Package 'gerda'

October 29, 2025

Title German Election Database (GERDA)

Version 0.3.0

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

Description Provides tools to download comprehensive datasets of local, state, and federal election results in Germany from 1990 to 2025. The package facilitates access to data on turnout, vote shares for major parties, and demographic information across different levels of government (municipal, state, and federal). It offers access to geographically harmonized datasets that account for changes in municipal boundaries over time and incorporate mail-in voting districts. Users can easily retrieve, clean, and standardize German electoral data, making it ready for analysis. Data is sourced from <https://github.com/awiedem/german_election_data>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports stringdist, dplyr, readr, knitr, tibble

Suggests 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 2025-10-29 05:10:02 UTC

Contents

`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)
```

Arguments

`election_data`	A data frame containing GERDA election data. Must contain a column with county or municipal codes (see Details) and `election_year`.

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.

Value

The input data frame with additional columns for all 20 county-level covariates. The number of rows remains unchanged (left join).

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

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
muni_data %>%
  group_by(county_code_21, 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)
```

`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 20 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: Median income, purchasing power, low-income households

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 22 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
covs <- gerda_covariates()

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

# 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"))
```

---

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 22 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.

Value

A tibble containing the available GERDA data with descriptions. When print_table = TRUE, the function prints a formatted table to the console and invisibly returns the data tibble. When print_table = FALSE, the function directly returns the data tibble.

Examples

```
gerda_data_list()
```

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")
```

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".

Value

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

Examples

```
# 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")
```

party_crosswalk *GERDA*

Description

This function creates a crosswalk between parties and their corresponding names using the ParlGov view_party table. In cases where the party name is not found in the view_party table, the function returns NA. Note that this function should be run on GERDA party names, and will likely not work on other party naming schemes.

Usage

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

Arguments

party_gerda	A character vector containing the GERDA party names to be converted.
destination	The name of the column in the view_party table to map to.

Value

A vector with the mapped party names.

Examples

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

Index