Package ‘coalitions’

June 30, 2021

Type Package

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

Version 0.6.20

Date 2021-06-21

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, jsonlite, RCurl, 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 7.1.1

LazyData true

NeedsCompilation no

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

Repository CRAN

Date/Publication 2021-06-30 08:40:06 UTC
Calculate coalition probability from majority table

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**

```r
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`.

---

**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.
Examples

```r
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**

```r
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`.

**See Also**

- `calculate_prob`

**Examples**

```r
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", "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
## Not run:
emnid <- scrape_wahlrecht()
emnid.long <- collapse_parties(emnid)
## End(Not run)

__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)
**draw_from_posterior**

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>Number of votes per party.</td>
</tr>
<tr>
<td>parties</td>
<td>Names of parties (must be same length as votes).</td>
</tr>
<tr>
<td>n_seats</td>
<td>Number of seats in parliament. Defaults to 183 (seats in Austrian parliament).</td>
</tr>
</tbody>
</table>

**Value**

A numeric vector containing the seats of all parties after redistribution via D’Hondt

**See Also**

`sls`

**Examples**

```r
library(coalitions)
library(dplyr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample) %>% 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**

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

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>survey</td>
<td>survey object as returned by as_survey or getSurveys</td>
</tr>
<tr>
<td>nsim</td>
<td>number of simulations</td>
</tr>
<tr>
<td>seed</td>
<td>sets seed</td>
</tr>
<tr>
<td>prior</td>
<td>optional prior information. Defaults to 1/2 (Jeffrey's prior).</td>
</tr>
</tbody>
</table>
get_probabilities

**correction**

A positive number. If not NULL, each sample from the Dirichlet distribution will be additionally "corrected" by a random number from \((-1 \times \text{correction}, 1 \times \text{correction})\). 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.
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.

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 \( U(-1*\text{correction}, 1*\text{correction}) \). 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).

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)

# 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")
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.
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)
# 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 = c("DE", "AT"))
get_surveys_by()
get_surveys_rp()
get_surveys_nds()
get_surveys_saxony()
get_surveys_brb()
get_surveys

get_surveys_thuringen()

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

Arguments

country Choose country from which surveys should be scraped. Currently "DE" (Germany) and "AT" (Austria) are 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

## Not run:
library(coalitions)
# scrape data for the German federal election
# get_surveys()

## End(Not run)
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**

```r
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`.

**Examples**

```r
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).
have_majority

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) %>% 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.

Examples

library(coalitions)
library(dplyr)
library(purrr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample)
# 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>surveys</td>
<td>A tibble containing survey results for multiple pollsters as returned by get_surveys.</td>
</tr>
<tr>
<td>last_date</td>
<td>Only surveys in the time-window from last_date to last_date - period will be considered for each pollster. Defaults to current date.</td>
</tr>
<tr>
<td>pollsters</td>
<td>Character vector of pollsters that should be considered for pooling.</td>
</tr>
<tr>
<td>period</td>
<td>See last_date argument.</td>
</tr>
</tbody>
</table>
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.

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.
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.

See Also

get_seats, sls
Examples

```r
library(coalitions)
library(dplyr)
# get the latest survey for a sample of German federal election polls
surveys <- get_latest(surveys_sample)
# 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_austria**

*Import Austrian survey results*

**Description**

Reads JSON file from neuwal.com and performs some preprocessing to bring data into standardized format. Returns a nested tibble.

**Usage**

```r
scrape_austria(
  address = "https://neuwal.com/wahlumfragen/data/neuwal-wahlumfragen-user.json"
)
```

**Arguments**

- **address** URL of the JSON file.

**scrape_wahlrecht**

*Scrape surveys for German general election*

**Description**

Scrapes survey tables and performs sanitation to output tidy data

**Usage**

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

```r
c scrape_by(
  address = "https://www.wahlrecht.de/umfragen/landtage/bayern.htm",
  parties = c("CSU", "SPD", "GRUENE", "FDP", "LINKE", "PIRATEN", "FW", "AFD",
```
"SONSTIGE")
}
sr <- 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)
)
slr <- scrape_ltw(
  address = "https://www.wahlrecht.de/umfragen/landtage/niedersachsen.htm",
  parties = c("CDU", "SPD", "GRUENE", "FDP", "LINKE", "PIRATEN", "FW", "AFD",
               "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.

Examples

## Not run:
library(coalitions)
library(dplyr)
# select a polling agency from .pollster_df that should be scraped ...
coalitions:::pollster_df
# ... here we choose Forsa
address <- coalitions:::pollster_df %>% filter(pollster == "forsa") %>% pull(address)
sr <- scrape_wahlrecht(address = address) %>% slice(1:5)

## End(Not run)
## Not run:
# Niedersachsen
slr %>% slice(1:5)
# Hessen
slr("https://www.wahlrecht.de/umfragen/landtage/hessen.htm", ind_row_remove=-c(1)) %>%
  slice(1:5)

## End(Not run)
surveys_sample

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 numeric vector giving the number of seats each party obtained.

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) %>% 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
**try_readHTML**

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

---

**Source**

https://www.wahlrecht.de/

**Description**

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

**Usage**

```python
try_readHTML(url)
```

**Arguments**

- `url` http-address that should be scraped.
# Index

* **datasets**
  * party_colors_de, 12
  * party_labels_de, 13
  * surveys_sample, 17

* **distribution**
  * get_seats, 7

* **seat**
  * get_seats, 7

* **seats**
  * get_seats, 7, 14

* **calculate_prob**
  * as_survey, 6
  * calculate_probs, 3
  * collapse_parties, 4

* **dHondt**
  * 4, 8, 17

* **draw_from_posterior**
  * 5, 8

* **get_latest**
  * (get_surveys), 8

* **get_probabilities**
  * 6

* **get_seats**
  * 7, 14

* **get_superior**
  * 2, 3

* **get_surveys**
  * 8, 9, 13

* **get_surveys_brb**
  * (get_surveys), 8

* **get_surveys_by**
  * (get_surveys), 8

* **get_surveys_nds**
  * (get_surveys), 8

* **get_surveys_rp**
  * (get_surveys), 8

* **get_surveys_saxony**
  * (get_surveys), 8

* **get_surveys_thuringen**
  * (get_surveys), 8

* **gg_survey**
  * 10

* **hare_niemeyer**
  * 10

* **have_majority**
  * 11

* **party_colors_de**
  * 12

* **party_labels_de**
  * 13

* **pool_surveys**
  * 13

* **redistribute**
  * 14

* **scrape_austria**
  * 15

* **scrape_by**
  * (scrape_wahlrecht), 15

* **scrape_ltw**
  * (scrape_wahlrecht), 15

* **scrape_rp**
  * (scrape_wahlrecht), 15

* **scrape_wahlrecht**
  * 15

* **sls**
  * 5, 6, 8, 11, 14, 16

* **surveys_sample**
  * 17

* **try_readHTML**
  * 18