thematic mapping of the entire US. Geometry was originally obtained from the albersusa R package. As of May 2021, we recommend using `tigris::shift_geometry()` instead. |
| summary_var | Character string of a "summary variable" from the ACS to be included in your output. Usually a variable (e.g. total population) that you'll want to use as a denominator or comparison. |
| key | Your Census API key. Obtain one at https://api.census.gov/data/key_signup.html |
| moe_level | The confidence level of the returned margin of error. One of 90 (the default), 95, or 99. |

| | |
|---|---|
| survey | The ACS contains one-year, three-year, and five-year surveys expressed as "acs1", "acs3", and "acs5". The default selection is "acs5." |
| show_call | if TRUE, display call made to Census API. This can be very useful in debugging and determining if error messages returned are due to tidycensus or the Census API. Copy to the API call into a browser and see what is returned by the API directly. Defaults to FALSE. |
| ... | Other keyword arguments |

## Value

A tibble or sf tibble of ACS data

## Examples

```
## Not run:
library(tidycensus)
library(tidyverse)
library(viridis)
census_api_key("YOUR KEY GOES HERE")

tarr <- get_acs(geography = "tract", variables = "B19013_001",
                state = "TX", county = "Tarrant", geometry = TRUE, year = 2020)

ggplot(tarr, aes(fill = estimate, color = estimate)) +
  geom_sf() +
  coord_sf(crs = 26914) +
  scale_fill_viridis(option = "magma") +
  scale_color_viridis(option = "magma")


vt <- get_acs(geography = "county", variables = "B19013_001", state = "VT", year = 2019)

vt %>%
mutate(NAME = gsub(" County, Vermont", "", NAME)) %>%
 ggplot(aes(x = estimate, y = reorder(NAME, estimate))) +
 geom_errorbar(aes(xmin = estimate - moe, xmax = estimate + moe), width = 0.3, size = 0.5) +
  geom_point(color = "red", size = 3) +
  labs(title = "Household income by county in Vermont",
       subtitle = "2015-2019 American Community Survey",
       y = "",
       x = "ACS estimate (bars represent margin of error)")


## End(Not run)
```

---

get_decennial *Obtain data and feature geometry for the decennial US Census*

---

## Description

Obtain data and feature geometry for the decennial US Census

## Usage

```
get_decennial(
  geography,
  variables = NULL,
  table = NULL,
  cache_table = FALSE,
  year = 2020,
  sumfile = NULL,
  state = NULL,
  county = NULL,
  geometry = FALSE,
  output = "tidy",
  keep_geo_vars = FALSE,
  shift_geo = FALSE,
  summary_var = NULL,
  pop_group = NULL,
  pop_group_label = FALSE,
  key = NULL,
  show_call = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| geography | The geography of your data. |
| variables | Character string or vector of character strings of variable IDs. |
| table | The Census table for which you would like to request all variables. Uses lookup tables to identify the variables; performs faster when variable table already exists through `load_variables(cache = TRUE)`. Only one table may be requested per call. |
| cache_table | Whether or not to cache table names for faster future access. Defaults to FALSE; if TRUE, only needs to be called once per dataset. If variables dataset is already cached via the `load_variables` function, this can be bypassed. |
| year | The year for which you are requesting data. Defaults to 2020; 2000, 2010, and 2020 are available. |
| sumfile | The Census summary file; if `NULL`, defaults to `"pl"` when the year is 2020 and `"sf1"` for 2000 and 2010. Not all summary files are available for each decennial Census year. Make sure you are using the correct summary file for your requested variables, as variable IDs may be repeated across summary files and represent different topics. |
| state | The state for which you are requesting data. State names, postal codes, and FIPS codes are accepted. Defaults to NULL. |

| | |
|---|---|
| county | The county for which you are requesting data. County names and FIPS codes are accepted. Must be combined with a value supplied to 'state'. Defaults to NULL. |
| geometry | if FALSE (the default), return a regular tibble of ACS data. if TRUE, uses the tigris package to return an sf tibble with simple feature geometry in the 'geometry' column. |
| output | One of "tidy" (the default) in which each row represents an enumeration unit-variable combination, or "wide" in which each row represents an enumeration unit and the variables are in the columns. |
| keep_geo_vars | if TRUE, keeps all the variables from the Census shapefile obtained by tigris. Defaults to FALSE. |
| shift_geo | (deprecated) if TRUE, returns geometry with Alaska and Hawaii shifted for thematic mapping of the entire US. Geometry was originally obtained from the albersusa R package. As of May 2021, we recommend using `tigris::shift_geometry()` instead. |
| summary_var | Character string of a "summary variable" from the decennial Census to be included in your output. Usually a variable (e.g. total population) that you'll want to use as a denominator or comparison. |
| pop_group | The population group code for which you'd like to request data. Applies to summary files for which population group breakdowns are available like the Detailed DHC-A file. |
| pop_group_label | If `TRUE`, return a `"pop_group_label"` column that contains the label for the population group. Defaults to `FALSE`. |
| key | Your Census API key. Obtain one at https://api.census.gov/data/key_signup.html |
| show_call | if TRUE, display call made to Census API. This can be very useful in debugging and determining if error messages returned are due to tidycensus or the Census API. Copy to the API call into a browser and see what is returned by the API directly. Defaults to FALSE. |
| ... | Other keyword arguments |

## Value

a tibble or sf tibble of decennial Census data

## Examples

```
## Not run:
# Plot of race/ethnicity by county in Illinois for 2010
library(tidycensus)
library(tidyverse)
library(viridis)
census_api_key("YOUR KEY GOES HERE")
vars10 <- c("P005003", "P005004", "P005006", "P004003")

il <- get_decennial(geography = "county", variables = vars10, year = 2010,
```

```
                    summary_var = "P001001", state = "IL", geometry = TRUE) %>%
  mutate(pct = 100 * (value / summary_value))

ggplot(il, aes(fill = pct, color = pct)) +
  geom_sf() +
  facet_wrap(~variable)



## End(Not run)
```

---

get_estimates *Get data from the US Census Bureau Population Estimates Program*

---

## Description

The get_estimates() function requests data from the US Census Bureau's Population Estimates Program (PEP) datasets. The PEP datasets are defined by the US Census Bureau as follows: "The Census Bureau's Population Estimates Program (PEP) produces estimates of the population for the United States, its states, counties, cities, and towns, as well as for the Commonwealth of Puerto Rico and its municipios. Demographic components of population change (births, deaths, and migration) are produced at the national, state, and county levels of geography. Additionally, housing unit estimates are produced for the nation, states, and counties. PEP annually utilizes current data on births, deaths, and migration to calculate population change since the most recent decennial census and produce a time series of estimates of population, demographic components of change, and housing units. The annual time series of estimates begins with the most recent decennial census data and extends to the vintage year. As each vintage of estimates includes all years since the most recent decennial census, the latest vintage of data available supersedes all previously-produced estimates for those dates."

## Usage

```
get_estimates(
 geography = c("us", "region", "division", "state", "county", "county subdivision",
    "place/balance (or part)", "place", "consolidated city", "place (or part)",
    "metropolitan statistical area/micropolitan statistical area", "cbsa",
    "metropolitan division", "combined statistical area"),
  product = NULL,
  variables = NULL,
  breakdown = NULL,
  breakdown_labels = FALSE,
  vintage = 2022,
  year = vintage,
  state = NULL,
  county = NULL,
  time_series = FALSE,
  output = "tidy",
  geometry = FALSE,
```

```
  keep_geo_vars = FALSE,
  shift_geo = FALSE,
  key = NULL,
  show_call = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| geography | The geography of your data. Available geographies for the most recent data vintage are listed here. "cbsa" may be used an alias for "metropolitan statistical area/micropolitan statistical area". |
| product | The data product (optional). "population", "components" "housing", and "characteristics" are supported.<br>For 2020 and later, the only supported product is "characteristics". |
| variables | A character string or vector of character strings of requested variables. For years 2020 and later, use variables = "all" to request all available variables. |
| breakdown | The population breakdown used when product = "characteristics". Acceptable values are "AGEGROUP", "RACE", "SEX", and "HISP", for Hispanic/Not Hispanic. These values can be combined in a vector, returning population estimates in the value column for all combinations of these breakdowns. For years 2020 and later, "AGE" is also available for single-year age when using geography = "state". |
| breakdown_labels | Whether or not to label breakdown elements returned when product = "characteristics". Defaults to FALSE. |
| vintage | It is recommended to use the most recent vintage available for a given decennial series (so, year = 2019 for the 2010s, and year = 2023 for the 2020s). Will default to 2022 until the full PEP for 2023 is released. |
| year | The data year (defaults to the vintage requested). Use time_series = TRUE to access time-series estimates. |
| state | The state for which you are requesting data. State names, postal codes, and FIPS codes are accepted. Defaults to NULL. |
| county | The county for which you are requesting data. County names and FIPS codes are accepted. Must be combined with a value supplied to 'state'. Defaults to NULL. |
| time_series | If TRUE, the function will return a time series of observations back to the decennial Census of 2010. The returned column is either "DATE", representing a particular estimate date, or "PERIOD", representing a time period (e.g. births between 2016 and 2017), and contains integers representing those values. Integer to date or period mapping is available at https://www.census.gov/data/developers/data-sets/popest-popproj/popest/popest-vars/2019.html. |
| output | One of "tidy" (the default) in which each row represents an enumeration unit-variable combination, or "wide" in which each row represents an enumeration unit and the variables are in the columns. |

| | |
|---|---|
| geometry | if FALSE (the default), return a regular tibble of ACS data. if TRUE, uses the tigris package to return an sf tibble with simple feature geometry in the 'geometry' column. |
| keep_geo_vars | if TRUE, keeps all the variables from the Census shapefile obtained by tigris. Defaults to FALSE. |
| shift_geo | (deprecated) if TRUE, returns geometry with Alaska and Hawaii shifted for thematic mapping of the entire US. As of May 2021, we recommend using `tigris::shift_geometry()` instead. |
| key | Your Census API key. Obtain one at https://api.census.gov/data/key_signup.html. Can be stored in your .Renviron with `census_api_key("YOUR KEY", install = TRUE)` |
| show_call | if TRUE, display call made to Census API. This can be very useful in debugging and determining if error messages returned are due to tidycensus or the Census API. Copy to the API call into a browser and see what is returned by the API directly. Defaults to FALSE. |
| ... | other keyword arguments |

## Details

`get_estimates()` requests data from the Population Estimates API for years 2019 and earlier; however the Population Estimates are no longer supported on the API as of 2020. For recent years, `get_estimates()` reads a flat file from the Census website and parses it. This means that arguments and output for 2020 and later datasets may differ slightly from datasets acquired for 2019 and earlier.

As of April 2022, variables available for 2020 and later datasets are as follows: ESTIMATESBASE, POPESTIMATE, NPOPCHG, BIRTHS, DEATHS, NATURALCHG, INTERNATIONALMIG, DOMESTICMIG, NETMIG, RESIDUAL, GQESTIMATESBASE, GQESTIMATES, RBIRTH, RDEATH, RNATURALCHG, RINTERNATIONALMIG, RDOMESTICMIG, and RNETMIG.

## Value

A tibble, or sf tibble, of population estimates data

## See Also

https://www.census.gov/programs-surveys/popest/about.html

---

`get_flows` *Obtain data and feature geometry for American Community Survey Migration Flows*

---

## Description

Obtain data and feature geometry for American Community Survey Migration Flows

## Usage

```
get_flows(
  geography,
  variables = NULL,
  breakdown = NULL,
  breakdown_labels = FALSE,
  year = 2018,
  output = "tidy",
  state = NULL,
  county = NULL,
  msa = NULL,
  geometry = FALSE,
  key = NULL,
  moe_level = 90,
  show_call = FALSE
)
```

## Arguments

| | |
|---|---|
| geography | The geography of your requested data. Possible values are "county", "county subdivision", and "metropolitan statistical area". MSA data is only available beginning with the 2009-2013 5-year ACS. |
| variables | Character string or vector of character strings of variable names. By default, get_flows() returns the GEOID and names of the geographies as well as the number of people who moved in, out, and net movers of each geography ("MOVEDIN", "MOVEDOUT", "MOVEDNET"). If additional variables are specified, they are pulled in addition to the default variables. The names of additional variables can be found in the Census Migration Flows API documentation at https://api.census.gov/data/2018/acs/flows/variables.html. |
| breakdown | A character vector of the population breakdown characteristics to be crossed with migration flows data. For datasets between 2006-2010 and 2011-2015, selected demographic characteristics such as age, race, employment status, etc. are available. Possible values are "AGE", "SEX", "RACE", "HSGP", "REL", "HHT", "TEN", "ENG", "POB", "YEARS", "ESR", "OCC", "WKS", "SCHL", "AHINC", "APINC", and "HISP_ORIGIN". For more information and to see which characteristics are available in each year, visit the Census Migration Flows documentation at https://www.census.gov/data/developers/data-sets/acs-migration-flows.html. Note: not all characteristics are available in all years. |
| breakdown_labels | Whether or not to add columns with labels for the breakdown characteristic codes. Defaults to FALSE. |
| year | The year, or endyear, of the ACS sample. The Migration Flows API is available for 5-year ACS samples from 2010 to 2018. Defaults to 2018. |
| output | One of "tidy" (the default) in which each row represents an enumeration unit-variable combination, or "wide" in which each row represents an enumeration unit and the variables are in the columns. |

| | |
|---|---|
| state | An optional vector of states for which you are requesting data. State names, postal codes, and FIPS codes are accepted. When requesting county subdivision data, you must specify at least one state. |
| county | The county for which you are requesting data. County names and FIPS codes are accepted. Must be combined with a value supplied to 'state'. |
| msa | The metropolitan statistical area for which you are requesting data. Specify a single value or a vector of values to get data for more than one MSA. Numeric or character MSA GEOIDs are accepted. When specifying MSAs, geography must be set to "metropolitan statistical area" and state and county must be NULL. |
| geometry | if FALSE (the default), return a tibble of ACS Migration Flows data. If TRUE, return an sf object with the centroids of both origin and destination as sfc_POINT columns. The origin point feature is returned in a column named centroid1 and is the active geometry column in the sf object. The destination point feature is returned in the centroid2 column. |
| key | Your Census API key. Obtain one at https://api.census.gov/data/key_signup.html |
| moe_level | The confidence level of the returned margin of error. One of 90 (the default), 95, or 99. |
| show_call | if TRUE, display call made to Census API. This can be very useful in debugging and determining if error messages returned are due to tidycensus or the Census API. Copy to the API call into a browser and see what is returned by the API directly. Defaults to FALSE. |

## Value

A tibble or sf tibble of ACS Migration Flows data

## Examples

```
## Not run:
get_flows(
  geography = "county",
  state = "VT",
  county = c("Washington", "Chittenden")
  )

get_flows(
  geography = "county subdivision",
  breakdown = "RACE",
  breakdown_labels = TRUE,
  state = "NY",
  county = "Westchester",
  output = "wide",
  year = 2015
  )

get_flows(
   geography = "metropolitan statistical area",
```

```
  variables = c("POP1YR", "POP1YRAGO"),
  geometry = TRUE,
  output = "wide",
  show_call = TRUE
 )

## End(Not run)
```

---

`get_pop_groups` *Get available population groups for a given Decennial Census year and summary file*

---

## Description

Get available population groups for a given Decennial Census year and summary file

## Usage

```
get_pop_groups(year, sumfile)
```

## Arguments

| | |
|---|---|
| `year` | The decennial Census year; one of 2000, 2010, or 2020. |
| `sumfile` | The summary file. Available summary files are `"ddhca"`, `"sf2"`, and `"sf4"`. |

## Value

A tibble containing codes (to be used with the `pop_group` argument of `get_decennial()`) and descriptive names.

---

`get_pums` *Load data from the American Community Survey Public Use Microdata Series API*

---

## Description

Load data from the American Community Survey Public Use Microdata Series API

## Usage

```
get_pums(
  variables = NULL,
  state = NULL,
  puma = NULL,
  year = 2023,
  survey = "acs5",
  variables_filter = NULL,
  rep_weights = NULL,
  recode = FALSE,
  return_vacant = FALSE,
  show_call = FALSE,
  key = NULL
)
```

## Arguments

| | |
|---|---|
| variables | A vector of variables from the PUMS API. Use `View(pums_variables)` to browse variable options. |
| state | A state, or vector of states, for which you would like to request data. The entire US can be requested with `state = "all"` - though be patient with the data download! |
| puma | A vector of PUMAs from a single state, for which you would like to request data. To get data from PUMAs in more than one state, specify a named vector of state/PUMA pairs and set `state = "multiple"`. |
| year | The data year of the 1-year ACS sample or the endyear of the 5-year sample. Defaults to 2023. Please note that 1-year data for 2020 is not available in tidycensus, so users requesting 1-year data should supply a different year. |
| survey | The ACS survey; one of either `"acs1"` or `"acs5"` (the default). |
| variables_filter | A named list of filters you'd like to return from the PUMS API. For example, passing `list(AGE = 25:50, SEX = 1)` will return only males aged 25 to 50 in your output dataset. Defaults to `NULL`, which returns all records. If a housing-only dataset is required, use `list(SPORDER = 1)` to only return householder records (taking care in your analysis to use the household weight `WGTP`). |
| rep_weights | Whether or not to return housing unit, person, or both housing and person-level replicate weights for calculation of standard errors; one of `"person"`, `"housing"`, or `"both"`. |
| recode | If TRUE, recodes variable values using Census data dictionary and creates a new `*_label` column for each variable that is recoded. Available for 2017 - 2022 data. Defaults to FALSE. |
| return_vacant | If TRUE, makes a separate request to the Census API to retrieve microdata for vacant housing units, which are handled differently in the API as they do not have person-level characteristics. All person-level columns in the returned dataset will be populated with NA for vacant housing units. Defaults to FALSE. |

show_call — If TRUE, display call made to Census API. This can be very useful in debugging and determining if error messages returned are due to tidycensus or the Census API. Copy to the API call into a browser and see what is returned by the API directly. Defaults to FALSE.

key — Your Census API key. Obtain one at https://api.census.gov/data/key_signup.html

## Value

A tibble of microdata from the ACS PUMS API.

## Examples

```
## Not run:
get_pums(variables = "AGEP", state = "VT")
get_pums(variables = "AGEP", state = "multiple", puma = c("UT" = 35008, "NV" = 00403))
get_pums(variables = c("AGEP", "ANC1P"), state = "VT", recode = TRUE)
get_pums(variables = "AGEP", state = "VT", survey = "acs1", rep_weights = "person")

## End(Not run)
```

---

# interpolate_pw *Use population-weighted areal interpolation to transfer information from one set of shapes to another*

---

## Description

A common use-case when working with time-series small-area Census data is to transfer data from one set of shapes (e.g. 2010 Census tracts) to another set of shapes (e.g. 2020 Census tracts). Population-weighted interpolation is one such solution to this problem that takes into account the distribution of the population within a Census unit to intelligently transfer data between incongruent units.

## Usage

```
interpolate_pw(
  from,
  to,
  to_id = NULL,
  extensive,
  weights,
  weight_column = NULL,
  weight_placement = c("surface", "centroid"),
  crs = NULL
)
```

## Arguments

| | |
|---|---|
| from | The spatial dataset from which numeric attributes will be interpolated to target zones. By default, all numeric columns in this dataset will be interpolated. |
| to | The target geometries (zones) to which numeric attributes will be interpolated. |
| to_id | (optional) An ID column in the target dataset to be retained in the output. For data obtained with tidycensus, this will be "GEOID" by convention. If NULL, the output dataset will include a column id that uniquely identifies each row. |
| extensive | if TRUE, return weighted sums; if FALSE, return weighted means. |
| weights | An input spatial dataset to be used as weights. If the dataset is not of geometry type POINT, it will be converted to points by the function with sf::st_point_on_surface(). For US-based applications, this will commonly be a Census block dataset obtained with the tigris or tidycensus packages. |
| weight_column | (optional) a column in weights used for weighting in the interpolation process. Typically this will be a column representing the population (or other weighting metric, like housing units) of the input weights dataset. If NULL (the default), each feature in weights is given an equal weight of 1. |
| weight_placement | (optional) One of "surface", where weight polygons are converted to points on polygon surfaces with sf::st_point_on_surface(), or "centroid", where polygon centroids are used instead with sf::st_centroid(). Defaults to "surface". This argument is not necessary if weights are already of geometry type POINT. |
| crs | (optional) The EPSG code of the output projected coordinate reference system (CRS). Useful as all input layers (from, to, and weights) must share the same CRS for the function to run correctly. |

## Details

The approach implemented here is based on Esri's data apportionment algorithm, in which an "apportionment layer" of points (referred to here as the weights) is used to determine how to weight areas of overlap between origin and target zones. Users must supply a "from" dataset as an sf object (the dataset from which numeric columns will be interpolated) and a "to" dataset, also of class sf, that contains the target zones. A third sf object, the "weights", may be an object of geometry type POINT or polygons from which points will be derived using sf::st_point_on_surface().

An intersection is computed between from and to, and a spatial join is computed between the intersection layer and the weights layer, represented as points. A specified weight_column in weights will be used to determine the relative influence of each point on the allocation of values between from and to; if no weight column is specified, all points will be weighted equally.

The extensive parameter (logical) should reflect the values being interpolated correctly. If TRUE, the function returns a weighted sum for each zone. If FALSE, a weighted mean will be returned. For Census data, extensive = TRUE should be used for transferring counts / estimated counts between zones. Derived metrics (e.g. population density, percentages, etc.) should use extensive = FALSE. Margins of error in the ACS will not be transferred correctly with this function, so please use with caution.

## Value

A dataset of class sf with the geometries and an ID column from `to` (the target shapes) but with numeric attributes of `from` interpolated to those shapes.

## Examples

```
## Not run:
# Example: interpolating work-from-home from 2011-2015 ACS
# to 2020 shapes
library(tidycensus)
library(tidyverse)
library(tigris)
options(tigris_use_cache = TRUE)

wfh_15 <- get_acs(
  geography = "tract",
  variables = "B08006_017",
  year = 2015,
  state = "AZ",
  county = "Maricopa",
  geometry = TRUE
) %>%
select(estimate)

wfh_20 <- get_acs(
  geography = "tract",
  variables = "B08006_017",
  year = 2020,
  state = "AZ",
  county = "Maricopa",
  geometry = TRUE
 )

maricopa_blocks <- blocks(
  "AZ",
  "Maricopa",
  year = 2020
)

wfh_15_to_20 <- interpolate_pw(
  from = wfh_15,
  to = wfh_20,
  to_id = "GEOID",
  weights = maricopa_blocks,
  weight_column = "POP20",
  crs = 26949,
  extensive = TRUE
)


## End(Not run)
```

load_variables *Load variables from a decennial Census or American Community Survey dataset to search in R*

## Description

Finding the right variables to use with `get_decennial()` or `get_acs()` can be challenging; `load_variables()` attempts to make this easier for you. Choose a year and a dataset to search for variables; those variables will be loaded from the Census website as an R data frame. It is recommended that RStudio users use the `View()` function to interactively browse and filter these variables to find the right variables to use.

## Usage

```
load_variables(
  year,
 dataset = c("sf1", "sf2", "sf3", "sf4", "pl", "dhc", "dp", "ddhca", "ddhcb", "sdhc",
  "as", "gu", "mp", "vi", "acsse", "dpas", "dpgu", "dpmp", "dpvi", "dhcvi", "dhcgu",
    "dhcvi", "dhcas", "acs1", "acs3", "acs5", "acs1/profile", "acs3/profile",
  "acs5/profile", "acs1/subject", "acs3/subject", "acs5/subject", "acs1/cprofile",
  "acs5/cprofile", "sf2profile", "sf3profile", "sf4profile", "aian", "aianprofile",
  "cd110h", "cd110s", "cd110hprofile", "cd110sprofile", "sldh", "slds", "sldhprofile",
    "sldsprofile", "cqr",
     "cd113", "cd113profile", "cd115", "cd115profile",
    "cd116", "plnat", "cd118"),
  cache = FALSE
)
```

## Arguments

| | |
|---|---|
| year | The year for which you are requesting variables. Either the year or endyear of the decennial Census or ACS sample. 5-year ACS data is available from 2009 through 2020. 1-year ACS data is available from 2005 through 2021, with the exception of 2020. |
| dataset | The dataset name as used on the Census website. See the Details in this documentation for a full list of dataset names. |
| cache | Whether you would like to cache the dataset for future access, or load the dataset from an existing cache. Defaults to FALSE. |

## Details

`load_variables()` returns three columns by default: `name`, which is the Census ID code to be supplied to the `variables` parameter in `get_decennial()` or `get_acs()`; `label`, which is a detailed description of the variable; and `concept`, which provides information about the table that a given variable belongs to. For 5-year ACS detailed tables datasets, a fourth column, `geography`, tells you the smallest geography at which a given variable is available.

Datasets available are as follows: "sf1", "sf2", "sf3", "sf4", "pl", "dhc", "dp", "dhca", "ddhca", "ddhcb", "sdhc", "as", "gu", "mp", "vi", "acsse", "dpas", "dpgu", "dpmp", "dpvi", "dhcvi", "dhcgu", "dhcvi", "dhcas", "acs1", "acs3", "acs5", "acs1/profile", "acs3/profile", "acs5/profile", "acs1/subject", "acs3/subject", "acs5/subject", "acs1/cprofile", "acs5/cprofile", "sf2profile", "sf3profile", "sf4profile", "aian", "aianprofile", "cd110h", "cd110s", "cd110hprofile", "cd110sprofile", "sldh", "slds", "sldh-profile", "sldsprofile", "cqr", "cd113", "cd113profile", "cd115", "cd115profile", "cd116", "cd118", and "plnat".

## Value

A tibble of variables from the requested dataset.

## Examples

```
## Not run:
v15 <- load_variables(2015, "acs5", cache = TRUE)
View(v15)

## End(Not run)
```

---

# mig_recodes *Dataset with Migration Flows characteristic recodes*

---

## Description

Built-in dataset for Migration Flows code label lookup.

- `characteristic`: Characteristic variable name
- `code`: Characteristic calue code
- `desc`: Characteristic calue label
- `ordered`: Whether or not recoded value should be ordered factor

## Usage

```
data(mig_recodes)
```

## Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 120 rows and 4 columns.

## Details

Dataset with Migration Flows characteristic recodes

Built-in dataset that is created from the Migration Flows API documentation. This dataset contains labels for the coded values returned by the Census API and is used when `breakdown_labels = TRUE` in `get_flows`.

## moe_product *Calculate the margin of error for a derived product*

### Description

Calculate the margin of error for a derived product

### Usage

```
moe_product(est1, est2, moe1, moe2)
```

### Arguments

| | |
|---|---|
| est1 | The first factor in the multiplication equation (an estimate) |
| est2 | The second factor in the multiplication equation (an estimate) |
| moe1 | The margin of error of the first factor |
| moe2 | The margin of error of the second factor |

### Value

A margin of error for a derived product

## moe_prop *Calculate the margin of error for a derived proportion*

### Description

Calculate the margin of error for a derived proportion

### Usage

```
moe_prop(num, denom, moe_num, moe_denom)
```

### Arguments

| | |
|---|---|
| num | The numerator involved in the proportion calculation (an estimate) |
| denom | The denominator involved in the proportion calculation (an estimate) |
| moe_num | The margin of error of the numerator |
| moe_denom | The margin of error of the denominator |

### Value

A margin of error for a derived proportion

## moe_ratio *Calculate the margin of error for a derived ratio*

### Description

Calculate the margin of error for a derived ratio

### Usage

```
moe_ratio(num, denom, moe_num, moe_denom)
```

### Arguments

| | |
|---|---|
| num | The numerator involved in the ratio calculation (an estimate) |
| denom | The denominator involved in the ratio calculation (an estimate) |
| moe_num | The margin of error of the numerator |
| moe_denom | The margin of error of the denominator |

### Value

A margin of error for a derived ratio

## moe_sum *Calculate the margin of error for a derived sum*

### Description

Generates a margin of error for a derived sum. The function requires a vector of margins of error involved in a sum calculation, and optionally a vector of estimates associated with the margins of error. If the associated estimates are not specified, the user risks inflating the derived margin of error in the event of multiple zero estimates. It is recommended to inspect your data for multiple zero estimates before using this function and setting the inputs accordingly.

### Usage

```
moe_sum(moe, estimate = NULL, na.rm = FALSE)
```

### Arguments

| | |
|---|---|
| moe | A vector of margins of error involved in the sum calculation |
| estimate | A vector of estimates, the same length as moe, associated with the margins of error |
| na.rm | A logical value indicating whether missing values (including NaN) should be removed |

## Value

A margin of error for a derived sum

## See Also

https://www2.census.gov/programs-surveys/acs/tech_docs/accuracy/MultiyearACSAccuracyofData2015.pdf

---

`pums_variables` *Dataset with PUMS variables and codes*

---

## Description

Built-in dataset for variable name and code label lookup. To access the data directly, issue the command `data(pums_variables)`.

- `survey`: acs1 or acs5
- `year`: Year of data. For 5-year data, last year in range.
- `var_code`: Variable name
- `var_label`: Variable label
- `data_type`: chr or num
- `level`: housing or person
- `val_min`: For numeric variables, the minimum value
- `val_max`: For numeric variables, the maximum value
- `val_label`: Value label
- `recode`: Use labels to recode values
- `val_length`: Length of value returned
- `val_na`: Value of NA value returned by API (if known)

## Usage

```
data(pums_variables)
```

## Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 69400 rows and 12 columns.

## Details

Dataset with PUMS variables and codes

Built-in dataset that is created from the Census PUMS data dictionaries. Use this dataset to lookup the names of variables to use in `get_pums`. This dataset also contains labels for the coded values returned by the Census API and is used when `recode = TRUE` in `get_pums`.

Because variable names and codes change from year to year, you should filter this dataset for the survey and year of interest. NOTE: 2017 - 2019 and 2021 acs1 and 2017 - 2021 acs5 variables are available.

## significance *Evaluate whether the difference in two estimates is statistically significant.*

### Description

Evaluate whether the difference in two estimates is statistically significant.

### Usage

```
significance(est1, est2, moe1, moe2, clevel = 0.9)
```

### Arguments

| | |
|---|---|
| est1 | The first estimate. |
| est2 | The second estimate |
| moe1 | The margin of error of the first estimate |
| moe2 | The margin of error of the second estimate |
| clevel | The confidence level. May by 0.9, 0.95, or 0.99 |

### Value

TRUE if the difference is statistically signifiant, FALSE otherwise.

### See Also

https://www.census.gov/content/dam/Census/library/publications/2018/acs/acs_general_handbook_2018_ch07.pdf

## state_laea *State geometry with Alaska and Hawaii shifted and re-scaled*

### Description

Built-in dataset for use with `shift_geo = TRUE`

Dataset of US states with Alaska and Hawaii shifted and re-scaled

### Usage

```
data(state_laea)

data(state_laea)
```

### Format

An object of class `sf` (inherits from `data.frame`) with 51 rows and 2 columns.

### Details

Dataset with state geometry for use when shifting Alaska and Hawaii

Built-in dataset for use with the `shift_geo` parameter, with the continental United States in a Lambert azimuthal equal area projection and Alaska and Hawaii shifted and re-scaled. The data were originally obtained from the albersusa R package (https://github.com/hrbrmstr/albersusa).

---

## summary_files *Identify summary files for a given decennial Census year*

---

### Description

Identify summary files for a given decennial Census year

### Usage

```
summary_files(year)
```

### Arguments

| | |
|---|---|
| `year` | The year of the decennial Census |

### Value

A vector of available summary files for a given decennial Census year. To access data for a given summary file, supply the desired value to the `sumfile` parameter in `get_decennial()`.

---

## to_survey *Convert a data frame returned by get_pums() to a survey object*

---

### Description

This helper function takes a data frame returned by `get_pums` and converts it to a tbl_svy from the srvyr `as_survey` package or a svyrep.design object from the `svrepdesign` package. You can then use functions from the srvyr or survey to calculate weighted estimates with replicate weights included to provide accurate standard errors.

### Usage

```
to_survey(
  df,
  type = c("person", "housing"),
  class = c("srvyr", "survey"),
  design = "rep_weights"
)
```

## Arguments

| | |
|---|---|
| df | A data frame with PUMS person or housing weight variables, most likely returned by get_pums. |
| type | Whether to use person or housing-level weights; either "housing" or "person" (the default). |
| class | Whether to convert to a srvyr or survey object; either "survey" or "srvyr" (the default). |
| design | The survey design to use when creating a survey object. Currently the only option is "rep_weights". |

## Value

A tbl_svy or svyrep.design object.

## Examples

```
## Not run:
pums <- get_pums(variables = "AGEP", state = "VT", rep_weights = "person")
pums_design <- to_survey(pums, type = "person", class = "srvyr")
survey::svymean(~AGEP, pums_design)

## End(Not run)
```

# Index

∗ **datasets**
acs5_geography, 2
county_laea, 7
fips_codes, 7
mig_recodes, 24
pums_variables, 27
state_laea, 28

acs5_geography, 2
as_dot_density, 3
as_survey, *29*

census_api_key, 5
check_ddhca_groups, 6
counties, *8*
county_laea, 7

fips_codes, 7

get_acs, 8
get_decennial, 10
get_estimates, 13
get_flows, 15, *24*
get_pop_groups, 18
get_pums, 18, *27*, *29*, *30*

interpolate_pw, 20

load_variables, 23

mig_recodes, 24
moe_product, 25
moe_prop, 25
moe_ratio, 26
moe_sum, 26

pums_variables, 27

significance, 28
state_laea, 28
summary_files, 29
svrepdesign, *29*

to_survey, 29