Package ‘eiCompare’

June 22, 2023

Type Package

Title Compares Different Ecological Inference Methods

Version 3.0.2

URL https://github.com/RPVote/eiCompare


License GPL-3

Depends R (>= 3.5.0), eiPack, ei, wru (>= 1.0.0)

Imports bayestestR, coda, data.table, doSNOW, dplyr, foreach, ggplot2, graphics, magrittr, mcmcse, methods, overlapping, purrr, rlang, sf, stringr, tidyselect

NeedsCompilation no

Suggests knitr, plyr, rmarkdown, reshape2, RColorBrewer, RJSONIO, testthat, tigris

RoxygenNote 7.2.1

Encoding UTF-8

VignetteBuilder knitr

Author Loren Collingwood [aut] (<https://orcid.org/0000-0002-4447-8204>),
Ari Decter-Frain [aut, cre] (<https://orcid.org/0000-0001-9635-3334>),
Hikari Murayama [aut] (<https://orcid.org/0000-0002-4067-4734>),
Pratik Sachdeva [aut] (<https://orcid.org/0000-0002-6809-2437>),
Juandelyn Burke [aut] (<https://orcid.org/0000-0002-6345-7505>),
Scott Henderson [ctb] (<https://orcid.org/0000-0003-0624-4965>),
Spencer Wood [ctb] (<https://orcid.org/0000-0002-5794-2619>),
Matt Barreto [ctb] (<https://orcid.org/0000-0003-2423-6395>)

Maintainer Ari Decter-Frain <adecterfrain@gmail.com>
LazyData: true
Repository: CRAN
Date/Publication: 2023-06-22 17:50:02 UTC

R topics documented:

- eiCompare-package .................................................. 3
- bayes_table_make .................................................. 4
- betas_for_return ................................................... 6
- check_args .......................................................... 6
- corona ............................................................... 7
- dedupe_precincts ................................................... 8
- dedupe_voter_file .................................................. 9
- ei_compare-class ................................................... 10
- ei_est_gen .......................................................... 11
- ei_good .............................................................. 14
- ei_homog ............................................................. 15
- ei_iter ............................................................... 16
- ei_rc_congruence .................................................. 18
- ei_rc_good_table ................................................... 19
- ei_reg_bayes_conf_int ............................................. 20
- ei_rxc ............................................................... 22
- elect_algebra ....................................................... 24
- empty_ei_df ........................................................ 26
- ersd_maps .......................................................... 26
- fips_extract ....................................................... 27
- ga_geo .............................................................. 27
- georgia_census ...................................................... 29
- get_ei_iter_se ...................................................... 30
- get_md_bayes_gen_output ......................................... 30
- get_multi_barreled_surnames .................................... 31
- get_results_table .................................................. 31
- get_special_character_surnames ................................ 32
- get_unique_special_characters .................................. 33
- get_word_count ..................................................... 33
- goodman_generalize ............................................... 34
- gwinnett ............................................................ 35
- gwinnett_ei ........................................................ 36
- gwin_fulton_shape ............................................... 36
- lac_10 .............................................................. 37
- lambda_two_compare ............................................... 38
- latlong2fips ....................................................... 39
- mbd_two ............................................................ 40
- mbd_two_minority .................................................. 41
- md_bayes_draw ...................................................... 41
- md_bayes_draw_lambda ............................................. 42
eiCompare-package

Compares EI, Goodman, RxC Estimates

Description

Compares estimates from three ecological inferences routines, based on King et. al.'s approach.

Details

See demo(demo, "eiCompare") for examples on how to use code

Author(s)

Loren Collingwood
Maintainer: Loren Collingwood <loren.collingwood@ucr.edu>
bayes_table_make

References


---

**bayes_table_make**

*EI:RxC Bayes Table Make*

**Description**

Creates data.frame() table from eiPack RxC output, in the same format as ei_est_gen.

**Usage**

```r
bayes_table_make(ei_bayes_object, cand_vector, table_names)
```

**Arguments**

- `ei_bayes_object`  
  Output from eiPack ei.reg.bayes() function
- `cand_vector`  
  Character vector of candidate name variables, usually "pct_johns" or something
- `table_names`  
  Character vector of column names, e.g., c("RxC: Pct Hisp", "RxC: Pct Asian")

**Value**

Data frame object in similar vein to ei_est_gen

**Author(s)**

Loren Collingwood <loren.collingwood@ucr.edu>

**References**


**Examples**

```r
# Toy data example
ca <- runif(5)
cb <- 1 - ca
w <- runif(5)
b <- 1 - w
t <- round(runif(5, min = 20, max = 40), 0)

toy <- data.frame(ca, cb, w, b, t)
```
cands <- c("canda", "candb")
table_names <- c("RxC: PCT Black", "RxC PCT White")

# generate formula for passage to ei.reg.bayes() function
form <- formula(cbind(canda, candb) ~ cbind(black, white))
# run bayesian model
suppressWarnings(
  ei_bayes <- ei.reg.bayes(form, data = toy, sample = 100, truncate = TRUE)
)
# table creation, using function bayes_table_make
ei_bayes_res <- bayes_table_make(ei_bayes,
  cand_vector = cands,
  table_names = table_names)
ei_bayes_res

# Example 2: Corona data

data(corona)
# create character vectors

cands <- c(
  "pct_husted",
  "pct_spiegel",
  "pct_ruth",
  "pct_button",
  "pct_montanez",
  "pct_fox"
)
table_names <- c("RxC: Pct Hisp", "RxC: Pct Asian", "RxC: Pct White")

# generate formula for passage to ei.reg.bayes() function
form <- formula(cbind(
  pct_husted, pct_spiegel, pct_ruth,
  pct_button, pct_montanez, pct_fox
) ~ cbind(pct_hisp, pct_asian, pct_white))
# run bayesian model
suppressWarnings(
  ei_bayes <- ei.reg.bayes(
    form,
    data = corona,
    sample = 10000,
    truncate = TRUE
  )
)
# table creation using bayes_table_make
ei_bayes_res <- bayes_table_make(ei_bayes,
  cand_vector = cands,
  table_names = table_names
)
ei_bayes_res
betas_for_return

Manipulate precinct results to get betas as from ei_est_gen

Description

Manipulate precinct results to get betas as from ei_est_gen

Usage

betas_for_return(precinct_results, race_cand_pairs)

Arguments

precinct_results
A list of betas from ei_iter()

race_cand_pairs
The set of race/candidate pairs tested in ei_iter

Author(s)

Ari Decter-Frain <agd75@cornell.edu>

check_args

Check for missing essential arguments from an ei function

Description

Check for missing essential arguments from an ei function

Usage

check_args(data, cand_cols, race_cols, totals_col, totals_null = FALSE)

Arguments

data
A dataframe upon which EI is to be performed

cand_cols
A column of candidate names passed from ei functions

race_cols
A column of race names passed from ei functions

totals_col
The name of a column passed from ei functions

totals_null
A boolean. If TRUE, ignore totals_col argument

Author(s)

Ari Decter-Frain <agd75@cornell.edu>
Description

This dataset contains precinct vote data and racial demographics from a 2014 election in Corona, CA.

Usage

data(corona)

Format

A data frame with 46 observations on the following 12 variables:

- **precinct** Precinct ID number.
- **totvote** The total vote, per precinct.
- **pct_husted** Percent of vote for Husted.
- **pct_spiegel** Percent of vote for Spiegel.
- **pct_ruth** Percent of vote for Ruth.
- **pct_button** Percent of vote for Button.
- **pct_montanez** Percent of vote for Montanez.
- **pct_fox** Percent of vote for Fox.
- **pct_hisp** Percent of voters identifying as Hispanic.
- **pct_asian** Percent of voters identifying as Asian.
- **pct_white** Percent of voters identifying as white.
- **pct_non_lat** Percent of voters identifying as non-Latino.

Source

Riverside County, CA Board of Elections
**cor_06  Corona 2006 Election Results**

**Description**

This dataset contains precinct vote data from a 2006 election in Corona, CA.

**Usage**

```r
data(cor_06)
```

**Format**

A data frame with 47 observations on the following 8 variables:

- **precinct**: Precinct ID number.
- **totvote**: The total vote, per precinct.
- **pct_latino**: Percent of voters identifying as Latino.
- **pct_other**: Percent of voters identifying as non-Latino.
- **pct_breitenbucher**: Percent of vote for Breitenbucher.
- **pct_montanez**: Percent of voters for Montanez.
- **pct_spiegel**: Percent of voters for Spiegel.
- **pct_skipworth**: Percent of voters for Skipworth.

**Source**

Riverside County, CA Board of Elections.

---

**dedupe_precincts  Remove or identify duplicated precincts**

**Description**

Removes any rows in the dataset that are fully duplicated. If necessary, adds ’ duplicates’ column indicating where precincts appear duplicated, for manual inspection by the user.

**Usage**

```r
dedupe_precincts(data, id_cols, verbose = TRUE)
```
Argument

data A data.frame() object containing precinct-level turnout data by race and candidate.
id_cols The name or index of the column in the data containing unique precinct identifiers. Can pass multiple column names or indices in a vector if precincts are identified over multiple columns (eg. c("precinctid", "countyid").
verbose A boolean. If true, messages are returned describing actions taken by the function.

Value
A new dataframe without duplicated rows, and (if any) a boolean column identifying duplicated precincts for further investigation.

Author(s)
Ari Decter-Frain <agd75@cornell.edu>

Description
Currently, this function removes all but the latest entries in a voter file according to voter ID. This assumes the voter file is sorted by voter ID chronologically.

Usage

dedupe_voter_file(voter_file, voter_id = "voter_id")

Arguments
voter_file The voter file, as a data frame or tibble.
voter_id The column denoting the voter ID.

Details
This function can be updated with more functionality to handle edge cases.

Value
The voter file with duplicates removed.
ei_compare-class

Class "ei_compare"

Description

An S4 class object stemming from ei_rc_good_table(), used for plotting, and examining comparison results.

Objects from the Class

Objects can, in principle, be created by calls of the form new("ei_compare", ...). However, the preferred form is to have them called ei_rc_good_table()

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>

Examples

```r
## Not run:
# TOY DATA EXAMPLE
canda <- c(.1, .09, .85, .9, .92)
candb <- 1 - canda
white <- c(.8, .9, .10, .08, .11)
black <- 1 - white
total <- c(30, 80, 70, 20, 29)
toy <- data.frame(canda, candb, white, black, total)

# CREATE VECTORS
cands <- c("canda")
race_group <- c("- black")  # only use one group for example
table_names <- c("EI: PCT Black", "EI: PCT White")

# RUN ei_est_gen()
# KEEP DATA TO JUST ONE ROW FOR EXAMPLE (time) ONLY!
results <- ei_est_gen(cands, race_group, "total",
                      data = toy[c(1, 3, 5), ], table_names = table_names, sample = 100)

# Generate formula for passage to ei.reg.bayes() function
form <- formula(cbind(canda, candb) ~ cbind(black, white))
# Run Bayesian model
suppressWarnings(
  ei_bayes <- ei.reg.bayes(form, data = toy, sample = 100, truncate = TRUE)
)

table_names <- c("RxC: PCT Black", "RxC: PCT White")
cands <- c("canda", "candb")
ei_bayes_res <- bayes_table_make(ei_bayes, cand_vector = cands, table_names = table_names)
```
ei_est_gen

Iterative EI Estimation

Description

Iteratively fits EI models for candidates and racial/ethnic groups
Usage

```r
ei_est_gen(
    cand_vector, 
    race_group, 
    total, 
    rho = 10, 
    data, 
    table_names, 
    sample = 1000, 
    tomog = FALSE, 
    density_plot = FALSE, 
    plot_path = NULL, 
    beta_yes = FALSE, 
    seed = NULL, 
    ...
)
```

Arguments

- **cand_vector**: Character vector of candidate names, taken from the dataset
- **race_group**: Character vector of formula, e.g., "~ pct_latino"
- **total**: Character vector (e.g., "totvote") of total variable name from data, variable in data is numeric
- **rho**: Rho parameter for ei() estimate, defaults to 10, numeric
- **data**: data.frame() object containing the data
- **table_names**: Character vector of table names with same length as race_group. Used for formatting output. If only one racial group, must provide "Pct. Other" as second element of vector
- **sample**: Number of samples used for EI calculation, default = 1000
- **tomog**: Logical to display tomography plot. If true will save pdf plot to working directory. Default is FALSE
- **density_plot**: Logical to display density plot of betab and betaw. If true will save pdf plot to working directory. Default is FALSE
- **plot_path**: Path to save. If NULL, plot is not saved.
- **beta_yes**: Logical to export betas (b, w) in list object in addition to table of results. Default is FALSE
- **seed**: An integer seed value for replicating estimate results across runs. If NULL, a random seed is chosen. Defaulted to NULL.
- **...**: Arguments passed onto ei() function

Value

Data frame/table object containing EI individually estimated results. If beta_yes=TRUE, two list items, first the data frame table of results, second dataframe of betas themselves.
Note

If this results in an error, "Error in .subset2(x, i, exact = exact) : invalid subscript type "list"", just rerun the algorithm again.

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>

References


Examples

# TOY DATA EXAMPLE

canda <- c(.1, .09, .85, .9, .92)
candb <- 1 - canda
white <- c(.8, .9, .10, .08, .11)
black <- 1 - white
total <- c(30, 80, 70, 20, 29)
toy <- data.frame(canda, candb, white, black, total)

# CREATE VECTORS

cands <- c("canda")
race_group <- c("black") # only use one group for example
table_names <- c("EI: PCT Black", "EI: PCT White")

# RUN ei_est_gen()
# KEEP DATA TO JUST ONE ROW FOR EXAMPLE (time) ONLY!
ei_est_gen(cands, race_group, "total",
  data = toy[c(1, 3, 5), ], table_names = table_names, sample = 100 )

# WARNING -- May take a little while to execute
# Load Package Data
data(corona)
# Create Character Vectors
cands <- c("pct_husted", "pct_spiegel", "pct_ruth", "pct_button", "pct_montanez", "pct_fox")
race_group3 <- c("pct_hisp", "pct_asian", "pct_white")
table_names <- c("EI: Pct Hisp", "EI: Pct Asian", "EI: Pct White")

# Run ei_est_gen function
results <- ei_est_gen(
  cand_vector = cands, race_group = race_group3,
  total = "totvote", data = corona, table_names = table_names
)

results
# Run ei_est_gen function; Exporting betas into data frame
ei_good

EI iterative estimation via Goodman’s Regression

Description

EI iterative estimation via Goodman’s Regression

Usage

ei_good(data, cand_cols, race_cols, totals_col)

Arguments

data          A data.frame() object containing precinct-level turnout data by race and candidate

  cand_cols    A character vector listing the column names for turnout for each candidate

  race_cols    A character vector listing the column names for turnout by race

  totals_col   The name of the column containing total votes cast in each precinct

Value

matrix with precinct results, columns = race groups, rows = candidates

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>
Ari Decter-Frain <agd75@cornell.edu>

References

eiPack King et. al. (http://gking.harvard.edu/eiR)
ei_homog

Homogeneous Precinct Analysis

Description

Creates matrix table of homogeneous precinct analysis results by racial/ethnic group. The idea, for example, is to get a basic sense of voting behavior by racial group, examine candidate preference in districts that are above 80

Usage

```r
ei_homog(
  data,
  cand_cols,
  race_cols,
  totals_col,
  cp = 0.8,
  warn_row = 5,
  verbose = FALSE
)
```

Arguments

- **data**: A data.frame() object containing precinct-level turnout data by race and candidate
- **cand_cols**: A character vector listing the column names for turnout for each candidate
- **race_cols**: A character vector listing the column names for turnout by race
- **totals_col**: The name of the column containing total votes cast in each precinct
- **cp**: numeric; homogeneous precinct cut-point, e.g., 0.80; default = 0.80
- **warn_row**: numeric; threshold number of precincts racial group must be above to conduct analysis; default = 5. For example, with three groups, whites, blacks, Hispanics, each group must have at least 5 precincts with at least 80 that group. All racial groups need to have at least n number of precincts at or above warn_row level or error will be thrown.
- **verbose**: A boolean indicating whether to print out status messages.

Details

- **ei_homog**

Value

matrix with homogeneous precinct results, columns = race groups, rows = candidates
Author(s)
Loren Collingwood <loren.collingwood@ucr.edu>; <loren.collingwood@gmail.com>
Stephen Popick

Examples

# Toy data example
cand_a <- c( rep(.8, 10), rep(.2, 10))
cand_b <- 1 - cand_a
white <- c(rep(.7, 5), rep(.85, 5), rep(.1, 5), rep(.05, 5))
black <- 1 - white
total <- c( rep(200, 5), rep(100, 5), rep(80, 5), rep(300, 5) )
toy <- data.frame(cand_a, cand_b, white, black, total)

# Default Example#
ei_homog(data = toy,
    race_cols = c("white", "black"),
    cand_cols = c("cand_a", "cand_b"),
    totals_col = "total")

# Verbosity Example#
ei_homog(data = toy,
    race_cols = c("white", "black"),
    cand_cols = c("cand_a", "cand_b"),
    totals_col = "total",
    verbose = TRUE)

# Adjust Cut Point (cp) to 0.70
ei_homog(data = toy,
    race_cols = c("white", "black"),
    cand_cols = c("cand_a", "cand_b"),
    totals_col = "total",
    cp = 0.70,
    verbose = TRUE)

# Set Precincts to anything above 3
ei_homog(data = toy,
    race_cols = c("white", "black"),
    cand_cols = c("cand_a", "cand_b"),
    totals_col = "total",
    warn_row = 3,
    verbose = TRUE)

---

**ei_iter**

Iterative EI Estimation

Description

This function runs enables running iterative ecological inference (EI) to estimate the proportion of votes by different race/ethnicity groups for different political candidates.
ei_iter

Usage

ei_iter(
  data,
  cand_cols,
  race_cols,
  totals_col,
  name = "",
  erho = 10,
  seed = NULL,
  plots = FALSE,
  eiCompare_class = TRUE,
  betas = FALSE,
  par_compute = FALSE,
  n_cores = NULL,
  verbose = FALSE,
  plot_path = NULL,
  ...
)

Arguments

data A data.frame() object containing precinct-level turnout data by race and candidate
cand_cols A character vector listing the column names for turnout for each candidate
race_cols A character vector listing the column names for turnout by race
totals_col The name of the column containing total votes cast in each precinct
name A unique identifier for the outputted eiCompare object.
erho A number passed directly to ei::ei(). Defaulted to 10. Can also pass in a vector of erho values
seed An integer seed value for replicating estimate results across runs. If NULL, a random seed is chosen. Defaulted to NULL.
plots A boolean indicating whether or not to include density and tomography plots
eiCompare_class default = TRUE
betas A boolean to return precinct-level betas for each 2x2 ei
par_compute A boolean to conduct ei using parallel processing
n_cores The number of cores to use in parallel computation. Defaulted to NULL, in which case parallel::detectCores() - 1 is used
verbose A boolean indicating whether to print out status messages.
plot_path A string to specify plot save location. If NULL, plot is not saved
... Additional arguments passed directly to ei::ei()
Details

Iterative EI iterates through all possible race-candidate pairs. For each pair, votes by other races and for other candidates are binned and 2x2 ecological inference is run.

This function wraps around the ei function from the ei R package. This function is unstable and can break in arbitrary ways. Errors often emerge with particular values of the erho parameter. If the function breaks, it will automatically try adjusting the erho parameter, first to 20, then to 0.5.

If problems persist, please submit an issue on the eiCompare github repository and include the error message you receive.

Value

If eiCompare_class = TRUE, an object of class eiCompare is returned. Otherwise, a dataframe is returned that matches the formatting of ei_est_gen output.

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>
Ari Decter-Frain <agd75@cornell.edu>
Hikari Murayama <hikari_murayama@berkeley.edu>

References


Description

Calculates congruence scores between EI and RxC for the 2x2 Scenario

Usage

ei_rc_congruence(ei_rc_table, cand_race, group_race)

Arguments

ei_rc_table Object produced from ei_rc_good_table(), where include_good=F, of class ei_compare
cand_race Numeric vector indicating race of the candidates in order they show up in table rownames, where 1=Latino; 2=Black; 3=Asian; 4=White/Non
group_race Numeric vector, taking similar values as cand_race where 1=Latino; 2=Black; 3=Asian; 4=White/Non
ei_rc_good_table

Value

   Table of congruence scores

Author(s)

   Loren Collingwood <loren.collingwood@ucr.edu>, Matt Barreto <barretom@ucla.edu>

---

ei_rc_good_table  Create EI Comparison Table

Description

Takes output from EI model, EI RxC model, Goodman regression, and puts them into a data frame table for useful analysis and comparison.

Usage

   ei_rc_good_table(ei, rc, good, groups, include_good = FALSE)

Arguments

   ei
       Table/data frame object result from ei_est_gen. This assumes beta_yes=FALSE
       in ei_est_gen(). See example below for beta_yes=TRUE in ei_est_gen().

   rc
       Table/data frame from EI:RxC process from bayes_table_make()

   good
       Table/data frame from Goodman regression, from goodman_generalize(). Default is nothing

   groups
       Character vector of voting blocks (e.g., c("Latino", "White"))

   include_good
       Logical, default is FALSE. Set to TRUE if including a Goodman table/data object

Value

   Object of class ei_compare containing a 1. data.frame() slot of comparisons across the three models;
   2. Character vector of group names used for later plotting

Note

   Most of the time the user will not include the Goodman table, as they are interested in the EI vs.
   EI:RxC comparison

Author(s)

   Loren Collingwood <loren.collingwood@ucr.edu>

References

   eiPack, King et. al. (http://gking.harvard.edu/eiR)
**ei_reg_bayes_conf_int**  
*Creates EI Reg Bayes Tables*

**Description**

Creates EI reg bayes tables with confidence bands

**Usage**

```r
ei_reg_bayes_conf_int(ei_bayes)
```

**Arguments**

- `ei_bayes`: Object result of call to `ei.reg.bayes()` function.

**Value**

Matrix object, table of results

**Author(s)**

Loren Collingwood <loren.collingwood@ucr.edu>

**References**

eiPack, King et. al. (http://gking.harvard.edu/eiR)

**Examples**

```r
# Toy data example
cand_a <- c(.1, .09, .85, .9, .92)
cand_b <- 1 - cand_a
white <- c(.8, .9, .10, .08, .11)
black <- 1 - white
total <- c(30, 80, 70, 20, 29)
toy <- data.frame(cand_a, cand_b, white, black, total)

# Create vectors for iterative EI function
cands <- c("cand_a")
race_group <- c("- black")
table_names <- c("EI: PCT Black", "EI: PCT White")

# Run iterative EI using only row for simplicity
results <- ei_est_gen(cands,
race_group,
"total",
data = toy[c(1, 3, 5), ],
```
table_names = table_names, sample = 100
)

# Generate formula for passage to ei.reg.bayes() function
form <- formula(cbind(cand_a, cand_b) ~ cbind(black, white))
# Run Bayesian model
suppressWarnings(
  ei_bayes <- ei.reg.bayes(form, data = toy, sample = 100, truncate = TRUE)
)

# Produce Table
ei_reg_bayes_conf_int(ei_bayes)
# An example using real election. Warning: this example takes a while to run.
# Load corona data
data(corona)
# Generate character vectors
cands <- c("pct_husted", "pct_spiegel", "pct_ruth", "pct_button", "pct_montanez", "pct_fox")
race_group3 <- c("~ pct_hisp", "~ pct_asian", "~ pct_white")
table_names <- c("EI: Pct Lat", "EI: Pct Asian", "EI: Pct White")
# Run EI iterative Fitting
results <- ei_est_gen(
  cand_vector = cands, race_group = race_group3,
  total = "totvote", data = corona, table_names = table_names
)

# EI: RxC model
# Generate formula
form <- formula(cbind(
  pct_husted,
  pct_spiegel,
  pct_ruth,
  pct_button,
  pct_montanez,
  pct_fox
)
  ~ cbind(pct_hisp, pct_asian, pct_white))
suppressWarnings(
  ei_bayes <- ei.reg.bayes(
    form,
    data = corona,
    sample = 10000,
    truncate = TRUE
  )
)
# Produce Table
ei_reg_bayes_conf_int(ei_bayes)
**Description**

EI Bayesian simultaneous estimation for multiple races and candidates

**Usage**

```r
ei_rxc(
  data,
  cand_cols,
  race_cols,
  totals_col,
  name = "",
  ntunes = 10,
  totaldraws = 10000,
  samples = 1e+05,
  thin = 1,
  burnin = 10000,
  ci_size = 0.95,
  seed = NULL,
  eiCompare_class = TRUE,
  ret_mcmc = FALSE,
  verbose = FALSE,
  diagnostic = FALSE,
  n_chains = 3,
  plot_path = NULL,
  par_compute = FALSE,
  n_cores = NULL,
  ...
)
```

**Arguments**

- **data** A data.frame() object containing precinct-level turnout data by race and candidate
- **cand_cols** A character vector listing the column names for turnout for each candidate
- **race_cols** A character vector listing the column names for turnout by race
- **totals_col** The name of the column containing total votes cast in each precinct
- **name** A unique identifier for the outputted eiCompare object.
- **ntunes** Integer number of pre-MCMC tuning runs, defaulted to 10
totaldraws  Integer number of iterations per run in pre-MCMC tuning runs, defaulted to 10000
samples  Integer number of draws saved and used to compute estimates. Total chain length is sample*thin + burnin
thin  Integer specifying the thinning interval for posterior draws. Eg. if thin = 2, every second draw gets added to the sample
burnin  Integer specifying the number of initial iterations to be discarded, defaulted to 10000

ci_size  Numeric desired probability within the upper and lower credible-interval bounds, defaulted to 0.95
seed  A numeric seed value for replicating estimate results across runs. If NULL, a random seed is chosen. Defaulted to NULL.
eiCompare_class  default = TRUE
ret_mcmc  Boolean. If true, the full sample chains are returned
verbose  A boolean indicating whether to print out status messages.
diagnostic  Boolean. If true, run diagnostic test to assess viability of MCMC parameters (will return all chain results)
n_chains  Number of chains for diagnostic test. Default is set to 3.
plot_path  A string to specify plot save location. If NULL, plot is not saved.
par_compute  Boolean. If true, diagnostic test will be run in parallel.
n_cores  The number of cores to use in parallel computation. Defaulted to NULL, in which case parallel::detectCores() - 1 is used
...
Additional parameters passed to eiPack::tuneMD()

Value

If ret_mcmc == TRUE, a list is returned containing results and a data frame of the full chains from the MCMC. If ret_mcmc == FALSE, results are returned in a data frame

A dataframe of ei results

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>, <loren.collingwood@gmail.com>
Hikari Murayama <hikari_murayama@berkeley.edu>
Ari Decter-Frain <agd75@cornell.edu>

References

eiPack, King et al., (http://gking.harvard.edu/eiR)
elect_algebra

Election Algebra for 2x2 Case

Description

Creates data.frame() table of algebraically defined white/non-white preferences for candidates. Typically used when analyst has high confidence in white turnout and voting behavior but needs to deduce minority voting behavior when only CVAP available. First, estimate white/non-white turnout using ei/rxc. Second, gather overall CVAP numbers. Third, estimate candidate preference by white/non-white using ei/rxc. Then enter values into function.

Usage

elect_algebra(totals, c1_ei_res, c2_ei_res, cand_names)

Arguments

totals       data.frame(), dimensions 2x2. Row 1 is white, row 2 is minority. First column is turnout (probably estimated from ei or rxc; e.g.: c(.2876, .1529)); second column is Citizen Voting Age Population (CVAP); e.g.: c(36472, 23851)
c1_ei_res   numeric vector of 2x2 EI candidate results by white voters, estimated from ei or rxc; e.g. c(0.2796, 0.7204) = whites voted 28% for candidate-a and 72% for candidate-b
c2_ei_res   numeric vector of 2x2 EI candidate results by non-white voters, estimated from ei or rxc
cand_names  Character vector of candidate names used for output, e.g.: c("Collingwood", "Barreto")

Details

elect_algebra

Value

Table with estimated candidate A/B votes by race, with columns for percent vote too

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>; <loren.collingwood@gmail.com>

Matt Barreto <barretom@ucla.edu>
Examples

toy <- data.frame(
  precinct = 1:10,
  cvap_white = c(3669, 3349, 5726, 5229, 3862, 2079, 6109, 2098, 2397, 1954),
  cvap_non_white = c(398, 2313, 449, 176, 3138, 6887, 3987, 831, 1493, 4179),
  voted = c(1028, 829, 2350, 1473, 2552, 1029, 2207, 723, 1053, 878),
  novote = c(3039, 4833, 3825, 3932, 4448, 7937, 7889, 2206, 2837, 5255),
  total = c(4067, 5662, 6175, 5405, 7000, 8966, 10096, 2929, 3890, 6133),
  pct_voted = c(0.2527662, 0.1464147, 0.3805668, 0.2725254, 0.3645714, 0.1147669, 0.2186014, 0.2468419, 0.2706941, 0.1431600),
  pct_novote = c(0.7472338, 0.8535853, 0.6194332, 0.7274746, 0.6354286, 0.8852331, 0.7813986, 0.7531581, 0.7293059, 0.8568400),
  pct_white = c(0.9021392, 0.5914871, 0.9272874, 0.9674376, 0.5517143, 0.2318760, 0.6050911, 0.7162854, 0.6161954, 0.3186043),
  pct_nonwhite = c(0.0978608, 0.4085129, 0.0727126, 0.0325624, 0.4482857, 0.7681240, 0.3949089, 0.2837146, 0.6813957),
  cand_a = c(326, 745, 46, 66, 620, 830, 534, 388, 792, 617),
  cand_b = c(702, 84, 2304, 1407, 1932, 199, 1673, 335, 261, 261),
  pct_cand_a_voters = c(0.31712062, 0.89867310, 0.01957447, 0.04480652, 0.24294671, 0.80660836, 0.24195741, 0.53665284, 0.75213675, 0.70273349),
  pct_cand_b_voters = c(0.6828794, 0.1013269, 0.9804255, 0.9551935, 0.04480652, 0.24294671, 0.80660836, 0.24195741, 0.53665284, 0.75213675, 0.70273349)
)

# NOT RUN: Estimate white/non-white Turnout#
#summary(ei_rxc(data = toy,
#  cand_cols = c("pct_voted","pct_novote"),
#  race_cols = c("pct_white", "pct_nonwhite"),
#  totals = "total",
#  seed = 973472)
# )

# Turnout by Race, Estimated: 27-28% White Turnout; 16-17% Minority Turnout
# Citizen Voting Age Population for Whole Jurisdiction; White, Non-White
#cit_totals <- data.frame(turnout = c(0.2786, 0.1663), cvap = c(36472, 23851))

# Not Run: Estimate Vote Choice
# set.seed(197485)
#summary(ei_rxc(data = toy,
#  cand_cols = c("pct_cand_a_voters", "pct_cand_b_voters"),
#  race_cols = c("pct_white", "pct_nonwhite"),
#  totals_col = "total")
# )
# Extract Results
# c1_ei_res <- c(0.2796, 0.7204)
# c2_ei_res <- c(0.7013, 0.2987)
# Set up vectors for function#
# cand_names <- c("Cand A", "Cand B")
# Execute elect_algebra()
# elect_algebra(totals = cit_totals, c1_ei_res, c2_ei_res, cand_names)
empty_ei_df  Create a dataframe with NA values for racial and candidate counts.

Description
Create a dataframe with NA values for racial and candidate counts.

Usage
empty_ei_df(ncand = 2, nrace = 2, nrow = 2)

Arguments
ncand  The number of candidates to include
nrace  The number of race/ethnicities to include
nrow   The number of rows for the dataframe

Value
A dataframe with columns for each candidate and race, all with NAs

ersd_maps  East Ramapo School District Proposed Maps

Description
This dataset contains proposed maps and Citizen Voting Age Population (CVAP) totals for East Ramapo School District.

Usage
data(ersd_maps)

Format
A data frame with 8 observations on the following 8 variables:

WARD  The ward ID number.
TOT.CVAP  The total vote, according to CVAP, per precinct.
WHI.CVAP  The number of white CVAP voters, per precinct.
BLA.CVAP  The number of Black CVAP voters, per precinct.
HIS.CVAP  The number of Hispanic CVAP voters, per precinct.
ASI.CVAP  The number of Asian CVAP voters, per precinct.
MIN.AGG.FRAC  The number of Black/Hispanic CVAP voters, per precinct.
geometry  The geometry for each ward.
**fips_extract**

*Extract geographic unit codes from FIPS codes.*

**Description**

This function will split up a column of FIPS codes into several columns, each containing the individual code at different units. It is agnostic to the level of the FIPS codes (i.e., FIPS codes are not required to be 15 digits long). However, this function assumes that all FIPS codes begin at the state level of precision.

**Usage**

```r
defines_paste0(fips_extract(df, fips_col = NULL, geo = NULL))
```

**Arguments**

- **df**: The dataframe, with one column containing FIPS codes.
- **fips_col**: A string denoting the column containing the FIPS codes.
- **geo**: A string denoting the smallest geographic unit in the FIPS code. If NULL, the smallest geographic unit is determined based off the length of the FIPS codes.

**Value**

A dataframe with additional columns containing the individual codes for different geographic units.

---

**ga_geo**

*Voter file information that has been geocoded*

**Description**

This dataset contains results from geocoding voter addresses using the U.S. Census Bureau. The geocoded voter file has 12 observations and 25 variables that include a geometry of latitude and longitude points and FIPS code values for state, county, tract, and block geographies.

**Usage**

```r
data(ga_geo)
```
Format

A data frame with 12 rows and 25 columns

- **county_code**: Unique identifier for counties in the state of Georgia
- **county_name**: A list of the county name matching the county_code
- **registration_number**: Unique identifier for registered voter identification
- **voter_status**: The registration status of the voter
- **last_name**: The last name of the voter
- **first_name**: The first name of the voter
- **str_num**: The street number of the voter address
- **str_name**: The name of the street of the voter address
- **str_suffix**: The suffix of the street that is commonly directional
- **city**: The city of the voter address
- **state**: The state of the voter address
- **zipcode**: The 5 or 9 digit zipcode of the voter address
- **street_address**: The street number and street name, concatenated
- **final_address**: The street_address, city, state, and zipcode concatenated
- **cxy_address**: The address generated and predicted by the US Census Geocoder
- **cxy_status**: The US Census Geocoder flag for whether an addresses was matched in the US Census Geocoder
- **cxy_quality**: The determinant of whether the addresses matched exctly
- **cxy_matched_address**: The address used to compare with the voter address inputted into the Geocoder API to determine whether a match has occurred
- **cxy_tiger_line_id**: unique identifier from the Tiger line database that captures geographic aras of interests like roads, railroads, rivers, etc.
- **cxy_tiger_side**: a directional identifier in the Tiger Line database
- **STATEFP10**: the FIPS code for the state geographic level
- **COUNTYFP10**: the FIPS code for the county geographic level
- **TRACTCE10**: the FIPS code for the tract geographic level
- **BLOCKCE10**: the FIPS code for the block geographic level
- **geometry**: latitude and longitude coordinates
This dataset contains the demographic information for Fulton and Gwinnett counties in Georgia.

Usage

data(georgia_census)

Format

A nested list which can be sent to the ‘wru_predict_race_wrapper’ function. Within "GA", the "block", "tract", and "county" keys contain the following columns.

- **state**: State FIPS code
- **county**: County FIPS code
- **tract**: Tract FIPS code
- **block**: Block FIPS code
- **P005003**: White alone population
- **P005004**: Black or African American alone population
- **P005005**: American Indian and Alaska Native alone population
- **P005006**: Asian alone population
- **P005007**: Native Hawaiian and Other Pacific Islander alone population
- **P005008**: Some other race alone population
- **P005009**: Two or more races population
- **P005010**: Hispanic or Latino population
- **r_whi**: White voters; from Census Bureau.
- **r_bla**: Black voters; from Census Bureau.
- **r_his**: Hispanic voters; from Census Bureau.
- **r_asi**: Asian voters; from Census Bureau.
- **r_oth**: Other voters; from Census Bureau.

Source

Census Bureau via the WRU package.
get_ei_iter_se

Get 2x2 ei standard errors from ei object Works according to the aggregate formula in King, 1997, section 8.3

Description

Get 2x2 ei standard errors from ei object Works according to the aggregate formula in King, 1997, section 8.3

Usage

get_ei_iter_se(aggs)

Arguments

aggs A dataframe of aggregate value draws, taken from eiread()

Author(s)

Ari Decter-Frain <agd75@cornell.edu>

get_md_bayes_gen_output

Get md_bayes_gen() output from ei_rxc() output

Description

Get md_bayes_gen() output from ei_rxc() output

Usage

get_md_bayes_gen_output(results_table, tag = "")

Arguments

results_table A results table from
tag A string added onto the columns names of each table. If empty string, no tag is added. Tags are separated by underscores.

Value

A list of tables, each keyed by the racial group. The table contains the mean, standard error, and confidence bounds for the EI estimate.

Author(s)

Ari Decter-Frain <agd75@cornell.edu>
get_multi_barreled_surnames

*Gets multi-barreled surnames from a voter file.*

**Description**

A multi-barreled surname is one containing a dash or a space. This function finds all multi-barreled surnames in a voter file.

**Usage**

```r
get_multi_barreled_surnames(
  voter_file,
  surname_col = "last_name",
  regex = "[ -]+"
)
```

**Arguments**

- **voter_file**  
The voter file, with each row consisting of a voter.
- **surname_col**  
A string denoting the surname column.
- **regex**  
A string denoting the regular expression to use for denoting the the special characters.

**Value**

A dataframe of voters whose surnames are multi-barreled.

---

get_results_table

*Get results dataframe from a list of results as from ei_est_gen*

**Description**

Get results dataframe from a list of results as from ei_est_gen

**Usage**

```r
get_results_table(
  district_results,
  cand_col,
  race_col,
  n_cand,
  n_race,
  n_iter,
  add_other = TRUE
)
```
32  

get_special_character_surnames

Arguments

district_results  
A list of dataframes computed in the midst of ei_iter

cand_col  
Passed through from ei_iter

race_col  
Passed through from ei_iter

n_cand  
Passed through from ei_iter

n_race  
Passed through from ei_iter

n_iter  
Passed through from ei_iter

add_other  
A boolean. If true, adds an 'other' column to the output when only one race group is included. Generally, set TRUE for ei_iter, FALSE for ei_good.

Value

a dataframe of results that will work with table comparison funcs.

Author(s)

Ari Decter-Frain <agd75@cornell.edu>

get_special_character_surnames

Description

Returns a subsetted voter file whose rows consist of voters that have special characters in their last name.

Usage

get_special_character_surnames(
  voter_file,
  surname_col = "last_name",
  regex = "[^A-Za-z]"
)

Arguments

voter_file  
The voter file, with each row consisting of a voter.

surname_col  
A string denoting the surname column.

regex  
A string denoting the regular expression to use for querying the the special characters.

Value

A dataframe of voters whose surname has special characters.
get_unique_special_characters

Gets special characters in a column of names.

Description

Returns a unique list of special characters found in a column of a dataframe. By default, these characters consist of any that are not upper- or lower-case letters. This preference can be overwritten by providing a new regular expression.

Usage

```r
get_unique_special_characters(
  voter_file,
  surname_col = "last_name",
  regex = "[A-Za-z]"
)
```

Arguments

- **voter_file**: The voter file, with each row consisting of a voter.
- **surname_col**: A string denoting the surname column.
- **regex**: A string denoting the regular expression to use for identifying non-special characters (by default, alphabetic characters).

Value

A vector of unique special characters found in the names.

get_word_count

Counts the number of words per row in the column of a dataframe.

Description

A “word” is defined as a string of alphabetical characters separated by either spaces or dashes (but not other special characters).

Usage

```r
get_word_count(voter_file, surname_col = "last_name", regex = "[ -]+")
```

Arguments

- **voter_file**: The voter file, with each row consisting of a voter.
- **surname_col**: A string denoting the surname column.
- **regex**: A string denoting the regular expression to use for querying the word count.
goodman_generalize

Value
A vector of word counts.

goodman_generalize  Goodman Regression Generalization

Description
Makes summary table out of multiple heckman regression results, for multiple candidates and groups

Usage
goodman_generalize(cand_vector, race_group, total, data, table_names, ...)

Arguments
- cand_vector: Character vector of candidate names, taken from the dataset
- race_group: Character vector of formula, e.g., "~ pct_latino"
- total: Character vector (e.g., "totvote") of total variable name from data, variable in data is numeric
- data: data.frame() object containing the data
- table_names: Character vector of table names with same length as race_group. Used for formatting output
- ... Arguments passed onto lm() function

Value
Object of class data.frame() returned containing table summary of all the Goodman regressions

Author(s)
Loren Collingwood <loren.collingwood@ucr.edu>

References

See Also
ei_rc_good_table
Examples

# Load corona data
## Not run:
data(corona)
# Generate character vectors
cands <- c("pct_husted", "pct_spiegel", "pct_ruth", "pct_button", "pct_montanez", "pct_fox")
race_group3 <- c("~ pct_hisp", "~ pct_asian", "~ pct_white")

# Goodman Regression
good_corona <- goodman_generalize(cands, race_group3, "totvote", corona, table_names)

## End(Not run)

gwinnett

Election results and racial turnout data for Gwinnett County, Georgia, US

Description
This dataset contains results of the 2018 Georgia gubernatorial election for precincts in Gwinnett County. Data includes counts of votes cast for each candidate and turnout by racial group.

Usage

data(gwinnett)

Format
A data frame with 157 rows and 9 columns

- **precinct** Unique precinct identifier
- **turnout** Count of voter turnout
- **kemp** Count of votes cast for Republican candidate Brian Kemp
- **abrams** Count of votes cast for Democratic candidate Stacey Abrams
- **metz** Count of votes cast for Libertarian candidate Ted Metz
- **white** Count of voters self-reporting as white
- **black** Count of voters self-reporting as black
- **hispanic** Count of voters self-reporting as hispanic
- **other** Count of voters self-reporting any other racial/ethnic group

Details
Data contain the following intentional errors mean for illustration in vignettes: Rows 35 and 36 split up election results for the same precinct. These should be collapsed.
**gwinnett_ei**

Stylized dataset of election results and turnout by race in Gwinnett county, 2018 Georgia gubernatorial election.

**Description**

Stylized dataset of election results and turnout by race in Gwinnett county, 2018 Georgia gubernatorial election.

**Usage**

data(gwinnett_ei)

**Format**

A data frame with 157 rows and 7 columns

- **kemp** Proportion of votes cast for candidate Brian Kemp
- **abrams** Proportion of votes cast for candidate Stacey Abrams
- **metz** Proportion of votes cast for candidate Jim Metz
- **white** Proportion of voters self-reporting as white
- **black** Proportion of voters self-reporting as black
- **other** Proportion of other voters
- **turnout** Count of voter turnout

**gwin_fulton_shape**

Shape file information for Gwinnett and Fulton counties in Georgia

**Description**

This dataset contains results for shape file FIPS codes and geometrie using the tigris package from the US Census Bureau. The values correspond to information about the multipolygon geometry and fips code values for state, county, tract, and block geographies.

**Usage**

data(gwin_fulton_shape)
**Format**

A data frame with 68 rows and 17 columns

- **STATEFP10** the 2010 FIPS code for the state geographic level
- **COUNTYFP10** the 2010 FIPS code for the county geographic level
- **TRACTCE10** the 2010 FIPS code for the tract geographic level
- **BLOCKCE10** the 2010 FIPS code for the block geographic level
- **GEOID10** the 2010 FIPS code for Census block identifier. a concatenation of 2010 Census state FIPS code, 2010 Census county FIPS code, 2010 Census tract code, and 2010 Census block number
- **NAME10** Census block identifier; a concatenation of 2010 Census state FIPS code, 2010 Census county FIPS code, 2010 Census tract code, and 2010 Census block number
- **MTFCC10** MAF/TIGER feature class code (G5040)
- **UR10** 2010 Census urban/rural indicator
- **UACE10** 2010 Census urban area code
- **UATYPE** 2010 Census urban area type
- **FUNCSTAT10** 2010 Census functional status
- **ALAND10** 2010 Census land area
- **AWATER10** 2010 Census water area
- **INTPTLAT10** 2010 Census latitude of the internal point
- **INTPTLON10** 2010 Census longitude of the internal point
- **geometry** latitude and longitude coordinates
- **COUNTYFP** the FIPS code for county
- **STATEFP** the FIPS code for state

---

**Description**

This dataset contains precinct vote data and racial demographics from a 2010 election in Los Angeles County.

**Usage**

data(lac_10)
**Format**

A data frame with 4980 observations on the following 10 variables:

- **precinct**: Precinct ID number.
- **tot_reg**: The total number of registered voters.
- **i_jones**: Number of votes for Jones.
- **i_delatore**: Number of votes for Delatorre.
- **votescast**: The total number of votes cast.
- **lat_voters**: Number of Latino voters.
- **pct_latino**: Percent of voters identifying as Latino.
- **pct_delatore**: Percent of vote for Delatorre.
- **pct_jones**: Percent of vote for Jones.
- **pct_other**: Percent of vote for other candidates.

**Source**

Los Angeles County

---

**lambda_two_compare**

*Lambda Two Compare*

**Description**

Compares two vectors of lambdas, usually one racial group’s support for two separate candidates, or two separate groups’ support for the same candidate.

**Usage**

```r
lambda_two_compare(lmd, cnames, group_name = "Latino", cand1or2 = 1)
```

**Arguments**

- **lmd**: data.frame() object returned from md_bayes_draw_lambda()
- **cnames**: Vector of character (column) names, needs to match relevant column names in md_bayes_draw_lambda return.
- **group_name**: Character string for name appearing in posterior plot. Default is "Latino")
- **cand1or2**: Numeric. Either 1 or 2. Default = 1. Which pairing over the other.

**Value**

Data frame of the probability of one scenario over the other by 10 pct., by 5 pct., greater than 0 (e.g., what is the probability that candidate 1 beats candidate 2 among Latinos by 10 percentage points, etc.)
latlong2fips

Description

Converts latitude/longitude coordinates to 15-digit FIPS code. Communicates with FCC API.

Usage

latlong2fips(latitude, longitude, number)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude</td>
<td>Numeric. Latitude coordinate.</td>
</tr>
<tr>
<td>longitude</td>
<td>Numeric. Longitude coordinate.</td>
</tr>
<tr>
<td>number</td>
<td>Numeric. Usually part of a loop index counter</td>
</tr>
</tbody>
</table>

Value

Character string 15-digit FIPS code corresponding to Lat/Long entry
mbd_two

Multinomial Dirichlet Bayes Draw Two Candidates

Description

Extract posterior means and credible intervals. Need to label candidate vote variables: V1, V2, when two=FALSE, add V3; Hispanic = VtdHVap_cor, White = VtdAVap_cor, Black = VtdB-Vap_cor

Usage

mbd_two(md, colnames, two = TRUE)

Arguments

- **md**: object from ei.MD.bayes() return
- **colnames**: Vector of candidate names. Stick to c(V1,V2) or c(V1,V2,V3)
- **two**: Logical. Two candidates (TRUE), or three (FALSE)

Value

- List with two data frames

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>
**mbd_two_minority**

*Multinomial Dirichlet Bayes Draw Two Candidates, and White/Minority voters*

---

**Description**

Extract posterior means and credible intervals. Need to call variables V1, V2. When two=FALSE, add in V3; Race names = VtdAVap_cor, VtdMVap_cor

**Usage**

```r
mbd_two_minority(md, colnames, two = TRUE)
```

**Arguments**

- `md`: object from `ei.MD.bayes()` return
- `colnames`: Vector of candidate names. Stick to c(V1,V2) or c(V1,V2,V3)
- `two`: Logical. Two candidates (TRUE), or three (FALSE)

**Value**

List with two data frames

**Author(s)**

Loren Collingwood <loren.collingwood@ucr.edu>

---

**md_bayes_draw**

*MD Bayes Draw*

---

**Description**

Tunes and estimates MD Bayes algorithm (ei.MD.bayes)

**Usage**

```r
md_bayes_draw(
  dat,
  race_vote_split,
  form,
  ntunes = 10,
  totaldraws = 1e+05,
  seed = 12345,
  sample = 1e+05,
  thin = 100,
  burnin = 1e+05,
  ret.mcmc = TRUE
)
```
md_bayes_draw_lambda

Arguments

dat          data.frame() object of just raw candidate vote and raw population counts. Put vote results in first set of columns, put population counts next

race_vote_split Numeric vector of length 2 indicating where vote column ends (e.g., 3), and population counts begin (e.g., 4): c(3,4)

form         Formula object, e.g.: cbind(V1, V2, novote) ~ cbind(VtdAVap_cor, VtdBVap_cor, VtdHVap_cor, VtdOVap_cor)

ntunes       Numeric; how much to tune tuneMD. Default = 10

totaldraws   Numeric; How many total draws from MD. Default = 100000

seed         Numeric. Default = 12345

sample       Numeric. Default = 100000

thin         Numeric. Default = 10

burnin       Numeric. Default = 100000

ret.mcmc     Logical. Default = TRUE

Value

Matrix object, of simulation reults

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>

References

eiPack, King et. al. (http://gking.harvard.edu/eiR)

Description

Tunes and estimates MD Bayes algorithm (ei.MD.bayes). Returns a data frame of lambda posterior distribution draws. Similar to md_bayes_draw, but used primarily for assessing posterior distribution tests.
**md_bayes_draw_lambda**

**Usage**

```r
md_bayes_draw_lambda(
  dat,
  race_vote_split,
  form,
  ntunes = 10,
  totaldraws = 1e+05,
  seed = 12345,
  sample = 1e+05,
  thin = 100,
  burnin = 1e+05,
  ret.mcmc = TRUE
)
```

**Arguments**

- `dat` data.frame() object of just raw candidate vote and raw population counts. Put vote results in first set of columns, put population counts next
- `race_vote_split` Numeric vector of length 2 indicating where vote column ends (e.g., 3), and population counts begin (e.g., 4): c(3,4)
- `form` Formula object, e.g.: `cbind(V1, V2, novote) ~ cbind(VtdAVap_cor, VtdBVap_cor, VtdHVap_cor, VtdOVap_cor)`
- `ntunes` Numeric; how much to tune tuneMD. Default = 10
- `totaldraws` Numeric; How many total draws from MD. Default = 100000
- `seed` Numeric. Default = 12345
- `sample` Numeric. Default = 100000
- `thin` Numeric. Default = 10
- `burnin` Numeric. Default = 100000
- `ret.mcmc` Logical. Default = TRUE

**Value**

Posterior distribution of lambdas. This is often used for assessing RPB in elections with a small number of precincts.

**Author(s)**

Loren Collingwood <loren.collingwood@ucr.edu>; Justin Gross <jhgross@umass.edu>

**References**

eiPack, King et al. (http://gking.harvard.edu/eiR)
Examples

```r
# TOY DATA EXAMPLE
canda <- c(10, 8, 10, 4, 8)
candb <- 20 - canda
white <- c(15, 12, 18, 6, 10)
black <- 20 - white
toy <- data.frame(canda, candb, white, black)

# Generate formula for passage to ei.reg.bayes() function
form <- formula(cbind(canda, candb) ~ cbind(black, white))
# Then excute md_bayes_draw(); not run here due to time
md_bayes_draw_lambda(toy, c(2,3), form)
```

**Description**

Tunes and estimates MD Bayes algorithm (ei.MD.bayes). This, combined with md_bayes_table() produces tables of results compatible with EI table of results.

**Usage**

```r
md_bayes_gen(
  dat,
  form,
  total_yes = TRUE,
  total,
  ntunes = 10,
  totaldraws = 10000,
  seed = 12345,
  sample = 1000,
  thin = 100,
  burnin = 10000,
  ret_mcmc = TRUE,
  ci = c(0.025, 0.975),
  ci_true = TRUE,
  produce_draws = FALSE,
  ...
)
```

**Arguments**

- `dat` data.frame() object of just raw candidate vote and raw population counts. Put vote results in first set of columns, put population counts next
form  Formula object, e.g.: cbind(V1, V2, novote) ~ cbind(VtdAVap_cor, VtdBVap_cor, VtdHVap_cor, VtdOVap_cor)
total_yes Logical, default=TRUE. Include total variable from data? Usually when data are stored in percents
total character, total variable column name
ntunes Numeric. How much to tune tuneMD. Default = 10
totaldraws Numeric. Number of total draws from MD. Default = 10000
seed Numeric. Default = 12345
sample Numeric. Default = 10000
thin Numeric. Default = 10
burnin Numeric. Default = 10000
ret_mcmc Logical. Default = TRUE
ci numeric vector of credible interval (low/high), default is 95 percent= c(0.025, 0.975)
ci_true Logical, default = TRUE. Include credible intervals in reported results.
produce_draws Logical, default is FALSE. Produces two-item list of table and md.bayes() mcmc draws (for additional testing and analysis)
... Additional arguments passed to tuneMD() and ei.MD.bayes()

Value

List object of length 1 (when produce_draws=FALSE). List object of length 2 (when produce_draws=TRUE). First item is list of race x candidate tabular results, with mean, SE, and credible intervals. Second item is mcmc draws.

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>

References

eiPack, King et. al. (http://gking.harvard.edu/eiR)

Examples

# TOY DATA EXAMPLE
canda <- c(10, 8, 10, 4, 8)
candb <- 20 - canda
white <- c(15, 12, 18, 6, 10)
black <- 20 - white
toy <- data.frame(canda, candb, white, black)

# Generate formula for passage to ei.reg.bayes() function #
form <- formula(cbind(canda, candb) ~ cbind(black, white))
# Then execute md_bayes_gen(); not run here due to time
md_bayes_gen(
    dat = toy,
    form = form,
    total_yes = FALSE,
    ntunes = 1,
    thin = 1,
    totaldraws = 100,
    sample = 10,
    burnin = 1
)

# Add in mcmc drawings
drawings <- md_bayes_gen(
    dat = toy,
    form = form,
    total_yes = FALSE,
    ntunes = 1,
    thin = 1,
    totaldraws = 100,
    sample = 10,
    burnin = 1,
    produce_draws = TRUE
)
head(drawings$draws)

 md_bayes_table \hspace{1cm} MD Bayes Generalize Table Creation

**Description**

This, combined with md_bayes_gen() produces tables of results compatible with EI table of results.

**Usage**

md_bayes_table(md_results)

**Arguments**

- **md_results** Results object from md_bayes_gen() function.

**Value**

Data.frame object of candidate (rows) and race (columns) RxC results. This, combined with results from ei_est_gen() sends to the ei_rc_good_table() function for combined table results and comparisons.
Author(s)
Loren Collingwood <loren.collingwood@ucr.edu>

References
eiPack, King et. al. (http://gking.harvard.edu/eiR)

Examples

# TOY DATA EXAMPLE
canda <- c(10, 8, 10, 4, 8)
candb <- 20 - canda
white <- c(15, 12, 18, 6, 10)
black <- 20 - white
toy <- data.frame(canda, candb, white, black)

# Generate formula for passage to ei.reg.bayes() function
form <- formula(cbind(canda, candb) ~ cbind(black, white))

# Then execute md_bayes_gen(); not run here due to time
res <- md_bayes_gen(
toy,
form,
total_yes = FALSE,
ntunes = 1,
thin = 1,
totaldraws = 100,
sample = 10,
burnin = 1,

mean_and_ci

mean_and_ci

Description
Internal

Usage

mean_and_ci(cbind_dat, ci = c(0.025, 0.975))
merge_voter_file_to_shape

Arguments

- cbind_dat: cbind object
- ci: Credible intervals. Default: c(.025, .975)

Value

Mean and credible interval

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>

Examples

```r
# EXAMPLE: NOT RUN #
```

```
merge_voter_file_to_shape

Merges a voter file to a shape file.
```

Description

This is achieved by determining the units (e.g., Census block, district, etc.) for which each voter's address lies within.

Usage

```r
merge_voter_file_to_shape(
  voter_file,
  shape_file,
  crs = NULL,
  coords = c("lon", "lat"),
  voter_id = "voter_id"
)
```

Arguments

- voter_file: A dataframe denoting the voter file. If it is not a geometry dataframe, it will be converted to one.
- shape_file: The shapefile for the region, as an sf object.
- crs: The PROJ4 string or int for the coordinate reference system.
- coords: The columns, as a list, that refer to the longitude and latitude.
- voter_id: The column for the Voter ID.
**ny_fips**

**Details**

This function assumes that the sf package was used to read in the shape files.

**Value**

The voter file with unit information attached.

<table>
<thead>
<tr>
<th>ny_fips</th>
<th>New York State FIPS codes</th>
</tr>
</thead>
</table>

**Description**

New York State FIPS codes for 500 voters.

**Usage**

data(ny_fips)

**Format**

A data frame with 500 observations on the following 2 variables:

- **row_id**  Unique identifier.
- **FIP**  The 15-digit FIPS code.

<table>
<thead>
<tr>
<th>ny_voter</th>
<th>New York Voter File Sample</th>
</tr>
</thead>
</table>

**Description**

This dataset contains a sample of 500 voters in East Ramapo School District, New York.

**Usage**

data(ny_voter)
Format

A data frame with 500 observations on the following 10 variables:

- **Voter.ID**  Anonymized voter ID.
- **SD.Poll**  Precinct ID.
- **fips**  The 15-digit FIPS code
- **st**  State FIPS code
- **county**  County FIPS code
- **tract**  Tract FIPS code
- **block**  Block FIPS code
- **st_cty**  State-county FIPS code
- **st_cty_tract**  State-county-tract FIPS code
- **Last.Name**  Voter surname.

Source

East Ramapo School District Board of Elections.

---

**Description**

Internal

**Usage**

`od_plot_create(race, cand_pair, dens_data, out, plot_path = NULL, cand_colors)`

**Arguments**

- **race**  Racial demographic of interest
- **cand_pair**  All possible candidate pairing combinations
- **dens_data**  Beta values long for each race and candidate pair
- **out**  Summary table from `overlay_density_plot` for every race candidate pair
- **plot_path**  Path to save plots. If NULL, plot is not saved.
- **cand_colors**  Colors for every candidate

**Value**

Comparison density plots

overlay density plot comparing candidates for votes by race
**overlay_density_plot**

**Author(s)**

Loren Collingwood <loren.collingwood@ucr.edu>
Hikari Murayama

---

**Description**

Internal

**Usage**

```r
overlay_density_plot(
  agg_betas,
  results_table,
  race_cols,
  cand_cols,
  plot_path,
  ei_type
)
```

**Arguments**

- `agg_betas`: Output for RxC and iterative ei
- `results_table`: Summary table for candidate race pair means and se’s
- `race_cols`: A character vector listing the column names for turnout by race
- `cand_cols`: A character vector listing the column names for turnout for each candidate
- `plot_path`: Path to save
- `ei_type`: Specify whether the data comes from iterative ei ("ei") or rxc ("rxc")

**Value**

Prep and run density plot creation iteratively

**Author(s)**

Loren Collingwood <loren.collingwood@ucr.edu>
Hikari Murayama
performance_analysis  Performs a performance analysis using a voter file, census shape, and district shape.

**Description**

Performs a performance analysis using a voter file, census shape, and district shape.

**Usage**

```r
performance_analysis(
  voter_file,
  district_shape,
  census_shape,
  census_data,
  join_census_shape = TRUE,
  join_district_shape = TRUE,
  state = NULL,
  voter_id = "voter_id",
  surname = "last_name",
  district = "district",
  census_state_col = "STATEFP10",
  census_county_col = "COUNTYFP10",
  census_tract_col = "TRACTCE10",
  census_block_col = "BLOCKCE10",
  crs = NULL,
  coords = c("lon", "lat"),
  census_geo = "block",
  use_surname = TRUE,
  surname_only = FALSE,
  surname_year = 2010,
  use_age = FALSE,
  use_sex = FALSE,
  normalize = TRUE,
  verbose = FALSE
)
```

**Arguments**

- **voter_file**  A dataframe containing the voter file.
- **district_shape**  The shapefiles for the new districts or precincts to consider.
- **census_shape**  The shapefiles for the Census blocks or tracts for which the voter file will be geocoded against.
- **census_data**  A dataframe containing the Census tracts or blocks in the region for the voter file.
join_census_shape
A logical denoting whether the voter file already has the Census block, tract, and county information. If TRUE, then column names for these items must be provided. If FALSE, then a Census shape must be provided in order to perform a spatial join.

join_district_shape
A logical denoting whether the voter file already has the district identity per voter. If TRUE, then a column names for the district must be provided. If FALSE, then a district shape must be provided in order to perform a spatial join.

state
The state in which the functionality analysis is performed, as a two character string.

voter_id
A string denoting the column name for the voter ID.

surname
A string denoting the column name for the surname.

district
A string denoting the column name for the district.

census_state_col
The column in the Census data that indicates state. If the voter file already has Census information, this should denote the column in the voter file containing the state FIPS code.

census_county_col
The column in the Census data that indicates county. If the voter file already has Census information, this should denote the column in the voter file containing the county FIPS code.

census_tract_col
The column in the Census data that indicates tract. If the voter file already has Census information, this should denote the column in the voter file containing the tract FIPS code.

census_block_col
The column in the Census data that indicates block. If the voter file already has Census information, this should denote the column in the voter file containing the block FIPS code.

crs
A string denoting the PROJ4 string for projecting maps.

coords
The columns for the coordinates.

census_geo
The geographic level at which to perform BISG.

use_surname
Whether to use the surname in calculating race probabilities. Passed to WRU.

surname_only
Whether to only use the surname in calculating race probabilities. Passed to WRU.

surname_year
Which Census year to use for surname matching. Passed to WRU.

use_age
Whether to use the age in the BISG calculation. Passed to WRU.

use_sex
Whether to use the sex in the BISG calculation. Passed to WRU.

normalize
If TRUE, normalizes the district percentages.

verbose
If TRUE, will output diagnostic strings.

Value
The processed voter file and a summary of district turnout across racial groups.
plot_eiCompare

Print a plot comparing the predictions of EI outputs.

Description

The output of this plot consists of error bars containing the mean for each candidate, racial group, and eiCompare object. Error bars represent one standard deviation from the mean of the posterior sampling distribution.

Usage

```r
## S3 method for class 'eiCompare'
plot(x, ...)
```

Arguments

- `x`: An eiCompare object, outputted from ei_iter() or ei_rxc().
- `...`: Additional eiCompare objects to summarize.

Value

A ggplot comparing eiCompare objects.

plot_bivariate

Plot bivariate relationships between all combinations of candidates and race/ethnicities

Description

Plot bivariate relationships between all combinations of candidates and race/ethnicities

Usage

```r
plot_bivariate(
  data,
  cand_cols,
  race_cols,
  corrs = FALSE,
  save = FALSE,
  path = NULL
)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A data.frame() object containing precinct-level turnout data by race and candidate</td>
</tr>
<tr>
<td>cand_cols</td>
<td>A character vector listing the column names for turnout for each candidate</td>
</tr>
<tr>
<td>race_cols</td>
<td>A character vector listing the column names for turnout by race</td>
</tr>
<tr>
<td>corrs</td>
<td>A boolean indicating whether to include correlation coefficients on the plot.</td>
</tr>
<tr>
<td>save</td>
<td>A boolean indicating whether to save the plot to a file.</td>
</tr>
<tr>
<td>path</td>
<td>A string to specify plot save location. If NULL, plot is not saved.</td>
</tr>
</tbody>
</table>

Value

ggplot object with bivariate plots faceted by candidate and race

Description

Obtains aggregated precinct counts of racial groups from a voter file. This function is usually applied after application of BISG, when the voter file has probabilistic estimates of race. However, it can be applied more generally, aggregating actual counts of race. This function can perform aggregation over probabilistic estimates of race and ground truth race at the same time.

Usage

```r
precinct_agg_combine(
  voter_file,
  group_col = "precinct",
  race_cols = NULL,
  true_race_col = NULL,
  true_race_keys = NULL,
  include_total = FALSE
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>voter_file</td>
<td>The voter file, as a dataframe. Should contain columns that denote the race probabilities or the actual race of the voter.</td>
</tr>
<tr>
<td>group_col</td>
<td>A string denoting the column to aggregate over (e.g., &quot;precinct&quot;).</td>
</tr>
<tr>
<td>race_cols</td>
<td>A list of strings denoting which columns contain probabilistic estimates of race. By default, it assumes output from WRU. This function does not require all WRU output columns be present in the voter file; rather, it checks which outputs are present and uses those in aggregation.</td>
</tr>
</tbody>
</table>
true_race_col A string denoting which (single) column in the voter file specifies the true race of the voter. If this variable is provided, then true_race_keys must also be provided, or an error is thrown.

true_race_keys A named list, with keys denoting the new race groups (e.g., "white", "black", "hispanic", etc.). The value of each key is a string or list of strings that denote which columns in the voter file map onto the new ground truth race column. This is useful, for example, in mapping multiracial and Native American voters onto an "other" race category. This variable should only be provided if aggregating over the true race.

include_total A logical denoting whether the total counts (potentially rounded) should be included in the output dataframe.

Value

Aggregated dataset of nrow() precinct size, including racial size precinct estimates. Dataset suitable for EI/RxC.

Examples

# Create synthetic voter file with typical BISG output
voter_file <- data.frame(
  precinct = c(1, 1, 2, 2),
  pred.whi = c(0.10, 0.20, 0.30, 0.40),
  pred.bla = c(0.40, 0.30, 0.20, 0.10),
  pred.his = c(0.10, 0.20, 0.30, 0.40),
  pred.asi = c(0.30, 0.20, 0.10, 0.00),
  pred.oth = c(0.10, 0.10, 0.10, 0.10)
)

# Function uses these column names by default
agg <- precinct_agg_combine(
  voter_file = voter_file,
  group_col = "precinct",
  include_total = FALSE
)

# Running aggregation with a ground truth race column
voter_file <- data.frame(
  precinct = c(1, 1, 1, 2, 2, 2, 2, 2),
  race = c("BL", "WH", "NA", "MR", "BL", "WH", "BL", "BL")
)

# Need to specify race keys for true race column
agg <- precinct_agg_combine(
  voter_file = voter_file,
  group_col = "precinct",
  true_race_col = "race",
  true_race_keys = list("whi" = "WH", "bla" = "BL", "oth" = c("NA", "MR")),
  include_total = TRUE
)

# Running aggregation for both predicted and true race columns. Note the # change in column names, which means we need to specify column names.
voter_file <- data.frame(
  precinct = c(1, 1, 2, 2),
  p.whi = c(0.10, 0.20, 0.30, 0.40),
  p.bla = c(0.40, 0.30, 0.20, 0.10),
  p.his = c(0.10, 0.20, 0.30, 0.40),
  p.asi = c(0.30, 0.20, 0.10, 0.00),
  p.oth = c(0.10, 0.10, 0.10, 0.10),
  race = c("BL", "WH", "BL", "WH")
)
agg <- precinct_agg_combine(
  voter_file = voter_file,
  group_col = "precinct",
  race_cols = c("p.whi", "p.bla", "p.his", "p.asi", "p.oth"),
  true_race_col = "race",
  true_race_keys = list("whi" = "WH", "bla" = "BL"),
  include_total = FALSE
)

predict_race_multi_barreled

Predicts, for one row in a voter file, the probability of a voter having a certain race by averaging over each "barrel" of the surname.

Description

Predicts, for one row in a voter file, the probability of a voter having a certain race by averaging over each "barrel" of the surname.

Usage

predict_race_multi_barreled(
  voter_file,
  surname_col = "last_name",
  surname_only = TRUE,
  census_data = NULL,
  census_geo = "block",
  surname_year = 2010,
  use_age = FALSE,
  use_sex = FALSE,
  state = NULL,
  county = NULL,
  tract = NULL,
  block = NULL,
  pattern = "[ -]+",
  remove_patterns = NULL
)
Arguments

- voter_file: The voter file, with each row consisting of a voter.
- surname_col: A string denoting the surname column.
- surname_only: Whether to obtain probabilities for surnames only.
- census_data: A data frame containing Census data corresponding to the geographic information for units in the voter file.
- census_geo: The census level at which to apply BISG. Passed to WRU.
- surname_year: Which Census year to use for surname matching. Passed to WRU.
- use_age: Whether to use the age in the BISG calculation. Passed to WRU.
- use_sex: Whether to use the sex in the BISG calculation. Passed to WRU.
- state: A string denoting the state for which the data is queried.
- county: A string denoting the column containing the county FIPS code.
- tract: A string denoting the column containing the tract FIPS code.
- block: A string denoting the column containing the block FIPS code.
- pattern: What pattern to split surnames on. By default, surnames are split on a space(s), which assumes hyphens have already been removed.
- remove_patterns: A list of strings which will be removed from the list of barrels.

Value

A vector of probabilities for each surname.

---

race_cand_cors

Table of bivariate correlations

Description

Table of bivariate correlations

Usage

race_cand_cors(data, cand_cols, race_cols)

Arguments

- data: A data.frame() object containing precinct-level turnout data by race and candidate
- cand_cols: A character vector listing the column names for turnout for each candidate
- race_cols: A character vector listing the column names for turnout by race

Value

A dataframe of correlation coefficients describing to correlation between the racial proportion of a precinct and the vote share of each candidate.
Description

Checks that both sides of the RxC equation for White/Minority and White, Black, Hispanic, Other, respectively, add up to the same values. If small rounding issues, adjusts the "other" race category.

Usage

```r
desc_check_2_3(
dat,
split = c(3, 4),
catch = FALSE,
catch_col = NULL,
print_sides = TRUE
)
```

Arguments

- **dat**: data.frame() object. One no vote/third party vote column, with candidate votes (for either 2 or 3 candidates), then up to four demographics with last as other
- **split**: Numeric vector of length 2. Default is c(3, 4), for two candidates and one catch-all. c(4,5) for three candidates and one catch all.
- **catch**: Logical (TRUE/FALSE). Catch negative values. Default is FALSE
- **catch_col**: Column names to be caught.
- **print_sides**: Logical (TRUE/FALSE). Print out evaluations. Default is TRUE

Value

Dataset of Left side (Votes) vs. Right side (Demographics). diff column can be tagged on to exiting 'other' category to expedite data preparation process.

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>

Examples

```r
# EXAMPLE: NOT RUN #
```
**ramapo2018**  
*East Ramapo School District 2018 Voter File*

**Description**

This dataset contains a subset of the voter file for voters in East Ramapo School District, in 2018. This file has been modified to protect the privacy of the voters. The voter IDs have been replaced, surnames have been replaced with "similar" surnames, and the file comes already geocoded into Census block and ward.

**Usage**

```r
data(ramapo2018)
```

**Format**

A data frame with 9401 observations on the following 7 variables:

- **voter_id**: The voter ID, recoded to protect voter privacy.
- **last_name**: The surname of the voter.
- **ward**: The ward, as a character, that the voter is located in.
- **state**: FIPS code of state for voter.
- **county**: FIPS code of Census county for voter.
- **tract**: FIPS code of Census tract for voter.
- **block**: FIPS code of Census block for voter.

**Source**

East Ramapo School District

---

**remove_nas**

*Remove missing values from dataset and return warning if any removed*

**Description**

Remove missing values from dataset and return warning if any removed

**Usage**

```r
remove_nas(data)
```

**Arguments**

- **data**: A dataframe on which ei is to be performed.
resolve_missing_vals

Author(s)

Ari Decter-Frain <agd75@cornell.edu>

Description

Remove / Impute NAs in an EI dataset

Usage

resolve_missing_vals(
data,  
cand_cols,  
race_cols,  
totals_col,  
na_action = "DROP",  
verbose = TRUE
)

Arguments

data  A data.frame() object containing precinct-level turnout data by race and candidate

cand_cols  A character vector listing the column names for turnout for each candidate

race_cols  A character vector listing the column names for turnout by race

totals_col  The name of the column containing total votes cast in each precinct

na_action  A string indicating how to handle missing values in EI columns. Possible values are "DROP" and "MEAN". "DROP" drops all rows where variables are missing. "MEAN" imputes missing values as the mean of the column

verbose  A boolean indicating whether to give status updates

Value

A dataframe of inputs to ecological inference without any missing values.
rockland_census

Rockland County, NY, Census demographic dataset.

Description

This dataset contains the demographic information for Rockland County in New York, which is where East Ramapo School District is located.

Usage

data(rockland_census)

Format

A nested list which can be sent to the 'predict_race' function in WRU. Within "NY", the "block", "tract", and "county" keys contain the following columns.

- **state**  State FIPS code
- **county** County FIPS code
- **tract**  Tract FIPS code
- **block**  Block FIPS code
- **P005003** White alone population
- **P005004** Black or African American alone population
- **P005005** American Indian and Alaska Native alone population
- **P005006** Asian alone population
- **P005007** Native Hawaiian and Other Pacific Islander alone population
- **P005008** Some other race alone population
- **P005009** Two or more races population
- **P005010** Hispanic or Latino population
- **r_whi** White voters; from Census Bureau.
- **r_bla** Black voters; from Census Bureau.
- **r_his** Hispanic voters; from Census Bureau.
- **r_asi** Asian voters; from Census Bureau.
- **r_oth** Other voters; from Census Bureau.

Source

Census Bureau via the WRU package.
**rpv_density**

Description

rpv_density

Usage

```r
rpv_density(agg_betas, plot_path = NULL)
```

Arguments

- `agg_betas`: Aggregated beta values
- `plot_path`: Path to save. If NULL, plot is not saved.

Value

Return density for every race/candidate pair for Bb-Bw

Author(s)

Loren Collingwood &lt;loren.collingwood@ucr.edu&gt;
Hikari Murayama

Examples

```r
# EXAMPLE: NOT RUN#
```

**rxc_formula**

Make rxc formula

Description

Make rxc formula

Usage

```r
rxc_formula(cand_cols, race_cols)
```

Arguments

- `cand_cols`: Character vector of candidate column names, passed from `ei_rxc`
- `race_cols`: Character vector of candidate race names, passed from `ei_rxc`
stdize_votes

Author(s)
Ari Decter-Frain <agd75@cornell.edu>

Description
Converts raw vote totals from different voter groups / candidates across precincts into proportions, checking for problematic differences between known vote totals and sums across race/ethnicities.

Usage
stdize_votes(
  data,
  cols,
  totals_col = NULL,
  max_dev = 0.1,
  avg_dev = 0.025,
  new_names = FALSE,
  verbose = TRUE,
  diagnostic = FALSE
)

Arguments
data A dataframe of election results, where each row represents a precinct or geographic voting unit
cols A character vector with the names of the columns indicating total votes cast by each race, or for each candidate
totals_col A character string with the name of the total vote count column in the data. If null, total votes are computed within the function
max_dev A numeric object setting the max allowable deviation of a precinct’s vote sum from totals
avg_dev A numeric object setting the max allowable average deviation difference of all precincts’ vote sums from totals
new_names A boolean indicating whether to return cand and race columns with the same names. If FALSE, names returned with "_prop" added on.
verbose A boolean indicating whether to print status messages
diagnostic A boolean. When true, an extra column of booleans is returned indicating whether each row had a deviation from totals
**stdize_votes_all**

**Details**

- If turnout columns sum row-wise to equal vote_totals, they are returned as proportions.
- If turnout columns sum row-wise to sufficiently close to vote_totals, they are returned as proportions of the sums.
- If turnout columns sum row-wise exceedingly far from vote_totals, the function stops and returns an error message.

**Value**

A dataframe with proportions corresponding to the turnout of each race/ethnicity group

**Author(s)**

Ari Decter-Frain

---

**stdize_votes_all**  

**Description**

Converts a dataframe with total votes for candidates and total votes by each racial/ethnic group into proportions that can be used for Ecological Inference analysis

**Usage**

```r
stdize_votes_all(
  data, 
  race_cols, 
  cand_cols, 
  totals_from = "cand", 
  totals_col = NULL, 
  max_dev_race = 0.1, 
  max_dev_cand = 0.1, 
  avg_dev_race = 0.025, 
  avg_dev_cand = 0.025, 
  new_names = FALSE, 
  ignore_devs = FALSE, 
  verbose = TRUE, 
  diagnostic = FALSE 
)
```

**Arguments**

- `data` A dataframe of election results, where each row represents a precinct or geographic voting unit
**strip_special_characters**

Strips special characters from a voter file.

---

**Description**

Given a voter file and a column, returns a voter file with special characters stripped from that column.

---

**Value**

A dataframe containing columns for each race and candidate converted to percentages and a totals column, ready for Ecological Inference.

---

**Author(s)**

Ari Decter-Frain
Usage

```
strip_special_characters(
    voter_file,
    surname_col = "last_name",
    regex = "[^A-Za-z]+",
    replace = " ",
)
```

Arguments

- `voter_file`: The voter file, with each row consisting of a voter.
- `surname_col`: A string denoting the surname column.
- `regex`: A string denoting the regular expression to use for denoting the special characters.
- `replace`: The replacement string for special characters.

Value

A dataframe of voters whose surname column is stripped of special characters.

---

**summary.eiCompare**

Print a summary of an eiCompare object

Description

Print a summary of an eiCompare object

Usage

```
## S3 method for class 'eiCompare'
summary(object, ...)
```

Arguments

- `object`: An eiCompare object, outputted from ei_iter() or ei_rxc()
- `...`: Additional eiCompare objects to summarize

Value

A nicely formatted dataframe for printing results
**sum_over_cols**  
*Sum row-wise over columns in a dataframe*

**Description**

Simple wrapper of rowSums for checking row sums of race, candidate columns

**Usage**

```r
sum_over_cols(data, cols)
```

**Arguments**

- `data` A data.frame() object containing precinct-level turnout data by race and candidate
- `cols` A set of columns to sum over. Typically, enter cand_cols or race_cols here.

**Value**

A vector of row-wise sums across the column vector entered as argument.

---

**surname_match**  
*Determines which surnames match to the Census list.*

**Description**

Determines which surnames match to the Census list.

**Usage**

```r
surname_match(voter_file, surname_col = "last_name", strip_special = FALSE)
```

**Arguments**

- `voter_file` The voter file, with each row consisting of a voter.
- `surname_col` A string denoting the surname column.
- `strip_special` Whether to strip special characters before matching in the surname database.

**Value**

A vector of logicals denoting a match or not.
surname_summary

Briefly summarizes the surnames in a voter file.

Description

Briefly summarizes the surnames in a voter file.

Usage

surname_summary(voter_file, surname_col)

Arguments

title file The voter file, with each row consisting of a voter.
surname_col A string denoting the surname column.

Value

No return value, called for side effects (message)

tidy_voter_file_wru

Tidies a voter file for WRU.

Description

Checks if columns exist in the original voter file and renames them so that WRU can process the new voter file. Only extract the information needed, tossing the remaining columns.

Usage

tidy_voter_file_wru(
  voter_file,
  voter_id = NULL,
  surname = NULL,
  state = NULL,
  county = NULL,
  tract = NULL,
  block = NULL
)
Arguments

- **voter_file** 
  The voter file, as a data frame or tibble.

- **voter_id** 
  A string denoting the column containing voter ID. Default is NULL, when the voter file does not have an ID or registration number.

- **surname** 
  A string denoting the column containing the surname.

- **state** 
  A string denoting the column containing the state FIPS code.

- **county** 
  A string denoting the column containing the county FIPS code.

- **tract** 
  A string denoting the column containing the tract FIPS code.

- **block** 
  A string denoting the column containing the block FIPS code.

Value

A new voter file that can be read in by WRU functions.

**Description**

This function assumes that the Census data is provided to the function. It does not provide the capability of downloading the Census data, since this is a time intensive process.

**Usage**

```r
wru_predict_race_wrapper(
  voter_file,
  census_data,
  voter_id = NULL,
  surname = "last_name",
  state = NULL,
  county = NULL,
  tract = NULL,
  block = NULL,
  census_geo = NULL,
  use_surname = TRUE,
  surname_only = FALSE,
  surname_year = 2010,
  use_age = FALSE,
  use_sex = FALSE,
  return_surname_flag = FALSE,
  return_geocode_flag = FALSE,
  verbose = FALSE
)
```
Arguments

voter_file  The voter file, containing columns with a surname and potentially geographic information.
census_data  A data frame containing Census data corresponding to the geographic information for units in the voter file.
voter_id  A string denoting the column containing voter ID. Default is NULL, if there is no voter ID in the file. In this case, a voter ID will be assigned.
surname  A string denoting the column containing the surname.
state  A string denoting the column containing the state FIPS code.
county  A string denoting the column containing the county FIPS code.
tract  A string denoting the column containing the tract FIPS code.
block  A string denoting the column containing the block FIPS code.
census_geo  The census level at which to apply BISG. Passed to WRU.
use_surname  Whether to use the surname in calculating race probabilities. Passed to WRU.
surname_only  Whether to only use the surname in calculating race probabilities. Passed to WRU.
surname_year  Which Census year to use for surname matching. Passed to WRU.
use_age  Whether to use the age in the BISG calculation. Passed to WRU.
use_sex  Whether to use the sex in the BISG calculation. Passed to WRU.
return_surname_flag  If TRUE, returns a flag indicating whether the surnames matched.
return_geocode_flag  If TRUE, returns a flag indicating whether the first level of geocode matched.
verbose  A flag indicating whether to print out status messages.

Value

The voter file component extracted from the provided data frame, with additional surname/geocode flags, as well as a data frame race prediction.

References

Imai and Khanna (2016) "Improving Ecological Inference by Predicting Individual Ethnicity from Voter Registration Records"
Index

* **classes**
  - ei_compare-class, 10

* **datasets**
  - cor_06, 8
  - corona, 7
  - ersd_maps, 26
  - ga_geo, 27
  - georgia_census, 29
  - gwin_fulton_shape, 36
  - gwinnett, 35
  - gwinnett_ei, 36
  - lac_10, 37
  - ny_fips, 49
  - ny_voter, 49
  - ramapo2018, 60
  - rockland_census, 62

* **package**
  - eiCompare-package, 3

  - bayes_table_make, 4
  - betas_for_return, 6

  - check_args, 6
  - cor_06, 8
  - corona, 7

  - dedupe_precincts, 8
  - dedupe_voter_file, 9

  - ei_compare-class, 10
  - ei_est_gen, 11
  - ei_good, 14
  - ei_homog, 15
  - ei_iter, 16
  - ei_rc_congruence, 18
  - ei_rc_good_table, 19, 34
  - ei_reg_bayes_conf_int, 20
  - ei_rxc, 22
  - eiCompare (eiCompare-package), 3
  - eiCompare-package, 3

  - elect_algebra, 24
  - empty_ei_df, 26
  - ersd_maps, 26

  - fips_extract, 27

  - ga_geo, 27
  - georgia_census, 29
  - get_ei_iter_se, 30
  - get_md_bayes_gen_output, 30
  - get_multi_barreled_surnames, 31
  - get_results_table, 31
  - get_special_character_surnames, 32
  - get_unique_special_characters, 33
  - get_word_count, 33
  - goodman_generalize, 34
  - gwin_fulton_shape, 36
  - gwinnett, 35
  - gwinnett_ei, 36

  - lac_10, 37
  - lambda_two_compare, 38
  - latlong2fips, 39

  - mbd_two, 40
  - mbd_two_minority, 41
  - md_bayes_draw, 41
  - md_bayes_draw_lambda, 42
  - md_bayes_gen, 44
  - md_bayes_table, 46
  - mean_and_ci, 47
  - merge_voter_file_to_shape, 48

  - ny_fips, 49
  - ny_voter, 49

  - od_plot_create, 50
  - overlay_density_plot, 51

  - performance_analysis, 52
  - plot.eiCompare, 54
INDEX

plot_bivariate, 54
precinct_agg_combine, 55
predict_race_multi_barreled, 57

race_cand_cors, 58
race_check_2_3, 59
ramapo2018, 60
remove_nas, 60
resolve_missing_vals, 61
rockland_census, 62
rpv_density, 63
rxc_formula, 63

stdize_votes, 64
stdize_votes_all, 65
strip_special_characters, 66
sum_over_cols, 68
summary.eiCompare, 67
surname_match, 68
surname_summary, 69

tidy_voter_file_wru, 69

wru_predict_race_wrapper, 70