Package: getACS (via r-universe)

October 21, 2024

Type Package

Title Help Wrangling American Community Survey Data from tidycensus

Version 0.1.1.9003

Maintainer Eli Pousson <eli.pousson@gmail.com>

Description A package with helper functions for working with Census data downloaded with the tidycensus package.

License MIT + file LICENSE

URL https://elipousson.github.io/getACS/

BugReports https://github.com/elipousson/getACS/issues

Depends R (>= 2.10)

Imports cli, dplyr, ggplot2, glue, gt, knitr, lifecycle, purrr, rappdirs, rlang (>= 1.1.0), scales, sf, stats, stringr, tibble, tidycensus, tidyr, tidyselect, tigris (>= 2.1.1), vctrs, withr

Suggests forcats, readr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Remotes elipousson/tigris

Repository https://elipousson.r-universe.dev

RemoteUrl https://github.com/elipousson/getACS

RemoteRef HEAD

RemoteSha 4139a8875c6d6b812d64ce49880f22ab0f0dc1e7

Contents

`acs_survey` *Assorted helpers for ACS survey types and labels*

Description

These simple functions allow validating ACS survey options, getting comparable years for time series analysis, and creating standard labels.

Usage

```
acs_survey_match(survey = "acs5", error_call = caller_env())

acs_survey_sample(survey = "acs5")

acs_survey_ts(survey = "acs5", year = 2022, call = caller_env())

acs_survey_label(
  survey = "acs5",
  year = 2022,
  pattern = "{year_start}-{year} ACS {sample}-year Estimates",
  prefix = ""
)

acs_survey_label_table(
  survey = "acs5",
  year = 2022,
  prefix = "",
  table = NULL,
  table_label = "Table",
  sep = ", ",
  and = " and ",
  before = "",
  after = before,
  end = ".",
  oxford_comma = TRUE
)
```

Arguments

survey	ACS survey, "acs5", "acs3", or "acs1".
error_call	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.
year	Based on the year and survey, `acs_survey_ts()` returns a vector of years for non-overlapping ACS samples to allow comparison.
call	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.
pattern	Pattern passed to `glue::glue()`. Allows use of the `year_start` variable which is the earliest year for a survey sample specified by the survey parameter.
prefix	Text to insert before ACS survey label.
table	One or more table IDs to include in label or source note.
table_label	Label to use when referring to table or tables. A "s" is appended to the end of the table_label if tables is more than length 1.
sep	Separator to be inserted between words.

and	Character string to be prepended to the last word.
before, after	A character string to be added before/after each word.
end	A character string appended to the end of the full label. Defaults to ".".
oxford_comma	Whether to insert the separator between the last two elements in the list.

Examples

```
acs_survey_match("acs1")

acs_survey_sample("acs3")

acs_survey_ts("acs5", 2020)

acs_survey_label()

acs_survey_label_table(table = c("B19013", "B01003"))
```

---

acs_table_race_iteration

Append a set of race iteration codes to an ACS table ID

Description

acs_table_race_iteration() uses the race_iteration reference data to create or validate race iteration codes and create race iteration table IDs.

Usage

```
acs_table_race_iteration(table, codes = NULL, error_call = caller_env())
```

Arguments

table	An ACS table ID string.
codes	Character vector of race iteration codes to return. If NULL (default), codes is set to c("", race_iteration[["code"]]).
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

A character vector of variable ID values for a single table.

See Also

acs_table_variables()

Examples

```
acs_table_race_iteration("B25003")
```

`acs_table_variables` *Convert an ACS table ID to a set of variable ID values*

Description

`acs_table_variables()` helps to make a vector of variable ID values based on a table ID string. The returned variable IDs use the format returned by `tidycensus::get_acs()`, e.g. "{table_id}_{line_number}" where the line_number is a width 3 string prefixed by "0". If variables is `NULL`, the function calls `get_acs_metadata()` with `metadata = "column"` and returns all available variables for the table for the supplied year and survey. Note that the `sep` and `width` parameters should *not* be changed if you are working with data from the `\{tidycensus\}` package.

Usage

```
acs_table_variables(
  table = NULL,
  variables = NULL,
  data = NULL,
  survey = "acs5",
  year = 2022,
  sep = "_",
  width = 3,
  error_call = caller_env()
)
```

Arguments

`table`	An ACS table ID string.
`variables`	A numeric vector corresponding to the line number of the variables.
`data`	If data is provided and table is `NULL`, table is set based on the unique values in the "table_id" column of data. If data contains more than one table_id value, the function will error
`survey`	Survey, "acs5", "acs3", or "acs1".
`year`	Sample year (between 2006 and 2022).
`sep`	A separator character between the table ID string and variable ID values.
`width`	Variable ID suffix width.
`error_call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

A character vector of variable ID values for a single table.

See Also

acs_table_race_iteration()

Examples

```
acs_table_variables(table = "B15003")

acs_table_variables(table = "B15003", variables = c(1:5))
```

---

assign_acs_reliability

Add columns for the coefficient of variation and reliability category

---

Description

assign_acs_reliability() tests the reliability of ACS estimate values based on the assigned MOE level and adds columns to the output with the reliability information.

Usage

```
assign_acs_reliability(
  data,
  value_col = "estimate",
  moe_col = "moe",
  moe_level = 90,
  type = c("census", "esri"),
  digits = 2,
  cv_col = "cv",
  reliability_col = "reliability"
)
```

Arguments

data	A data frame with a column of estimate values. Typically created with tidycensus::get_acs() or a function in this package such as get_acs_tables() or get_acs_geographies().
value_col, moe_col	Value and margin of error column names (default to "estimate" and "moe").
moe_level	The confidence level of the margin of error. Defaults to 90 (which is the same default as tidycensus::get_acs()).
type	Type of reliability rating to assign. Either "census" (default) or "esri". In both cases, the added reliability column values are "high", "medium", or "low".
digits	Number of digits to use for values in the coefficient of variation column. Passed to base::round().
cv_col	Coefficient of variation column name. Defaults to "cv".
reliability_col	Reliability category column name. Defaults to "reliability".

Value

A data frame with an added columns using the names assigned to `cv_col` and `reliability_col`

`collapse_acs_variables` *Collapse variables into a new label column using* `forcats::fct_collapse()`

Description

`collapse_acs_variables()` uses `forcats::fct_collapse()` to aggregated variables while creating a new label column. Other variables are retained in list columns of unique values. The aggregated values for `perc_moe` may not be accurate after transformation with this function. To group by additional variables, passed a grouped data frame to data and set `.add = TRUE`.

Usage

```
collapse_acs_variables(
  data,
  ...,
  other_level = NULL,
  name_col = "NAME",
  variable_col = "variable",
  label_col = "label",
  value_col = "estimate",
  moe_col = "moe",
  moe_level = 90,
  reliability = FALSE,
  na.rm = TRUE,
  na_zero = TRUE,
  digits = 2,
  .add = FALSE,
  extensive = TRUE
)
```

Arguments

`data`	ACS data frame input.
`...`	<dynamic-dots> A series of named character vectors. The levels in each vector will be replaced with the name.
`other_level`	Value of level used for "other" values. Always placed at end of levels.
`name_col`	Name column name, Default: 'NAME'
`variable_col`	Variable column name, Default: 'variable'
`label_col`	Label column name, Default: 'label'. Label is a factor column added to the returned data frame.

value_col, moe_col
: Value and margin of error column names (default to "estimate" and "moe").

moe_level
: The confidence level of the margin of error. Defaults to 90 (which is the same default as tidycensus::get_acs()).

reliability
: If TRUE, use assign_acs_reliability() to assign a reliability value to estimate values based on the specified moe_level.

na.rm
: Passed to sum(), Default: TRUE

na_zero
: If TRUE, and the collapsed sum of a MOE is 0, replaced MOE value with NA. This is beneficial for percent estimates with the margin of error falls below 1% and is rounded to 0 with the default number of digits.

digits
: Passed to round(), Default: 2

.add
: When FALSE, the default, group_by() will override existing groups. To add to the existing groups, use .add = TRUE.

  This argument was previously called add, but that prevented creating a new grouping variable called add, and conflicts with our naming conventions.

extensive
: Must be TRUE. If FALSE (not currently supported), summarize collapsed variables using a weighted mean.

See Also

forcats::fct_collapse(), camiller::add_grps()

Examples

```
## Not run:
if (interactive()) {
  edu_data <- get_acs_tables(
    "county",
    table = "B15003",
    state = "MD",
    county = "Baltimore city"
  )

  table_vars <- acs_table_variables("B15003")

  collapse_acs_variables(
    edu_data,
    "Total" = table_vars[1],
    "5th Grade or less" = table_vars[5:9],
    "6th to 8th Grade" = table_vars[10:12],
    "9th to 11th Grade" = table_vars[13:15],
    other_level = "Other"
  )
}

## End(Not run)
```

`fmt_acs_county` *Format place names or column titles in a gt table or data frame with ACS data*

Description

`fmt_acs_county()` is helpful for stripping the state name from county-level ACS data and `fmt_acs_minutes()` does the same for a column with a duration (e.g. commute times). If data is not a `gt_tbl` object, both function can use `dplyr::mutate()` to transform a standard data frame.

Usage

```
fmt_acs_county(
  data,
  state = NULL,
  pattern = ", {state}",
  replacement = "",
  name_col = "NAME",
  columns = all_of(name_col),
  ...
)

fmt_acs_minutes(
  data,
  pattern = "[:space:]minutes$",
  replacement = "",
  column_title_col = "column_title",
  columns = all_of(column_title_col),
  ...
)
```

Arguments

`data`	*The gt table data object* `obj:<gt_tbl>` // **required** This is the **gt** table object that is commonly created through use of the `gt()` function.
`state`	State name. Required if state is included in pattern.
`pattern`	Passed to `glue::glue()` and `stringr::str_replace()` for `fmt_acs_county()` or just to `stringr::str_replace()` by `fmt_acs_minutes()`. Defaults to `", {state}"` which strips the state name from a column of county-level name values or `"[:space:]minutes$"` which strips the trailing text for minutes.
`replacement`	Passed to `stringr::str_replace()`. Defaults to `""`.
`name_col`	Name for column with place name values. Defaults to "NAME"

columns *Columns to target*
<column-targeting expression> // *default:* everything()
Can either be a series of column names provided in c(), a vector of column indices, or a select helper function (e.g. starts_with(), ends_with(), contains(), matches(), num_range() and everything()).

... Arguments passed on to gt::fmt

rows *Rows to target*
<row-targeting expression> // *default:* everything()
In conjunction with columns, we can specify which of their rows should undergo formatting. The default everything() results in all rows in columns being formatted. Alternatively, we can supply a vector of row captions within c(), a vector of row indices, or a select helper function (e.g. starts_with(), ends_with(), contains(), matches(), num_range(), and everything()).
We can also use expressions to filter down to the rows we need (e.g., [colname_1] > 100 & [colname

compat *Formatting compatibility*
vector<character> // *default:* NULL (optional)
An optional vector that provides the compatible classes for the formatting. By default this is NULL.

fns *Formatting functions*
function|list of functions // **required**
Either a single formatting function or a named list of functions.

column_title_col
Column title column.

---

fmt_acs_estimate *Format estimate and margin of error columns in a gt table*

Description

fmt_acs_estimate() formats estimate and margin of error columns for a gt table created with ACS data. fmt_acs_percent() does the same for the perc_estimate and perc_moe columns calculated by join_acs_percent(). Both functions are used internally by gt_acs().

Usage

```
fmt_acs_estimate(
  gt_object,
  col_est = "estimate",
  col_moe = "moe",
  columns = NULL,
  col_labels = "Est.",
  spanner = NULL,
  decimals = 0,
  use_seps = TRUE,
  ...,
  call = caller_env()
```

```
)

fmt_acs_percent(
  gt_object,
  col_est = "perc_estimate",
  col_moe = "perc_moe",
  columns = NULL,
  col_labels = "% share",
  spanner = NULL,
  decimals = 0,
  use_seps = TRUE,
  ...,
  call = caller_env()
)

cols_label_ext(
  gt_object,
  columns = NULL,
  col_labels = NULL,
  call = caller_env()
)
```

Arguments

gt_object	A gt object.
col_est, col_moe	Column names for the estimate and margin of error values in the table data.
columns	If NULL (default), columns is set to c(col_est, col_moe). If spanner is NULL, columns is passed to cols_merge_uncert_ext() and must be a length 2 character vector.
col_labels	Column name used for one or more columns passed to cols_label_ext()
spanner	If NULL, gt table is passed to cols_merge_uncert_ext(). If not NULL, spanner is passed to the label parameter of gt::tab_spanner().
decimals	*Number of decimal places* scalar<numeric|integer>(val>=0) // *default:* 2 This corresponds to the exact number of decimal places to use. A value such as 2.34 can, for example, be formatted with 0 decimal places and it would result in "2". With 4 decimal places, the formatted value becomes "2.3400".
use_seps	*Use digit group separators* scalar<logical> // *default:* TRUE An option to use digit group separators. The type of digit group separator is set by sep_mark and overridden if a locale ID is provided to locale. This setting is TRUE by default.
...	Additional parameters passed to gt::fmt_number() by fmt_acs_estimate() or to gt::fmt_percent() by fmt_acs_percent().

call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Details

Using cols_label_ext cols_label_ext() is a variant on gt::cols_label() used by fmt_acs_estimate() and fmt_acs_percent().

See Also

Other gt table: gt_acs(), gt_acs_compare(), tab_acs_source_note()

---

fmt_acs_jam_values *Format jam values in an estimate column of a gt table or ACS data frame*

Description

Currently only supports variable B25035_001 from the Median Year Structure Built table.

Usage

```
fmt_acs_jam_values(data)
```

Arguments

data	Data frame with ACS data

See Also

- fmt_acs_county()
- fmt_acs_minutes()

---

geom_acs_col *Creating a bar chart with error bar and scale*

Description

Create a bar chart with ggplot2::geom_col() and apply an errorbar (using geom_acs_errorbar), scale (using scale_x_acs or scale_y_acs).

Usage

```
geom_acs_col(
  mapping = NULL,
  data = NULL,
  position = "stack",
  ...,
  x = "estimate",
  y = "column_title",
  fill = y,
  value_col = "estimate",
  moe_col = "moe",
  perc_prefix = "perc",
  perc_sep = "_",
  perc = TRUE,
  orientation = NA,
  errorbar_value = TRUE,
  errorbar_params = list(linewidth = 0.5, height = 0.35, position = "identity"),
  scale_value = TRUE,
  scale_params = list()
)
```

Arguments

mapping	Aesthetic mapping. Recommend leaving this as NULL.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`x`, `y`, `fill` String values with column names mapped to aesthetics. Optional if `mapping` is supplied.

`value_col` Column name for estimate value column. Defaults to "estimate".

`moe_col` Column name for margin of error column. Defaults to "moe".

`perc_prefix` Prefix string for percent value columns.

`perc_sep` Separator string between `perc_prefix` and the `value_col` and `moe_col` strings.

`perc` If `TRUE`, return percent value and margin of error columns.

`orientation` The orientation of the layer. The default (`NA`) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting `orientation` to either `"x"` or `"y"`. See the *Orientation* section for more detail.

`errorbar_value` If `TRUE` (default), apply `geom_acs_errorbar()` function to geom.

`errorbar_params`
Parameters passed to `geom_acs_errorbar()` if `errorbar_value = TRUE`. Defaults to `list(linewidth = 0.5, height = 0.35)`

`scale_value` If `TRUE` (default), apply `scale_x_acs()` or `scale_y_acs()` function to geom.

`scale_params` Parameters passed to `scale_x_acs()` or `scale_y_acs()` function if `scale_value = TRUE`. Defaults to `list()`.

`get_acs_tables` *Get multiple tables or multiple geographies of ACS data*

Description

[Experimental] These functions wrap `tidycensus::get_acs()` and `label_acs_metadata()` to support downloading multiple tables and combining tables into a single data frame or downloading data for multiple geographies. Note that while the Census API does not have a specific rate or request limit when using a Census API key, using these functions with a large number of tables or geographies may result in errors or failed requests.

CRAN policies require that tidycensus avoid caching by default, however, this package sets `cache_table = TRUE` by default to avoid unecessary load on the Census API.

Usage

```
get_acs_tables(
  geography,
  table = NULL,
  cache_table = TRUE,
  year = 2022,
  survey = "acs5",
  variables = NULL,
  moe_level = 90,
  ...,
  crs = NULL,
  label = TRUE,
  perc = TRUE,
  reliability = FALSE,
  keep_geography = TRUE,
  geoid_col = "GEOID",
  quiet = FALSE,
  call = caller_env()
)

get_acs_geographies(
  geography = c("county", "state"),
  variables = NULL,
  table = NULL,
  cache_table = TRUE,
  year = 2022,
  state = NULL,
  county = NULL,
  msa = NULL,
  survey = "acs5",
  ...,
  label = TRUE,
  perc = TRUE,
  geoid_col = "GEOID",
  quiet = FALSE
)

get_acs_geography(
```

```
  geography,
  variables = NULL,
  table = NULL,
  cache_table = TRUE,
  year = 2022,
  state = NULL,
  county = NULL,
  msa = NULL,
  survey = "acs5",
  ...,
  label = TRUE,
  perc = TRUE,
  geoid_col = "GEOID",
  call = caller_env()
)
```

Arguments

geography	Required character vector of one or more geographies. See https://walker-data.com/tidycensus/articles/basic-usage.html#geography-in-tidycensus for supported options. Defaults to c("county", "state") for get_acs_geographies(). If a supplied geography does not support county and state parameters, these options are dropped before calling tidycensus::get_acs(). Any required parameters are also bound to the returned data frame as new columns.
table	A character vector of tables.
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 2022; 1-year ACS data is available from 2005 through 2022, with the exception of 2020. Defaults to 2022.
survey	The ACS contains one-year, three-year, and five-year surveys expressed as "acs1", "acs3", and "acs5". The default selection is "acs5."
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.
moe_level	The confidence level of the returned margin of error. One of 90 (the default), 95, or 99.
...	Arguments passed on to tidycensus::get_acs 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. 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

`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.

`crs` Coordinate reference system to use for returned sf tibble when `geometry = TRUE` is passed to `tidycensus::get_acs()`. Defaults to `NULL`.

`label` If `TRUE` (default), label the returned ACS data with `label_acs_metadata()` before returning the data frame.

`perc` If `TRUE` (default), use the denominator column ID to calculate each estimate as a percent share of the denominator value and use `tidycensus::moe_prop()` to calculate a new margin of error for the percent estimate.

`reliability` If `TRUE`, use `assign_acs_reliability()` to assign a reliability value to estimate values based on the specified `moe_level`.

`keep_geography` If `TRUE` (default), bind geography and any supplied county or state columns to the returned data frame.

`geoid_col` A GeoID column name to use if perc is `TRUE`, Defaults to 'GEOID'.

`quiet` If `FALSE` (default), leave `cli.default_handler` option unchanged. If `TRUE`, set `cli.default_handler` to suppressMessages temporarily with `rlang::local_options()`

`call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

`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.

`msa` Name or GeoID of a metro area that should be filtered from the overall list of metro areas returned when geography or geographies is "metropolitan/micropolitan statistical area", "cbsa", or "metropolitan statistical area/micropolitan statistical area".

Examples

```
## Not run:
if (interactive()) {
  get_acs_tables(
    geography = "county",
    county = "Baltimore city",
    state = "MD",
    table = c("B01003", "B19013")
  )

  get_acs_geographies(
    geography = c("county", "state"),
    state = "MD",
    table = c("B01003", "B19013")
  )
}

## End(Not run)
```

---

get_acs_ts *Get multiple years of ACS data for time series analysis*

Description

get_acs_ts() is a variant on get_acs_geographies() that supports downloading data for multiple years in addition to multiple tables or multiple geographies. The year is appended as an additional column in the returned data frame. The intended use is to provide the latest year needed and the function will download data for all non-overlapping survey periods. For example, 2021 ACS data using the 5-year sample can be compared to 5-year data from 2016 and 2011. Not all variables can be compared across different years and caution is recommended when using ACS data for time series analysis.

Usage

```
get_acs_ts(
  geography,
  variables = NULL,
  table = NULL,
  cache_table = TRUE,
  year = 2022,
  state = NULL,
  county = NULL,
  survey = "acs5",
  ...,
  quiet = FALSE
)
```

Arguments

geography	Required character vector of one or more geographies. See https://walker-data.com/tidycensus/articles/basic-usage.html#geography-in-tidycensus for supported options. Defaults to c("county", "state") for get_acs_geographies(). If a supplied geography does not support county and state parameters, these options are dropped before calling tidycensus::get_acs(). Any required parameters are also bound to the returned data frame as new columns.
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	A character vector of tables.
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	A numeric vector of years. If length 1, the function uses acs_survey_ts() to get data for all comparable survey years back to the start of the ACS. This is the recommended approach for using get_acs_ts(). If length is greater than 1, return the selected years even if those years may not be valid to compare.
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.
survey	The ACS contains one-year, three-year, and five-year surveys expressed as "acs1", "acs3", and "acs5". The default selection is "acs5."
...	Other keyword arguments
quiet	If FALSE (default), leave cli.default_handler option unchanged. If TRUE, set cli.default_handler to suppressMessages temporarily with rlang::local_options()

Value

A data frame or sf object.

---

get_decennial_ts *Get multiple years of decennial US Census data for time series analysis*

Description

get_decennial_ts() is a wrapper for tidycensus::get_decennial() to handle time series data.

Usage

```
get_decennial_ts(
  geography,
  variables = NULL,
  table = NULL,
  cache_table = TRUE,
  year = 2020,
  sumfile = NULL,
  state = NULL,
  county = NULL,
  geometry = FALSE,
  summary_var = NULL,
  label = TRUE,
  ...
)
```

Arguments

geography	The geography of your data.
variables	If any year value is 2020, variables must be the same length as year with each value corresponding to one of the years requested. This is a temporary requirement to address the mismatch between the available data for 2000 and 2010 relative to 2020. Default: NULL
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	If year is length 1, it is treated as the max year and decennial Census years back to 2000, are added to the vector of requested years. Default: 2020
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.

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.

label If TRUE (default), use label_decennial_data() to add formatted label columns to the decennial Census data frame.

... Arguments passed on to tidycensus::get_decennial

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.

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.

Value

A data frame with decennial Census data.

See Also

tidycensus::get_decennial()

Examples

```
## Not run:
if (interactive()) {
  md_counties <- get_decennial_ts(
    geography = "county",
    variables = c("P001001", "P001001", "P1_001N"),
    year = 2020,
    county = "Baltimore city",
    state = "MD",
    geometry = FALSE
  )
}
```

```
## End(Not run)
```

`gt_acs` *Create a gt table with formatted ACS estimate and percent estimate columns*

Description

Create or format a gt table with an estimate and margin of error and (optionally) percent estimate and margin of error value. Use in combination with the `select_acs()` helper function to prep data before creating a table.

Usage

```
gt_acs(
  data,
  rownames_to_stub = FALSE,
  row_group_as_column = FALSE,
  ...,
  value_col = "estimate",
  moe_col = "moe",
  perc_prefix = "perc",
  perc_sep = "_",
  perc = FALSE,
  column_title_col = "column_title",
  name_col = "NAME",
  perc_value_label = "% share",
  value_label = "Est.",
  column_title_label = NULL,
  name_label = NULL,
  est_spanner = NULL,
  perc_spanner = NULL,
  combined_spanner = NULL,
  decimals = 0,
  source_note = NULL,
  append_note = FALSE,
  drop_geometry = TRUE,
  hide_na_cols = TRUE,
  currency_value = FALSE,
  survey = "acs5",
  year = 2022,
  table = NULL,
  prefix = "Source: ",
  end = ".",
  est_cols = NULL,
  perc_cols = NULL
)
```

Arguments

data	*Input data table* obj:<data.frame>|obj:<tbl_df> // **required** A data.frame object or a tibble (tbl_df).
rownames_to_stub	*Use data frame row labels in the stub* scalar<logical> // *default:* FALSE An option to take rownames from the input data table (should they be available) as row labels in the display table stub.
row_group_as_column	*Mode for displaying row group labels in the stub* scalar<logical> // *default:* FALSE An option that alters the display of row group labels. By default this is FALSE and row group labels will appear in dedicated rows above their respective groups of rows. If TRUE row group labels will occupy a secondary column in the table stub.
...	Additional parameters passed to gt::fmt_number() by fmt_acs_estimate() or to gt::fmt_percent() by fmt_acs_percent().
value_col	Column name for estimate value column. Defaults to "estimate".
moe_col	Column name for margin of error column. Defaults to "moe".
perc_prefix	Prefix string for percent value columns.
perc_sep	Separator string between perc_prefix and the value_col and moe_col strings.
perc	If TRUE, return percent value and margin of error columns.
column_title_col, column_title_label	Column title and label. If column_title_label is a string, column_title_col is required. column_title_label can also be a named vector in the format of c("label" = "column"). column_title_col defaults to "column_title". If column_title_label is "from_table", the label is set based on the simple_table_title column in the table metadata.
name_col, name_label	Place name column and label. name_label can be a string or a named vector (similar to column_title_label). name_col defaults to "NAME"
perc_value_label	Percent value column label.
value_label	Value column label. Defaults to "Est.".
est_spanner, perc_spanner	Spanner labels for estimate and percent estimate columns.
combined_spanner	If not NULL, combined_spanner is passed to label parameter of gt::tab_spanner() using the value columns and percent columns as the columns parameter.
decimals	*Number of decimal places* scalar<numeric|integer>(val>=0) // *default:* 2 This corresponds to the exact number of decimal places to use. A value such as 2.34 can, for example, be formatted with 0 decimal places and it would result in "2". With 4 decimal places, the formatted value becomes "2.3400".

source_note *Source note text*
scalar<character> // **required**
Text to be used in the source note. We can optionally use `md()` and `html()` to style the text as Markdown or to retain HTML elements in the text.

append_note If TRUE, add source_note to the end of the generated ACS data label. If FALSE, any supplied source_note will be used instead of an ACS label.

drop_geometry If TRUE (default) and data is an sf object, drop geometry before turning the data frame into a table.

hide_na_cols If TRUE (default), hide columns where all values are NA.

currency_value If TRUE, use `gt::fmt_currency()` to format value columns instead of `gt::fmt_number()`.

survey ACS survey, "acs5", "acs3", or "acs1".

year Based on the year and survey, `acs_survey_ts()` returns a vector of years for non-overlapping ACS samples to allow comparison.

table One or more table IDs to include in label or source note.

prefix Text to insert before ACS survey label.

end A character string appended to the end of the full label. Defaults to ".".

est_cols, perc_cols
Deprecated. Estimate and percent estimate columns.

See Also

Other gt table: `fmt_acs_estimate()`, `gt_acs_compare()`, `tab_acs_source_note()`

Examples

```
## Not run:
if (interactive()) {
  data <- get_acs_tables(
    geography = "county",
    county = "Baltimore city",
    state = "MD",
    table = "B08134"
  )

  tbl_data <- filter_acs(data, indent == 1, line_number <= 10)
  tbl_data <- select_acs(tbl_data)

  gt_acs(
    tbl_data,
    column_title_label = "Commute time",
    table = "B08134"
  )
}

## End(Not run)
```

gt_acs_compare *Create a gt table with values compared by name, geography, or variable*

Description

`gt_acs_compare()` is a variant of `gt_acs()` that uses `pivot_acs_wider()` to support comparisons of multiple named areas or multiple geographies side-by-side in a combined gt table.

Usage

```
gt_acs_compare(
  data,
  name_col = "NAME",
  value_col = "estimate",
  moe_col = "moe",
  perc_prefix = "perc",
  perc_sep = "_",
  perc = TRUE,
  variable_col = "variable",
  column_title_col = "column_title",
  value_label = "Est.",
  moe_label = "MOE",
  perc_value_label = "% share",
  perc_moe_label = "% MOE",
  column_title_label = NULL,
  id_cols = column_title_col,
  id_expand = FALSE,
  names_from = name_col,
  values_from = NULL,
  names_vary = "slowest",
  names_glue = NULL,
  names_sep = "_",
  decimals = 0,
  currency_value = FALSE,
  merge_moe = TRUE,
  split = "last",
  limit = 1,
  reverse = TRUE,
  source_note = NULL,
  append_note = FALSE,
  hide_na_cols = TRUE,
  survey = "acs5",
  year = 2022,
  table = NULL,
  prefix = "Source: ",
  end = ".",
```

```
  use_md = FALSE,
  use_spanner = TRUE,
  ...
)

gt_acs_compare_vars(
  data,
  name_col = "NAME",
  value_col = "estimate",
  moe_col = "moe",
  perc_prefix = "perc",
  perc_sep = "_",
  variable_col = "variable",
  column_title_col = "column_title",
  value_label = NULL,
  moe_label = "MOE",
  id_cols = name_col,
  names_from = variable_col,
  values_from = c(value_col, moe_col),
  use_spanner = FALSE,
  ...
)
```

Arguments

`data`	A data frame to pivot.
`name_col`	Name column. Defaults to "NAME". Ignored if names_from is not set to name_col.
`value_col`	Column name for estimate value column. Defaults to "estimate".
`moe_col`	Column name for margin of error column. Defaults to "moe".
`perc_prefix`	Prefix string for percent value columns.
`perc_sep`	Separator string between `perc_prefix` and the `value_col` and `moe_col` strings.
`perc`	If `TRUE`, return percent value and margin of error columns.
`variable_col`	Variable column name. Defaults to "variable".
`column_title_col, column_title_label`	Column title column name and label. Defaults to "column_title" and `NULL`.
`value_label`	Value column label. Defaults to "Est.".
`moe_label`	Margin of error column label. Defaults to "MOE".
`perc_value_label`	Percent value column label.
`perc_moe_label`	Percent margin of error column label.
`id_cols`	Defaults to `column_title_col`. See `tidyr::pivot_longer()` for details.
`id_expand`	Should the values in the `id_cols` columns be expanded by `expand()` before pivoting? This results in more rows, the output will contain a complete expansion of all possible values in `id_cols`. Implicit factor levels that aren't represented

in the data will become explicit. Additionally, the row values corresponding to the expanded `id_cols` will be sorted.

`names_from`, `values_from`
: <tidy-select> A pair of arguments describing which column (or columns) to get the name of the output column (`names_from`), and which column (or columns) to get the cell values from (`values_from`).

If `values_from` contains multiple values, the value will be added to the front of the output column.

`names_vary`
: When `names_from` identifies a column (or columns) with multiple unique values, and multiple `values_from` columns are provided, in what order should the resulting column names be combined?

- `"fastest"` varies `names_from` values fastest, resulting in a column naming scheme of the form: `value1_name1, value1_name2, value2_name1, value2_name2`. This is the default.
- `"slowest"` varies `names_from` values slowest, resulting in a column naming scheme of the form: `value1_name1, value2_name1, value1_name2, value2_name2`.

`names_glue`
: Instead of `names_sep` and `names_prefix`, you can supply a glue specification that uses the `names_from` columns (and special `.value`) to create custom column names.

`names_sep`
: If `names_from` or `values_from` contains multiple variables, this will be used to join their values together into a single string to use as a column name.

`decimals`
: *Number of decimal places*

`scalar<numeric|integer>(val>=0)` // *default:* `2`

This corresponds to the exact number of decimal places to use. A value such as `2.34` can, for example, be formatted with `0` decimal places and it would result in `"2"`. With `4` decimal places, the formatted value becomes `"2.3400"`.

`currency_value`
: If `TRUE`, use `gt::fmt_currency()` to format value columns instead of `gt::fmt_number()`.

`merge_moe`
: If `TRUE`, use `gt::cols_merge_uncert()` to merge the value_col and moe_col and the percent value and margin of error columns.

`split`
: *Splitting side*

`singl-kw:[last|first]` // *default:* `"last"`

Should the delimiter splitting occur from the `"last"` instance of the `delim` character or from the `"first"`? The default here uses the `"last"` keyword, and splitting begins at the last instance of the delimiter in the column name. This option only has some consequence when there is a `limit` value applied that is lesser than the number of delimiter characters for a given column name (i.e., number of splits is not the maximum possible number).

`limit`
: *Limit for splitting*

`scalar<numeric|integer|character>` // *default:* `NULL` (`optional`)

An optional limit to place on the splitting procedure. The default `NULL` means that a column name will be split as many times are there are delimiter characters. In other words, the default means there is no limit. If an integer value is given to `limit` then splitting will cease at the iteration given by `limit`. This works in tandem with `split` since we can adjust the number of splits from either the right side (`split = "last"`) or left side (`split = "first"`) of the column name.

reverse	*Reverse vector of split names* scalar<logical> // *default:* FALSE Should the order of split names be reversed? By default, this is FALSE.
source_note	*Source note text* scalar<character> // **required** Text to be used in the source note. We can optionally use md() and html() to style the text as Markdown or to retain HTML elements in the text.
append_note	If TRUE, add source_note to the end of the generated ACS data label. If FALSE, any supplied source_note will be used instead of an ACS label.
hide_na_cols	If TRUE (default), hide any columns with all NA values using gt::cols_hide().
survey	ACS survey, "acs5", "acs3", or "acs1".
year	Based on the year and survey, acs_survey_ts() returns a vector of years for non-overlapping ACS samples to allow comparison.
table	One or more table IDs to include in label or source note.
prefix	Text to insert before ACS survey label.
end	A character string appended to the end of the full label. Defaults to ".".
use_md	If TRUE, pass source_note to gt::md() first.
use_spanner	If TRUE (default), create spanners for the comparison geographies.
...	Additional arguments passed on to methods.

See Also

Other gt table: fmt_acs_estimate(), gt_acs(), tab_acs_source_note()

---

jam_values *ACS Jam Values for Medians*

Description

Reference table of ACS "jam values" for medians from "Table 5.2. Jam Values for Medians," *Understanding and Using American Community Survey Data: What All Data Users Need to Know* (2020). type and units values are added. year is included to account for the possibility of alternate jam values for earlier or later years but annual variation in values has not been checked.

Usage

```
jam_values
```

Format

A data frame with 20 rows and 6 variables:

`value` Estimate value

`meaning` Meaning of estimate value

`use` Subjects/tables where jam value is used

`type` Type (minimum or maximum jam value)

`units` Units. Note year is for a specific year, years is for duration.

`year` Year applicable

Details

https://docs.google.com/spreadsheets/d/1YX3NBDkkoDXHs88KDfPS_QoS9-1j_C_q8UAyjPznfzA/edit?usp=sharing

---

join_acs_denominator *Join denominator values based on a supplied denominator column*

Description

Note that this function and the related `join_acs_percent()` function depends on the column-level metadata supplied by `label_acs_metadata()`.

Usage

```
join_acs_denominator(
  data,
  geoid_col = "GEOID",
  value_col = "estimate",
  moe_col = "moe",
  column_id_col = "column_id",
  column_title_col = "column_title",
  denominator_col = NULL,
  denominator_prefix = "denominator_",
  na_matches = "never",
  digits = 2,
  call = caller_env()
)
```

Arguments

`data`	A data frame with column names including "column_id", "column_title", "denominator_column_id", "estimate", and "moe".
`geoid_col`	A GeoID column name to use if perc is `TRUE`, Defaults to 'GEOID'.
`value_col`	Value column name

moe_col — Margin of error column name

column_id_col — Column ID column name from Census Reporter metadata. Defaults to "column_id"

column_title_col — Column title column name. Defaults to "column_title".

denominator_col — Denominator column ID name from Census Reporter metadata. Defaults to NULL

denominator_prefix — Prefix to use for denominator column names.

na_matches — Should two NA or two NaN values match?

- "na", the default, treats two NA or two NaN values as equal, like %in%, match(), and merge().
- "never" treats two NA or two NaN values as different, and will never match them together or to any other values. This is similar to joins for database sources and to base::merge(incomparables = NA).

digits — integer indicating the number of decimal places (round) or significant digits (signif) to be used. For round, negative values are allowed (see 'Details').

call — The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

---

join_acs_geography_ratio

Join ACS data from a single reference geography by variable to calculate a ratio value based on the reference geography data

---

Description

join_acs_geography_ratio() uses data from get_acs_geographies() to support the calculation of proportions join parent column titles to a data frame of ACS data.

Usage

```
join_acs_geography_ratio(
  data,
  variable_col = "variable",
  value_col = "estimate",
  moe_col = "moe",
  geography = "county",
  na_matches = "never",
  digits = 2
)
```

Arguments

data	A data frame with column names matching the supplied parameters.
variable_col	Variable column name to join as join variable, Default: 'variable'
value_col, moe_col	Estimate and margin of error column names, Default: 'estimate' and 'moe'
geography	Value in geography column to use as comparison values, Default: 'county'
na_matches	Should two NA or two NaN values match? • "na", the default, treats two NA or two NaN values as equal, like %in%, match(), and merge(). • "never" treats two NA or two NaN values as different, and will never match them together or to any other values. This is similar to joins for database sources and to base::merge(incomparables = NA).
digits	integer indicating the number of decimal places (round) or significant digits (signif) to be used. For round, negative values are allowed (see 'Details').

Value

A data frame with new estimate and moe columns prefixed with "ratio_".

See Also

tidycensus::moe_ratio()

---

join_acs_parent_column

Join parent column titles to ACS data based on parent column ID values

---

Description

join_acs_parent_column() uses data labelled with parent_column_id values to join parent column titles to a data frame of ACS data.

Usage

```
join_acs_parent_column(
  data,
  column_id_col = "column_id",
  column_title_col = "column_title",
  parent_id_col = "parent_column_id",
  suffix = c("", "_parent"),
  na_matches = "never",
  relationship = "many-to-one"
)
```

Arguments

data	A data frame with the specified column names. Expected to be labelled using label_acs_metadata().
column_id_col, column_title_col, parent_id_col	Column ID, column title, and parent column ID.
suffix	Suffix passed to dplyr::left_join(), Default: c("", "_parent")
na_matches	Should two NA or two NaN values match? • "na", the default, treats two NA or two NaN values as equal, like %in%, match(), and merge(). • "never" treats two NA or two NaN values as different, and will never match them together or to any other values. This is similar to joins for database sources and to base::merge(incomparables = NA).
relationship	Handling of the expected relationship between the keys of x and y. If the expectations chosen from the list below are invalidated, an error is thrown. • NULL, the default, doesn't expect there to be any relationship between x and y. However, for equality joins it will check for a many-to-many relationship (which is typically unexpected) and will warn if one occurs, encouraging you to either take a closer look at your inputs or make this relationship explicit by specifying "many-to-many". See the *Many-to-many relationships* section for more details. • "one-to-one" expects: – Each row in x matches at most 1 row in y. – Each row in y matches at most 1 row in x. • "one-to-many" expects: – Each row in y matches at most 1 row in x. • "many-to-one" expects: – Each row in x matches at most 1 row in y. • "many-to-many" doesn't perform any relationship checks, but is provided to allow you to be explicit about this relationship if you know it exists. relationship doesn't handle cases where there are zero matches. For that, see unmatched.

Value

A data frame with added parent column title.

`join_acs_percent` *Join percent estimates to ACS data based on denominator values*

Description

join_acs_percent() uses the denominator_column_id value from the column metadata added with label_acs_metadata() to calculate the estimate as a percent share of the denominator value. tidycensus::moe_prop() is used to calculate the margin of error for the percentage. join_acs_percent_parent() is a variation that, by default, calculates the percentage values based on the "parent_column_id" instead of the "denomination_column_id".

Usage

```
join_acs_percent(
  data,
  geoid_col = "GEOID",
  column_id_col = "column_id",
  denominator_col = NULL,
  denominator_prefix = "denominator_",
  value_col = "estimate",
  moe_col = "moe",
  perc = TRUE,
  perc_prefix = "perc",
  perc_sep = "_",
  na_matches = "never",
  digits = 2
)

join_acs_percent_parent(
  data,
  geoid_col = "GEOID",
  column_id_col = "column_id",
  denominator_col = NULL,
  denominator_prefix = "parent_",
  value_col = "estimate",
  moe_col = "moe",
  perc_prefix = "perc_parent",
  perc_sep = "_",
  na_matches = "never",
  digits = 2
)
```

Arguments

data	A data frame with column names including "column_id", "column_title", "denominator_column_id", "estimate", and "moe".
geoid_col	A GeoID column name to use if perc is TRUE, Defaults to 'GEOID'.
column_id_col	Column ID column name from Census Reporter metadata. Defaults to "column_id"
denominator_col	Denominator column ID name from Census Reporter metadata. Defaults to NULL
denominator_prefix	Prefix to use for denominator column names.
value_col	Value column name
moe_col	Margin of error column name
perc	If FALSE, return data joined with join_acs_denominator() and skip joining percent values. Defaults to TRUE.

perc_prefix — Prefix string for percent value columns.

perc_sep — Separator string between perc_prefix and the value_col and moe_col strings.

na_matches — Should two NA or two NaN values match?

- "na", the default, treats two NA or two NaN values as equal, like %in%, match(), and merge().
- "never" treats two NA or two NaN values as different, and will never match them together or to any other values. This is similar to joins for database sources and to base::merge(incomparables = NA).

digits — integer indicating the number of decimal places (round) or significant digits (signif) to be used. For round, negative values are allowed (see 'Details').

See Also

tidycensus::moe_prop(), camiller::calc_shares()

---

labs_acs_survey *Label a ggplot2 plot and add a caption based on an ACS survey year*

Description

labs_acs_survey() uses acs_survey_label_table() to create a label for a ggplot2 plot passed to the caption parameter of ggplot2::labs().

Usage

```
labs_acs_survey(
  ...,
  caption = NULL,
  survey = "acs5",
  year = 2022,
  prefix = "Source: ",
  table = NULL,
  .data = NULL
)
```

Arguments

... — Arguments passed on to ggplot2::labs

title The text for the title.

subtitle The text for the subtitle for the plot which will be displayed below the title.

tag The text for the tag label which will be displayed at the top-left of the plot by default.

alt,alt_insight Text used for the generation of alt-text for the plot. See get_alt_text for examples.

caption	The text for the caption which will be displayed in the bottom-right of the plot by default.
survey	ACS survey, "acs5", "acs3", or "acs1".
year	Based on the year and survey, acs_survey_ts() returns a vector of years for non-overlapping ACS samples to allow comparison.
prefix	Text to insert before ACS survey label.
table	One or more table IDs to include in label or source note.
.data	Optional data frame with "table_id" column used in place of table if table is NULL. Ignored if table is supplied.

---

load_acs_vars *Load ACS variables with* tidycensus::load_variables()

Description

load_acs_vars() calls tidycensus::load_variables() and then combines the returned data frame with the Census Reporter metadata from label_acs_table_metadata(). The function can optionally filter the variable definitions to a set of tables and variables or drop variables from the results.

Usage

```
load_acs_vars(
  year = 2022,
  survey = "acs5",
  cache = TRUE,
  variable_col = "variable",
  geography_levels = c("block", "block group", "tract", "county", "state", "us"),
  table = NULL,
  vars = NULL,
  drop_vars = NULL
)
```

Arguments

year	Sample year (between 2006 and 2022).
survey	Survey, "acs5", "acs3", or "acs1".
cache	Whether you would like to cache the dataset for future access, or load the dataset from an existing cache. Defaults to FALSE.
variable_col	Variable column name. Defaults to "variable"
geography_levels	Ordered vector of geography levels used to convert the geography column returned by tidycensus::load_variables() into a factor. Default: c("block", "block group", "tract", "county", "state", "us")

table	Table ID to return.
vars, drop_vars	Variable IDs to keep or to drop. If table is supplied (or if data only contains data for a single table), numeric values are allowed for vars and drop_vars (e.g. if table is "B14001" and vars is 2 data is filtered to variable "B14001_002").

Value

A data frame with ACS variables definitions.

See Also

tidycensus::load_variables()

---

make_area_xwalk *Make and use crosswalk data based on U.S. Census block-level weights for U.S. Census tracts and non-Census geographic areas*

Description

make_area_xwalk() creates a crosswalk data frame based on the weight_col parameter (if year = 2020, use "POP20" for population, "HOUSING20" for households, or "ALAND20" for land area). Using this function with other years, requires users to add population data to the block_xwalk as the tigris::blocks() function only includes population and household count data for the 2020 year. This function has also not been tested when areas include overlapping geometry and the results may be invalid for those overlapping areas if that is the case.

Usage

```
make_area_xwalk(
  area,
  block_xwalk = NULL,
  state = NULL,
  county = NULL,
  year = 2020,
  name_col = "NAME",
  weight_col = "HOUSING20",
  geoid_col = "GEOID",
  tract_col = "TRACTCE20",
  by = c(TRACTCE20 = "TRACTCE"),
  suffix = c("_block", "_tract"),
  placement = c("largest", "surface", "centroid"),
  digits = 2,
  extensive = TRUE,
  coverage = TRUE,
  erase = FALSE,
  area_threshold = 0.75,
  keep_geometry = FALSE,
```

```
  crs = NULL,
  make_valid = TRUE,
  ...
)

use_area_xwalk(
  data,
  area_xwalk,
  geography = "area",
  name_col = "NAME",
  geoid_col = "GEOID",
  suffix = c("_area", ""),
  weight_col = "perc_HOUSING20",
  variable_col = "variable",
  value_col = "estimate",
  moe_col = "moe",
  digits = 0,
  perc = TRUE,
  extensive = TRUE,
  reliability = FALSE,
  moe_level = 90
)
```

Arguments

area	A sf object with an arbitrary geography overlapping with the block_xwalk. Required. If area only partly overlaps with block_xwalk, coverage should be set to TRUE (default).
block_xwalk	Block-tract crosswalk sf object. If NULL, state is required to create a crosswalk using make_block_xwalk()
state	The two-digit FIPS code (string) of the state you want. Can also be state name or state abbreviation.
county	The three-digit FIPS code (string) of the county you'd like to subset for, or a vector of FIPS codes if you desire multiple counties. Can also be a county name or vector of names.
year	the data year; defaults to 2022
name_col	Name column in area.
weight_col	Column name in input block_xwalk to use for weighting. Generated weight_col used by use_area_xwalk() should be the same as the weight_col for make_area_xwalk() but include the "perc_" prefix. Defaults to "HOUSING20" for make_block_xwalk() and "perc_HOUSING20" for use_area_xwalk().
geoid_col, tract_col	GeoID for Census tract and Census tract ID column in block_xwalk
by	Specification of join variables in the format of c("block column name for tract" = "tract column name"). Passed to dplyr::left_join().
suffix	Suffixes added to the output to disambiguate column names from the block and tract data. Unused for 2020 data.

placement — String with option for joining area and block_xwalk: "largest", "surface", or "centroid". "largest" joins the two using sf::st_join() with largest set to TRUE. "surface" first transforms block_xwalk using sf::st_point_on_surface() and "centroid" uses sf::st_centroid().

digits — Digits to use for percent share of weight value.

extensive — If TRUE (default) calculate new estimate values as weighted sums and re-calculate margin of error with tidycensus::moe_sum(). If FALSE, calculate new estimate values as weighted means (appropriate for ACS median variables) and drop the margin of error. perc is also always set to FALSE if extensive is FALSE.

coverage — If TRUE (default), it is assumed that area does not cover the full extent of the block_xwalk and an additional feature is added with the difference between the unioned area geometry and unioned block_xwalk geometry. This additional coverage ensures that blocks are accurately assigned to this alternate geography but it is excluded from the returned data frame. If coverage is TRUE and all features in area overlap with block_xwalk, the function issues a warning and then resets coverage to FALSE. The reverse option is applied if any features from area do not overlap. coverage can also be a sf or sfc object which may be useful in some limited cases.

erase — If TRUE, apply tigris::erase_water() to input area and block_xwalk before joining. Defaults to FALSE. If erase is a sf object, the geometry of the input sf is erased from area and block_xwalk. This option is intended to support erasing open space or other non-developed land as well as water areas.

area_threshold — The percentile rank cutoff of water areas to use in the erase operation, ranked by size. Defaults to 0.75, representing the water areas in the 75th percentile and up (the largest 25 percent of areas). This value may need to be modified by the user to achieve optimal results for a given location.

keep_geometry — If TRUE, area_xwalk is a sf object with the same geometry as the input area. Defaults to FALSE.

crs — Coordinate reference system to use for input data. Recommended to set to a projected CRS if input area data is in a geographic CRS.

make_valid — Default TRUE. If TRUE, apply sf::st_make_valid() to the input area geometry and to any sf or sfc object passed to the erase parameter. If this has any unexpected results, set make_valid = FALSE and prepare any invalid geometry before passing to this function.

... — Passed to make_block_xwalk().

data — A data frame downloaded with tidycensus::get_acs().

area_xwalk — A area crosswalk data frame created with make_area_xwalk(). Required for use_area_xwalk().

geography — A character string used as general description for area geography type. Defaults to "area" but typical values could include "neighborhood", "planning district", or "service area".

variable_col — Variable column name. Defaults to "variable"

value_col, moe_col — Value and margin of error column names (defaults to "estimate" and "moe").

perc	If TRUE (default), use the denominator column ID to calculate each estimate as a percent share of the denominator value and use tidycensus::moe_prop() to calculate a new margin of error for the percent estimate.
reliability	If TRUE, use assign_acs_reliability() to assign a reliability value to estimate values based on the specified moe_level.
moe_level	The confidence level of the margin of error. Defaults to 90 (which is the same default as tidycensus::get_acs()).

Details

Using an area crosswalk

After creating an area crosswalk with make_area_xwalk(), you can pass the crosswalk to use_area_xwalk() along with a data frame from tidycensus::get_acs() or get_acs_tables(). At a minimum, the data must have a column with the same name as geoid_col along with columns named "variable", "estimate", and "moe".

Please note that this approach to aggregation does *not* work well if your data contains "jam" values, e.g. the substitution of 0 for "1939 or older" for the Median Year Built variable. Ideally, the weight used for aggregation should be based on household counts when aggregating a household-level variable and population counts when aggregating a individual-level variable.

Value

A tibble or a sf object.

See Also

tidycensus::interpolate_pw(), areal::aw_interpolate()

---

make_block_xwalk *Make crosswalk data for U.S. Census blocks and tracts*

Description

make_block_xwalk() joined U.S. Census blocks data from tigris::blocks() to a data frame from tigris::tracts() to provide a crosswalk between both geographies. If year = 2020, the suffix parameter is not used. If year is any other year than 2020, the by parameter must be changed from the default value of c("TRACTCE20" = "TRACTCE"). 2020 is also the only year where tigris::blocks() includes the population and household count data required to use this crosswalk data frame with make_area_xwalk().

Usage

```
make_block_xwalk(
  state,
  county = NULL,
  year = 2020,
```

```
  by = c(TRACTCE20 = "TRACTCE"),
  keep_zipped_shapefile = TRUE,
  suffix = c("_block", "_tract"),
  crs = NULL,
  ...
)
```

Arguments

state	The two-digit FIPS code (string) of the state you want. Can also be state name or state abbreviation.
county	The three-digit FIPS code (string) of the county you'd like to subset for, or a vector of FIPS codes if you desire multiple counties. Can also be a county name or vector of names.
year	the data year; defaults to 2022
by	Specification of join variables in the format of c("block column name for tract" = "tract column name"). Passed to dplyr::left_join().
keep_zipped_shapefile	Passed to tigris::blocks() and tigris::tracts() to keep and re-use the zipped shapefile.
suffix	Suffixes added to the output to disambiguate column names from the block and tract data. Unused for 2020 data.
crs	Coordinate reference system to return.
...	Arguments passed on to tigris::blocks

---

pivot_acs_wider *Pivot a ACS data frame into a wider format by name or other columns*

Description

pivot_acs_wider() wraps tidyr::pivot_wider() and makes it easy to convert an ACS data frame into a wide format by changing the value of the names_from parameter. The default parameter value vary from the tidyr version with names_vary = "slowest" and values_from = NULL (replaced by using the .col_fn {tidyselect} function on the named value and percent value columns). You may need to retain the variable column and set id_cols = "variable" if the column_title does not uniquely identify rows after widening the input data.

Usage

```
pivot_acs_wider(
  data,
  name_col = "NAME",
  value_col = "estimate",
  moe_col = "moe",
```

```
  perc_prefix = "perc",
  perc_sep = "_",
  perc = TRUE,
  .col_fn = any_of,
  ...,
  id_cols = NULL,
  id_expand = FALSE,
  names_from = name_col,
  names_sep = "_",
  names_glue = NULL,
  names_vary = "slowest",
  names_repair = "check_unique",
  values_from = NULL
)
```

Arguments

data	A data frame to pivot.
name_col	Name column. Defaults to "NAME". Ignored if names_from is not set to name_col.
value_col	Column name for estimate value column. Defaults to "estimate".
moe_col	Column name for margin of error column. Defaults to "moe".
perc_prefix	Prefix string for percent value columns.
perc_sep	Separator string between `perc_prefix` and the `value_col` and `moe_col` strings.
perc	If `TRUE`, return percent value and margin of error columns.
.col_fn	tidyselect function to use with column names. Defaults to tidyselect::starts_with,
...	Arguments passed on to `tidyr::pivot_wider`

`names_from,values_from` <tidy-select> A pair of arguments describing which column (or columns) to get the name of the output column (`names_from`), and which column (or columns) to get the cell values from (`values_from`). If `values_from` contains multiple values, the value will be added to the front of the output column.

`names_prefix` String added to the start of every variable name. This is particularly useful if `names_from` is a numeric vector and you want to create syntactic variable names.

`names_sort` Should the column names be sorted? If `FALSE`, the default, column names are ordered by first appearance.

`names_expand` Should the values in the `names_from` columns be expanded by `expand()` before pivoting? This results in more columns, the output will contain column names corresponding to a complete expansion of all possible values in `names_from`. Implicit factor levels that aren't represented in the data will become explicit. Additionally, the column names will be sorted, identical to what `names_sort` would produce.

`values_fill` Optionally, a (scalar) value that specifies what each `value` should be filled in with when missing.

This can be a named list if you want to apply different fill values to different value columns.

`values_fn` Optionally, a function applied to the value in each cell in the output. You will typically use this when the combination of `id_cols` and `names_from` columns does not uniquely identify an observation.
This can be a named list if you want to apply different aggregations to different `values_from` columns.

`unused_fn` Optionally, a function applied to summarize the values from the unused columns (i.e. columns not identified by `id_cols`, `names_from`, or `values_from`).
The default drops all unused columns from the result.
This can be a named list if you want to apply different aggregations to different unused columns.
`id_cols` must be supplied for `unused_fn` to be useful, since otherwise all unspecified columns will be considered `id_cols`.
This is similar to grouping by the `id_cols` then summarizing the unused columns using `unused_fn`.

`id_cols` <tidy-select> A set of columns that uniquely identify each observation. Typically used when you have redundant variables, i.e. variables whose values are perfectly correlated with existing variables.

Defaults to all columns in `data` except for the columns specified through `names_from` and `values_from`. If a tidyselect expression is supplied, it will be evaluated on `data` after removing the columns specified through `names_from` and `values_from`.

`id_expand` Should the values in the `id_cols` columns be expanded by `expand()` before pivoting? This results in more rows, the output will contain a complete expansion of all possible values in `id_cols`. Implicit factor levels that aren't represented in the data will become explicit. Additionally, the row values corresponding to the expanded `id_cols` will be sorted.

`names_from`, `values_from` <tidy-select> A pair of arguments describing which column (or columns) to get the name of the output column (`names_from`), and which column (or columns) to get the cell values from (`values_from`).

If `values_from` contains multiple values, the value will be added to the front of the output column.

`names_sep` If `names_from` or `values_from` contains multiple variables, this will be used to join their values together into a single string to use as a column name.

`names_glue` Instead of `names_sep` and `names_prefix`, you can supply a glue specification that uses the `names_from` columns (and special `.value`) to create custom column names.

`names_vary` When `names_from` identifies a column (or columns) with multiple unique values, and multiple `values_from` columns are provided, in what order should the resulting column names be combined?

- `"fastest"` varies `names_from` values fastest, resulting in a column naming scheme of the form: `value1_name1, value1_name2, value2_name1, value2_name2`. This is the default.

- `"slowest"` varies `names_from` values slowest, resulting in a column naming scheme of the form: `value1_name1, value2_name1, value1_name2, value2_name2`.

`names_repair` What happens if the output has invalid column names? The default, `"check_unique"` is to error if the columns are duplicated. Use `"minimal"` to allow duplicates in the output, or `"unique"` to de-duplicated by adding numeric suffixes. See `vctrs::vec_as_names()` for more options.

`race_iteration` *Race or Latino Origin Table Codes*

Description

For selected tables, an alphabetic suffix follows to indicate that a table is repeated for the nine major race and Hispanic or Latino groups.

Usage

```
race_iteration
```

Format

A data frame with 9 rows and 3 variables:

`code` Code

`group` Race or Ethnic group

`label` Short label

Details

https://www.census.gov/programs-surveys/acs/data/data-tables/table-ids-explained.html

---

`scale_acs` *Scales for plotting ACS data with ggplot2*

Description

Scales for plotting ACS data with ggplot2

Usage

```
scale_x_acs(..., perc = FALSE)

scale_y_acs(..., perc = FALSE)

scale_x_acs_estimate(name = "Estimate", ..., labels = scales::label_comma())

scale_y_acs_percent(
  name = "Est. % of total",
  ...,
  labels = scales::label_percent()
)

scale_x_acs_percent(
  name = "Est. % of total",
  ...,
  labels = scales::label_percent()
)

scale_y_acs_estimate(name = "Estimate", ..., labels = scales::label_comma())

scale_x_acs_ts(name = "Year", ..., breaks = NULL, survey = "acs5", year = 2022)

scale_y_acs_ts(name = "Year", ..., breaks = NULL, survey = "acs5", year = 2022)
```

Arguments

`...`	Other arguments passed on to `scale_(x|y)_continuous()`
`perc`	If `TRUE`, use the `scale_x_acs_percent` or `scale_y_acs_percent`. Defaults to `FALSE`.
`name`	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
`labels`	One of: • `NULL` for no labels • `waiver()` for the default labels computed by the transformation object • A character vector giving labels (must be same length as `breaks`) • An expression vector (must be the same length as breaks). See ?plotmath for details. • A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.
`breaks`	One of: • `NULL` for no breaks • `waiver()` for the default breaks computed by the transformation object • A numeric vector of positions

- A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Note that for position scales, limits are provided after scale expansion. Also accepts rlang lambda function notation.

`survey` ACS survey, "acs5", "acs3", or "acs1".

`year` Based on the year and survey, `acs_survey_ts()` returns a vector of years for non-overlapping ACS samples to allow comparison.

`select_acs` *Keep or drop columns from an ACS data frame using* `dplyr::select()`

Description

[Experimental]

Usage

```
select_acs(
  .data,
  ...,
  .name_col = "NAME",
  .column_title_col = "column_title",
  .value_col = "estimate",
  .moe_col = "moe",
  .perc_prefix = "perc",
  .perc_sep = "_",
  .perc = TRUE,
  .fn = any_of
)
```

Arguments

`.data` A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.

`...` <`tidy-select`> One or more unquoted expressions separated by commas. Variable names can be used as if they were positions in the data frame, so expressions like `x:y` can be used to select a range of variables.

`.name_col`, `.column_title_col`, `.value_col`, `.moe_col`
ACS data column names to select using the Tidyverse selection helper in `.fn`. Set any parameter to `NULL` to avoid selecting columns.

`.perc_prefix`, `.perc_sep`
Percent value prefix and separator. Set .perc_prefix to `NULL` or `.perc = FALSE` to drop the percent value and percent margin of error columns.

`.perc` If `TRUE`, select the percent value and percent margin of error columns along with the supplied column values.

`.fn` Tidyverse selection helper to use with named ACS columns. Defaults to tidyselect::any_of. See `dplyr::select()` for an overview of selection features.

Details

select_acs() is a wrapper for dplyr::select() designed to select the appropriate columns for a gt table created with gt_acs(). Set any named parameter to NULL to drop the respective column or use the additional ... parameter to modify the selection.

Examples

```
## Not run:
if (interactive()) {
  edu_data <- get_acs_tables(
    "county",
    table = "B15003",
    state = "MD",
    county = "Baltimore city"
  )

  select_acs(edu_data)
}

## End(Not run)
```

---

tab_acs_source_note *Add a Census data source note to a gt table*

Description

tab_acs_source_note() adds a source note to a gt table using acs_survey_label_table() and gt::tab_source_note().

Usage

```
tab_acs_source_note(
  gt_object,
  source_note = NULL,
  append_note = FALSE,
  survey = "acs5",
  year = 2022,
  table = NULL,
  table_label = "Table",
  prefix = "Source: ",
  end = ".",
  use_md = FALSE,
  ...
)
```

Arguments

gt_object	A gt object.
source_note	*Source note text* scalar<character> // **required** Text to be used in the source note. We can optionally use md() and html() to style the text as Markdown or to retain HTML elements in the text.
append_note	If TRUE, add source_note to the end of the generated ACS data label. If FALSE, any supplied source_note will be used instead of an ACS label.
survey	ACS survey, "acs5", "acs3", or "acs1".
year	Based on the year and survey, acs_survey_ts() returns a vector of years for non-overlapping ACS samples to allow comparison.
table	One or more table IDs to include in label or source note.
table_label	Label to use when referring to table or tables. A "s" is appended to the end of the table_label if tables is more than length 1.
prefix	Text to insert before ACS survey label.
end	A character string appended to the end of the full label. Defaults to ".".
use_md	If TRUE, pass source_note to gt::md() first.
...	For tab_acs_source_note(), additional parameters passed to acs_survey_label_table(). For cols_merge_uncert_ext(), additional parameters passed to gt::cols_merge_uncert(). For fmt_acs_percent(), additional parameters passed to gt::fmt_percent().

See Also

Other gt table: fmt_acs_estimate(), gt_acs(), gt_acs_compare()

---

tigerweb_geo_index *U.S. Census Bureau ArcGIS Services Index*

Description

Index created with esri2sf::esriIndex() listing all services located at https://tigerweb.geo.census.gov/arcgis/rest/services. Access ArcGIS services using the esri2sf package https://github.com/elipousson/esri2sf or arcpullr https://github.com/pfrater/arcpullr/.

Usage

```
tigerweb_geo_index
```

Format

A data frame with 7081 rows and 15 variables:

`name` Name
`type` Service/layer type
`url` Folder/service/layer URL
`urlType` URL type
`folderPath` Index type
`serviceName` Service name
`serviceType` Service type
`id` integer Layer ID number
`parentLayerId` integer Parent layer ID number
`defaultVisibility` logical Layer default visibility
`subLayerIds` list Sublayer ID numbers
`minScale` double Minimum scale
`maxScale` integer Maximum scale
`geometryType` Geometry type
`supportsDynamicLegends` logical Supports dynamic legends

Details

https://tigerweb.geo.census.gov/arcgis/rest/services

---

usa_states *U.S. States Reference Data*

Description

A reference table of state names, abbreviations, regions, and divisions.

Usage

```
usa_states
```

Format

A data frame with 56 rows and 7 variables:

`state` State name
`state_abb` State USPS abbreviation
`STATE_GEOID` State GeoID
`division` Census Division name
`DIVISION_GEOID` Census Division GeoID
`region` Census Region name
`REGION_GEOID` Census Region GeoID

vec_get_acs *Vectorized variant of tidycensus::get_acs*

Description

Vectorized variant of tidycensus::get_acs

Usage

```
vec_get_acs(..., .fn = tidycensus::get_acs, .size = NULL, .call = caller_env())
```

Arguments

`...`	Additional parameters passed to .fn.
`.fn`	Function to call with parameters, Defaults to `tidycensus::get_acs`. Function must require a geography parameter and return a data frame.
`.size`	Desired output size.
`.call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

A list of data frames (using default .fn value or another function that returns a data frame).

A list of data frames.

Examples

```
## Not run:
if (interactive()) {
  # TODO: Add examples
}

## End(Not run)
```

Index