Package: coalitions (via r-universe)

May 8, 2026

Type Package

Title Bayesian ``Now-Cast'' Estimation of Event Probabilities in Multi-Party Democracies

Version 0.6.27

Date 2026-04-12

Maintainer Andreas Bender <bender.at.R@gmail.com>

Description An implementation of a Bayesian framework for the opinion poll based estimation of event probabilities in multi-party electoral systems (Bender and Bauer (2018) <doi:10.21105/joss.00606>).

Depends R (>= 3.2.1)

Imports checkmate, gtools, rvest, xml2, rlang, magrittr, lubridate, stringr, tidyr (>= 1.0.0), purrr (> 0.2.2), dplyr (> 0.5.0), ggplot2, tibble (>= 3.0.0)

Suggests testthat, covr,

Encoding UTF-8

License MIT + file LICENSE

URL https://adibender.github.io/coalitions/

BugReports https://github.com/adibender/coalitions/issues

RoxygenNote 7.3.2

LazyData true

NeedsCompilation no

Author Andreas Bender [aut, cre] (ORCID: <https://orcid.org/0000-0001-5628-8611>), Alexander Bauer [aut] (ORCID: <https://orcid.org/0000-0003-3495-5131>), Rebekka Schade [ctb]

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

Date/Publication 2026-05-08 15:55:30 UTC

RemoteUrl https://github.com/cran/coalitions

RemoteRef HEAD

RemoteSha 8b37f731e2b2b04c24351ae5c6306fd4ea93f63b

Contents

`calculate_prob` *Calculate coalition probability from majority table*

Description

Given a table with simulations in the rows and coalitions in the columns, this function returns the coalition probabilities for a specified coalition, by default excluding superior coalitions first

Usage

```
calculate_prob(majority_df, coalition, exclude_superior = TRUE, ...)
```

Arguments

`majority_df`	A data frame containing logical values indicating if the coalitions (columns) have a majority (rows).
`coalition`	The coalition of interest for which superior coalitions will be obtained by `get_superior`.
`exclude_superior`	Logical. If `TRUE`, superior coalitions will be excluded, otherwise total coalition probabilities will be returned. Usually it makes sense to exclude superior coalitions.
`...`	Further arguments passed to `get_superior`

Value

A data frame with one numeric column giving the coalition probability (percentage of simulations in which the coalition obtained a majority, after optionally excluding superior coalitions).

Examples

```
test_df <- data.frame(
 cdu            = c(rep(FALSE, 9), TRUE),
 cdu_fdp        = c(rep(FALSE, 8), TRUE, TRUE),
 cdu_fdp_greens = c(TRUE, TRUE, rep(FALSE, 6), TRUE, TRUE))
calculate_prob(test_df, "cdu_fdp_greens") # exclude_superior defaults to TRUE
calculate_prob(test_df, "cdu_fdp_greens", exclude_superior=FALSE)
```

---

calculate_probs *Calculate coalition probabilities for multiple coalitions*

Description

Given a table with simulations in the rows and coalitions in the columns, this function returns the coalition probabilities for a specified coalition, by default excluding superior coalitions first

Usage

```
calculate_probs(majority_df, coalitions, exclude_superior = TRUE, ...)
```

Arguments

majority_df	A data frame containing logical values indicating if the coalitions (columns) have a majority (rows).
coalitions	A list of coalitions for which coalition probabilities should be calculated. Each list entry must be a vector of party names. Those names need to correspond to the names in majority_df.
exclude_superior	Logical. If TRUE, superior coalitions will be excluded, otherwise total coalition probabilities will be returned. Usually it makes sense to exclude superior coalitions.
...	Further arguments passed to get_superior

Value

A data frame with columns coalition (character) and probability (numeric, 0–100), one row per coalition.

See Also

calculate_prob

Examples

```
test_df <- data.frame(
 cdu            = c(rep(FALSE, 9), TRUE),
 cdu_fdp        = c(rep(FALSE, 8), TRUE, TRUE),
 cdu_fdp_greens = c(TRUE, TRUE, rep(FALSE, 6), TRUE, TRUE))
calculate_probs(test_df, list("cdu", "cdu_fdp", "cdu_fdp_greens"))
calculate_probs(test_df, list("cdu", "cdu_fdp", "cdu_fdp_greens"), exclude_superior=FALSE)
```

`collapse_parties` *Transform surveys in long format*

Description

Given a data frame containing multiple surveys (one row per survey), transforms the data into long format with one row per party.

Usage

```
collapse_parties(
  surveys,
 parties = c("cdu", "spd", "greens", "fdp", "left", "pirates", "fw", "afd", "bsw",
    "others")
)
```

Arguments

`surveys`	A data frame with one survey per row.
`parties`	A character vector containing names of parties to collapse.

Value

Data frame in long format

Examples

```
emnid <- scrape_wahlrecht()
emnid.long <- collapse_parties(emnid)
```

dHondt *Seat Distribution by D'Hondt*

Description

Calculates number of seats for the respective parties according to the method of d'Hondt.

Usage

```
dHondt(votes, parties, n_seats = 183)
```

Arguments

votes	Number of votes per party.
parties	Names of parties (must be same length as votes).
n_seats	Number of seats in parliament. Defaults to 183 (seats in Austrian parliament).

Value

A named integer vector of seat counts, one entry per party, in the same order as `parties`. The vector has a logical attribute `ties`: `TRUE` if two or more parties had equal claim to the last seat (i.e. the result is not uniquely determined and was resolved randomly), `FALSE` otherwise. When `ties = TRUE`, re-running with a different random seed may produce a different but equally valid seat distribution.

See Also

sls

Examples

```
library(coalitions)
library(dplyr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample) %>% ungroup() %>% slice(1) %>% tidyr::unnest("survey")
# calculate the seat distribution based on D'Hondt for a parliament with 300 seats
dHondt(surveys$votes, surveys$party, n_seats = 300)
```

`draw_from_posterior` *Draw random numbers from posterior distribution*

Description

Draw random numbers from posterior distribution

Usage

```
draw_from_posterior(
  survey,
  nsim = 10000,
  seed = as.numeric(now()),
  prior = NULL,
  correction = NULL
)
```

Arguments

`survey`	survey object as returned by `as_survey` or `getSurveys`
`nsim`	number of simulations
`seed`	sets seed
`prior`	optional prior information. Defaults to 1/2 (Jeffrey's prior).
`correction`	A positive number. If not `NULL`, each sample from the Dirichlet distribution will be additionally "corrected" by a random number from U(-1*correction, 1*correction). This can be used to introduce extra variation which might be useful due to rounding errors from reported survey results (or add an additional source of variation in general).

Value

`data.frame` containing random draws from Dirichlet distribution which can be interpreted as election results.

See Also

`as_survey`

`get_probabilities` *Wrapper for calculation of coalition probabilities from survey*

Description

Given a table with simulations in the rows and coalitions in the columns, this function returns the coalition probabilities for a specified coalition, by default excluding superior coalitions first

Usage

```
get_probabilities(
  x,
 coalitions = list(c("cdu"), c("cdu", "fdp"), c("cdu", "fdp", "greens"), c("spd"),
    c("spd", "left"), c("spd", "left", "greens")),
  nsim = 1e+05,
  distrib.fun = sls,
  seats_majority = 300L,
  seed = as.numeric(now()),
  correction = NULL
)
```

Arguments

`x`	A table containing one row per survey and survey information in long format in a separate column named `survey`.
`coalitions`	A list of coalitions for which coalition probabilities should be calculated. Each list entry must be a vector of party names. Those names need to correspond to the names in `majority_df`.
`nsim`	number of simulations
`distrib.fun`	Function to calculate seat distribution. Defaults to `sls` (Sainte-Lague/Schepers).
`seats_majority`	The number of seats needed to obtain majority.
`seed`	sets seed
`correction`	A positive number. If not `NULL`, each sample from the Dirichlet distribution will be additionally "corrected" by a random number from U(-1*correction, 1*correction). This can be used to introduce extra variation which might be useful due to rounding errors from reported survey results (or add an additional source of variation in general).

Value

A tibble with the same rows as `x` (one per survey) and an additional list-column `probabilities` containing a data frame of coalition names and their probabilities (0–100) for each survey.

See Also

`calculate_prob`

Examples

```
library(coalitions)
library(dplyr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample) %>% ungroup() %>% slice(1)
# calculate probabilities for two coalitions
probs <- get_probabilities(surveys,
                           coalitions = list(c("cdu", "fdp"),
                                             c("spd", "left", "greens")),
                           nsim = 100) # ensure fast runtime with only 100 simulations
probs %>% tidyr::unnest("probabilities")
```

`get_seats` *Calculate seat distribution from draws from posterior*

Description

Calculate seat distribution from draws from posterior

Usage

```
get_seats(
  dirichlet.draws,
  survey,
  distrib.fun = sls,
  samplesize = NULL,
  hurdle = 0.05,
  others = "others",
  ...
)
```

Arguments

`dirichlet.draws`	Matrix containing random draws from posterior.
`survey`	The actual survey results on which `dirichlet.draws` were based on.
`distrib.fun`	Function to calculate seat distribution. Defaults to `sls` (Sainte-Lague/Schepers).
`samplesize`	Number of individuals participating in the `survey`.
`hurdle`	The percentage threshold which has to be reached by a party to enter the parliament. Any party called "ssw" will be exempt from the hurdle.
`others`	A string indicating the name under which parties not listed explicitly are subsumed.
`...`	Further arguments passed to `distrib.fun`.

Value

A data frame containing seat distributions for each simulation in `dirichlet.draws`

See Also

draw_from_posterior, sls, dHondt

Examples

```
library(coalitions)
library(dplyr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample) %>% ungroup() %>% slice(1)
# simulate 100 seat distributions
surveys <- surveys %>% mutate(draws = purrr::map(survey, draw_from_posterior, nsim = 100),
                              seats = purrr::map2(draws, survey, get_seats))
surveys$seats
```

---

get_surveys *Scrape surveys from all pollsters*

Description

Given a specific date, extract the survey from this date or the last one before this date.

Usage

```
get_surveys(country = "DE")

get_surveys_by()

get_surveys_rp()

get_surveys_nds()

get_surveys_saxony()

get_surveys_brb()

get_surveys_thuringen()

get_latest(surveys = NULL, max_date = Sys.Date())
```

Arguments

country	Choose country from which surveys should be scraped. Currently "DE" (Germany) is supported.
surveys	If provided, latest survey will be obtained from this object, otherwise calls get_surveys.
max_date	Specifies the date, relative to which latest survey will be searched for. Defaults to Sys.Date.

Value

Nested tibble. When fully unnested, the dataset contains the following columns:

pollster Character name of the polling institute.

date Publication date of the poll.

start, end Start and end date of the field period, i.e. the dates during which the poll was conducted.

respondents Number of respondents in the poll.

party Character name of an individual party.

percent Percentage of respondents that chose the party. Given in percentage points, i.e. `38%` is given as `38`.

votes Number of respondents that chose the party.

Examples

```
library(coalitions)
get_surveys()

library(coalitions)
### Scrape the newest poll for the German federal election
# Possibility 1: Calling get_latest without arguments scrapes surveys from the web
# Possibility 2: Use get_latest() on an already scraped dataset
surveys <- get_latest(surveys_sample)
```

---

gg_survey *Plot voter shares observed in one survey*

Description

Bar chart of the raw voter shares observed in one survey. Additionally to plotting positive voter shares, the function can be used to plot party-specific differences (e.g. between a survey and the election result), including negative numbers.

Usage

```
gg_survey(data, colors = NULL, labels = NULL, annotate_bars = TRUE, hurdle = 5)
```

Arguments

`data`	Scraped dataset containing one row per party in the column `party` and the observed voter share in the column `percent`
`colors`	Named vector containing party colors. If `NULL` (default) tries to guess color based on party names, gray otherwise.
`labels`	Named vector containing party labels. If `NULL` (default) tries to guess party names from `data`.
`annotate_bars`	If `TRUE` (default) bars are annotated by the respective vote share (percentage).
`hurdle`	Hurdle for single parties to get into the parliament, e.g. '5' for '5%'. If set to NULL no horizontal line is plotted. The horizontal line can be suppressed using `NULL`.

Value

A ggplot object displaying voter shares as a bar chart.

Examples

```
library(tidyr)
library(dplyr)
library(coalitions)

survey <- surveys_sample$surveys[[1]]$survey[[1]]

gg_survey(survey)
```

---

hare_niemeyer *Seat Distribution by Hare/Niemeyer*

Description

Calculates number of seats for the respective parties that have received more than hurdle percent of votes (according to the method of Hare/Niemeyer)

Usage

```
hare_niemeyer(votes, parties, n_seats = 183)
```

Arguments

votes	Number of votes per party.
parties	Names of parties (must be same length as votes).
n_seats	Number of seats in parliament. Defaults to 183 (seats in Austrian parliament).

Value

A data.frame containing parties above the hurdle and the respective seats/percentages after redistribution via Hare/Niemeyer

See Also

sls

Examples

```
library(coalitions)
library(dplyr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample) %>% ungroup() %>% slice(1) %>% tidyr::unnest("survey")
# calculate the seat distribution based on Hare/Niemeyer for a parliament with 300 seats
hare_niemeyer(surveys$votes, surveys$party, n_seats = 300)
```

have_majority *Do coalitions have a majority*

Description

Do coalitions have a majority

Usage

```
have_majority(
  seats_tab,
 coalitions = list(c("cdu"), c("cdu", "fdp"), c("cdu", "fdp", "greens"), c("spd"),
    c("spd", "left"), c("spd", "left", "greens")),
  seats_majority = 300L,
  collapse = "_"
)
```

Arguments

seats_tab	A data frame containing number of seats obtained by a party. Must have columns party and seats.
coalitions	A list of coalitions for which coalition probabilities should be calculated. Each list entry must be a vector of party names. Those names need to correspond to the names in majority_df.
seats_majority	The number of seats needed to obtain majority.
collapse	Character string passed to base::paste.

Value

A data frame with one column per coalition. Each column is logical indicating whether the coalition obtained a majority in each simulation row.

Examples

```
library(coalitions)
library(dplyr)
library(purrr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample) %>% ungroup() %>% slice(1)
# check for majorities of two coalitions
coals <- list(c("cdu", "fdp"),
              c("spd", "left", "greens"))
# only use 100 simulations for a fast runtime
surveys <- surveys %>% mutate(draws = map(survey, draw_from_posterior, nsim = 100),
                              seats = map2(draws, survey, get_seats),
                             majorities = map(seats, have_majority, coalitions = coals))
surveys$majorities
```

party_colors_de *Colors for German parties*

Description

A vector of colors associated with German parties.

Usage

```
party_colors_de
```

Format

A named character vector. Names indicate parties. Values contain color strings for the respective parties

party_labels_de *Labels for German parties*

Description

A vector of labels associated with German parties.

Usage

```
party_labels_de
```

Format

A named character vector. Names indicate parties. Values contain party names suitable for plot labels.

pool_surveys *Obtain pooled survey during specified period*

Description

Per default, pools surveys starting from current date and going 14 days back. For each pollster within the defined time-frame, only the most recent survey is used.

Usage

```
pool_surveys(
  surveys,
  last_date = Sys.Date(),
 pollsters = c("allensbach", "emnid", "forsa", "fgw", "gms", "infratest", "dimap",
    "infratestdimap", "insa"),
  period = 14,
  period_extended = NA,
  corr = 0.5,
  weights = NULL
)
```

Arguments

surveys	A tibble containing survey results for multiple pollsters as returned by get_surveys.
last_date	Only surveys in the time-window from last_date to last_date - period will be considered for each pollster. Defaults to current date.
pollsters	Character vector of pollsters that should be considered for pooling.
period	See last_date argument.
period_extended	Optional. If specified, all surveys in the time-window from last_date - period_extended to last_date - period will also be considered for each pollster, but only after down-weighting them by halving their true sample size.
corr	Assumed correlation between surveys (of different pollsters). Defaults to 0.5.
weights	Additional weights for individual surveys.

Value

A data frame with one row per party containing columns pollster (set to "pooled"), date, start, end, respondents (effective sample size), party, percent, and votes.

Examples

```
library(coalitions)
library(dplyr)
latest <- get_latest(surveys_sample)
pool_surveys(surveys_sample, last_date=as.Date("2017-09-02"))
```

`redistribute` *Calculate percentage of votes/seats after excluding parties with* `votes < hurdle`

Description

Calculate percentage of votes/seats after excluding parties with `votes < hurdle`

Usage

```
redistribute(survey, hurdle = 0.05, others = "others", epsilon = 1e-05)
```

Arguments

`survey`	The actual survey results on which `dirichlet.draws` were based on.
`hurdle`	The percentage threshold which has to be reached by a party to enter the parliament. Any party called "ssw" will be exempt from the hurdle.
`others`	A string indicating the name under which parties not listed explicitly are subsumed.
`epsilon`	Percentages should add up to 1. If they do not, within accuracy of `epsilon`, an error is thrown.

Value

A data frame with the same structure as `survey` but with parties below the hurdle removed and vote percentages renormalized.

See Also

`get_seats`, `sls`

Examples

```
library(coalitions)
library(dplyr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample) %>% ungroup() %>% slice(1)
# redistribute the shares of 'others' parties and parties with a share of under 5\%
surveys <- surveys %>% mutate(survey_redist = purrr::map(survey, redistribute))
surveys$survey # results before redistribution
surveys$survey_redist # results after redistribution
```

scrape_wahlrecht *Scrape surveys for German general election*

Description

Scrapes survey tables and performs sanitation to output tidy data

Usage

```
scrape_wahlrecht(
  address = "https://www.wahlrecht.de/umfragen/emnid.htm",
  parties = c("CDU", "SPD", "GRUENE", "FDP", "LINKE", "PIRATEN", "AFD", "BSW",
    "SONSTIGE")
)

scrape_by(
  address = "https://www.wahlrecht.de/umfragen/landtage/bayern.htm",
 parties = c("CSU", "SPD", "GRUENE", "FDP", "LINKE", "PIRATEN", "FW", "AFD", "SONSTIGE")
)

scrape_rp(
  address = "https://www.wahlrecht.de/umfragen/landtage/rheinland-pfalz.htm",
  parties = c("CDU", "SPD", "GRUENE", "FDP", "LINKE", "AFD", "FW", "SONSTIGE"),
  ind_row_remove = -c(1:3)
)

scrape_ltw(
  address = "https://www.wahlrecht.de/umfragen/landtage/niedersachsen.htm",
 parties = c("CDU", "SPD", "GRUENE", "FDP", "LINKE", "PIRATEN", "FW", "AFD", "BSW",
    "SONSTIGE"),
  ind_row_remove = -c(1:2)
)
```

Arguments

address	http-address from which tables should be scraped.
parties	A character vector containing names of parties to collapse.
ind_row_remove	Negative vector of rows that will be skipped at the beginning.

Value

A tibble with one row per survey date and columns for date, respondents, and one column per party containing the percentage of votes.

Examples

```
library(coalitions)
library(dplyr)
scrape_wahlrecht() %>% slice(1:5)


# Niedersachsen
scrape_ltw() %>% slice(1:5)
# Hessen
scrape_ltw("https://www.wahlrecht.de/umfragen/landtage/hessen.htm", ind_row_remove=-c(1)) %>%
 slice(1:5)
```

`sls` *Seat Distribution by Sainte-Lague/Schepers*

Description

Calculates number of seats for the respective parties that have received more than 5% of votes (according to the method of Sainte-Lague/Schepers, see https://www.wahlrecht.de/verfahren/rangmasszahlen.html).

Usage

```
sls(votes, parties, n_seats = 598L)
```

Arguments

`votes`	A numeric vector giving the redistributes votes
`parties`	A character vector indicating the names of parties with respective `votes`.
`n_seats`	The total number of seats that can be assigned to the different parties.

Value

A named integer vector of seat counts, one entry per party, in the same order as `parties`. The vector has a logical attribute `ties`: `TRUE` if two or more parties had equal claim to the last seat (i.e. the result is not uniquely determined and was resolved randomly), `FALSE` otherwise. When `ties = TRUE`, re-running with a different random seed may produce a different but equally valid seat distribution.

See Also

`dHondt`

Examples

```
library(coalitions)
library(dplyr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample) %>% ungroup() %>% slice(1) %>% tidyr::unnest("survey")
# calculate the seat distribution based on Sainte-Lague/Schepers for a parliament with 300 seats
sls(surveys$votes, surveys$party, n_seats = 300)
```

---

surveys_sample *Sample of selected surveys*

Description

A data set with surveys from seven different pollsters, three surveys per pollster. Surveys report support for different parties in the running for the German Bundestag prior to the 2017 election.

Usage

```
surveys_sample
```

Format

A nested data frame with 7 rows and 2 columns:

institute name of the pollster

surveys a list of data frames, each containing one survey

Source

https://www.wahlrecht.de/

---

try_readHTML *Try call of read_html that throws an error if the url cannot be resolved*

Description

Try call of read_html that throws an error if the url cannot be resolved

Usage

```
try_readHTML(url)
```

Arguments

url	http-address that should be scraped.

Value

An xml_document object as returned by xml2::read_html.

Index