Package ‘censable’

October 5, 2021

Title  Making Census Data More Usable
Version  0.0.3
Date  2021-10-05

URL  https://www.christophertkenny.com/censable/,
     https://github.com/christopherkenny/censable

BugReports  https://github.com/christopherkenny/censable/issues

Description  Creates a common framework for organizing, naming, and gathering population, age, race, and ethnicity data from the Census Bureau. Accesses the API <https://www.census.gov/data/developers/data-sets.html> via the package tidycensus. Provides tools for adding information to existing data to line up with Census data.

License  MIT + file LICENSE
Encoding  UTF-8
LazyData  true
RoxygenNote  7.1.1
Suggests  roxygen2, testthat (>= 3.0.0)
Imports  magrittr, dplyr (>= 1.0.4), rlang (>= 0.1.2), tidycensus, sf
         (>= 1.0.0), tibble, stringr, memoise
Depends  R (>= 2.10)
Config/testthat/edition  3

NeedsCompilation  no

Author  Christopher T. Kenny [aut, cre]
         (<https://orcid.org/0000-0002-9386-6860>)

Maintainer  Christopher T. Kenny <christopherkenny@fas.harvard.edu>

Repository  CRAN

Date/Publication  2021-10-05 20:40:02 UTC
R topics documented:

- add_r_environ ........................................ 3
- breakdown_geoid ...................................... 4
- build_acs ............................................. 4
- build_dec ............................................. 7
- collapse4 ............................................. 9
- collapse4_pop ....................................... 9
- collapse4_vap ....................................... 10
- collapse5 ............................................. 10
- collapse5_pop ....................................... 11
- collapse5_vap ....................................... 12
- construct_geoid ..................................... 12
- custom_geoid ......................................... 13
- fips_2000 ............................................ 14
- fips_2010 ............................................ 14
- fips_2020 ............................................ 15
- join_abb_ansi ....................................... 15
- join_abb_fips ....................................... 16
- join_abb_name ....................................... 16
- join_ansi_abb ....................................... 17
- join_ansi_fips ....................................... 18
- join_ansi_name ....................................... 18
- join_fips_abb ....................................... 19
- join_fips_ansi ....................................... 19
- join_fips_name ....................................... 20
- join_name_abb ....................................... 20
- join_name_ansi ....................................... 21
- join_name_fips ....................................... 22
- match_abb ............................................ 22
- match_ansi ........................................... 23
- match_fips ........................................... 23
- match_name ........................................... 24
- mt_county ............................................. 24
- recode_abb_ansi ..................................... 25
- recode_abb_fips ..................................... 26
- recode_abb_name ..................................... 26
- recode_ansi_abb ..................................... 27
- recode_ansi_fips ..................................... 27
- recode_ansi_name ..................................... 28
- recode_fips_abb ..................................... 28
- recode_fips_ansi ..................................... 29
- recode_fips_name ..................................... 30
- recode_name_abb ..................................... 30
- recode_name_ansi ..................................... 31
- recode_name_fips ..................................... 31
- stata .................................................... 32

Index 33
Add Entry to Renviron

Description

Adds a value to the Renvironment of the form name=value. Designed for flexibly adding API keys for future sessions. Defaults are set up for entering a Census API key to work with tidycensus.

Usage

```r
add_r_environ(
  value,
  name = "CENSUS_API_KEY",
  overwrite = FALSE,
  install = FALSE
)
```

Arguments

- `value`: Character. Value to add.
- `name`: Defaults to `CENSUS_API_KEY`. Character. Name to give `value`.
- `overwrite`: Defaults to `FALSE`. Boolean. Should existing item with name `name` in Renviron be overwritten?
- `install`: Defaults to `FALSE`. Boolean. Should this be added `~/.Renviron` file?

Value

`value`, invisibly

Examples

```r
## Not run:
add_r_environ("1234", "SECRET_API_KEY")

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

**Breakdown Census GEOID into Components**

**Description**

Breakdown Census GEOID into Components

**Usage**

`breakdown_geoid(.data, GEOID = "GEOID", area_type = "spine")`

**Arguments**

- `.data`: dataframe, tibble, or sf tibble
- `GEOID`: Column in `.data` with Census GEOID
- `area_type`: String, default is `"spine"` with type of GEOID. Options are `"spine"` for states, counties, tracts, block groups, and blocks. `"shd"` for lower state legislative districts, `"ssd"` for upper state legislative districts, `"cd"` for congressional districts, or `"zcta"` for zip code tabulation areas.

**Value**

`.data` with added identifying columns based on `area_type`

**Examples**

```r
data(mt_county)
mt_county <- mt_county %>% breakdown_geoid()
```

### build_acs

**Build Data from the Decennial Census**

**Description**

Creates a dataset, using the decennial census information, with the standard variables used for redistricting. Creates a stable base for getting data from tidycensus for common calls in redistricting.

```r
# Output columns are:
• GEOID: Geographic Identifier
• NAME: Name of County
• pop: total population
• pop_white: total population, Non-Hispanic White
• pop_black: total population, Non-Hispanic Black
• pop_hisp: total population, Hispanic
```
• pop_ian: total population, Non-Hispanic American Indian and Alaskan Native
• pop_asian: total population, Non-Hispanic Asian
• pop_nhpi: total population, Non-Hispanic Native Hawaiian and Pacific Islander
• pop_other: total population, Non-Hispanic Other
• pop_two: total population, Non-Hispanic Two Plus Races
• vap: voting age population
• vap_white: voting age population, Non-Hispanic White
• vap_black: voting age population, Non-Hispanic Black
• vap_hisp: voting age population, Hispanic
• vap_ian: voting age population, Non-Hispanic American Indian and Alaskan Native
• vap_asian: voting age population, Non-Hispanic Asian
• vap_nhpi: voting age population, Non-Hispanic Native Hawaiian and Pacific Islander
• vap_other: voting age population, Non-Hispanic Other
• vap_two: voting age population, Non-Hispanic Two Plus Races
• geometry: sf geometry

Arguments for geography are not checked, so will fail with tidyCensus errors if invalid. This is by design to avoid blocking usage that could become valid, especially following the 2020 Census data release.

Currently valid options for geography:

• 'state'
• 'county'
• 'tract'
• 'block group'
• 'block'
• 'county subdivision'
• 'zcta'
• 'congressional district'
• 'state legislative district (upper chamber)'
• 'state legislative district (lower chamber)'
• 'school district (unified)'

Full options for geography that may or may not be valid depending on year and geometry are listed at: Kyle Walker’s tidyCensus site.
Usage

build_acs(
  geography, 
  state, 
  county = NULL, 
  geometry = TRUE, 
  year = 2010, 
  survey = "acs5", 
  groups = "all"
)

mem_build_acs(
  geography, 
  state, 
  county = NULL, 
  geometry = TRUE, 
  year = 2010, 
  survey = "acs5", 
  groups = "all"
)

Arguments

geography Required. The geography level to use.
state Required. Two letter state postal code.
county Optional. Name of county. If not provided, returns blocks for the entire state.
geometry Defaults to TRUE. Whether to return the geometry or not.
year year, must be 2000, 2010, or 2020 (after August 2021)
survey whether the get estimates from the 5-year ('acs5'), 3-year ('acs3'), or 1-year ('acs1') survey. Default is 'acs5'.
groups defaults to 'all', which gets pop and vap. If 'pop', only gets pop. If 'vap', only gets vap. Any other strings default to 'all'.

Value
tibble with observations for each observation of the geography in the state or county. Data includes up to 3 sets of columns for each race or ethnicity category: population (pop), voting age population (vap), and citizen voting age population (cvap)

Examples

## Not run: 
# uses the Census API 
tb <- build_dec(geography = 'block', state = 'NY', county = 'Rockland', geometry = TRUE)

## End(Not run)
**Build Data from the Decennial Census**

**Description**

Creates a dataset, using the decennial census information, with the standard variables used for redistricting. Creates a stable base for getting data from tidycensus for common calls in redistricting.

**Usage**

```r
build_dec(
  geography,
  state,
  county = NULL,
  geometry = TRUE,
  year = 2020,
  groups = "all"
)
```

```r
cmem_build_dec(
  geography,
  state,
  county = NULL,
  geometry = TRUE,
  year = 2020,
  groups = "all"
)
```

**Arguments**

- `geography`: Required. The geography level to use.
- `state`: Required. Two letter state postal code.
- `county`: Optional. Name of county. If not provided, returns blocks for the entire state.
- `geometry`: Defaults to TRUE. Whether to return the geometry or not.
- `year`: year, must be 2000, 2010, or 2020 (after August 2021)
- `groups`: defaults to 'all', which gets pop and vap. If 'pop', only gets pop. If 'vap', only gets vap. Any other strings default to 'all'.

**Value**

tibble with observations for each observation of the geography in the state or county. Data includes 2 sets of columns for each race or ethnicity category: population (pop) and voting age population (vap)
Output columns are:

- GEOID: Geographic Identifier
- NAME: Name of County
- pop: total population
- pop_white: total population, Non-Hispanic White
- pop_black: total population, Non-Hispanic Black
- pop_hisp: total population, Hispanic
- pop_ain: total population, Non-Hispanic American Indian and Alaskan Native
- pop_asian: total population, Non-Hispanic Asian
- pop_nhpi: total population, Non-Hispanic Native Hawaiian and Pacific Islander
- pop_other: total population, Non-Hispanic Other
- pop_two: total population, Non-Hispanic Two Plus Races
- vap: voting age population
- vap_white: voting age population, Non-Hispanic White
- vap_black: voting age population, Non-Hispanic Black
- vap_hisp: voting age population, Hispanic
- vap_ain: voting age population, Non-Hispanic American Indian and Alaskan Native
- vap_asian: voting age population, Non-Hispanic Asian
- vap_nhpi: voting age population, Non-Hispanic Native Hawaiian and Pacific Islander
- vap_other: voting age population, Non-Hispanic Other
- vap_two: voting age population, Non-Hispanic Two Plus Races
- geometry: sf geometry

Arguments for geography are not checked, so will fail with tidycensus errors if invalid. This is by design to avoid blocking usage that could become valid, especially following the 2020 Census data release.

Currently valid options for geography:

- 'state'
- 'county'
- 'tract'
- 'block group'
- 'block'
- 'county subdivision'
- 'zcta'
- 'congressional district'
- 'state legislative district (upper chamber)'
- 'state legislative district (lower chamber)'
- 'school district (unified)'

Full options for geography that may or may not be valid depending on year and geometry are listed at: Kyle Walker’s tidycensus site.
Examples

```r
## Not run:
# uses the Census API
tb <- build_dec(geography = 'block', state = 'NY', county = 'Rockland', geometry = TRUE)

## End(Not run)
```

collapse4	 Collapse Full Race Categories into 4 Categories

Description

Collapses Other, AIAN, Asian, NHPI, and Two+ into other, by prefix.

Usage

```r
collapse4(.data, prefix)
```

Arguments

- `.data`: tibble, data.frame, or sf tibble
- `prefix`: The prefix(es) for the race categories. Must be a character vector.

Value

`.data` with columns collapsed

Examples

```r
data(mt_county)
mt_county <- mt_county %>% collapse4(prefix = c("pop_", "vap_"))
```

collapse4_pop	 Collapse Population Race Categories into 4 Categories

Description

Collapses Other, AIAN, Asian, NHPI, and Two+ into other.

Usage

```r
collapse4_pop(.data, prefix = "pop_")
```

Arguments

- `.data`: tibble, data.frame, or sf tibble
- `prefix`: Default is "pop_". The prefix for the race categories.
Value

.data with columns collapsed

Examples

data(mt_county)
mt_county <- mt_county %>% collapse4_pop()

collapse4_vap  

**Collapse Voting Age Population Race Categories into 4 Categories**

Description

Collapses Other, AIAN, Asian, NHPI, and Two+ into other.

Usage

collapse4_vap(.data, prefix = "vap_")

Arguments

- `.data`  
  tibble, data.frame, or sf tibble
- `prefix`  
  Default is vap_. The prefix for the race categories.

Value

.data with columns collapsed

Examples

data(mt_county)
mt_county <- mt_county %>% collapse4_vap()

collapse5  

**Collapse Full Race Categories into 5 Categories**

Description

Collapses Other, AIAN, NHPI, and Two+ into Other, by prefix.

Usage

collapse5(.data, prefix)
collapse5_pop

Arguments

.data tibble, data.frame, or sf tibble
prefix The prefix(es) for the race categories. Must be a character vector.

Value

.data with columns collapsed

Examples

data(mt_county)
mt_county <- mt_county %>% collapse5(prefix = c('pop_', 'vap_'))

collapse5_pop

Collapse Population Race Categories into 5 Categories

Description

Collapses Other, AIAN, NHPI, and Two+ into other.

Usage

collapse5_pop(.data, prefix = "pop_")

Arguments

.data tibble, data.frame, or sf tibble
prefix Default is pop_. The prefix for the race categories.

Value

.data with columns collapsed

Examples

data(mt_county)
mt_county <- mt_county %>% collapse5_pop()
### collapse5_vap

**Collapse Voting Age Population Race Categories into 5 Categories**

**Description**

Collapses Other, AIAN, NHPI, and Two+ into other.

**Usage**

```r
collapse5_vap(.data, prefix = "vap_")
```

**Arguments**

- `.data` tibble, data.frame, or sf tibble
- `prefix` Default is `vap_`. The prefix for the race categories.

**Value**

.data with columns collapsed

**Examples**

```r
data(mt_county)
mt_county <- mt_county %>% collapse5_vap()
```

### construct_geoid

**Create GEOID from Default Columns**

**Description**

Create GEOID from Default Columns

**Usage**

```r
construct_geoid(
  .data,
  area_type,
  state = "state",
  county = "county",
  tract = "tract",
  block_group = "block group",
  block = "block",
  cd = "cd",
  shd = "shd",
  ssd = "ssd",
  zcta = "zcta"
)
```
**custom_geoid**

Create a GEOID from Columns

**Description**

Create a GEOID from Columns

**Usage**

`custom_geoid(.data, ...)`

**Arguments**

- `.data` dataframe, tibble, or sf tibble
- `...` columns of `.data` in the order you want to make the GEOID

**Value**

`.data` with new column GEOID

**Examples**

```r
data(mt_county)
mt_county <- mt_county %>% breakdown_geoid()
mt_county <- mt_county %>% dplyr::select(-dplyr::all_of('GEOID'))
mt_county <- mt_county %>% construct_geoid()
```
fips_2010 Counties FIPS 2010

Description
Contains three columns:

- state: state FIPS
- county: county FIPS
- name: county name

Usage

data('fips_2010')

Value
tibble

Examples

data('fips_2010')

fips_2010 Counties FIPS 2010

Description
Contains three columns:

- state: state FIPS
- county: county FIPS
- name: county name

Usage

data('fips_2010')

Value
tibble

Examples

data('fips_2010')
### fips_2020

**Value**

- tibble

**Examples**

```r
data('fips_2010')
```

---

### Counties FIPS 2020

**Description**

Contains three columns:
- state: state FIPS
- county: county FIPS
- name: county name

**Usage**

```r
data('fips_2020')
```

**Value**

- tibble

**Examples**

```r
data('fips_2020')
```

---

### join_abb_ansi

**Join Abb by ANSI**

**Description**

Adds a column with state abbreviation joining by a column with state ansi

**Usage**

```r
join_abb_ansi(.data, .ansi)
```

**Arguments**

- `.data` data.frame or tibble
- `.ansi` column with state ansi
Value
.value with column .ansi replaced with state abbreviation

Examples
.data('stata')
  stata %>% join_abb_ansi(ansi)

join_abb_fips                
**Join Abb by FIPS**

Description
Adds a column with state abbreviation joining by a column with state fips

Usage
  join_abb_fips(.data, .fips)

Arguments
  .data     data.frame or tibble
  .fips     column with state fips

Value
.value with column .fips replaced with state abb

Examples
.data('stata')
  stata %>% join_abb_fips(fips)

join_abb_name                
**Join Abb by Name**

Description
Adds a column with state abbs joining by a column with state names

Usage
  join_abb_name(.data, .name)
**join_ansi_abb**

**Arguments**

- `.data` data.frame or tibble
- `.name` column with state name

**Value**

.data with column .name replaced with abbreviation

**Examples**

```r
data('stata')
stata %>% join_abb_name(name)
```

---

**join_ansi_abb**

*Join ANSI* by *Abb*

**Description**

Adds a column with state ansi joining by a column with state abbreviation

**Usage**

```r
join_ansi_abb(.data, .abb)
```

**Arguments**

- `.data` data.frame or tibble
- `.abb` column with state abbreviation

**Value**

.data with column .abb replaced with state ansi

**Examples**

```r
data('stata')
stata %>% join_ansi_abb(abb)
```
### join_ansi_fips

**Join ANSI by FIPS**

**Description**

Adds a column with state ansi joining by a column with state fips

**Usage**

```r
join_ansi_fips(.data, .fips)
```

**Arguments**

<table>
<thead>
<tr>
<th>.data</th>
<th>data.frame or tibble</th>
</tr>
</thead>
<tbody>
<tr>
<td>.fips</td>
<td>column with state fips</td>
</tr>
</tbody>
</table>

**Value**

.data with column .fips replaced with state ansi

**Examples**

```r
data('stata')
stata %>% join_ansi_fips(fips)
```

### join_ansi_name

**Join ANSI by Name**

**Description**

Adds a column with state ansi joining by a column with state name

**Usage**

```r
join_ansi_name(.data, .name)
```

**Arguments**

<table>
<thead>
<tr>
<th>.data</th>
<th>data.frame or tibble</th>
</tr>
</thead>
<tbody>
<tr>
<td>.name</td>
<td>column with state name</td>
</tr>
</tbody>
</table>

**Value**

.data with column .name replaced with ansi
join_fips_abb

Examples

data(' stata')
  stata %>% join_ansi_name(name)

Description

Adds a column with state fips joining by a column with state abbreviation

Usage

  join_fips_abb(.data, .abb)

Arguments

  .data  data.frame or tibble
  .abb   column with state abbreviation

Value

  .data with column .abb replaced with state name

Examples

  data(' stata')
  stata %>% join_fips_abb(abb)

join_fips_ansi

Description

Adds a column with state fips joining by a column with state ansi

Usage

  join_fips_ansi(.data, .ansi)

Arguments

  .data  data.frame or tibble
  .ansi  column with state ansi
**join_fips_ansi**

### Description

Adds a column with state fips joining by a column with state name.

### Usage

```
join_fips_ansi(.data, .ansi)
```

### Arguments

- `.data` data.frame or tibble
- `.ansi` column with state name

### Value

`.data` with column `.ansi` replaced with state fips

### Examples

```r
data('stata')
stata %>% join_fips_ansi(ansi)
```

---

**join_fips_name**

### Description

**Join FIPS by Name**

Adds a column with state fips joining by a column with state name.

### Usage

```
join_fips_name(.data, .name)
```

### Arguments

- `.data` data.frame or tibble
- `.name` column with state name

### Value

`.data` with column `.name` replaced with fips

### Examples

```r
data('stata')
stata %>% join_fips_name(name)
```

---

**join_name_abb**

### Description

**Join Name by Abb**

Adds a column with state name joining by a column with state abbreviation.

### Usage

```
join_name_abb(.data, .abb)
```

### Examples

```r
data('stata')
stata %>% join_name_abb(abb)
```
join_name_ansi

Arguments

.data data.frame or tibble
.abb column with state abbreviation

Value

.data with column .abb replaced with state name

Examples

data('stata')
stata %>% join_name_abb(abb)

join_name_ansi

Join Name by ANSI

Description

Adds a column with state name joining by a column with state ansi

Usage

join_name_ansi(.data, .ansi)

Arguments

.data data.frame or tibble
.ansi column with state ansi

Value

.data with column .ansi replaced with state name

Examples

data('stata')
stata %>% join_name_ansi(name)
### join_name_fips  
**Join Name by FIPS**

**Description**

Adds a column with state name joining by a column with state fips

**Usage**

```r
join_name_fips(.data, .fips)
```

**Arguments**

- `.data`  
  data.frame or tibble
- `.fips`  
  column with state fips

**Value**

`.data` with column `.fips` replaced with state name

**Examples**

```r
data(" stata")
stata %>% join_name_fips(fips)
```

### match_abb  
**Try to Match to State Abbreviation**

**Description**

Searches for an exact match and offers the best match if no exact match

**Usage**

```r
match_abb(state)
```

**Arguments**

- `state`  
  character with state FIPS, Abbreviation, Name, or ANSI

**Value**

Abbreviation if a match is found or `character(0)` if no match is found

**Examples**

```r
match_abb(\"NY\")  
match_abb(\"Ø1\")
```
**match_ansi**

**Try to Match to State ANSI**

**Description**

Searches for an exact match and offers the best match if no exact match.

**Usage**

```
match_ansi(state)
```

**Arguments**

- `state` character with state FIPS, Abbreviation, Name, or ANSI

**Value**

ANSI if a match is found or character(0) if no match is found

**Examples**

```
match_ansi('NY')
match_ansi('01')
```

---

**match_fips**

**Try to Match to State FIPS**

**Description**

Searches for an exact match and offers the best match if no exact match.

**Usage**

```
match_fips(state)
```

**Arguments**

- `state` character with state FIPS, Abbreviation, Name, or ANSI

**Value**

FIPS code if a match is found or character(0) if no match is found

**Examples**

```
match_fips('NY')
match_fips('01')
```
match_name  

**Try to Match to State Name**

**Description**

Searches for an exact match and offers the best match if no exact match

**Usage**

```
match_name(state)
```

**Arguments**

```
state character with state FIPS, Abbreviation, Name, or ANSI
```

**Value**

Name if a match is found or character(0) if no match is found

**Examples**

```
match_name('NY')
match_name('01')
```

---

mt_county  

**Montana County Data**

**Description**

- GEOID: Geographic Identifier
- NAME: Name of County
- pop: total population
- pop_white: total population, Non-Hispanic White
- pop_black: total population, Non-Hispanic Black
- pop_hisp: total population, Hispanic
- pop_aiian: total population, Non-Hispanic American Indian and Alaskan Native
- pop_asian: total population, Non-Hispanic Asian
- pop_nhpi: total population, Non-Hispanic Native Hawaiian and Pacific Islander
- pop_other: total population, Non-Hispanic Other
- pop_two: total population, Non-Hispanic Two Plus Races
- vap: voting age population
- vap_white: voting age population, Non-Hispanic White
\texttt{recode\_abb\_ansi}

- \texttt{vap\_black}: voting age population, Non-Hispanic Black
- \texttt{vap\_hisp}: voting age population, Hispanic
- \texttt{vap\_aiian}: voting age population, Non-Hispanic American Indian and Alaskan Native
- \texttt{vap\_asian}: voting age population, Non-Hispanic Asian
- \texttt{vap\_nhpi}: voting age population, Non-Hispanic Native Hawaiian and Pacific Islander
- \texttt{vap\_other}: voting age population, Non-Hispanic Other
- \texttt{vap\_two}: voting age population, Non-Hispanic Two Plus Races
- \texttt{geometry}: sf geometry

\textbf{Usage}

\begin{verbatim}
data('mt_county')
\end{verbatim}

\textbf{Value}

sf tibble with one observation for each county in Montana

\textbf{Examples}

\begin{verbatim}
data('mt_county')
\end{verbatim}

\begin{verbatim}
<table>
<thead>
<tr>
<th>recode_abb_ansi</th>
<th>Recode Abb by ANSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>.data</td>
<td>.ansi</td>
</tr>
</tbody>
</table>

\textbf{Description}

Replaces state ansi with state abbreviation

\textbf{Usage}

\begin{verbatim}
recode\_abb\_ansi(.data, .ansi)
\end{verbatim}

\textbf{Arguments}

- \texttt{.data} data.frame or tibble
- \texttt{.ansi} column with state ansi

\textbf{Value}

\begin{verbatim}
.data with column \texttt{.ansi} replaced with state abbreviation
\end{verbatim}

\textbf{Examples}

\begin{verbatim}
data('stata')
stata %>% recode\_abb\_ansi(ansi)
\end{verbatim}
recode_abb_fips        Recode Abb by FIPS

Description
Replaces state fips with state abb

Usage
recode_abb_fips(.data, .fips)

Arguments
  .data       data.frame or tibble
  .fips       column with state fips

Value
.data with column .fips replaced with state abb

Examples
  data(' stata')
  stata %>% recode_abb_fips(fips)

recode_abb_name        Recode Abb by Name

Description
Replaces state name with state abbreviation

Usage
recode_abb_name(.data, .name)

Arguments
  .data       data.frame or tibble
  .name       column with state name

Value
.data with column .name replaced with abbreviation
**recode_ansi_abb**

**Examples**

```r
data('stata')
stata %>% recode_abb_name(name)
```

---

**Description**

Replaces state abbreviation with state ansi

**Usage**

```r
recode_ansi_abb(.data, .abb)
```

**Arguments**

- `.data`: data.frame or tibble
- `.abb`: column with state abbreviation

**Value**

`.data` with column `.abb` replaced with state ansi

**Examples**

```r
data('stata')
stata %>% recode_ansi_abb(abb)
```

---

**recode_ansi_fips**

**Recode ANSI by FIPS**

**Description**

Replaces state fips with state ansi

**Usage**

```r
recode_ansi_fips(.data, .fips)
```

**Arguments**

- `.data`: data.frame or tibble
- `.fips`: column with state fips
Value
.data with column .fips replaced with state ansi

Examples
data('stata')
stata %>% recode_ansi_fips(fips)

---

recode_ansi_name   Recode ANSI by Name

Description
Replaces state name with state ansi

Usage
recode_ansi_name(.data, .name)

Arguments
.data data.frame or tibble
.name column with state name

Value
.data with column .name replaced with ansi

Examples
data('stata')
stata %>% recode_ansi_name(name)

---

recode_fips_abb   Recode FIPS by Abb

Description
Replaces state abbreviation with state fips

Usage
recode_fips_abb(.data, .abb)
recode_fips_ansi

Arguments

.data data.frame or tibble
.abb column with state abbreviation

Value

.data with column .abb replaced with state name

Examples

data('stata')
stata %>% recode_fips_abb(abb)

recode_fips_ansi  Recode FIPS by ANSI

Description

Replaces state ansi with state fips

Usage

recode_fips_ansi(.data, .ansi)

Arguments

.data data.frame or tibble
.ansi column with state ansi

Value

.data with column .ansi replaced with state fips

Examples

data('stata')
stata %>% recode_fips_ansi(ansi)
recode_fips_name | Recode FIPS by Name

**Description**
Replaces state name with state fips

**Usage**
recode_fips_name(.data, .name)

**Arguments**
- **.data**: data.frame or tibble
- **.name**: column with state name

**Value**
.data with column .name replaced with fips

**Examples**
```r
data('stata')
stata %>% recode_fips_name(name)
```

recode_name_abb | Recode Name by Abb

**Description**
Replaces state abbreviation with state name

**Usage**
recode_name_abb(.data, .abb)

**Arguments**
- **.data**: data.frame or tibble
- **.abb**: column with state abbreviation

**Value**
.data with column .abb replaced with state name
**recode_name_ansi**

**Examples**
```r
data(' stata')
stata %>% recode_name_abb(abb)
```

**Description**
Replaces state ansi with state name

**Usage**
```r
recode_name_ansi(.data, .ansi)
```

**Arguments**
- `.data` data.frame or tibble
- `.ansi` column with state ansi

**Value**
`.data` with column `.ansi` replaced with state name

**Examples**
```r
data(' stata')
stata %>% recode_name_ansi(name)
```

---

**recode_name_fips**  
*Recode Name by FIPS*

**Description**
Replaces state fips with state name

**Usage**
```r
recode_name_fips(.data, .fips)
```

**Arguments**
- `.data` data.frame or tibble
- `.fips` column with state fips
Value

.data with column .fips replaced with state name

Examples

data('stata')
stata %>% recode_name_fips(fips)

<table>
<thead>
<tr>
<th>stata</th>
<th>stata (State Data)</th>
</tr>
</thead>
</table>

Description

tibble with columns:

- fips: Federal Information Processing Standards codes
- abb: two letter postal abbreviations
- name: title case state name
- ansi: American National Standards Institute codes
- region: Census Regions (for 50 states and D.C.)
- division: Census Divisions (for 50 states and D.C.)

Usage

data('stata')

Value

tibble with state identifying information

Examples

data('stata')
Index

* **build**
  - build_acs, 4
  - build_dec, 7
* **collapse**
  - collapse4, 9
  - collapse4_pop, 9
  - collapse4_vap, 10
  - collapse5, 10
  - collapse5_pop, 11
  - collapse5_vap, 12
* **data**
  - fips_2000, 14
  - fips_2010, 14
  - fips_2020, 15
  - mt_county, 24
  - stata, 32
* **geoid**
  - breakdown_geoid, 4
  - construct_geoid, 12
  - custom_geoid, 13
* **join**
  - join_abb_ansi, 15
  - join_abb_fips, 16
  - join_abb_name, 16
  - join_ansi_abb, 17
  - join_ansi_fips, 18
  - join_ansi_name, 18
  - join_fips_abb, 19
  - join_fips_ansi, 19
  - join_fips_name, 20
  - join_name_abb, 20
  - join_name_ansi, 21
  - join_name_fips, 22
* **match**
  - match_abb, 22
  - match_ansi, 23
  - match_fips, 23
  - match_name, 24
* **recode**
  - recode_abb_ansi, 25
  - recode_abb_fips, 26
  - recode_abb_name, 26
  - recode_ansi_abb, 27
  - recode_ansi_fips, 27
  - recode_ansi_name, 28
  - recode_fips_abb, 28
  - recode_fips_ansi, 29
  - recode_fips_name, 30
  - recode_name_abb, 30
  - recode_name_ansi, 31
  - recode_name_fips, 31

add_r_environ, 3
breakdown_geoid, 4
build_acs, 4
build_dec, 7
collapse4, 9
collapse4_pop, 9
collapse4_vap, 10
collapse5, 10
collapse5_pop, 11
collapse5_vap, 12
fips_2000, 14
fips_2010, 14
fips_2020, 15
join_abb_ansi, 15
join_abb_fips, 16
join_abb_name, 16
join_ansi_abb, 17
join_ansi_fips, 18
join_ansi_name, 18
join_fips_abb, 19
join_fips_ansi, 19
join_fips_name, 20
join_name_abb, 20
join_name_ansi, 21
join_name_fips, 22
match_abb, 22
match_ansi, 23
match_fips, 23
match_name, 24
recode_abb_ansi, 25
recode_abb_fips, 26
recode_abb_name, 26
recode_ansi_abb, 27
recode_ansi_fips, 27
recode_ansi_name, 28
recode_fips_abb, 28
recode_fips_ansi, 29
recode_fips_name, 30
recode_name_abb, 30
recode_name_ansi, 31
recode_name_fips, 31

33
INDEX

join_name_abb, 20
join_name_ansi, 21
join_name_fips, 22

match_abb, 22
match_ansi, 23
match_fips, 23
match_name, 24
mem_build_acs (build_acs), 4
mem_build_dec (build_dec), 7
mt_county, 24

recode_abb_ansi, 25
recode_abb_fips, 26
recode_abb_name, 26
recode_ansi_abb, 27
recode_ansi_fips, 27
recode_ansi_name, 28
recode_fips_abb, 28
recode_fips_ansi, 29
recode_fips_name, 30
recode_name_abb, 30
recode_name_ansi, 31
recode_name_fips, 31

stata, 32