Package: tidycensuskr (via r-universe)

November 14, 2026

Title Easy Access for South Korea Census Data and Boundaries

Version 0.4.2

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>) 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

LazyDataCompression xz

Depends R (>= 4.2.0)

Roxygen list(markdown = TRUE)

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

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-11-14 11:34:49 UTC

RemoteUrl https://github.com/sigmafelix/tidycensuskr

RemoteRef HEAD

RemoteSha 8c4753c1ae365d01fca41ee0bb2a39fc5f2f7f01

Contents

adm2_sf_2020 2
anycensus 3
catalog 5
censuskor 6
detect_adm2_type 7
kr_grid_adm2_sgis_2020 8
load_districts 9
set_kosis_key 9
spatial_weights 10

Index **12**

`adm2_sf_2020` *South Korea Census Boundary in 2020*

Description

District level boundary data in 2020, repaired using 2015 historical adjacency evidence and 2020 geometry only. Original attributes and row order are retained. Polygon overlaps are removed and the Guro/Yeongdeungpo/Dongjak/Gwanak junction is one shared point. Eight unresolved historical reference pairs and two historically unconfirmed contacts remain documented for review.

Usage

```
adm2_sf_2020
```

Format

An sf object with 250 rows and three fields: `year`, `adm2_code`, and `geometry`. Coordinates use EPSG:5179.

Details

Validated 2010, 2015 and 2025 boundaries are registered datasets in the companion `tidycensuskr.sf` package; `load_districts()` selects their package automatically. Codes can be used to join `anycensus()` output.

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

year	integer(1). One of 2010, 2015, 2020, or 2025. 2025 data is currently available only for ADM3 population and housing.
codes	integer vector of admin codes (e.g. c(11, 26)) or character administrative area names (e.g. c("Seoul", "Daejeon")).
type	character vector. One or more of "population", "housing", "tax", "economy", "medicine", "migration", "environment", "mortality", "social security", or "landuse". Defaults to "population".
level	character(1). "adm1" for province-level, "adm2" for municipal-level, or "adm3" for neighborhood/town-level. ADM3 population and housing data is available for 2015, 2020, and 2025; ADM3 economy and mortality data is available for 2020. Defaults to "adm2".
adm2_type	character(1). Municipal level to return or aggregate. For a city and its non-autonomous Gu with the same indicator, "all" removes the source city count when Gu values are available, but retains city and Gu rates; the city count is NA in a mixed wide row. "atn" returns city rows, summing available Gu counts and retaining the city rate; "non" returns Gu rows where they are available. Incomplete Gu sums emit a warning. When a city rate is missing, "atn" uses the Gu mean; with weights, it recalculates city rates from Gu values. Rates are identified by units per 100,000 population, births per 1,000, births per woman, and percent; other units are treated as additive counts. Selecting a city code with

	"atn" or "non" also selects its Gu. "all" retains both city and Gu rates, so choose the intended geographic level and a suitable aggregator when aggregating rates to "adm1".
aggregator	function to aggregate values when level = "adm1" or when weighted adm2_type = "atn" recalculates autonomous/basic local government rows.
weight_type	character(1). Optional data type used to supply weights when aggregating. For example, rate variables in type = "mortality" can be aggregated with population weights from weight_type = "population".
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 ADM2 geometries attached. Geometry is available for 2010, 2015, and 2020; 2025 census data is tabular only. 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 stats::weighted.mean().

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"
)

# Query 2025 neighborhood-level housing data (without geometry)
## Not run:
anycensus(
  codes = 11010,
  type = "housing",
```

```
  year = 2025,
  level = "adm3"
)

## End(Not run)

# 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
## Not run:
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
)

## End(Not run)

# 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
)
```

---

catalog *Catalog available census variables*

Description

Return the combinations of year, geographic level, `type`, `class1`, and `class2` available in `censuskor`, with units and row counts.

Usage

```
catalog()
```

Value

A data frame with columns `year`, `level`, `type`, `class1`, `class2`, `unit`, and `n`.

Examples

```
catalog()
```

`censuskor` *South Korea Census Data*

Description

District and neighborhood level data including tax, population, business entities, housing, economy, medicine and mortality in South Korea in 2010, 2015, 2020, and 2025. The available years and variables depend on the type and geographic level of data.

Usage

```
censuskor
```

Format

A data.frame with 836,707 rows and 12 columns, out of which 103,572 rows for adm2, and 733,135 rows for adm3 data.

Details

- year Year of the census data, e.g., 2010, 2015, 2020, or 2025
- 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` *Detect adm2 type from adm2_code field then return the exact codes*

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, each year in the data frame is processed separately.
`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 *Load district boundaries for a specific year*

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, 2020, or 2025)
level	The administrative level, one of "adm2" and "adm3"

Value

An sf object containing district boundaries for the specified year

Note

Other ADM2 years and ADM3 boundaries require the tidycensuskr.sf package to be installed. No explicit dependency is defind; 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 *Set KOSIS API Key from a File*

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.

`spatial_weights` *Binary district spatial-weight matrices*

Description

Symmetric, unstandardized integer matrices for 2020. `w_mat_YYYY` combines exact Queen adjacency with documented marine road bridges and subsea tunnels open by December 31 of that year. `w_mat_o_YYYY` contains exact Queen adjacency alone.

Usage

```
w_mat_2020

w_mat_o_2020
```

Format

Square 250 by 250 integer matrices. Values are 0 or 1, the diagonal is zero, and dimnames are five-digit SGIS ADM2 codes.

An object of class `matrix` (inherits from `array`) with 250 rows and 250 columns.

An object of class `matrix` (inherits from `array`) with 250 rows and 250 columns.

Details

Rows follow adm2_sf_2020, not sorted code order. Align observations by code before analysis. The 2010, 2015 and 2025 versions are registered in `tidycensuskr.sf`, with their corresponding boundaries.

Parallel facilities between the same districts contribute one edge. Within-district facilities contribute no diagonal entries. Ordinary inland river bridges, dykes, railway tunnels and facilities opening after the cutoff are not added. Marine bay and estuary crossings are included. Islands may have zero neighbors; row standardization is left to the user.

Exact Queen adjacency uses `sf::st_touches()` with no snapping tolerance. The 2020 coverage has no polygon overlaps. Unresolved historical contacts are documented with the repair artifacts. These matrices represent the supplied geometry plus the researched transport inventory, rather than a guarantee of complete real-world connectivity.

The connection inventory and code-order metadata are distributed under `system.file("extdata", "spatial_weights", package = "tidycensuskr.sf")`.

Source

SGIS/KOSTAT boundaries; Ministry of Land, Infrastructure and Transport, local government and National Archives facility records.

Examples

```
data(w_mat_2020)
data(adm2_sf_2020)
stopifnot(identical(rownames(w_mat_2020), as.character(adm2_sf_2020$adm2_code)))
```

Index

∗ **datasets**
adm2_sf_2020, 2
censuskor, 6
kr_grid_adm2_sgis_2020, 8
spatial_weights, 10

adm2_sf_2020, 2, *10*
anycensus, 3
anycensus(), *2*

catalog, 5
censuskor, *3*, 6

detect_adm2_type, 7

kr_grid_adm2_sgis_2020, 8

load_districts, 9
load_districts(), *2*

set_kosis_key, 9
spatial_weights, 10
stats::weighted.mean(), *4*

w_mat_2020 *(*spatial_weights*)*, 10
w_mat_o_2020 *(*spatial_weights*)*, 10