Package ‘eiCompare’

September 18, 2020

Type Package
Title Compares Ecological Inference, Goodman, Rows by Columns Estimates
Version 3.0.0
URL https://github.com/RPVote/eiCompare
License GPL-3
Depends R (>= 3.5.0), eiPack, ei, wru
Imports censusxy, bayestestR, coda, data.table, doParallel, dplyr, foreach, ggplot2, graphics, magrittr, mcmcse, methods, overlapping, purrr, rlang, sf, stringr, tidyselect
NeedsCompilation no
Suggests knitr, open Cage, plyr, rmarkdown, reshape2, RColorBrewer, RJSPIONI, testthat, tigris
RoxygenNote 7.1.1
Encoding UTF-8
VignetteBuilder knitr
Author Loren Collingwood [aut, cre] (<https://orcid.org/0000-0002-4447-8204>), Ari Decter-Frain [aut] (<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>), Juandalyn 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 Loren Collingwood <loren.collingwood@gmail.com>
Repository CRAN
Date/Publication 2020-09-18 15:00:02 UTC
R topics documented:

- eiCompare-package ........................................ 3
- add_split_comma ........................................... 4
- bayes_table_make ........................................ 4
- betas_for_return .......................................... 6
- check_args .................................................. 7
- concat_final_address .................................... 7
- concat_streetname ........................................ 8
- corona ....................................................... 9
- cor_06 ....................................................... 10
- dedupe_precincts ......................................... 10
- dedupe_voter_file ........................................ 11
- ei_compare-class ......................................... 12
- ei_est_gen .................................................. 13
- ei_good ..................................................... 16
- ei_homog ................................................... 17
- ei_iter ...................................................... 18
- ei_rc_congruence ........................................ 20
- ei_rc_good_table ......................................... 21
- ei_reg_bayes_conf_int ................................... 24
- ei_rxc ....................................................... 26
- elect_algebra .............................................. 28
- empty_ei_df ............................................... 30
- ersd_maps .................................................. 30
- fips_extract ............................................... 31
- ga_geo ....................................................... 32
- georgia_census ........................................... 33
- get_ei_iter_se ............................................ 34
- get_md_bayes_gen_output ................................. 34
- get_multi_barreled_surnames ............................ 35
- get_results_table ........................................ 35
- get_special_character_surnames ........................ 36
- get_unique_special_characters ........................... 37
- get_word_count ............................................ 37
- goodman_generalize ...................................... 38
- gwinnett .................................................... 39
- gwinnett_ei ................................................ 40
- gwin_fulton_shape ........................................ 40
- lac_10 ....................................................... 41
- lambda_two_compare ...................................... 42
- latlong2fips .............................................. 43
- map_interactive ............................................ 44
- map_shape_file ............................................ 45
- map_shape_points ......................................... 46
- mbd_two .................................................... 47
- mbd_two_minority ......................................... 47
- md_bayes_draw ............................................. 48
eiCompare-package

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>

References

add_split_comma
Pre-processes voter file by checking zipcode, and any special characters or typos within the address.

Description
Pre-processes voter file by checking zipcode, and any special characters or typos within the address.

Usage
add_split_comma(voter_file, address = "address", delimiter = "comma")

Arguments
voter_file A voter file containing the address of the voter.
address Either "single" or "split". single addresses are one line address that include street, city, state, and zipcode on one line.
delimiter The type of delimiter (comma, hyphen, dash) that the address parts are split into.

Value
The voter file with pre-processed format for each address variable.

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
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

# Toy data example
canda <- runif(5)
candb <- 1 - canda
white <- runif(5)
black <- 1 - white
total <- round(runif(5, min = 20, max = 40), 0)
toy <- data.frame(canda, candb, white, black, total)
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
## Not run:
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

## End(Not run)

---

**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()
**check_args**

race_cand_pairs

The set of race/candidate pairs tested in ei_iter

**Author(s)**

Ari Decter-Frain <agd75@cornell.edu>

---

**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>

---

**concat_final_address**

This function concatenate the final address

**Description**

This function concatenate the final address

**Usage**

```
concat_final_address(
    voter_file,
    street_address = "street_address",
    city = "city",
    state = "state",
    zipcode = "zipcode"
)
```
concat_streetname

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>voter_file</td>
<td>A voter file containing the address of the voter.</td>
</tr>
<tr>
<td>street_address</td>
<td>The street number and street name of the voters address. Ex. 1442 Market Street</td>
</tr>
<tr>
<td>city</td>
<td>The name of the city that the voter lives in.</td>
</tr>
<tr>
<td>state</td>
<td>The state (based on the United States 50 states) that the voter lives in.</td>
</tr>
<tr>
<td>zipcode</td>
<td>The United States Postal Service (USPS) postal code.</td>
</tr>
</tbody>
</table>

Value

The voter file with pre-processed format for each address variable.

concat_streetname

Pre-processes voter file by checking zipcode, and any special characters or typos within the address.

Description

Pre-processes voter file by checking zipcode, and any special characters or typos within the address.

Usage

```r
concat_streetname(
    voter_file,
    street_number = "street_number",
    street_name = "street_name",
    street_suffix = "street_suffix"
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>voter_file</td>
<td>A voter file containing the address of the voter.</td>
</tr>
<tr>
<td>street_number</td>
<td>The number attached to the street name. Ex. 1442</td>
</tr>
<tr>
<td>street_name</td>
<td>The name of the place in which a voter lives. Ex. Market Street</td>
</tr>
<tr>
<td>street_suffix</td>
<td>A directional abbreviation such as NE for northeast or SW for southwest.</td>
</tr>
</tbody>
</table>

Value

The voter file with pre-processed format for each address variable.
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)
```
dedupe_voter_file

Arguments

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 (e.g. 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>

---

dedupe_voter_file De-duplicates a voter file.

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

ei_est_gen(
  cand_vector,
  race_group,
  total,
  rho = 10,
  data,
  table_names,
  sample = 1000,
  tomog = F,
  density_plot = F,
  beta_yes = F,
  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 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
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
## Not run:
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
results_w_betas <- ei_est_gen(
    cand_vector = cands, race_group = race_group3,
    total = "totvote", data = corona, table_names = table_names, beta_yes = TRUE
)
res1 <- results_w_betas[[1]] # table of mean estimates
res1
res2 <- results_w_betas[[2]] # betas of estimates for each precinct

## End(Not run)

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

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% white, 80% black, etc.

Usage

 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% share of the population for 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.
Usage

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

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
- `verbose`: A boolean indicating whether to print out status messages.
- `plot_path`: A string to specify plot save location. Defaulted to working directory.
- `...`: 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**

```r
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

**Value**

Table of congruence scores
Author(s)
Loren Collingwood <loren.collingwood@ucr.edu>, Matt Barreto <barretom@ucla.edu>

Examples

```r
# LA County 2010 Insurance Commissioner Race
# ei_rc_combine <- ei_rc_good_table(results, ei_bayes_res,
#    groups= c("Latino", "Non Latino"))
## Not run:
load(system.file("extdata/congruence_table.RData", package = "eiCompare"))
ei_rc_congruence(ei_rc_combine2_2, c(1, 4), c(1, 4))
## End(Not run)
```

---

**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

```r
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)

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_bayes_res <- ei_bayes_res[c(1, 2, 5), ]

# Combine Results, results in object of class ei_compare
ei_rc_combine <- ei_rc_good_table(results, ei_bayes_res,
groups = c("Black", "White"))
```
# Produces data and character vector, which can be sent to plot()
ei_rc_combine

# 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
    )
)

# RxC table names
table_names <- c("RxC: Pct Hisp", "RxC: Pct Asian", "RxC: Pct White")

# Table Creation, using function bayes_table_make in ei_est_generalize.R file
ei_bayes_res <- bayes_table_make(ei_bayes,
    cand_vector = cands,
    table_names = table_names
)
ei_reg_bayes_conf_int

# Combine Results, results in object of class ei_compare
ei_rc_combine <- ei_rc_good_table(results,
           ei_bayes_res,
           groups = c("Latino", "Asian", "White")
)
ei_rc_combine

# If set beta_yes = TRUE in ei_est_gen():
ei_rc_combine2 <- ei_rc_good_table(results[[1]],
           ei_bayes_res,
           groups = c("Black", "White")
)

## End(Not run)

---

ei_reg_bayes_conf_int  Creates EI Reg Bayes Tables

Description

Creates EI reg bayes tables with confidence bands

Usage

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)
## Not run:

Toy data example

```r
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)

cands <- c("cand_a")
race_group <- c("- black")
table_names <- c("EI: PCT Black", "EI: PCT White")

ei_est_gen(cands, race_group, "total", data = toy[c(1, 3, 5), ],
table_names = table_names, sample = 100)

ei_reg_bayes(formula = cbind(cand_a, cand_b) ~ cbind(black, white),
data = toy, sample = 100)

ei_reg_bayes_conf_int(ei_bayes)
```

Real election example

```r
load(corona)
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")
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)

# End(Not run)

---

**ei_rxc**  
*EI Bayesian simultaneous estimation for multiple races and candidates*

## 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,
)```

---
ei_rxc

ret_mcmc = FALSE,
verbose = FALSE,
diagnostic = FALSE,
n_chains = 3,
plot_path = "",
par_compute = 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
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 Integery specifying the number of initial iterations to be discarded, defaulted to 10000

Arguments

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. Defaulted to working directory
par_compute Boolean. If true, diagnostic test will be run in parallel.
... 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 dataframe 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)

Description

Election Algebra for 2x2 Case

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, 2079, 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, 0.7837146, 0.3838046, 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, 0.8066836, 0.24195741, 0.53665284,
                        0.75213675, 0.70273349),
  pct_cand_b_voters = c(0.6828794, 0.1013269, 0.9804255, 0.9551935, 0.7570533,
                        0.1933916, 0.7580426, 0.4633472, 0.2478632, 0.2972665)
)

# 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
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 = "total",  #
#  seed = 973472)  #
#)
# 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 = 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/ethnicitiesto 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.

**Source**

East Ramapo School District

---

**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
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.
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 exactly
- `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 areas 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
gometry latitude and longitude coordinates

---

**Description**

This dataset contains the demographic information for Fulton and Gwinnett counties in Georgia.

**Usage**

```r
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 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
)
```
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

Gets surnames containing special characters.

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.
**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`
**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

**Description**

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

**Usage**

```r
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

**Description**

Shape file information for Gwinnett and Fulton counties in Georgia

**Usage**

```r
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**

```r
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_delatorre**: 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
Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>

References

https://geo.fcc.gov/api/census/block/

Examples

```r
## Not run:
# EXAMPLE: NOT RUN #
# census_block <- list()
# num_catch <- rep(NA, nrow(nom_geo))

# for (i in 1:nrow(nom_geo)) {
#  census_block[[i]] <- latlong2fips(nom_geo$lat[i], nom_geo$lon[i], i)
# }

# Row Bind the list into a data.frame object #
# fips_df <- rbindlist(census_block)

## End(Not run)
```

map_interactive

This function allows the user to plot an interactive map of the voter longitude and latitude points.

Description

This function allows the user to plot an interactive map of the voter longitude and latitude points.

Usage

```r
map_interactive(
  voter_file,
  voter_id = "registration_number",
  f_name = "first_name",
  l_name = "last_name",
  fips_code = "county_code",
  latitude = "lat",
  longitude = "lon"
)
```
map_shape_file

Arguments

voter_file  a dataframe with a geometry column for latitude and longitudes created after original voter file was processed with a select geocoder.
voter_id  a unique identifier on the voter registration file.
f_name  the column with first names of voters.
l_name  the column with last names of voters.
fips_code  the column with the fips code for the designated geographic. unit of interest (i.e. state, county, block, tract).
latitude  the column of the of the voter_file that corresponds to latitude coordinates.
longitude  the column of the of the voter_file that corresponds to longitude coordinates.

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>
Juandalyn Burke <jcburke@uw.edu>

Description

Function for making basic choropleth maps of shape_file using the tigris package and ggplot

Usage

map_shape_file(
  shape_file,
  crs = "+proj=latlong +ellps=GRS80 +no_defs",
  title = "Title of the Shapefile"
)

Arguments

shape_file  a shape file based on desired ecological unit (i.e. state, county, block, tract)
crs  the Coordinate reference system, default is crs="+proj=latlong +ellps=GRS80 +no_defs"
title  the tile of the map
map_shape_points

Value

Plots of mapped ecological units desired shape

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>
Juandalyn Burke <jcburke@uw.edu>

map_shape_points

Function for making basic choropleth maps of longitude and latitude points using the tigris package and ggplot

Description

Function for making basic choropleth maps of longitude and latitude points using the tigris package and ggplot

Usage

map_shape_points(
  voter_file,
  shape_file,
  crs = "+proj=longlat +ellps=GRS80",
  title = "title"
)

Arguments

voter_file  a voter file with latitude, longitude, or latitude/longitude and other geographic data.
shape_file  a shape file based on desired ecological unit (i.e. state, county, block, tract)
crs  the Coordinate reference system, default is crs="+proj=latlong +ellps=GRS80 +no_defs"
title  the title of the map

Value

Plots of mapped ecological units desired and voter latitude and longitudes
**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 = VtdBVap_cor

**Usage**

```r
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
)
```

**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(VtdA_Vap_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 results

Author(s)

Loren Collingwood <loren.collingwood@ucr.edu>

References

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

Examples

```r
## Not run:
# 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(toy, c(2,3), form )

## End(Not run)
```
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

## Not run:
# 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 )
## End(Not run)

md_bayes_gen MD Bayes Generalize

## 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
**md_bayes_gen**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample</td>
<td>Numeric. Default = 10000</td>
</tr>
<tr>
<td>thin</td>
<td>Numeric. Default = 10</td>
</tr>
<tr>
<td>burnin</td>
<td>Numeric. Default = 10000</td>
</tr>
<tr>
<td>ret_mcmc</td>
<td>Logical. Default = TRUE</td>
</tr>
<tr>
<td>ci</td>
<td>numeric vector of credible interval (low/high), default is 95 percent= c(0.025, 0.975)</td>
</tr>
<tr>
<td>ci_true</td>
<td>Logical, default = TRUE. Include credible intervals in reported results.</td>
</tr>
<tr>
<td>produce_draws</td>
<td>Logical, default is FALSE. Produces two-item list of table and md.bayes() mcmc draws (for additional testing and analysis)</td>
</tr>
<tr>
<td></td>
<td>... Additional arguments passed to tuneMD() and ei.MD.bayes()</td>
</tr>
</tbody>
</table>

**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**

```r
## Not run:
# 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)

## End(Not run)

---

**md_bayes_table**  
*MD Bayes Generalize Table Creation*

**Description**

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

**Usage**

```r
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

```r
## Not run:
# 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,
  ci_true = FALSE
)
md_bayes_table(res)

## End(Not run)
```

### mean_and_ci

**Description**

Internal

**Usage**

```r
mean_and_ci(cbind_dat, ci = c(0.025, 0.975))
```

**Arguments**

- `cbind_dat` cbind object
- `ci` Credible intervals. Default: `c(0.025, 0.975)`

**Value**

Mean and credible interval
merge_voter_file_to_shape

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

Examples

# 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

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.

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

Value
The voter file with unit information attached.
ny_fips

New York State FIPS codes

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.

ny_voter

New York Voter File Sample

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.
**od_plot_create**

<table>
<thead>
<tr>
<th>Description</th>
<th>Internal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td><strong>od_plot_create(race, cand_pair, dens_data, out, plot_path = &quot;&quot;, cand_colors)</strong></td>
</tr>
</tbody>
</table>

**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
- **cand_colors**: Colors for every candidate

**Value**

Comparison density plots
overlay density plot comparing candidates for votes by race

**Author(s)**

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

---

**overlay_density_plot**

<table>
<thead>
<tr>
<th>Description</th>
<th>Internal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td><strong>overlay_density_plot(agg_betas, results_table, race_cols, cand_cols, plot_path, ei_type)</strong></td>
</tr>
</tbody>
</table>

---
performance_analysis

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

description

Describes

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.
plot.eiCompare

### Value

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

#### 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 = ""
)
```

**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  
- `corrs`: A boolean indicating whether to include correlation coefficients on the plot.  
- `save`: A boolean indicating whether to save the plot to a file.  
- `path`: A string to specify plot save location. Defaulted to working directory

**Value**

A ggplot object with bivariate plots faceted by candidate and race

---

**precinct_agg_combine**  
*Aggregates racial estimates across geographic units*

**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.
precinct_agg_combine(precinct_file,
group_col = "precinct",
race_cols = NULL,
true_race_col = NULL,
true_race_keys = NULL,
include_total = FALSE)

Arguments

voter_file The voter file, as a dataframe. Should contain columns that denote the race probabilities or the actual race of the voter.
group_col A string denoting the column to aggregate over (e.g., "precinct").
race_cols 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.
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
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.
predict_race_multi_barreled

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.
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

race_check_2_3

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

race_check_2_3(
    dat,
    split = c(3, 4),
    catch = FALSE,
    catch_col = NULL,
    print_sides = TRUE
)

Arguments

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

# 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

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
rem**ove**_nas

**Description**

Remove missing values from dataset and return warning if any removed

**Usage**

remove_nas(data)

**Arguments**

data A dataframe on which ei is to be performed.

**Author(s)**

Ari Decter-Frain <agd75@cornell.edu>

---

resolve_missing_vals

**Remove / Impute NAs in an EI dataset**

**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.

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.
**Description**

rpv_density

**Usage**

```
rpv_density(agg_betas, plot_path)
```

**Arguments**

- `agg_betas`  
  Aggregated beta values
- `plot_path`  
  Path to save

**Value**

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

**Author(s)**

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

**Examples**

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

---

**Description**

Geocoding voter file addresses with coordinates (latitude and longitude) and/or census geographies.
Usage

run_geocoder(
  voter_file,
  geocoder = "census",
  parallel = FALSE,
  voter_id = "voter_id",
  street = "street",
  city = "city",
  state = "state",
  zipcode = "residence_zipcode",
  country = NULL,
  census_return = NULL,
  census_benchmark = "Public_AR_Current",
  census_vintage = 4,
  census_output = "single",
  census_class = "sf",
  opencage_key = NULL
)

Arguments

voter_file A data frame contain the voter addresses, separated into columns for street, city, state, and zipcode

geocoder The options for selecting geocoders are "censusxy" and "opencage".

parallel TRUE or FALSE. The option to run parallel processing on the data. Running parallel processing requires the user to have at least 4 CPU cores. Use detectCores() to determine the number of CPUs on your device.

evoter_id the unique identifier

street the street number, street name, and/or street suffix. Ex. 555 Main Street SW

city the location/town

state the abbreviated state (U.S. state categories such as "GA")

zipcode the 5 or 9 digit number in the format XXXXX or XXXXX-XXXX.

country the abbreviated a nation or territory

census_return either "locations" or "geographies". "locations" returns the latitude and longitude coordinates. "geographies" returns the latitude, longitude, and FIPS codes for county, state, tract, and block.

census_benchmark a dataset of the snapshot of the US Census data. Data is collected two times a year. Public_AR_Current is the time period when we created the snapshot of the data (usually done twice yearly). For example, Public_AR_Current is the most recent snapshot of our dataset.

census_vintage a dataset that details the survey or census that the census_benchmark uses.

census_output "single" or "full". "single" indicates that only latitude and longitude are returned. "full" indicates that latitude, longitude, and FIPS codes are returned.
census_class  "sf" indicates returning a shape file in the R class, sf. Other file types like "json" and "csv" can also be used.

opencage_key  the Opencage Geocoder API key needed to run the Opencage Geocoder. The use of the key is limited to the level of membership on Opencage. Only 2500 requests per day for free membership.

Value
The geocoded voter file with either added simple (latitude and longitude coordinates) or other geographies.

Author(s)
Loren Collingwood <loren.collingwood@ucr.edu>
Juandalyn Burke <jcburke@uw.edu>

---

**rxc_formula**

**Make rxc formula**

**Description**
Make rxc formula

**Usage**
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

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

Pre-processes voter file by checking zipcode, and any special characters or typos within the address.

**Description**

Pre-processes voter file by checking zipcode, and any special characters or typos within the address.

**Usage**

```r
split_add_nocommas(voter_file, address = "address", delimiter = "space")
```

**Arguments**

- `voter_file` A voter file containing the address of the voter.
- `address` The voter address in the structure of street number-street name-city-state-zipcode
- `delimiter` The symbol/character used to parse address."space" is used to indicate that the address is parsed by spaces.

**Value**

The voter file with pre-processed format for each address variable.

**Author(s)**

Loren Collingwood <loren.collingwood@ucr.edu>
Juandalyn Burke <jcburke@uw.edu>

---

**stdize_votes**

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.

**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**

```r
stdize_votes(
  data,
  cols,
  totals_col = NULL,
  max_dev = 0.1,
  avg_dev = 0.025,
  new_names = FALSE,
```


```r
stdize_votes

  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

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,  # A dataframe of election results, where each row represents a precinct or geographic voting unit
  race_cols,  # A character vector of colnames corresponding to turnout counts of each race/ethnicity group
  cand_cols,  # A character vector of colnames corresponding to turnout counts of voters for each candidate
  totals_from = "cand",  # A character string, either "cand" or "race" to set whether totals are computed from candidate turnout or race/ethnicity turnout columns. Ignored if totals_col provided.
  totals_col = NULL,  # 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_race = 0.1,  # A numeric object setting the max allowable deviation of any one precincts’ sum of race columns from totals
  max_dev_cand = 0.1,  # A numeric object setting the max allowable deviation of any one precincts’ sum of candidate columns from totals
  avg_dev_race = 0.025,  # A numeric object setting the max allowable mean deviation of all precincts’ sum of race columns from totals
  avg_dev_cand = 0.025,
  new_names = FALSE,  # A character string, either "cand" or "race" to set whether totals are computed from candidate turnout or race/ethnicity turnout columns. Ignored if totals_col provided.
  ignore_devs = FALSE,  # A character string, either "cand" or "race" to set whether totals are computed from candidate turnout or race/ethnicity turnout columns. Ignored if totals_col provided.
  verbose = TRUE,  # A character string, either "cand" or "race" to set whether totals are computed from candidate turnout or race/ethnicity turnout columns. Ignored if totals_col provided.
  diagnostic = FALSE
)
```

Arguments

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

race_cols A character vector of colnames corresponding to turnout counts of each race/ethnicity group

cand_cols A character vector of colnames corresponding to turnout counts of voters for each candidate

totals_from A character string, either "cand" or "race" to set whether totals are computed from candidate turnout or race/ethnicity turnout columns. Ignored if totals_col provided.

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_race A numeric object setting the max allowable deviation of any one precincts’ sum of race columns from totals

max_dev_cand A numeric object setting the max allowable deviation of any one precincts’ sum of candidate columns from totals

avg_dev_race A numeric object setting the max allowable mean deviation of all precincts’ sum of race columns from totals
strip_special_characters

**avg_dev_cand**  A numeric object setting the max allowable mean deviation of all precincts' sum of candidate columns 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.

**ignore_devs**  A boolean. When true, columns are standardized ignoring all deviations from totals

**verbose**  A boolean. When true, function returns progress messages.

**diagnostic**  A boolean. When true, an extra column of booleans is returned indicating whether each row had a deviation from totals

**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

---

**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 stripped from that column.

**Usage**

```r
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 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

```r
## 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**
Determined which surnames match to the Census list.

**Usage**
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**
- voter_file: The voter file, with each row consisting of a voter.
- surname_col: A string denoting the surname column.
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**

```r
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.

`wru_predict_race_wrapper`

*Prepares a voter file for the WRU predict_race function, and then predicts race.*

**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

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.
zip_hyphen

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"

---

zip_hyphen

Pre-processes voter file by checking zipcode, and any special characters or typos within the address.

Description
Pre-processes voter file by checking zipcode, and any special characters or typos within the address.

Usage
zip_hyphen(voter_file, voter_id = "registration_number", zipcode = "zipcode")

Arguments

voter_file A voter file containing the address of the voter.
voter_id The unique identifier linked to the voter.
zipcode The United States Postal Service (USPS) postal code.

Value
The voter file with pre-processed format for each address variable.
# Index

* **classes**
  - ei_compare-class, 12

* **datasets**
  - cor_06, 10
  - corona, 9
  - ersd_maps, 30
  - ga_geo, 32
  - georgia_census, 33
  - gwin_fulton_shape, 40
  - gwinnett, 39
  - gwinnett_ei, 40
  - lac_10, 41
  - ny_fips, 56
  - ny_voter, 56
  - ramapo2018, 66
  - rockland_census, 68

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

  add_split_comma, 4

  bayes_table_make, 4

  betas_for_return, 6

  check_args, 7

  concat_final_address, 7

  concat_streetname, 8

  cor_06, 10

  corona, 9

  dedupe_precincts, 10

  dedupe_voter_file, 11

  ei_compare-class, 12

  ei_est_gen, 13

  ei_good, 16

  ei_homog, 17

  ei_iter, 18

  ei_rc_congruence, 20

  ei_rc_good_table, 21, 38

  ei_reg_bayes_conf_int, 24

  ei_rxc, 26

  eiCompare (eiCompare-package), 3

  eiCompare-package, 3

  elect_algebra, 28

  empty_ei_df, 30

  ersd_maps, 30

  fips_extract, 31

  ga_geo, 32

  georgia_census, 33

  get_ei_iter_se, 34

  get_md_bayes_gen_output, 34

  get_multi_barreled_surnames, 35

  get_results_table, 35

  get_special_character_surnames, 36

  get_unique_special_characters, 37

  get_word_count, 37

  goodman_generalize, 38

  gwin_fulton_shape, 40

  gwinnett, 39

  gwinnett_ei, 40

  lac_10, 41

  lambda_two_compare, 42

  latlong2fips, 43

  map_interactive, 44

  map_shape_file, 45

  map_shape_points, 46

  mbd_two, 47

  mbd_two_minority, 47

  md_bayes_draw, 48

  md_bayes_draw_lambda, 49

  md_bayes_gen, 51

  md_bayes_table, 53

  mean_and_ci, 54

  merge_voter_file_to_shape, 55

  ny_fips, 56

  ny_voter, 56
od_plot_create, 57
overlay_density_plot, 57

performance_analysis, 58
plot.eiCompare, 60
plot_bivariate, 61
precinct_agg_combine, 61
predict_race_multi_barreled, 63

race_cand_cors, 65
race_check_2_3, 65
ramapo2018, 66
remove_nas, 67
resolve_missing_vals, 67
rockland_census, 68
rpv_density, 69
run_geocoder, 69
rcx_formula, 71

split_add_nocommas, 72
stdize_votes, 72
stdize_votes_all, 74
strip_special_characters, 75
sum_over_cols, 76
summary.eiCompare, 76
surname_match, 77
surname_summary, 77

tidy_voter_file_wru, 78

wru_predict_race_wrapper, 78

zip_hyphen, 80