Package 'gerda'

July 27, 2026

Title German Election Database (GERDA)

Version 0.8.1

Author Hanno Hilbig [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5849-9172>)

Description Provides tools to download datasets of German elections covering local, state, federal, mayoral, European Parliament, and county (Kreistag) elections, with federal county-level coverage from 1953 and other families extending through 2025. The package supplies turnout, vote shares, and derived indicators at the municipal and county level, including geographically harmonized datasets that account for changes in municipal boundaries over time and incorporate mail-in voting districts. Bundled data includes county-level INKAR covariates (1995-2022) and municipality-level Zensus 2022 indicators. Data is sourced from <https://github.com/awiedem/german_election_data>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports dplyr, readr, stats, stringdist, tibble, tools, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/hhilbig/gerda,
https://github.com/awiedem/german_election_data

BugReports https://github.com/hhilbig/gerda/issues

VignetteBuilder knitr

Maintainer Hanno Hilbig <hhilbig@ucdavis.edu>

NeedsCompilation no

Repository CRAN

Date/Publication 2026-07-27 11:40:09 UTC

Contents

`add_gerda_census` *Add Census 2022 Data to GERDA Election Data*

Description

Convenience function to merge Zensus 2022 municipality-level data with GERDA election data. The census provides a cross-sectional snapshot (2022), so the same values are attached to all election years.

The function works with both municipality-level and county-level election data:

- **Municipality-level data**: Direct merge using 8-digit AGS codes
- **County-level data**: Census data is aggregated to the county level (population-weighted means for shares, sums for counts) before merging

Usage

```
add_gerda_census(election_data, unmatched = "warn")
```

Arguments

`election_data`	A data frame containing GERDA election data. Must contain either an `ags` column (municipality level) or a `county_code` column (county level).
`unmatched`	How to handle input rows whose geographic identifier does not match Census 2022. One of `"warn"` (default), `"error"`, or `"ignore"`.

Details

Required Columns:

The input data must contain one of:

- `ags`: 8-digit municipal code for municipality-level data
- `county_code`: 5-digit county code for county-level data

Merge Behavior:

Since the census is a 2022 cross-section, census values are the same for all election years. The merge is on geography only (no year join).

For county-level data, municipality-level census data is first aggregated:

- Share variables: Population-weighted means
- Count variables (population_census22, total_dwellings_census22): Sums
- Other variables (avg_household_size_census22, avg_rent_per_m2_census22): Population-weighted means

Validation and Diagnostics:

Geographic identifiers must be character vectors containing exactly eight digits (`ags`) or five digits (`county_code`). Numeric identifiers are rejected because leading zeros may already have been lost. Before joining, the function verifies reference-key uniqueness and rejects output-column conflicts. It then verifies that the input row count is unchanged and reports the exact number of unmatched rows and geographic units. Missing join keys are classified as unexpected. Use `unmatched = "error"` in unattended pipelines.

Value

The input data frame with additional census columns appended. The number of rows remains unchanged. A machine-readable join report is attached and can be retrieved with `gerda_join_diagnostics()`.

See Also

- `gerda_census` for direct access to the census data
- `gerda_census_codebook` for variable descriptions
- `gerda_join_diagnostics` for inspecting match results

Examples

```
## Not run:
library(gerda)

# Municipality-level merge
muni_data <- load_gerda_web("federal_muni_harm_21") |>
  add_gerda_census()

# County-level merge (aggregated from municipalities)
county_data <- load_gerda_web("federal_cty_harm") |>
  add_gerda_census()

## End(Not run)
```

`add_gerda_covariates` *Add County-Level Covariates to GERDA Election Data*

Description

Convenience function to merge INKAR county-level (Kreis) covariates with GERDA election data. This is the recommended way to add covariates, as it automatically uses the correct join keys and prevents common merge errors.

The function works with both county-level and municipal-level election data:

- **County-level data**: Direct merge using county codes
- **Municipal-level data**: Automatically extracts county code from municipal AGS (first 5 digits) and merges

Important: Covariates are always at the county level. When merging with municipal data, all municipalities within the same county will receive identical covariate values.

The function performs a left join, keeping all rows from the election data and adding covariates where available. This automatically retains only election years.

Usage

```
add_gerda_covariates(election_data, unmatched = "warn")
```

Arguments

`election_data`	A data frame containing GERDA election data. Must contain a column with county or municipal codes (see Details) and `election_year`.
`unmatched`	How to handle rows within the 1995-2022 INKAR coverage window whose join keys do not match. One of `"warn"` (default), `"error"`, or `"ignore"`. Rows outside 1995-2022 are reported separately and do not trigger this action.

Details

Required Columns:

The input data must contain `election_year` and one of:

- `county_code`: 5-digit county code (AGS) for county-level data
- `ags`: 8-digit municipal code (AGS) for municipal-level data

The function automatically detects which column is present and performs the appropriate merge. For municipal data, the county code is extracted from the first 5 digits of the AGS.

Data Level:

Covariates are at the county (Kreis) level:

- **County-level merge**: One-to-one match, each county gets its covariates
- **Municipal-level merge**: Many-to-one match, all municipalities in the same county receive identical covariate values

Data Availability:

Covariates are available from 1995-2022. For GERDA federal elections:

- Elections 1990, 1994: No covariates (before 1995)
- Elections 1998-2021: Covariates available

Missing Data:

Some covariates have missing values. Use `gerda_covariates_codebook()` to check data availability for specific variables.

Validation and Diagnostics:

Geographic identifiers must be character vectors containing exactly five digits (`county_code`) or eight digits (`ags`). Numeric identifiers are rejected because leading zeros may already have been lost. Before joining, the function verifies reference-key uniqueness and rejects output-column conflicts. It then verifies that the join has not changed the input row count and reports unexpected unmatched rows separately from years outside INKAR coverage. Missing join keys are always classified as unexpected. Use `unmatched = "error"` in unattended pipelines.

Value

The input data frame with additional columns for all 30 county-level covariates. The number of rows remains unchanged. A machine-readable join report is attached and can be retrieved with `gerda_join_diagnostics()`.

See Also

- `gerda_covariates` for direct access to the covariate data
- `gerda_covariates_codebook` for variable descriptions
- `load_gerda_web` for loading GERDA election data
- `gerda_join_diagnostics` for inspecting match results

Examples

```
## Not run:
library(gerda)
library(dplyr)

# Example 1: County-level election data
county_data <- load_gerda_web("federal_cty_harm") %>%
  add_gerda_covariates()

# Check the result
names(county_data) # See new covariate columns
table(county_data$election_year) # Only election years

# Example 2: Municipal-level election data
# Note: All municipalities in the same county will get identical covariates
muni_data <- load_gerda_web("federal_muni_harm_21") %>%
  add_gerda_covariates()
```

```
# Verify: municipalities in same county have same covariate values.
# The county code is the first 5 digits of the 8-digit municipal AGS.
muni_data %>%
  mutate(county_code = substr(ags, 1, 5)) %>%
  group_by(county_code, election_year) %>%
  summarize(
    n_munis = n(),
    unemp_range = max(unemployment_rate) - min(unemployment_rate)
  )

# Analyze with covariates
county_data %>%
  filter(election_year == 2021) %>%
  filter(!is.na(unemployment_rate)) %>%
  summarize(cor_unemployment_afd = cor(unemployment_rate, afd))

## End(Not run)
```

`clear_gerda_cache` *Clear the GERDA download cache*

Description

Removes files written to the GERDA cache by `load_gerda_web()` (`cache = TRUE`). Because the upstream data can be updated in place, use this to force fresh downloads, or pass `cache = TRUE, refresh = TRUE` to `load_gerda_web()` for a one-off refresh.

Usage

```
clear_gerda_cache(data_name = NULL)
```

Arguments

`data_name`	Optional dataset name. If supplied, only that dataset's cached CSV/RDS files are removed; otherwise the entire cache is cleared.

Value

Invisibly, a character vector of the removed file paths.

See Also

`gerda_cache_dir()`, `load_gerda_web()`

Examples

```
# Not run: this deletes files from the user's cache directory.
## Not run:
clear_gerda_cache()

## End(Not run)
```

`gerda_cache_dir` *GERDA cache directory*

Description

Returns the path to the directory used by `load_gerda_web()` to cache downloaded datasets when `cache = TRUE` (or `options(gerda.cache = TRUE)`). The location follows `tools::R_user_dir()` conventions and can be redirected with the standard `R_USER_CACHE_DIR` environment variable. The directory is not created until a cached download is actually written.

Usage

```
gerda_cache_dir()
```

Value

A single character string: the cache directory path.

See Also

`clear_gerda_cache()`, `load_gerda_web()`

Examples

```
gerda_cache_dir()
```

`gerda_census` *Get Municipality-Level Census 2022 Data*

Description

Returns municipality-level demographic and socioeconomic data from the German Census 2022 (Zensus 2022). This is a cross-sectional snapshot covering all German municipalities.

For most users, we recommend using `add_gerda_census` instead, which automatically merges census data with GERDA election data.

Usage

```
gerda_census()
```

Details

The dataset includes:

- Demographics: Population and population shares for ages under 18, 18-29, 30-49, 50-59, and 60+
- Migration: Migration background, foreign nationals
- Households: Average household size
- Housing: Dwellings, vacancy, ownership, rents, building types

Municipality codes are 8-digit AGS codes. Since the census is a single 2022 snapshot, there is no year dimension.

The age variables follow the bins published in the Destatis Regionaltabellen. In particular, `share_50to59_census22` covers ages 50-59 and `share_60plus_census22` covers ages 60 and older. The source groups ages 60-74 together, so it cannot support separate 50-64 and 65+ measures.

Value

A data frame with approximately 10,800 rows (one per municipality) and 16 columns containing census indicators. See `gerda_census_codebook` for variable descriptions.

See Also

- `add_gerda_census` for automatic merging with election data
- `gerda_census_codebook` for variable descriptions

Examples

```
# Get the census data
census <- gerda_census()
head(census)

# Check available municipalities
nrow(census)
```

---

gerda_census_codebook *Get Codebook for Census 2022 Data*

Description

Returns the data dictionary for municipality-level Census 2022 indicators. Provides variable names, labels, units, and data sources.

Usage

```
gerda_census_codebook()
```

Value

A data frame with 16 rows documenting all variables in the census dataset.

See Also

gerda_census for the actual census data

Examples

```
# View the codebook
codebook <- gerda_census_codebook()
print(codebook)
```

---

gerda_covariates *Get County-Level Covariates from INKAR*

Description

Returns county-level socioeconomic and demographic covariates from INKAR. This function provides flexible access to the raw covariate data for advanced users who want to inspect or manipulate it before merging with county-level election data.

For most users, we recommend using add_gerda_covariates instead, which automatically performs the merge with correct join keys.

Note: These covariates are at the county (Kreis) level and should be merged with county-level GERDA data (e.g., federal_cty_harm).

Usage

```
gerda_covariates()
```

Details

The dataset includes 30 socioeconomic and demographic variables:

- Demographics: Age structure, foreign population, gender
- Economy: GDP, sectoral composition, enterprise structure
- Labor Market: Unemployment rates (overall, youth, long-term)
- Education: School completion rates, students, apprentices
- Income: Purchasing power, low-income households
- Healthcare: Physician density, hospital beds, GP density
- Childcare: Coverage rates for under-3 and 3-6 age groups
- Housing: Building permits, rent levels, living space
- Transport: Cars per capita
- Public Finances: Municipal debt, tax revenue

County codes are formatted as 5-digit AGS codes matching GERDA's harmonized county codes (2021 boundaries).

Value

A data frame with 11,200 rows and 32 columns containing county-level covariates for 400 German counties from 1995 to 2022. See gerda_covariates_codebook for variable descriptions.

See Also

- add_gerda_covariates for automatic merging (recommended)
- gerda_covariates_codebook for variable descriptions

Examples

```
# Get the covariates data (bundled, no network call)
covs <- gerda_covariates()

# Inspect the data
head(covs)
summary(covs)

# Not run: downloads election data from GitHub.
## Not run:
# Manual merge (advanced)
library(dplyr)
elections <- load_gerda_web("federal_cty_harm")
merged <- elections %>%
  left_join(covs, by = c("county_code" = "county_code", "election_year" = "year"))

## End(Not run)
```

---

gerda_covariates_codebook

Get Codebook for County-Level Covariates

Description

Returns the data dictionary for county-level (Kreis) covariates from INKAR. Provides variable names, labels, units, categories, original INKAR codes, and missing data information for all county-level socioeconomic and demographic indicators.

Usage

```
gerda_covariates_codebook()
```

Value

A data frame with 32 rows documenting all variables in the county covariates dataset.

See Also

gerda_covariates for the actual covariate data

Examples

```
# View the full codebook
codebook <- gerda_covariates_codebook()
print(codebook)

# Find variables by category
library(dplyr)
codebook %>%
  filter(category == "Demographics")

# Find variables with good coverage
codebook %>%
  filter(missing_pct < 5)
```

---

gerda_data_list *List of GERDA Data*

Description

This function lists the available GERDA data sets. The purpose of this function is to quickly provide a list of available data sets and their descriptions.

Usage

```
gerda_data_list(print_table = TRUE)
```

Arguments

print_table	A logical value indicating whether to print the table in the console (TRUE) or return the data as a tibble (FALSE). Default is TRUE.

Details

In addition to downloadable datasets, the package includes bundled covariate data accessible via dedicated functions:

- gerda_covariates: County-level INKAR covariates (1995-2022)
- gerda_census: Municipality-level Census 2022 data

The returned tibble carries structured metadata beyond the name and description. Use print_table = FALSE to access these columns:

election_type municipal, state, federal, county-kreistag, european, mayoral, landrat, crosswalk, or covariate.

geographic_level `municipality`, `county`, `wahlkreis`, or `person`.

year_start, year_end Election year range, or `NA` where the dataset has no explicit stated span (e.g. crosswalks and covariates).

boundary Harmonization target (`unharmonized`, `harmonized`, `current`, `raw`, `2021`, `2023`, `2025`, or `NA`).

formats Available download formats (`"csv,rds"` or `"rds"`).

candidate_info `TRUE` if the dataset includes person-level candidate / office-holder identities.

Value

A tibble containing the available GERDA data with descriptions and structured metadata. When print_table = TRUE, the function prints a formatted table (name and description) to the console and invisibly returns the full tibble. When print_table = FALSE, the function directly returns the tibble.

Examples

```
gerda_data_list()

# Access the structured metadata (geographic level, years, formats, ...)
meta <- gerda_data_list(print_table = FALSE)
meta[meta$election_type == "federal", ]
```

`gerda_join_diagnostics` *Inspect GERDA Join Diagnostics*

Description

Returns the machine-readable reports attached by `add_gerda_covariates()` and `add_gerda_census()`. Both helpers preserve earlier reports, so a pipe that performs both joins returns one row per join in execution order.

Usage

```
gerda_join_diagnostics(x)
```

Arguments

`x`	A data frame returned by `add_gerda_covariates()` or `add_gerda_census()`.

Details

The diagnostics are stored as an attribute and are intended to be inspected immediately after the join pipeline. The GERDA join helpers preserve the full history, but unrelated data-frame transformations may remove custom attributes.

Value

A tibble with one row per GERDA join and the following columns:

helper, data_level, identifier, join_keys The helper, inferred geographic level, identifier column, and join-key mapping. `join_keys` is a list-column.

input_rows, output_rows Row counts before and after the join.

eligible_rows, matched_rows, unmatched_rows Rows eligible for matching after excluding temporal out-of-coverage rows, plus total matched and unmatched rows. Census joins treat every row as eligible.

outside_coverage_rows, unexpected_unmatched_rows, missing_key_rows Unmatched-row classifications. A missing join key is unexpected, even when another key would fall outside coverage.

matched_units, unmatched_units, outside_coverage_units, unexpected_unmatched_units Counts of distinct non-missing geographic identifiers in each category.

eligible_match_rate Matched eligible rows divided by all eligible rows, or `NA` if there are no eligible rows.

coverage_start, coverage_end Temporal reference-data coverage; `NA` for the time-invariant Census join.

unmatched_action The requested `unmatched` mode.

unexpected_unmatched_identifiers, outside_coverage_years List-columns containing the distinct values in each category. Missing identifiers are counted in `missing_key_rows` but cannot appear in the identifier list.

See Also

`add_gerda_covariates()`, `add_gerda_census()`

Examples

```
census <- gerda_census()
election_data <- data.frame(ags = head(census$ags, 2))
joined <- add_gerda_census(election_data, unmatched = "error")
gerda_join_diagnostics(joined)
```

`load_gerda_web` *Load GERDA Data*

Description

This function loads GERDA data from a web source.

Usage

```
load_gerda_web(
  file_name,
  verbose = FALSE,
  file_format = "rds",
  on_error = getOption("gerda.on_error", "warn"),
  timeout = getOption("gerda.timeout", 300),
  max_retries = getOption("gerda.max_retries", 2),
  cache = getOption("gerda.cache", FALSE),
  refresh = FALSE
)
```

Arguments

file_name	A character string specifying the name of the file to load. For a list of available data, see gerda_data_list.
verbose	A logical value indicating whether to print additional messages to the console. Default is FALSE.
file_format	A character string specifying the format of the file. Must be either "csv" or "rds". Default is "rds".
on_error	How to handle errors (unknown dataset name, failed download, corrupt file, invalid file_format). Either "warn" (default) to emit a warning and return NULL, or "stop" to raise an error. Use "stop" inside scripts or pipelines where silent NULL returns would produce confusing downstream failures. The global default can also be overridden with options(gerda.on_error = "stop").
timeout	Download timeout in seconds. Defaults to getOption("gerda.timeout", 300). The value is applied only for the duration of the download (raising R's default of 60s, which is too short for the larger GERDA files) and restored afterwards.
max_retries	Number of additional download attempts after the first, with exponential back-off, if a download fails or returns a Git LFS pointer instead of data. Defaults to getOption("gerda.max_retries", 2) (so up to three attempts in total).
cache	Logical; if TRUE, downloaded datasets are cached on disk (in gerda_cache_dir()) and reused on subsequent calls instead of being re-downloaded. Defaults to getOption("gerda.cache", FALSE). Caching is opt-in so the package never writes to user filespace without consent. See clear_gerda_cache() to purge the cache.
refresh	Logical; if TRUE, ignore any cached copy and force a fresh download (updating the cache when cache = TRUE). Default is FALSE.

Value

A tibble containing the loaded data, or NULL if the data could not be loaded.

Vote-share columns

Election datasets expose one column per party (e.g. cdu, spd, gruene, afd). These columns hold the party's share of valid votes and are expressed as fractions of 1. They do **not** sum to 1 across the

named major parties: the remainder is held by smaller parties with their own columns and, at the tail, an `other` category. For example, in `federal_cty_harm` for 2021, `cdu + csu + spd + gruene + fdp + linke_pds + afd` is typically around 0.91 and ranges roughly 0.78 to 0.97 across counties. To reconstruct a full 1.0 share, include every party column or use `other` together with turnout and invalid-vote columns.

See Also

`gerda_data_list()` for the dataset catalog, `clear_gerda_cache()` and `gerda_cache_dir()` for cache management.

Examples

```
# These examples download datasets from GitHub (up to ~125 MB each) and are
# therefore not run automatically.
## Not run:
# Load harmonized municipal elections data
data_municipal_harm <- load_gerda_web("municipal_harm", verbose = TRUE, file_format = "rds")

# Load federal election data harmonized to 2025 boundaries (includes 2025 election)
data_federal_2025 <- load_gerda_web("federal_muni_harm_25", verbose = TRUE, file_format = "rds")

# Cache the download so repeated calls in the same project reuse it. Caching
# is opt-in because it writes to gerda_cache_dir() outside the session's
# temporary directory.
data_federal_2025 <- load_gerda_web("federal_muni_harm_25", cache = TRUE)

## End(Not run)
```

---

party_crosswalk *Map GERDA Party Names to ParlGov Attributes*

Description

Creates a crosswalk between GERDA party names and ParlGov's `view_party` attributes. If a party name is not found, the corresponding output element is `NA`. This function expects GERDA party names (lowercase, underscores); other naming schemes will mostly return `NA`.

Usage

```
party_crosswalk(party_gerda, destination)
```

Arguments

`party_gerda`	A character vector containing the GERDA party names to be converted.
`destination`	A single string naming the target column. Available destinations:

- **Names**: `party_name`, `party_name_ascii`, `party_name_short`, `party_name_english`

- **Party family**: `family_name`, `family_name_short`
- **Ideology scales (ParlGov)**: `left_right`, `state_market`, `liberty_authority`, `eu_anti_pro`
- **External ideology scores**: `cmp`, `euprofiler`, `ees`, `castles_mair`, `huber_inglehart`, `ray`, `benoit_laver`, `chess`
- **Identifiers**: `country_id`, `party_id`, `family_id`

Value

A vector of the same length as `party_gerda` with the mapped values.

Examples

```
party_crosswalk(c("cdu", "spd", "linke_pds", NA), "left_right")
party_crosswalk(c("cdu", "afd"), "family_name_short")
```

Index