Package 'inedemogR'

August 23, 2026

Title Tidy Access to Spanish INE Demographic Data

Version 0.1.0

Description Provides tidy, harmonized access to demographic data from the Spanish National Statistics Institute (INE) fenomenos demograficos domain, including population, births, and deaths, retrieved live via the official 'ineapir' API wrapper, with optional spatial integration at municipality and province level via 'mapSpain'. Mortality indicators follow the Human Mortality Database Methods Protocol, including the average age at death in infancy method of Andreev and Kingkade (2015) <doi:10.4054/DemRes.2015.33.13>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports dplyr, ggplot2, httr2, ineapir, mapSpain, purrr, readr, rlang, scales, sf, stringi, stringr, tibble, tidyr

Suggests mockery, R.rsp, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Depends R (>= 4.1.0)

LazyData true

Collate 'helpers.R' 'death_rates.R' 'life_tables.R' 'abridged_life_tables.R' 'births.R' 'data.R' 'deaths.R' 'decomposition.R' 'download_ine_data.R' 'exposure.R' 'get_ine_demog.R' 'get_ine_geo.R' 'hmd_write.R' 'indicators.R' 'inedemogR-package.R' 'lexis.R' 'list_ine_indicators.R' 'plot_demog.R' 'plot_ine_map.R' 'population.R' 'update_ine_data.R'

VignetteBuilder R.rsp

NeedsCompilation no

Author J. R. Caro-Barrera [aut, cre, cph]

Maintainer J. R. Caro-Barrera <jrcaro@uco.es>

Repository CRAN

Date/Publication 2026-08-23 10:30:08 UTC

Contents

`age_dependency_ratio` *Age dependency ratios*

Description

Computes youth, old-age, and total age dependency ratios per province and year from age-disaggregated population data. Requires an age breakdown, so it operates on `get_ine_population()`'s output, not `get_ine_demog()`'s `population_total` indicator (which has no `age` column and cannot be used here).

Usage

```
age_dependency_ratio(pop_df, young_max = 14L, old_min = 65L)
```

Arguments

`pop_df`	Population tibble as returned by `get_ine_population()$data` (columns `nuts3_code`, `province_name`, `year`, `age`, `total`).
`young_max`	Integer, the maximum age counted as "young" (default `14`, i.e. ages 0-14).
`old_min`	Integer, the minimum age counted as "old" (default `65`).

Value

A tibble with `nuts3_code`, `province_name`, `year`, `youth_dependency_ratio`, `old_age_dependency_ratio`, `total_dependency_ratio` (all per 100 working-age population).

Examples

```
pop <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 20, 70), total = c(100, 300, 150)
)
age_dependency_ratio(pop)
```

`age_specific_fertility_rate` *Age-specific fertility rates (ASFR)*

Description

Computes age-specific fertility rates (births per 1,000 women of that age) per province, year, and single year of age, joining `get_ine_births_by_age()` and `get_ine_population()` output. This is the schedule underlying `total_fertility_rate()`, `mean_age_at_childbearing()`, `gross_reproduction_rate()`, and `net_reproduction_rate()` - unlike `general_fertility_rate()`, which collapses reproductive age into one aggregate rate, `age_specific_fertility_rate()` keeps the full age schedule, the input a true total fertility rate requires.

Usage

```
age_specific_fertility_rate(
  births_age_df,
  pop_df,
  age_min = 15L,
  age_max = 49L
)
```

Arguments

`births_age_df`	Tibble as returned by `get_ine_births_by_age()` `$data` (columns `nuts3_code`, `province_name`, `year`, `age`, `female`, `total`).
`pop_df`	Population tibble as returned by `get_ine_population()` `$data` (columns `nuts3_code`, `year`, `age`, `female`).
`age_min`	Integer, the youngest single-year age included (default `15`).
`age_max`	Integer, the oldest single-year age included (default `49`).

Value

A tibble with `nuts3_code`, `province_name`, `year`, `age`, `asfr` (all births per 1,000 women of that age), `asfr_female` (births of female newborns per 1,000 women of that age - the input `gross_reproduction_rate()` and `net_reproduction_rate()` need).

Examples

```
births_age <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(20, 30), female = c(10, 25), total = c(20, 50)
)
pop <- tibble::tibble(
  nuts3_code = "ES111", year = 2023, age = c(20, 30), female = c(2000, 2500)
)
age_specific_fertility_rate(births_age, pop)
```

aging_index *Aging index*

Description

Computes the aging index (elderly per 100 children) per province and year from age-disaggregated population data. Requires an age breakdown, so it operates on `get_ine_population()`'s output, not `get_ine_demog()`'s `population_total` indicator.

Usage

```
aging_index(pop_df, young_max = 14L, old_min = 65L)
```

Arguments

`pop_df`	Population tibble as returned by `get_ine_population()$data` (columns `nuts3_code`, `province_name`, `year`, `age`, `total`).
`young_max`	Integer, the maximum age counted as "young" (default `14`, i.e. ages 0-14).
`old_min`	Integer, the minimum age counted as "old" (default `65`).

Value

A tibble with `nuts3_code`, `province_name`, `year`, `aging_index`.

Examples

```
pop <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 20, 70), total = c(100, 300, 150)
)
aging_index(pop)
```

andreev_kingkade_a0 *Andreev-Kingkade a0 (mean age at death in the first year of life)*

Description

Implements the Andreev-Kingkade (2015) piecewise regression specified in HMD Methods Protocol V6 (replacing the older Coale-Demeny approximation).

Usage

```
andreev_kingkade_a0(m0, sex = c("female", "male"))
```

Arguments

m0	Central death rate at age 0.
sex	"female" or "male" (case-insensitive).

Value

Numeric a0 value.

Examples

```
andreev_kingkade_a0(0.004, "female")
andreev_kingkade_a0(0.15, "male")
```

---

birth_death_ratio *Birth-to-death ratio*

Description

Computes the birth-to-death ratio (live births per death) per geography and year. Unlike age_dependency_ratio()/aging_i sex_ratio(), this needs no age breakdown, so it operates directly on get_ine_demog()'s totals (System A) rather than get_ine_population() (System B) — fetch both births_total and deaths_total in one call and pass the result straight through.

Usage

```
birth_death_ratio(df)
```

Arguments

df	Output of get_ine_demog(indicator = c("births_total", "deaths_total")) (columns GEOID, NAME, year, births_total, deaths_total).

Value

A tibble with GEOID, NAME, year, birth_death_ratio. A value of 2 means 2 live births per death; 0.5 means 1 birth per 2 deaths.

Examples

```
df <- tibble::tibble(
  GEOID = c("15", "28"), NAME = c("A Coruna", "Madrid"), year = 2023,
  births_total = c(3000, 30000), deaths_total = c(6000, 25000)
)
birth_death_ratio(df)
```

`build_abridged_life_table` *Build an abridged (5-year age group) period life table*

Description

Constructs a full abridged period life table (age groups `"00-04"`, `"05-09"`, ..., `"95-99"`, `"100+"`) directly from grouped central death rates, per the standard discrete abridged life-table method (Chiang's nqx formula; Preston, Heuveline & Guillot 2001, Ch. 3). Complements `build_life_table()`'s single-year (1x1) tables - use this when comparing against other agencies' published abridged tables, or when only grouped-age data is available. The youngest group (`"00-04"`) is the one exception to using `mx_5x1` alone: it is built from an exact single-year sub-table using `mx_1x1`'s Andreev-Kingkade `a0` (see `andreev_kingkade_a0()`), avoiding the external nax(0-4) regression a from-mx_5x1-alone method would otherwise need for that interval's sharp infant-mortality gradient.

Usage

```
build_abridged_life_table(mx_1x1, mx_5x1, sex = c("female", "male"))
```

Arguments

`mx_1x1`	tibble with columns `age` (single years, at least `0:4`), `mx`, for one year and one sex - see `compute_death_rates()`'s `mx_1x1` output, filtered to one province/year/sex.
`mx_5x1`	tibble with columns `age_group` (`"00-04"`, `"05-09"`, ..., `"100+"`), `mx`, for the same year and sex - see `compute_death_rates()`'s `mx_5x1` output, filtered the same way.
`sex`	`"female"` or `"male"` - selects the Andreev-Kingkade a0 branch for the youngest group, same as `build_life_table()`.

Value

tibble: `age_group`, `age_start`, `n` (interval width, `NA` for the open interval), `mx`, `ax`, `qx`, `lx`, `dx`, `Lx`, `Tx`, `ex`.

Examples

```
mx1 <- data.frame(age = 0:4, mx = c(0.004, 0.0003, 0.0002, 0.0002, 0.0002))
mx5 <- data.frame(
  age_group = c("00-04", "05-09", "100+"), mx = c(0.0006, 0.0001, 0.35)
)
build_abridged_life_table(mx1, mx5, "female")
```

`build_abridged_life_tables` *Build abridged period life tables for every province*

Description

Constructs female, male, and both-sex abridged (5-year age group) period life tables for every province and year present in `mx_5x1`, from `compute_death_rates()`'s `mx_1x1`/`mx_5x1` output. See `build_abridged_life_table()` for the method; `build_life_tables()` for the single-year (1x1) equivalent this complements.

Usage

```
build_abridged_life_tables(mx_1x1, mx_5x1)
```

Arguments

`mx_1x1`	Output of the `mx_1x1` element of `compute_death_rates()`.
`mx_5x1`	Output of the `mx_5x1` element of `compute_death_rates()`.

Value

`list(fltper, mltper, bltper, qc)`: each a tibble with `nuts3_code`, `province_name`, `year`, `age_group`, `age_start`, `n`, `mx`, `ax`, `qx`, `lx`, `dx`, `Lx`, `Tx`, `ex`; `qc` is a named list of `validate_abridged_life_table()` results.

Examples

```
mx_1x1 <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = 0:4, mx_female = c(0.004, 0.0003, 0.0002, 0.0002, 0.0002),
  mx_male = c(0.005, 0.0004, 0.0003, 0.0003, 0.0003),
  mx_total = c(0.0045, 0.00035, 0.00025, 0.00025, 0.00025)
)
mx_5x1 <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age_group = c("00-04", "100+"),
  mx_female = c(0.0006, 0.35), mx_male = c(0.0008, 0.4), mx_total = c(0.0007, 0.38)
)
alt <- build_abridged_life_tables(mx_1x1, mx_5x1)
alt$fltper
```

build_life_table *Build a period life table from a mortality schedule*

Description

Constructs a full 1x1 period life table from a smoothed mx series spanning ages `0:110`, per HMD Methods Protocol V6: Andreev-Kingkade a0 at age 0, a(1) = 0.4, a(x) = 0.5 elsewhere; the open age interval closes with `qx = 1` and `Lx = lx/mx`.

Usage

```
build_life_table(mx_df, sex = c("female", "male"))
```

Arguments

`mx_df`	tibble with columns `age`, `mx` for one year and one sex (typically Kannisto-smoothed at older ages, see `build_life_tables()`).
`sex`	`"female"` or `"male"` — selects the Andreev-Kingkade a0 branch.

Value

tibble: `age`, `mx`, `qx`, `ax`, `lx`, `dx`, `Lx`, `Tx`, `ex`.

Examples

```
mx <- data.frame(age = 0:5, mx = c(0.004, 0.0003, 0.0002, 0.0002, 0.0002, 0.0002))
build_life_table(mx, "female")
```

build_life_tables *Build period life tables for every province*

Description

Constructs female, male, and both-sex 1x1 period life tables for every province and year present in `mx_1x1`, per SHMD Protocol Section 12, Step 6: Kannisto smoothing/extrapolation of old-age mortality (fitted on ages 80-95, extended to 110+), then standard life-table recursion with the Andreev-Kingkade a0.

Usage

```
build_life_tables(mx_1x1)
```

Arguments

`mx_1x1`	Output of the `mx_1x1` element of `compute_death_rates()`.

Value

`list(fltper, mltper, bltper, qc)`: each a tibble with `nuts3_code`, `province_name`, `year`, `age`, `mx`, `qx`, `ax`, `lx`, `dx`, `Lx`, `Tx`, `ex`; `qc` is a named list of `validate_life_table()` results, one per province/year/sex table that failed validation.

Examples

```
mx_1x1 <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = 0:5,
  mx_female = c(0.004, 0.0003, 0.0002, 0.0002, 0.0002, 0.0002),
  mx_male = c(0.005, 0.0004, 0.0003, 0.0003, 0.0003, 0.0003),
  mx_total = c(0.0045, 0.00035, 0.00025, 0.00025, 0.00025, 0.00025)
)
lt <- build_life_tables(mx_1x1)
lt$fltper
```

---

compute_death_rates *Compute central death rates for every province*

Description

Computes 1x1 (single-year age/period), 5x1 (5-year age group), and age-standardised (ESP 2013) death rates for every province present in both `deaths` and `exposure`, per SHMD Protocol Section 12, Step 5.

Usage

```
compute_death_rates(deaths, exposure)
```

Arguments

`deaths`	Output of the `data_provinces` element of `get_ine_deaths()`.
`exposure`	Output of the `data` element of `compute_exposure()`.

Value

`list(mx_1x1, mx_5x1, asdr, qc)`, each a tibble with `nuts3_code`, `province_name` plus the rate columns; `qc` is the output of `validate_death_rates()` run on `mx_1x1`.

Examples

```
deaths <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 1), female = c(1, 0), male = c(1, 1), total = c(2, 1)
)
exposure <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
```

```
  age = c(0, 1), female = c(500, 490), male = c(510, 505), total = c(1010, 995)
)
rates <- compute_death_rates(deaths, exposure)
rates$mx_1x1
rates$asdr
```

`compute_exposure` *Compute exposure-to-risk for every province*

Description

Computes E(x,t) = 1/2[P(x,t) + P(x,t+1)] + 1/2[D_upper(x,t) - D_lower(x,t)] for every province present in both `population` and `deaths`, per SHMD Protocol Section 12, Step 4. INE's Tempus3 death tables don't carry a birth-cohort split, so the Lexis triangle is derived via the documented 50/50 fallback (HMD Methods Protocol V6 Appendix A) unless `deaths` already carries a `cohort` column.

Usage

```
compute_exposure(population, deaths)
```

Arguments

`population`	Output of the `data` element of `get_ine_population()`.
`deaths`	Output of the `data_provinces` element of `get_ine_deaths()`.

Details

Death registration lags population estimates by about a year, so `population` commonly includes a most-recent year (e.g. a Jan-1 stock snapshot) with no corresponding rows anywhere in `deaths` yet. That year is still used as the `P(x,t+1)` boundary for computing the *previous* year's exposure, but is **not** included in the returned exposure rows itself — treating an entirely-absent year as zero deaths (rather than missing data) would mechanically produce zero mortality for that year, and downstream a nonsensical inflated life expectancy.

Value

`list(data, qc)`: `data` is a tibble with columns `nuts3_code`, `province_name`, `year`, `age`, `female`, `male`, `total` (exposure), `is_boundary_year`, `is_missing_source_pop`; `qc` is the output of `validate_exposure()`.

Examples

```
population <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna",
  year = c(2022, 2023), age = 0,
  female = c(100, 105), male = c(105, 110), total = c(205, 215)
)
deaths <- tibble::tibble(
```

```
  nuts3_code = "ES111", province_name = "A Coruna",
  year = c(2022, 2023), age = 0,
  female = c(1, 1), male = c(1, 2), total = c(2, 3)
)
compute_exposure(population, deaths)
```

`crude_birth_rate` *Crude birth rate*

Description

Computes the crude birth rate (births per 1,000 population) per province and year, joining `get_ine_births()` and `get_ine_population()` output.

Usage

```
crude_birth_rate(births_df, pop_df)
```

Arguments

`births_df`	Births tibble as returned by `get_ine_births()$data` (columns `nuts3_code`, `province_name`, `year`, `total`).
`pop_df`	Population tibble as returned by `get_ine_population()$data` (columns `nuts3_code`, `year`, `age`, `total`).

Details

This is a *crude* birth rate (total births / total population), **not** a total fertility rate (TFR). `get_ine_births()` itself has no age-of-mother breakdown, so `crude_birth_rate()` cannot compute a TFR on its own — for a true TFR, use `get_ine_births_by_age()` with `age_specific_fertility_rate()`/`total_fertility_rate()` instead. Do not substitute `crude_birth_rate()` for TFR in fertility analysis.

Value

A tibble with `nuts3_code`, `province_name`, `year`, `cbr`.

Examples

```
births <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023, total = 100
)
pop <- tibble::tibble(
  nuts3_code = "ES111", year = 2023, age = c(0, 1), total = c(5000, 5000)
)
crude_birth_rate(births, pop)
```

crude_death_rate *Crude death rate*

Description

Computes the crude death rate (deaths per 1,000 population) per province and year, joining `get_ine_deaths()` and `get_ine_population()` output. The symmetric counterpart to `crude_birth_rate()` — see `rate_of_natural_increase()` to combine the two.

Usage

```
crude_death_rate(deaths_df, pop_df)
```

Arguments

deaths_df	Deaths tibble as returned by `get_ine_deaths()` `$data_provinces` (columns `nuts3_code`, `province_name`, `year`, `age`, `total`). Unlike `crude_birth_rate()`'s `births_df`, this is age-specific, so it is summed over age internally before computing the rate.
pop_df	Population tibble as returned by `get_ine_population()$data` (columns `nuts3_code`, `year`, `age`, `total`).

Value

A tibble with `nuts3_code`, `province_name`, `year`, `cdr`.

Examples

```
deaths <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 1), total = c(1, 49)
)
pop <- tibble::tibble(
  nuts3_code = "ES111", year = 2023, age = c(0, 1), total = c(5000, 5000)
)
crude_death_rate(deaths, pop)
```

decompose_life_expectancy *Decompose a difference in life expectancy at birth by age*

Description

Attributes the difference in life expectancy at birth (`e0`) between two life tables to age-specific contributions, using either Arriaga's (1984) or Pollard's (1988) method. Arriaga's is an exact discrete decomposition: `sum(contribution)` recovers `e2(0) - e1(0)` exactly. Pollard's is exact only in the continuous limit; applied to a single-year life table, `sum(contribution)` is a close but not exact approximation of `e2(0) - e1(0)` - the approximation error grows with how steeply mortality changes with age (a few percent of the total gap under a fast-rising hazard is possible), so treat Arriaga's (the default) as the primary result and Pollard's as a cross-check, per Ponnapalli (2005)'s finding that the two methods' age patterns agree closely without being identical. Use this to answer "how much of the life-expectancy gap between province A and B (or between year Y1 and Y2) comes from mortality at each age?" - complementing `life_expectancy_summary()`/ `life_expectancy()`, which report `e0`/`e65` but don't explain *why* two values differ.

Usage

```
decompose_life_expectancy(lt1, lt2, method = c("arriaga", "pollard"))
```

Arguments

`lt1`	A single life table (the "before"/reference table) - one province-year-sex slice of `build_life_tables()`'s `fltper`, `mltper`, or `bltper` (or `build_life_table()`'s direct output). Must have one row per age, sorted or unsorted, with columns `age`, `mx`, `lx`, `Lx`, `Tx`, `ex`.
`lt2`	The "after"/comparison life table, same shape as `lt1`, over the *same set of ages* as `lt1`.
`method`	`"arriaga"` (default) or `"pollard"`.

Value

A tibble with `age`, `contribution` (years of the `e2(0) - e1(0)` gap attributable to mortality differences at that age; positive means that age group contributed to a *gain* from `lt1` to `lt2`).

Examples

```
age <- 0:100
lt1 <- build_life_table(
  data.frame(age = age, mx = 0.0004 * exp(0.075 * age)), "female"
)
lt2 <- build_life_table(
  data.frame(age = age, mx = 0.0003 * exp(0.070 * age)), "female"
)
decomp <- decompose_life_expectancy(lt1, lt2)
sum(decomp$contribution)
lt2$ex[1] - lt1$ex[1]
```

`download_ine_data` *Download SHMD pipeline data to a local folder*

Description

Runs the requested stage(s) of the SHMD (Spanish subnational Human Mortality Database) protocol pipeline — retrieval of province-level births/deaths/population from INE, exposure-to-risk, central death rates, and period life tables — and writes the results into `out_dir` as `.csv` and/or HMD-format `.txt` files, so they can be inspected, shared, or fed into other HMD-compatible tooling without staying in the R session.

Usage

```
download_ine_data(
  out_dir,
  stages = c("births", "deaths", "population", "exposure", "mx", "life_tables"),
  format = c("csv", "txt"),
  n_periods = 30
)
```

Arguments

`out_dir`	Directory to write into (created if it doesn't exist).
`stages`	Character vector of stages to include. One or more of `"births"`, `"deaths"`, `"population"`, `"exposure"`, `"mx"`, `"life_tables"`. Default: all of them.
`format`	Character vector, one or both of `"csv"` and `"txt"`. `"csv"` writes one combined file per data table; `"txt"` writes HMD-format files, one per province. Default: both.
`n_periods`	Number of most recent years to fetch for the births/deaths/population retrieval stages.

Details

Later stages depend on earlier ones (`exposure` needs `population` and `deaths`; `mx` needs `deaths` and `exposure`; `life_tables` needs `mx`). Requesting a later stage automatically fetches/computes and writes its dependencies too, since they were already retrieved along the way.

Value

Invisibly, a list with the in-memory results of every stage that was run (`births`, `deaths`, `population`, `exposure`, `death_rates`, `life_tables`, each `NULL` if not run) and `files`, a character vector of every path written.

Examples

```
## Not run:
# Not run: requires live network access to the INE API, and writes files
# to disk.
# Everything, both formats, into ./ine_data
download_ine_data("ine_data")

# Just births and deaths, CSV only
download_ine_data("ine_data", stages = c("births", "deaths"), format = "csv")

# Full pipeline through life tables
download_ine_data("ine_data", stages = "life_tables", n_periods = 40)

## End(Not run)
```

`general_fertility_rate` *General fertility rate*

Description

Computes the general fertility rate (births per 1,000 women of reproductive age) per province and year, joining `get_ine_births()` and `get_ine_population()` output.

Usage

```
general_fertility_rate(births_df, pop_df, age_min = 15L, age_max = 49L)
```

Arguments

`births_df`	Births tibble as returned by `get_ine_births()$data` (columns `nuts3_code`, `province_name`, `year`, `total`).
`pop_df`	Population tibble as returned by `get_ine_population()$data` (columns `nuts3_code`, `year`, `age`, `female`).
`age_min`	Integer, the youngest age counted as reproductive-age (default `15`).
`age_max`	Integer, the oldest age counted as reproductive-age (default `49`).

Details

GFR improves on `crude_birth_rate()` by dividing births by the female population actually at risk of childbearing (age `age_min`-`age_max`, default 15-49) instead of the total population — this removes the distortion `crude_birth_rate()` suffers when two provinces have different age/sex structures (e.g. one with proportionally more elderly residents) despite similar underlying fertility behavior.

GFR is still **not** a total fertility rate (TFR): it is a single aggregate rate over all reproductive-age women, not an age-specific schedule. For a true TFR, use `get_ine_births_by_age()` with `age_specific_fertility_rate()`/`total_fertility_rate()` instead.

Value

A tibble with `nuts3_code`, `province_name`, `year`, `gfr`.

Examples

```
births <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023, total = 100
)
pop <- tibble::tibble(
  nuts3_code = "ES111", year = 2023, age = c(10, 25, 40, 60),
  female = c(2000, 2500, 2500, 3000)
)
general_fertility_rate(births, pop)
```

`get_ine_births` *Retrieve province-level births*

Description

Retrieves, cleans, and validates Spanish annual live-birth counts by province and sex from INE's Tempus3 JSON API, per the SHMD Protocol (Section 12, Step 1: "Birth Count Assembly"). Unlike `get_ine_demog()` (which returns a `Total`-sex-only province count), this returns sex-disaggregated counts joined to NUTS-3 codes, suitable as an HMD-format Births input file - see `download_ine_data()` to write it to disk.

Usage

```
get_ine_births(
  table_id = 6506,
  n_periods = 100,
  use_cache = TRUE,
  force = FALSE,
  cache_dir = NULL
)
```

Arguments

`table_id`	INE Tempus3 table ID. Default `6506`, confirmed for "Nacimientos por lugar de residencia de la madre y sexo. Total nacional y provincias" (operation MNPN). Pass `NULL` to auto-discover.
`n_periods`	Number of most recent years to fetch.
`use_cache`	Logical (default `TRUE`). If a previous call already cached a result for this `table_id`/`n_periods` combination, return it directly instead of re-fetching from INE — this function's live fetch is one of the package's slowest. Set `FALSE` to always hit the network.
`force`	Logical (default `FALSE`). If `TRUE`, ignore any existing cache and re-fetch from INE, refreshing the cache afterwards.

cache_dir — Directory for cached data. Default NULL means no persistent caching (each call re-fetches from INE). Pass a directory, e.g. tools::R_user_dir("inedemogR", "cache"), to enable caching.

Value

list(data, qc): data is a tibble with columns ine_code, nuts3_code, nuts2_code, province_name, year, female, male, total; qc is the output of validate_births().

Examples

```
## Not run:
# Not run: requires live network access to the INE API.
births <- get_ine_births(n_periods = 30)
births$data
births$qc$issues

# Re-fetch even if a cached copy exists:
births <- get_ine_births(n_periods = 30, force = TRUE)

## End(Not run)
```

---

get_ine_births_by_age *Retrieve province-level births by age of mother*

Description

Retrieves, cleans, and validates Spanish annual live-birth counts by province, single year of age of the mother (15-49, plus the two open intervals under15/50plus), and sex of the newborn, from INE's raw Tempus3 JSON API. Unlike get_ine_births() (which has no age-of-mother breakdown), this is the data source needed for age-specific fertility rates - see age_specific_fertility_rate(), total_fertility_rate(), mean_age_at_childbearing(), gross_reproduction_rate(), and net_reproduction_rate().

Usage

```
get_ine_births_by_age(
  table_id = 6508,
  n_periods = 100,
  use_cache = TRUE,
  force = FALSE,
  cache_dir = NULL
)
```

Arguments

table_id	INE Tempus3 table ID. Default 6508, confirmed for "Nacimientos por lugar de residencia de la madre, sexo y edad de la madre. Total nacional y provincias" (operation MNPN).
n_periods	Number of most recent years to fetch. This table has ~6,000 series (province x age x newborn-sex), so fetches are slower than get_ine_births()'s; caching (see use_cache) matters more here.
use_cache	Logical (default TRUE). See get_ine_births().
force	Logical (default FALSE). See get_ine_births().
cache_dir	Directory for cached data. Default NULL means no persistent caching (each call re-fetches from INE). Pass a directory, e.g. tools::R_user_dir("inedemogR", "cache"), to enable caching.

Value

list(data, qc): data is a tibble with columns ine_code, nuts3_code, nuts2_code, province_name, year, age (integer, NA for the under15 group), age_group (character: "under15", "15".."49", "50plus"), female, male, total (births to mothers of that age, by sex of the newborn); qc is the output of validate_births_by_age().

Examples

```
## Not run:
# Not run: requires live network access to the INE API.
births_age <- get_ine_births_by_age(n_periods = 10)
births_age$data

## End(Not run)
```

---

get_ine_deaths *Retrieve province-level deaths*

Description

Retrieves, cleans, and validates Spanish annual death counts by province and sex from INE's Tempus3 JSON API, per the SHMD Protocol (Section 12, Step 1: "Death Count Assembly"). Fetches both an age-less table (for the national total and a national-vs-provincial reconciliation check) and an age-specific table (needed by compute_exposure() / compute_death_rates(), which require D(x,t), not just D(t)).

Usage

```
get_ine_deaths(
  table_id = 6545,
  age_table_id = 6547,
  n_periods = 100,
```

```
  use_cache = TRUE,
  force = FALSE,
  cache_dir = NULL
)
```

Arguments

table_id	INE table ID for the age-less (province + sex) series. Default 6545, confirmed for "Defunciones por lugar de residencia y sexo. Total nacional y provincias" (operation MNPD).
age_table_id	INE table ID for the age-specific series. Default 6547.
n_periods	Number of most recent years to fetch.
use_cache	Logical (default TRUE). If a previous call already cached a result for this table_id/age_table_id/n_peri combination, return it directly instead of re-fetching from INE — this function's live fetch (two tables) is one of the package's slowest. Set FALSE to always hit the network.
force	Logical (default FALSE). If TRUE, ignore any existing cache and re-fetch from INE, refreshing the cache afterwards.
cache_dir	Directory for cached data. Default NULL means no persistent caching (each call re-fetches from INE). Pass a directory, e.g. tools::R_user_dir("inedemogR", "cache"), to enable caching.

Value

list(data_provinces, data_national, qc, qc_age): data_provinces is age-specific (ine_code, nuts3_code, nuts2_code, province_name, year, age, female, male, total); data_national is Spain's age-less total.

Examples

```
## Not run:
# Not run: requires live network access to the INE API.
deaths <- get_ine_deaths(n_periods = 30)
deaths$data_provinces

# Re-fetch even if a cached copy exists:
deaths <- get_ine_deaths(n_periods = 30, force = TRUE)

## End(Not run)
```

---

get_ine_demog *Get INE Demographic Data*

Description

Retrieves demographic data from INE via the official ineapir::get_data_table() API wrapper, and tidies it into a data frame with one row per geography x year and one column per indicator.

Usage

```
get_ine_demog(
  indicator,
  geo_level = NULL,
  year = NULL,
  region = NULL,
  sex = "Total",
  geometry = FALSE
)
```

Arguments

indicator	Character vector of indicator codes (see list_ine_indicators()).
geo_level	Character, the geographic level to fetch. Must match the level at which every requested indicator is actually published (see list_ine_indicators()). If NULL (default), inferred from indicator.
year	Numeric or character, the year of data. Default (NULL) returns the latest available period only.
region	Optional character vector to subset by region name (regular expressions allowed, matched against the geography name).
sex	Character, one of "Total", "Hombres", "Mujeres".
geometry	Logical, join geometries from get_ine_geo() (default FALSE).

Details

Each indicator is mapped to a real INE table id via the ine_variables registry (see list_ine_indicators()). Different indicators are only available at different geographic granularities in INE's source tables (e.g. population is published by municipality, but births/deaths only down to province) — all requested indicators must share the same geo_level, or request them separately.

Value

A tidy tibble with columns GEOID, NAME, year, and one column per requested indicator; an sf object if geometry = TRUE.

Examples

```
## Not run:
# Not run: requires live network access to the INE API.
pop_data <- get_ine_demog(indicator = "population_total", year = 2023)
births_deaths <- get_ine_demog(
  indicator = c("births_total", "deaths_total"), year = 2023
)

## End(Not run)
```

get_ine_geo *Get INE Geographic Data*

Description

Retrieves spatial geometries for INE geographic levels via the mapSpain package, which sources boundaries from Instituto Geografico Nacional / GISCO. Geometries are returned with a `GEOID` column that matches the geographic codes returned by `get_ine_demog()`, so the two can be joined directly.

Usage

```
get_ine_geo(
  geo_level = c("municipality", "province"),
  region = NULL,
  moveCAN = TRUE,
  can_gap_km = 60
)
```

Arguments

`geo_level`	Character, one of `"municipality"` or `"province"`.
`region`	Optional character vector to subset by region name (regular expressions allowed).
`moveCAN`	Logical, shift the Canary Islands next to the mainland for compact national maps (default `TRUE`). Pass `FALSE` to keep their real geographic coordinates.
`can_gap_km`	Numeric, the gap in kilometers left between the shifted Canary Islands and the closest point on the mainland coast (default `60`). Only used when `moveCAN = TRUE`. Smaller values move the islands closer.

Value

An `sf` object with columns `GEOID`, `NAME`, and `geometry`, projected to ETRS89 / UTM zone 30N (EPSG:25830). When `moveCAN = TRUE` and the shift was applied, an sfc rectangle marking the shifted islands' extent is attached as the `"can_box"` attribute (used by `plot_ine_map()` to draw an inset frame).

Examples

```
## Not run:
# Not run: requires live network access to fetch boundaries via mapSpain.
prov_geo <- get_ine_geo(geo_level = "province")
mun_geo <- get_ine_geo(geo_level = "municipality", region = "Sevilla")

## End(Not run)
```

`get_ine_population` *Retrieve province-level population by age and sex*

Description

Retrieves, cleans, and validates Spanish January-1st provincial population estimates by single-year age and sex from INE's Padron Municipal Continuo, per the SHMD Protocol (Section 12, Step 3: "Population Estimates"). Covers 1996-present; pre-1996 years require the intercensal survival method applied to the 1981/1991 censuses, which is out of scope here.

Usage

```
get_ine_population(
  table_id = 56945,
  n_periods = 30,
  use_cache = TRUE,
  force = FALSE,
  cache_dir = NULL
)
```

Arguments

`table_id`	INE table ID. Default `56945` ("Poblacion residente por fecha, sexo y edad (desde 1971)", provincial variant, operation ECP). Pass `NULL` to auto-discover.
`n_periods`	Number of most recent years to fetch.
`use_cache`	Logical (default `TRUE`). If a previous call already cached a result for this `table_id`/`n_periods` combination, return it directly instead of re-fetching from INE, since this function's live fetch (often via the slower CSV bulk-export fallback) is one of the package's slowest. Set `FALSE` to always hit the network.
`force`	Logical (default `FALSE`). If `TRUE`, ignore any existing cache and re-fetch from INE, refreshing the cache afterwards.
`cache_dir`	Directory for cached data. Default `NULL` means no persistent caching (each call re-fetches from INE). Pass a directory, e.g. `tools::R_user_dir("inedemogR", "cache")`, to enable caching.

Details

Some province x age x sex population tables are too large for INE's JSON API and are rejected outright; this function automatically falls back to INE's full CSV bulk export in that case (a larger download, but with no series-count limit).

Value

`list(data, qc)`: `data` is a tibble with columns `ine_code`, `nuts3_code`, `nuts2_code`, `province_name`, `year`, `age`, `female`, `male`, `total`; `qc` is the output of `validate_population()`.

Examples

```
## Not run:
# Not run: requires live network access to the INE API.
pop <- get_ine_population(n_periods = 10)
pop$data

# Re-fetch even if a cached copy exists:
pop <- get_ine_population(n_periods = 10, force = TRUE)

## End(Not run)
```

`gross_reproduction_rate` *Gross reproduction rate (GRR)*

Description

Computes the gross reproduction rate (expected daughters per woman over her reproductive lifetime, ignoring mortality) per province and year, summing `age_specific_fertility_rate()`'s `asfr_female` column - the actual female-newborn-specific rate INE's age-of-mother table provides, rather than the more common approximation of scaling `total_fertility_rate()` by an assumed constant sex ratio at birth. See `net_reproduction_rate()` to additionally account for mortality before/during reproductive age.

Usage

```
gross_reproduction_rate(asfr_df)
```

Arguments

`asfr_df`	Output of `age_specific_fertility_rate()`.

Value

A tibble with `nuts3_code`, `province_name`, `year`, `grr`.

Examples

```
asfr <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(20, 30), asfr = c(30, 80), asfr_female = c(15, 40)
)
gross_reproduction_rate(asfr)
```

ine_variables *INE demographic indicator reference table*

Description

A registry mapping indicator codes used throughout inedemogR to the real INE table each is retrieved from via `ineapir::get_data_table()`.

Usage

```
ine_variables
```

Format

A tibble with 3 rows and 6 variables:

indicator Indicator code, e.g. `"population_total"`.

name Short human-readable name.

description Longer description of what the indicator measures.

id_table INE `idTable` identifier, or `NA` if not yet wired to a live table.

geo_var Name of the geographic dimension column returned by `ineapir::get_data_table(..., metanames = TRUE)` for this table.

geo_level Geographic granularity the table is published at (`"municipality"` or `"province"`).

Source

Instituto Nacional de Estadistica (INE), "fenomenos demograficos".

infant_mortality_rate *Infant mortality rate*

Description

Computes the infant mortality rate (deaths under age 1 per 1,000 live births) per province and year, joining `get_ine_deaths()` and `get_ine_births()` output.

Usage

```
infant_mortality_rate(deaths_df, births_df)
```

Arguments

`deaths_df`	Deaths tibble as returned by `get_ine_deaths()$data_provinces` (columns `nuts3_code`, `province_name`, `year`, `age`, `total`).
`births_df`	Births tibble as returned by `get_ine_births()$data` (columns `nuts3_code`, `year`, `total`).

Details

This is the standard demographic IMR: infant deaths in a year divided by live births *in that same year* (the birth cohort those deaths are drawn from), not by population — a different denominator convention from `crude_death_rate()`. It is closely related to, but not identical to, `q0` (probability of dying before age 1) in `build_life_table()`'s output: `q0` is derived from the exposure-based central death rate `m0` via the Andreev-Kingkade `a0` (per HMD Methods Protocol V6), while IMR here uses the simpler, more standard deaths/births ratio directly. The two will usually be close but need not match exactly.

Value

A tibble with `nuts3_code`, `province_name`, `year`, `imr`.

Examples

```
deaths <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 1, 65), total = c(3, 1, 40)
)
births <- tibble::tibble(nuts3_code = "ES111", year = 2023, total = 1000)
infant_mortality_rate(deaths, births)
```

`life_expectancy` *Life expectancy at birth and at age 65, for one province*

Description

Convenience wrapper for a single province: builds a period life table (Andreev-Kingkade a0, Kannisto old-age smoothing, per HMD Methods Protocol V6) from already-computed central death rates and extracts life expectancy at birth (`e0`) and at age 65 (`e65`) for every year present. Operates on `compute_death_rates()`'s `mx_1x1` output directly — data already retrieved/computed via the mortality pipeline — the same way `plot_lexis_diagram()` does, rather than requiring `build_life_tables()` to be run for every province first and `life_expectancy_summary()` applied afterwards.

Usage

```
life_expectancy(mx_df, province, sex = c("total", "female", "male"))
```

Arguments

`mx_df`	Output of the `mx_1x1` element of `compute_death_rates()` (columns `nuts3_code`, `province_name`, `year`, `age`, `mx_female`, `mx_male`, `mx_total`). All three `mx_*` columns must be present regardless of `sex`, since the underlying life-table construction builds all three sex tables together.
`province`	Character, a province name (regular expressions allowed, matched against `province_name` — same convention as `province` in `plot_lexis_diagram()`). Must match exactly one province.
`sex`	Character, one of `"total"`, `"female"`, `"male"`.

Value

A tibble with nuts3_code, province_name, year, e0, e65, sex.

Examples

```
# A Gompertz-like mx curve spanning to a realistic terminal age (100):
# build_life_table()'s open-interval formula at the terminal age
# (Lx = lx / mx) assumes that age is genuinely old (high mx), so a
# short age range with a low terminal mx would produce a nonsensical
# "remaining life expectancy" in the thousands of years.
age <- 0:100
mx <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023, age = age,
  mx_female = 0.0003 * exp(0.07 * age),
  mx_male = 0.00035 * exp(0.072 * age),
  mx_total = 0.00033 * exp(0.071 * age)
)
life_expectancy(mx, province = "A Coruna", sex = "female")
```

---

life_expectancy_summary

Life expectancy summary

Description

Extracts life expectancy at birth (e0) and at age 65 (e65) per province and year from a build_life_tables() period life table. Feeds map_life_expectancy(), which bridges this summary with get_ine_geo()'s province geometries.

Usage

```
life_expectancy_summary(lt_df, sex = c("female", "male", "both"))
```

Arguments

lt_df	One of build_life_tables()'s fltper, mltper, or bltper tibbles (columns nuts3_code, province_name, year, age, ex).
sex	Character label to attach to the output ("female", "male", or "both") — purely descriptive, matching which life table (fltper/mltper/bltper) was passed in.

Value

A tibble with nuts3_code, province_name, year, e0, e65, sex.

Examples

```
lt <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 65), ex = c(86.97, 20.5)
)
life_expectancy_summary(lt, sex = "female")
```

`list_ine_indicators` *List Available INE Indicators*

Description

Returns a data frame of available demographic indicators.

Usage

```
list_ine_indicators()
```

Value

A data frame with indicator codes, names, and descriptions.

Examples

```
list_ine_indicators()
```

`map_indicator` *Map any indicator by province or municipality*

Description

General-purpose choropleth: joins a tidy indicator tibble (keyed by `GEOID`, matching `get_ine_geo()`'s output — e.g. `get_ine_demog()`'s output already has it) onto province or municipality geometry and renders it, either as a binned/discrete-legend map (the default, matching the classic "N provinces above/below a threshold" style of choropleth) or a continuous gradient. `map_life_expectancy()` is a thin wrapper around this function for System B's life-table output.

Usage

```
map_indicator(
  df,
  value_col,
  geo_level = c("province", "municipality"),
  region = NULL,
  binned = TRUE,
  breaks = NULL,
```

```
  palette = if (binned) "PiYG" else "D",
  moveCAN = TRUE,
  can_gap_km = 60,
  legend_title = NULL,
  title = NULL
)
```

Arguments

df	A tibble with a GEOID column (matching get_ine_geo()'s GEOID for the requested geo_level) and value_col.
value_col	Character, the name of the numeric column to map.
geo_level	Character, one of "province" or "municipality".
region	Optional character vector to subset the geometry by region name (regular expressions allowed). See get_ine_geo().
binned	Logical (default TRUE). If TRUE, uses a discrete, binned color scale (ggplot2::scale_fill_fermenter() — a fixed number of color bands with a legend showing each band's range, matching the classic choropleth style. If FALSE, uses a continuous gradient (ggplot2::scale_fill_viridis_c() by default).
breaks	Numeric vector of bin edges when binned = TRUE. If NULL (default), computed automatically via pretty().
palette	Character, a RColorBrewer palette name when binned = TRUE (default "PiYG", a pink-to-green diverging palette well suited to ratios centered on 1) or a viridis option letter when binned = FALSE (default "D").
moveCAN	Logical, shift the Canary Islands next to the mainland (default TRUE). See get_ine_geo().
can_gap_km	Numeric, gap in km for the shifted Canary Islands (default 60). See get_ine_geo().
legend_title	Optional legend title (defaults to value_col).
title	Optional plot title.

Value

A ggplot object.

Examples

```
## Not run:
# Not run: get_ine_geo() requires live network access to fetch boundaries.
pop_data <- get_ine_demog(indicator = "population_total", year = 2023)
map_indicator(pop_data, "population_total", geo_level = "province")

## End(Not run)
```

`map_life_expectancy` *Map life expectancy by province*

Description

Bridges System B's mortality pipeline (`build_life_tables()` via `life_expectancy_summary()`) with System A's spatial layer (`get_ine_geo()`) to render a province-level choropleth of life expectancy at birth. A thin wrapper around `map_indicator()` (a continuous gradient, since life expectancy has no natural discrete bins) that handles the `nuts3_code` -> `GEOID` bridge via `province_lookup`. See `plot_ine_map()` for the analogous region-highlighting map built purely on `get_ine_geo()`.

Usage

```
map_life_expectancy(
  le_df,
  year,
  sex = "both",
  moveCAN = TRUE,
  can_gap_km = 60,
  title = NULL
)
```

Arguments

`le_df`	Output of `life_expectancy_summary()` (columns `nuts3_code`, `year`, `e0`, `sex`).
`year`	Numeric, the year to plot.
`sex`	Character, one of `"female"`, `"male"`, `"both"`, matching the `sex` value in `le_df` to plot.
`moveCAN`	Logical, shift the Canary Islands next to the mainland (default `TRUE`). See `get_ine_geo()`.
`can_gap_km`	Numeric, gap in km for the shifted Canary Islands (default `60`). See `get_ine_geo()`.
`title`	Optional plot title.

Value

A `ggplot` object.

Examples

```
## Not run:
# Not run: get_ine_geo() requires live network access to fetch province
# boundaries.
lt <- build_life_tables(rates$mx_1x1)
le <- life_expectancy_summary(lt$fltper, sex = "female")
map_life_expectancy(le, year = max(le$year), sex = "female")

## End(Not run)
```

mean_age_at_childbearing *Mean age at childbearing (MAC)*

Description

Computes the mean age at childbearing (the ASFR-weighted average age of mothers at birth) per province and year, from `age_specific_fertility_rate()` output.

Usage

```
mean_age_at_childbearing(asfr_df)
```

Arguments

`asfr_df`	Output of `age_specific_fertility_rate()`.

Value

A tibble with `nuts3_code`, `province_name`, `year`, `mac`.

Examples

```
asfr <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(20, 30), asfr = c(30, 80), asfr_female = c(15, 40)
)
mean_age_at_childbearing(asfr)
```

net_reproduction_rate *Net reproduction rate (NRR)*

Description

Computes the net reproduction rate (expected daughters per woman over her reproductive lifetime, accounting for the mother's own survival to each reproductive age) per province and year, weighting `age_specific_fertility_rate()`'s `asfr_female` by the female life-table survivorship function `Lx` from `build_life_tables()`'s `fltper` (person-years lived in the age interval, per HMD Methods Protocol V6 - the same `Lx` `life_expectancy_summary()`'s `ex` is derived from). Unlike `gross_reproduction_rate()`, which assumes every woman survives to bear children at every age, NRR is the mortality- adjusted figure: NRR < GRR whenever there is any mortality before the end of the reproductive span.

Usage

```
net_reproduction_rate(asfr_df, lt_df)
```

Arguments

asfr_df	Output of age_specific_fertility_rate().
lt_df	build_life_tables()'s fltper tibble (columns nuts3_code, year, age, Lx) - must be the *female* life table, since NRR tracks mothers' own survival.

Value

A tibble with nuts3_code, province_name, year, nrr.

Examples

```
asfr <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(20, 30), asfr = c(30, 80), asfr_female = c(15, 40)
)
fltper <- tibble::tibble(
  nuts3_code = "ES111", year = 2023, age = c(20, 30), Lx = c(99000, 98500)
)
net_reproduction_rate(asfr, fltper)
```

---

plot_demog_trend *Plot a demographic indicator trend over time*

Description

Generic time-series line chart for any indicator tibble keyed by nuts3_code/province_name/year, such as the output of age_dependency_ratio(), aging_index(), sex_ratio(), crude_birth_rate(), or life_expectancy_summary().

Usage

```
plot_demog_trend(df, value_col, region = NULL, title = NULL, ylab = NULL)
```

Arguments

df	A tibble with province_name, year, and value_col.
value_col	Character, the name of the column to plot.
region	Optional character vector to subset by province name (regular expressions allowed, matched against province_name); one line is drawn per matched province. If NULL (default), every province present in df is plotted.
title	Optional plot title.
ylab	Optional y-axis label (defaults to value_col).

Value

A ggplot object.

Examples

```
trend <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna",
  year = c(2022, 2023), aging_index = c(150, 160)
)
plot_demog_trend(trend, "aging_index")
```

`plot_ine_map` *Plot a Map of Spain with Regions Highlighted*

Description

Plots the full map of Spain at the requested geographic level (provinces or municipalities), with one or more regions highlighted in a different color. Useful for confirming *which* regions a `region` filter passed to `get_ine_demog()` or `get_ine_geo()` actually matches, since `get_ine_geo()` uses regex substring matching by default (see `vignette` examples in `test_cordoba.R`).

Usage

```
plot_ine_map(
  geo_level = c("province", "municipality"),
  highlight = NULL,
  base_color = "grey90",
  highlight_color = "firebrick",
  title = NULL,
  can_box = TRUE,
  can_gap_km = 60
)
```

Arguments

`geo_level`	Character, one of `"province"` or `"municipality"`.
`highlight`	Character vector of region names to highlight (regular expressions allowed, matched against the geography name — same matching rules as `region` in `get_ine_demog()`/`get_ine_geo()`). If `NULL` (default), the whole map is plotted in `base_color` with no highlighting.
`base_color`	Fill color for non-highlighted regions. Default `"grey90"`.
`highlight_color`	Fill color for highlighted regions. Default `"firebrick"`.
`title`	Optional plot title.
`can_box`	Logical, draw an inset separator box around the Canary Islands, which are shifted next to the mainland for a compact map (default `TRUE`).
`can_gap_km`	Numeric, how close (in km) the shifted Canary Islands sit next to the mainland (default `60`). Smaller values move them closer. See `get_ine_geo()`.

Value

A ggplot object.

Examples

```
## Not run:
# Not run: get_ine_geo() requires live network access to fetch boundaries.
plot_ine_map(geo_level = "province", highlight = "^Córdoba$")
plot_ine_map(geo_level = "municipality", highlight = "Córdoba")

## End(Not run)
```

---

plot_lexis_diagram *Calculate and plot a Lexis diagram for one province*

Description

Builds the classic demographic Lexis surface — age (y-axis) by calendar year (x-axis), shaded by mortality rate, with birth-cohort diagonals overlaid — for one province, from the age x year central death rates already computed by compute_death_rates(). Operates on data already retrieved/computed via the mortality pipeline (get_ine_deaths() -> compute_exposure() -> compute_death_rates()); it does not fetch anything itself.

Usage

```
plot_lexis_diagram(
  mx_df,
  province,
  sex = c("total", "female", "male"),
  log_scale = TRUE,
  rate_per = 1000,
  cohort_lines = TRUE,
  cohort_step = 10,
  title = NULL
)
```

Arguments

mx_df	Output of the mx_1x1 element of compute_death_rates() (columns nuts3_code, province_name, year, age, mx_female, mx_male, mx_total).
province	Character, a province name (regular expressions allowed, matched against province_name — same convention as region in plot_population_pyramid()). Must match exactly one province.
sex	Character, one of "total", "female", "male".
log_scale	Logical (default TRUE), color-scale mortality rates on a log10 scale — the standard convention for Lexis surfaces, since mx spans several orders of magnitude across ages.

`rate_per`	Numeric, express the mortality rate as deaths per this many people (default `1000`, the standard demographic convention — e.g. a legend value of `10` reads as "10 deaths per 1,000 population"). Use `1` for the raw per-person rate.
`cohort_lines`	Logical (default `TRUE`), overlay birth-cohort diagonals (age = year - cohort).
`cohort_step`	Numeric, spacing in years between overlaid cohort diagonals (default `10`).
`title`	Optional plot title (defaults to the matched province name).

Details

"Calculate" here means deriving the tidy age x year surface for the requested province/sex and the cohort-diagonal positions to overlay (age = year - cohort, for cohorts every `cohort_step` years across the data's range) — not fitting a new mortality model; the rates themselves come straight from `compute_death_rates()`.

The white diagonal lines are intentional, not an artifact. They are birth-cohort lines (age = year - cohort): every point along one line represents the same group of people born in the same year, aging one year for every calendar year that passes — the defining feature of a Lexis diagram, letting you trace one cohort's mortality experience across its lifetime instead of only reading across a fixed year or up a fixed age. Set `cohort_lines = FALSE` to turn them off.

Grey cells mean no rate could be computed for that age x year cell, almost always because the source population data has no single-year age breakdown for the oldest ages in the earliest years of the series (so exposure, the rate's denominator, is missing there) — not a bug in the calculation. This typically shows up as a rectangular block in the upper-left (oldest ages, earliest years), clearing up once the source data's age detail becomes complete. When present, grey gets its own "Missing data" legend key (a continuous color scale can't show this on its own, so it's added via a small dummy layer).

Value

A `ggplot` object.

Examples

```
mx <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna",
  year = rep(2020:2023, each = 3), age = rep(c(0, 50, 90), 4),
  mx_total = c(
    0.004, 0.003, 0.15, 0.0038, 0.0031, 0.14,
    0.0035, 0.0029, 0.16, 0.0033, 0.0028, 0.15
  )
)
plot_lexis_diagram(mx, province = "A Coruna")
```

plot_population_pyramid *Plot a population pyramid*

Description

Draws a population pyramid for one province and year from age-disaggregated population data. Requires an age/sex breakdown, so it operates on `get_ine_population()`'s output, not `get_ine_demog()`'s `population_total` indicator.

Usage

```
plot_population_pyramid(pop_df, year, region = NULL, title = NULL)
```

Arguments

`pop_df`	Population tibble as returned by `get_ine_population()$data` (columns `province_name`, `year`, `age`, `female`, `male`).
`year`	Numeric, the year to plot.
`region`	Optional character vector to subset by province name (regular expressions allowed, matched against `province_name`). If more than one province matches, counts are summed across them. If `NULL` (default), all provinces are summed (national pyramid).
`title`	Optional plot title.

Value

A `ggplot` object.

Examples

```
pop <- tibble::tibble(
  province_name = "A Coruna", year = 2023,
  age = c(0, 1), female = c(100, 90), male = c(105, 95)
)
plot_population_pyramid(pop, year = 2023)
```

rate_of_natural_increase *Rate of natural increase*

Description

Computes the rate of natural increase (RNI = CBR - CDR, per 1,000 population) per province and year, combining already-computed `crude_birth_rate()` and `crude_death_rate()` output rather than re-deriving from raw births/deaths/population — consistent with how `life_expectancy_summary()` consumes `build_life_tables()`'s output rather than recomputing it.

Usage

```
rate_of_natural_increase(cbr_df, cdr_df)
```

Arguments

cbr_df	Output of `crude_birth_rate()` (columns `nuts3_code`, `province_name`, `year`, `cbr`).
cdr_df	Output of `crude_death_rate()` (columns `nuts3_code`, `year`, `cdr`).

Value

A tibble with `nuts3_code`, `province_name`, `year`, `rni`. A positive value means births outnumber deaths (population growing from natural change alone, ignoring migration); negative means the reverse.

Examples

```
cbr <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023, cbr = 6.5
)
cdr <- tibble::tibble(nuts3_code = "ES111", year = 2023, cdr = 12.3)
rate_of_natural_increase(cbr, cdr)
```

sex_ratio *Sex ratio*

Description

Computes the sex ratio (males per 100 females) per province and year from age-disaggregated population data. Requires the `female`/`male` columns `get_ine_population()` provides; `get_ine_demog()`'s `population_total` indicator has no sex breakdown and cannot be used here.

Usage

```
sex_ratio(pop_df, by_age = FALSE)
```

Arguments

pop_df	Population tibble as returned by `get_ine_population()$data` (columns `nuts3_code`, `province_name`, `year`, `age`, `female`, `male`).
by_age	Logical, keep the age column and compute a sex ratio per age (default `FALSE`, which aggregates over all ages first).

Value

A tibble with `nuts3_code`, `province_name`, `year` (and `age` if `by_age = TRUE`), `sex_ratio`.

Examples

```
pop <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 1), female = c(100, 95), male = c(105, 100)
)
sex_ratio(pop)
sex_ratio(pop, by_age = TRUE)
```

`total_fertility_rate` *Total fertility rate (TFR)*

Description

Computes the total fertility rate (expected live births per woman over her reproductive lifetime, under a given year's fertility schedule) per province and year, by summing `age_specific_fertility_rate()` over single years of age. This is the true TFR that `crude_birth_rate()`'s and `general_fertility_rate()`'s documentation both note the package previously could not compute, since it requires the age-of-mother breakdown `get_ine_births_by_age()` retrieves.

Usage

```
total_fertility_rate(asfr_df)
```

Arguments

asfr_df	Output of `age_specific_fertility_rate()`.

Value

A tibble with `nuts3_code`, `province_name`, `year`, `tfr`.

Examples

```
asfr <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(20, 30), asfr = c(30, 80), asfr_female = c(15, 40)
)
total_fertility_rate(asfr)
```

update_ine_data *Update INE Data*

Description

Checks each indicator wired in ine_variables for its latest published year (via `get_ine_demog()`) and, if newer than the local cache, downloads and caches it for offline use.

Usage

```
update_ine_data(force = FALSE, data_dir)
```

Arguments

`force`	Logical, force update even if data is current.
`data_dir`	Character, directory for cached data. No default: this function only writes when a directory is explicitly supplied, per CRAN policy against writing to the user's home filespace by default. Pass e.g. `tools::R_user_dir("inedemogR", "cache")` for a persistent cache.

Value

Invisibly, a named list of the latest cached year per indicator.

Examples

```
## Not run:
# Not run: requires live network access to the INE API, and writes to
# the supplied cache directory.
update_ine_data(data_dir = tools::R_user_dir("inedemogR", "cache"))

## End(Not run)
```

validate_abridged_life_table

Validate a constructed abridged life table

Description

Checks: `lx` strictly non-increasing by age group; `Lx > 0`; `ex > 0`; `qx` in [0, 1]. Mirrors `validate_life_table()` for the abridged (5-year age group) case.

Usage

```
validate_abridged_life_table(alt)
```

Arguments

alt	Output of build_abridged_life_table() for one year.

Value

list(passed = logical, issues = named list of flagged tibbles).

Examples

```
mx1 <- data.frame(age = 0:4, mx = c(0.004, 0.0003, 0.0002, 0.0002, 0.0002))
mx5 <- data.frame(
  age_group = c("00-04", "05-09", "100+"), mx = c(0.0006, 0.0001, 0.35)
)
alt <- build_abridged_life_table(mx1, mx5, "female")
validate_abridged_life_table(alt)
```

---

validate_births *Validate cleaned provincial birth data*

Description

Checks: sex ratio at birth (SRB) implausibly far from the expected value, accounting for sample size; no negative/missing counts; no duplicate province-year rows; continuous year coverage.

Usage

```
validate_births(df)
```

Arguments

df	Output of the data element of get_ine_births().

Details

The SRB check does not use a single fixed band (an earlier version used [1.030, 1.070], which flagged ~60% of all province-years, including 38% of Madrid/Barcelona's — clearly miscalibrated: annual province-level SRB is naturally noisy for small birth counts, and this data's own pooled SRB runs about 1.06, already above that band's upper bound even before any noise). Instead it flags a province-year only if *both* (a) the observed SRB is statistically implausible given that year's birth count, assuming a true underlying SRB of 1.06 (a two-tailed z-test on the male-birth proportion at $z > 4$, i.e. roughly a 1-in-16,000 event under the null - deliberately strict, since thousands of province-years are tested at once), and (b) the deviation is also practically meaningful (SRB more than 0.03 from 1.06), so statistically-significant-but-tiny deviations in large provinces aren't flagged just because their sample size makes the test hyper-sensitive. An SRB outside [0.90, 1.25] is additionally always flagged as a sanity floor for gross data errors, but only once total births reach 200: below that, even a large swing (e.g. 23 vs. 17 births) is unremarkable binomial noise, not a data problem, and the z-test above already accounts for the small sample correctly.

Value

list(passed = logical, issues = named list of flagged tibbles).

Examples

```
births_df <- tibble::tibble(
  nuts3_code = c("ES611", "ES611", "ES612", "ES612"),
  province_name = c("Almeria", "Almeria", "Cadiz", "Cadiz"),
  year = c(2020, 2021, 2020, 2021),
  female = c(950, 970, 1200, 1210),
  male = c(1000, 1020, 1260, 1270),
  total = female + male
)
validate_births(births_df)
```

---

validate_births_by_age

Validate cleaned age-of-mother birth data

Description

Checks: no negative/missing counts; no duplicate province-year-age rows; single-year ages (15-49) present for every province-year found in the data (the two open intervals, under15/50plus, are not required to be present since they are typically very small or zero).

Usage

```
validate_births_by_age(df)
```

Arguments

df	Output of the data element of get_ine_births_by_age().

Value

list(passed = logical, issues = named list of flagged tibbles).

Examples

```
# Only 3 of the 35 required single-year ages (15-49) are present here,
# so this deliberately triggers the incomplete age coverage check.
births_age_df <- tibble::tibble(
  nuts3_code = "ES611",
  province_name = "Almeria",
  year = 2020,
  age = c(20, 21, 22),
  age_group = c("20", "21", "22"),
  female = c(30, 28, 25),
  male = c(32, 29, 27),
```

```
  total = female + male
)
validate_births_by_age(births_age_df)
```

`validate_deaths` *Validate cleaned provincial death data*

Description

Checks: sex ratio at death in [0.90, 1.50]; no negative/missing counts; no duplicate province-year rows; continuous year coverage; provincial sum vs. national total (if `nat_df` supplied).

Usage

```
validate_deaths(prov_df, nat_df = tibble::tibble())
```

Arguments

`prov_df`	Province-level cleaned deaths tibble.
`nat_df`	National-level cleaned deaths tibble (optional).

Value

`list(passed = logical, issues = named list of flagged tibbles)`.

Examples

```
deaths_df <- tibble::tibble(
  nuts3_code = c("ES611", "ES611", "ES612", "ES612"),
  province_name = c("Almeria", "Almeria", "Cadiz", "Cadiz"),
  year = c(2020, 2021, 2020, 2021),
  female = c(480, 500, 610, 630),
  male = c(510, 520, 640, 655),
  total = female + male
)
validate_deaths(deaths_df)
```

`validate_deaths_age` *Validate cleaned age-specific provincial death data*

Description

Checks: no negative/missing counts; no duplicate province-age-year rows; continuous year coverage per province.

Usage

```
validate_deaths_age(df)
```

Arguments

df	Age-specific province deaths tibble.

Value

list(passed = logical, issues = named list of flagged tibbles).

Examples

```
deaths_age_df <- tibble::tibble(
  nuts3_code = c("ES611", "ES611"),
  province_name = c("Almeria", "Almeria"),
  year = c(2020, 2020),
  age = c(0, 1),
  female = c(2, 0),
  male = c(3, 1),
  total = female + male
)
validate_deaths_age(deaths_age_df)
```

---

validate_death_rates *Validate computed central death rates*

Description

Checks: no negative/NA/infinite rates; no rates >= 1 for ages below 90 (flagged for review, not treated as a hard error).

Usage

```
validate_death_rates(mx_df)
```

Arguments

mx_df	Output of the mx_1x1 element of compute_death_rates().

Value

list(passed = logical, issues = named list of flagged tibbles).

Examples

```
deaths <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 1), female = c(1, 0), male = c(1, 1), total = c(2, 1)
)
exposure <- tibble::tibble(
  nuts3_code = "ES111", province_name = "A Coruna", year = 2023,
  age = c(0, 1), female = c(500, 490), male = c(510, 505), total = c(1010, 995)
)
```

```
rates <- compute_death_rates(deaths, exposure)
validate_death_rates(rates$mx_1x1)
```

`validate_exposure` *Validate computed exposure-to-risk values*

Description

Checks: cells with no usable source population (flagged for audit, not treated as an error); no other negative/NA exposures; exposure should not exceed 2x population (a generous plausibility bound).

Usage

```
validate_exposure(exposure_df, population_df)
```

Arguments

`exposure_df`	Output of the `data` element of `compute_exposure()`.
`population_df`	The population tibble originally passed in.

Value

`list(passed = logical, issues = named list of flagged tibbles).`

Examples

```
exposure_df <- tibble::tibble(
  nuts3_code = "ES611", year = 2020, age = c(0, 1),
  female = c(495, 480), male = c(520, 500), total = c(1015, 980),
  is_boundary_year = FALSE, is_missing_source_pop = FALSE
)
population_df <- tibble::tibble(
  nuts3_code = "ES611", year = 2020, age = c(0, 1), total = c(1000, 970)
)
validate_exposure(exposure_df, population_df)
```

`validate_life_table` *Validate a constructed life table*

Description

Checks: lx strictly non-increasing in age; Lx > 0; ex > 0; qx in [0, 1].

Usage

```
validate_life_table(lt)
```

Arguments

lt	Output of build_life_table() for one year.

Value

list(passed = logical, issues = named list of flagged tibbles).

Examples

```
mx <- data.frame(age = 0:5, mx = c(0.004, 0.0003, 0.0002, 0.0002, 0.0002, 0.0002))
lt <- build_life_table(mx, "female")
validate_life_table(lt)
```

---

validate_population *Validate cleaned provincial population data*

Description

Checks: no negative/missing counts; full age range (0..100) present per province-year; no duplicate province-age-year rows; continuous year coverage; no implausible age-to-age jumps (>50%, informational only).

Usage

```
validate_population(df)
```

Arguments

df	Output of the data element of get_ine_population().

Value

list(passed = logical, issues = named list of flagged tibbles).

Examples

```
population_df <- tibble::tibble(
  nuts3_code = c("ES611", "ES611"),
  province_name = c("Almeria", "Almeria"),
  year = c(2020, 2020),
  age = c(0, 1),
  female = c(480, 490),
  male = c(500, 505),
  total = female + male
)
validate_population(population_df)
```

Index