

Package ‘tidycensuskr’

July 23, 2026

Title Easy Access for South Korea Census Data and Boundaries

Version 0.3.0

Description Census and administrative data in South Korea are a basic source of quantitative and mixed-methods research for social and urban scientists. This package provides a 'sf' (Pebesma et al., 2024 <[doi:10.32614/CRAN.package.sf](https://doi.org/10.32614/CRAN.package.sf)>) based standardized workflow based on direct open API access to the major census and administrative data sources and pre-generated files in South Korea.

License MIT + file LICENSE

LazyData true

Depends R (>= 4.2.0)

Imports sf, dplyr, tidyr, kosis, utils, rlang

Suggests knitr, withr, purrr, rmarkdown, janitor, geofacet, ggplot2, biscale, cowplot, tmap, geodata, testthat (>= 3.0.0), ggspatial, spdep

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

URL <https://github.com/sigmafelix/tidycensuskr>,
<https://sigmafelix.github.io/tidycensuskr/>,
<https://sigmafelix.r-universe.dev/tidycensuskr/>

BugReports <https://github.com/sigmafelix/tidycensuskr/issues>

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Insang Song [aut, cre] (ORCID: <<https://orcid.org/0000-0001-8732-3256>>),
Sohyun Park [aut, ctb] (ORCID: <<https://orcid.org/0000-0002-1231-5662>>),
Hyesop Shin [aut, ctb] (ORCID: <<https://orcid.org/0000-0003-2637-7933>>)

Maintainer Insang Song <geoissong@snu.ac.kr>

Repository CRAN

Date/Publication 2026-07-23 06:20:02 UTC

Contents

adm2_sf_2020 2

anycensus 3

censuskor 5

detect_adm2_type 6

kr_grid_adm2_sgis_2020 7

load_districts 8

set_kosis_key 8

Index 10

adm2_sf_2020	South Korea Census Boundary in 2020
--------------	-------------------------------------

Description

District level boundary data in South Korea in 2020. adm2_code column can be used to join with an [anycensus\(\)](#) output.

Usage

adm2_sf_2020

Format

A sf object with 250 rows and 3 variables:

Details

- year Year of the census data, e.g., 2010, 2015, or 2020
- adm2_code Code of the district/municipal-level (Si-Gun-Gu) administrative unit
- geometry Geometry list-column

Source

- Statistical Geographic Information Service (SGIS)

anycensus

*Query Korean census data by administrative code and year***Description**

The function queries a long format census data frame ([censuskor](#)) for specific administrative codes (if provided)

Usage

```
anycensus(
  year = 2020,
  codes = NULL,
  type = c("population", "housing", "tax", "mortality", "economy", "medicine",
    "migration", "environment", "welfare", "social security", "landuse"),
  level = c("adm2", "adm3", "adm1"),
  adm2_type = c("all", "atn", "non"),
  aggregator = sum,
  weight_type = NULL,
  weight_column = NULL,
  geometry = FALSE,
  ...
)
```

Arguments

<code>year</code>	integer(1). One of 2010, 2015, or 2020.
<code>codes</code>	integer vector of admin codes (e.g. <code>c(11, 26)</code>) or character administrative area names (e.g. <code>c("Seoul", "Daejeon")</code>).
<code>type</code>	character vector. One or more of "population", "housing", "tax", "economy", "medicine", "migration", "environment", "mortality", "social security", or "landuse". Defaults to "population".
<code>level</code>	character(1). "adm1" for province-level, "adm2" for municipal-level, or "adm3" for neighborhood/town-level. Defaults to "adm2".
<code>adm2_type</code>	character(1). Which municipal code type to keep before returning adm2 results or aggregating to adm1. "all" keeps the current data, "atn" keeps autonomous/basic local government rows, and "non" keeps non-autonomous rows where they are available. For weighted aggregation with "atn", autonomous/basic local government rate rows are recalculated from their non-autonomous component rows using the supplied weights before returning adm2 or aggregating to adm1.
<code>aggregator</code>	function to aggregate values when <code>level = "adm1"</code> or when weighted <code>adm2_type = "atn"</code> recalculates autonomous/basic local government rows.
<code>weight_type</code>	character(1). Optional data type used to supply weights when aggregating. For example, rate variables in <code>type = "mortality"</code> can be aggregated with population weights from <code>weight_type = "population"</code> .

weight_column	character(1). Optional column name used as weights when aggregating. If weight_type = "population" and weight_column is omitted, "all households_total_prs" is used.
geometry	logical(1). If TRUE, returns an sf object with geometries attached. Defaults to FALSE.
...	additional arguments passed to the aggregator function. (e.g., na.rm = TRUE). When weight_type or weight_column is supplied, aggregator must accept a w argument such as <code>stats::weighted.mean()</code> .

Value

A data.frame object containing census data for the specified codes and year.

Note

Character names are resolved to their administrative codes before filtering. The 'wide' table is returned with separate columns for each class1 and class2 and unit (abbreviated whereof) combination.

Examples

```
# Query mortality data for adm2_code 21 (Busan)
anycensus(codes = 21, type = "mortality")

# Query population data for adm1 "Seoul" or "Daejeon"
anycensus(codes = c("Seoul", "Daejeon"), type = "housing", year = 2015)

# Query adm3 population data within Jongno-gu
anycensus(
  codes = 11010,
  type = "population",
  year = 2020,
  level = "adm3"
)

# Aggregate to adm1 level tax (province-level) using sum
anycensus(
  codes = c(11, 23, 31),
  type = "tax",
  year = 2020,
  level = "adm1",
  aggregator = sum,
  na.rm = TRUE
)

# Aggregate mortality rates to adm1 using population weights
anycensus(
  codes = "Seoul",
  type = "mortality",
  year = 2020,
  level = "adm1",
```

```

    aggregator = stats::weighted.mean,
    weight_type = "population",
    weight_column = "all households_total_prs",
    na.rm = TRUE
  )

  # Aggregate rates to adm1 after cleaning to autonomous/basic local governments
  anycensus(
    codes = "Gyeonggi-do",
    type = "mortality",
    year = 2020,
    level = "adm1",
    adm2_type = "atn",
    aggregator = stats::weighted.mean,
    weight_type = "population",
    weight_column = "all households_total_prs",
    na.rm = TRUE
  )

  # Recalculate adm2 rates after cleaning to autonomous/basic local governments
  anycensus(
    codes = "Gyeonggi-do",
    type = "mortality",
    year = 2020,
    level = "adm2",
    adm2_type = "atn",
    aggregator = stats::weighted.mean,
    weight_type = "population",
    weight_column = "all households_total_prs",
    na.rm = TRUE
  )

```

censuskor

South Korea Census Data

Description

District level data including tax, population, business entities, housing, economy, medicine and mortality in South Korea in 2010, 2015, and/or 2020. The available years and variables depend on the type of data.

Usage

```
censuskor
```

Format

A data.frame with 155,291 rows and 12 columns, out of which 103,626 rows for adm2, and 51,665 rows for adm3 data.

Details

- year Year of the census data, e.g., 2010, 2015, or 2020
- adm1 Name of the province-level (Si-Do) administrative unit
- adm1_code Code of the province-level (Si-Do) administrative unit
- adm2 Name of the district/municipal-level (Si-Gun-Gu) administrative unit
- adm2_code Code of the district/municipal-level (Si-Gun-Gu) administrative unit
- type Type of variable, e.g., "population", "tax", "mortality", "housing", "medicine", "migration", "environment", "welfare", or "economy"
- adm3 Name of neighborhood/town-level (Eup-Myeon-Dong) administrative unit
- adm3_code Code of the neighborhood/town-level (Eup-Myeon-Dong) administrative unit
- class1 First-level classification of the variable depending on the type
- class2 Second-level classification of the variable depending on the type
- unit Unit of measurement for the variable
- value Value of the variable

Note

NA values in the value field indicate that the data was omitted or suppressed. We kept these NA values as-is to reflect the original data from the source. For temporal comparison, province names in adm1 field are standardized to the common names with no suffix in metropolitan cities and "-do" suffix in provinces. For example, "Seoul" instead of "Seoul Metropolitan City", and "Jeollabuk-do" instead of "Jeonbuk State". "KRW" in the unit field stands for South Korean Won. Values are as-is unless otherwise noted in the unit field (e.g., "per 100k population" or "million KRW").

Source

- KOSIS (Korean Statistical Information Service)

detect_adm2_type	<i>Detect adm2 type from adm2_code field then return the exact codes</i>
------------------	--

Description

adm2_code often refers to the codes of autonomous _Gu_s or non-autonomous _Gu_s. The head table of the data frame may contain either or both of the two types of codes. This function detects the type of the codes in the adm2_code field and returns the exact codes accordingly.

Usage

```
detect_adm2_type(df, year = NULL, mode = "non", adm2_code = "adm2_code")
```

Arguments

df	A head data frame containing the full dataset. i.e., censuskor
year	The year for which to filter the data. If not specified, the function will use the data.frame as is.
mode	A character vector of "atn" (autonomous) and "non" (non-autonomous).
adm2_code	A character vector of adm2_code field Default is "adm2_code".

Value

filtered data frame with exact codes

Examples

```
# Load 2020 census population
pop20 <- anycensus(year = 2020, type = "population")
pop20_nonauto <- detect_adm2_type(pop20, mode = "non")
pop20_auto <- detect_adm2_type(pop20, mode = "atn")
unique(pop20_nonauto$adm2_code)
unique(pop20_auto$adm2_code)
```

kr_grid_adm2_sgis_2020

geofacet Grid for South Korea Administrative Districts (SGIS Standard, 2020)

Description

A geofacet grid for South Korea administrative districts (*Si-Gun-Gu*) based on the Statistical Geographic Information Service (SGIS) standard in 2020. Non-autonomous districts in cities are retained as separate entities. This grid can be used with the [geofacet](#) package to create faceted visualizations based on geographic layout.

Usage

```
kr_grid_adm2_sgis_2020
```

Format

A data.frame with 250 rows and 6 variables

Details

- name Name of the district/municipal-level (Si-Gun-Gu) administrative unit
- code SGIS code of the district/municipal-level (Si-Gun-Gu) administrative unit
- row Row position in the geofacet grid
- col Column position in the geofacet grid

Source

- Statistical Geographic Information Service (SGIS)
- GitHub username chichead in [GitHub geofacet issue page](#)

load_districts	<i>Load district boundaries for a specific year</i>
----------------	---

Description

Load district boundaries for a specific year

Usage

```
load_districts(year = 2020, level = "adm2")
```

Arguments

year	The year for which to load district boundaries (2010, 2015, or 2020)
level	The administrative level, one of "adm2" and "adm3"

Value

An sf object containing district boundaries for the specified year

Note

This function requires the `tidycensuskr.sf` package to be installed. No explicit dependency is defined; but users should install the package following the instructions at `vignette('v01_intro')` or more succinctly: `install.packages('tidycensuskr.sf', repos = 'https://sigmafelix.r-universe.dev')`

set_kosis_key	<i>Set KOSIS API Key from a File</i>
---------------	--------------------------------------

Description

This function reads a KOSIS API key from a specified file and sets it for use in KOSIS API calls.

Usage

```
set_kosis_key(file)
```

Arguments

file	A character string specifying the path to the file containing the KOSIS API key.
------	--

Details

The file should contain the API key as a single line of text. If the file does not exist, an error will be raised.

Value

No return value. A message will be printed to confirm that the key has been set.

Index

* datasets

adm2_sf_2020, [2](#)

censuskor, [5](#)

kr_grid_adm2_sgis_2020, [7](#)

adm2_sf_2020, [2](#)

anycensus, [3](#)

anycensus(), [2](#)

censuskor, [3](#), [5](#)

detect_adm2_type, [6](#)

kr_grid_adm2_sgis_2020, [7](#)

load_districts, [8](#)

set_kosis_key, [8](#)

stats::weighted.mean(), [4](#)