Package 'wru'

May 7, 2026

Title Who are You? Bayesian Prediction of Racial Category Using Surname, First Name, Middle Name, and Geolocation

Version 3.1.0

Date 2026-04-30

Description Predicts individual race/ethnicity using surname, first name, middle name, geolocation, and other attributes, such as gender and age. The method utilizes Bayes' Rule (with optional measurement error correction) to compute the posterior probability of each racial category for any given individual. The package implements methods described in Imai and Khanna (2016) ``Improving Ecological Inference by Predicting Individual Ethnicity from Voter Registration Records'' Political Analysis <DOI:10.1093/pan/mpw001> and Imai, Olivella, and Rosenman (2022) ``Addressing census data problems in race imputation via fully Bayesian Improved Surname Geocoding and name supplements'' <DOI:10.1126/sciadv.adc9824>. The package also incorporates the data described in Rosenman, Olivella, and Imai (2023) ``Race and ethnicity data for first, middle, and surnames'' <DOI:10.1038/s41597-023-02202-2>.

License GPL (>= 3)

URL https://github.com/kosukeimai/wru

BugReports https://github.com/kosukeimai/wru/issues

Depends R (>= 4.1.0), utils

Imports cli, dplyr, furrr, piggyback (>= 0.1.4), PL94171, purrr, Rcpp, rlang

Suggests covr, future, reticulate, testthat (>= 3.0.0), tidycensus, tidyr

LinkingTo Rcpp, RcppArmadillo

Config/testthat/edition 3

Encoding UTF-8

LazyData yes

LazyDataCompression xz

LazyLoad yes

RoxygenNote 7.3.1

NeedsCompilation yes

Author Kabir Khanna [aut],
Brandon Bertelsen [aut, cre],
Santiago Olivella [aut],
Evan Rosenman [aut],
Alexander Rossell Hayes [aut],
Kosuke Imai [aut],
Noah Dasanaike [aut]

Maintainer Brandon Bertelsen <brandon@bertelsen.ca>

Repository CRAN

Date/Publication 2026-05-07 11:40:12 UTC

Contents

`ebisg` *Utilities for eBISG (embedding-supplemented BISG)*

Description

These functions support the eBISG model, which uses pre-trained text embeddings to predict race probabilities for surnames not found in Census surname lists.

format_legacy_data *Legacy data formatting function.*

Description

format_legacy_data formats legacy data from the U.S. census to allow for Bayesian name geocoding.

Usage

```
format_legacy_data(legacyFilePath, state, outFile = NULL)
```

Arguments

legacyFilePath	A character vector giving the location of a legacy census data folder, sourced from https://www2.census.gov/programs-surveys/decennial/2020/data/01-Redistricting_File–PL_94-171/. These file names should end in ".pl".
state	The two letter state postal code.
outFile	Optional character vector determining whether the formatted RData object should be saved. The filepath should end in ".RData".

Details

This function allows users to construct datasets for analysis using the census legacy data format. These data are available for the 2020 census at https://www2.census.gov/programs-surveys/decennial/2020/data/01-Redistricting_File–PL_94-171/. This function returns data structured analogously to data from the Census API, which is not yet available for the 2020 Census as of September 2021.

Examples

```
## Not run:
gaCensusData <- format_legacy_data(PL94171::pl_url('GA', 2020))
predict_race_new(ga.voter.file, namesToUse = 'last, first, mid', census.geo = 'block',
     census.data = gaCensusData)

## End(Not run)
```

get_census_data *Multilevel Census data download function.*

Description

get_census_data returns county-, tract-, and block-level Census data for specified state(s). Using this function to download Census data in advance can save considerable time when running predict_race and census_helper.

Usage

```
get_census_data(
  key = Sys.getenv("CENSUS_API_KEY"),
  states,
  age = FALSE,
  sex = FALSE,
  year = "2020",
  census.geo = c("tract", "block", "block_group", "county", "place", "zcta"),
  retry = 3,
  county.list = NULL
)
```

Arguments

key	A character string containing a valid Census API key, which can be requested from the U.S. Census API key signup page. By default, attempts to find a census key stored in an environment variable named CENSUS_API_KEY.
states	which states to extract Census data for, e.g., c("NJ", "NY").
age	A TRUE/FALSE object indicating whether to condition on age or not. If FALSE (default), function will return Pr(Geolocation | Race). If TRUE, function will return Pr(Geolocation, Age | Race). If *sex* is also TRUE, function will return Pr(Geolocation, Age, Sex | Race).
sex	A TRUE/FALSE object indicating whether to condition on sex or not. If FALSE (default), function will return Pr(Geolocation | Race). If TRUE, function will return Pr(Geolocation, Sex | Race). If *age* is also TRUE, function will return Pr(Geolocation, Age, Sex | Race).
year	A character object specifying the year of U.S. Census data to be downloaded. Use "2010", or "2020". Default is "2020". Warning: 2020 U.S. Census data is downloaded only when *age* and *sex* are both FALSE.
census.geo	An optional character vector specifying what level of geography to use to merge in U.S. Census 2010 geographic data. Currently "county", "tract", "block", and "place" are supported.
retry	The number of retries at the census website if network interruption occurs.
county.list	A named list of character vectors of counties present in your *voter.file*, per state.

Value

Output will be an object of class `list` indexed by state. Output will contain a subset of the following elements: `state`, `age`, `sex`, `county`, `tract`, `block_group`, `block`, and `place`.

Examples

```
## Not run: get_census_data(states = c("NJ", "NY"), age = TRUE, sex = FALSE)
## Not run: get_census_data(states = "MN", age = FALSE, sex = FALSE, year = "2020")
```

`predict_race` *Race prediction function.*

Description

`predict_race` makes probabilistic estimates of individual-level race/ethnicity.

Usage

```
predict_race(
  voter.file,
  census.surname = TRUE,
  surname.only = FALSE,
  census.geo = c("tract", "block", "block_group", "county", "place", "zcta"),
  census.key = Sys.getenv("CENSUS_API_KEY"),
  census.data = NULL,
  age = FALSE,
  sex = FALSE,
  year = "2020",
  party = NULL,
  retry = 3,
  impute.missing = TRUE,
  skip_bad_geos = FALSE,
  use.counties = FALSE,
  model = "BISG",
  race.init = NULL,
  name.dictionaries = NULL,
  names.to.use = "surname",
  control = NULL,
  ebisg.model = "intfloat/multilingual-e5-large"
)
```

Arguments

`voter.file`	An object of class `data.frame`. Must contain a row for each individual being predicted, as well as a field named *surname* containing each individual's surname. If using geolocation in predictions, *voter.file* must contain a field named *state*, which contains the two-character abbreviation for each individual's state

of residence (e.g., "nj" for New Jersey). If using Census geographic data in race/ethnicity predictions, *voter.file* must also contain at least one of the following fields: *county*, *tract*, *block_group*, *block*, and/or *place*. These fields should contain character strings matching U.S. Census categories. County is three characters (e.g., "031" not "31"), tract is six characters, block group is usually a single character and block is four characters. Place is five characters. See below for other optional fields.

`census.surname` A TRUE/FALSE object. If TRUE, function will call `merge_surnames` to merge in Pr(Race | Surname) from U.S. Census Surname List (2000, 2010, or 2020) and Spanish Surname List. If FALSE, user must provide a `name.dictionary` (see below). Default is TRUE.

`surname.only` A TRUE/FALSE object. If TRUE, race predictions will only use surname data and calculate Pr(Race | Surname). Default is FALSE.

`census.geo` An optional character vector specifying what level of geography to use to merge in U.S. Census geographic data. Currently "county", "tract", "block_group", "block", and "place" are supported. Note: sufficient information must be in user-defined *voter.file* object. If *census.geo* = "county", then *voter.file* must have column named county. If *census.geo* = "tract", then *voter.file* must have columns named county and tract. And if *census.geo* = "block", then *voter.file* must have columns named county, tract, and block. If *census.geo* = "place", then *voter.file* must have column named place. If census.geo = "zcta", then voter.file must have column named zcta. Specifying *census.geo* will call census_helper function to merge Census geographic data at specified level of geography.

`census.key` A character object specifying user's Census API key. Required if census.geo is specified, because a valid Census API key is required to download Census geographic data.

If NULL, the default, attempts to find a census key stored in an environment variable named CENSUS_API_KEY.

`census.data` A list indexed by two-letter state abbreviations, which contains pre-saved Census geographic data. Can be generated using get_census_data function.

`age` An optional TRUE/FALSE object specifying whether to condition race predictions on age (in addition to surname and geolocation). Default is FALSE. Must be same as *age* in *census.data* object. May only be set to TRUE if census.geo option is specified. If TRUE, *voter.file* should include a numerical variable *age*.

`sex` optional TRUE/FALSE object specifying whether to condition race predictions on sex (in addition to surname and geolocation). Default is FALSE. Must be same as *sex* in *census.data* object. May only be set to TRUE if census.geo option is specified. If TRUE, *voter.file* should include a numerical variable *sex*, where *sex* is coded as 0 for males and 1 for females.

`year` An optional character vector specifying the year of U.S. Census geographic data to be downloaded. Use "2010", or "2020". Default is "2020".

`party` An optional character object specifying party registration field in *voter.file*, e.g., *party* = "PartyReg". If specified, race/ethnicity predictions will be conditioned on individual's party registration (in addition to geolocation). Whatever the name of the party registration field in *voter.file*, it should be coded as 1 for Democrat, 2 for Republican, and 0 for Other.

`retry` The number of retries at the census website if network interruption occurs.

`impute.missing` Logical, defaults to TRUE. Should missing be imputed?

`skip_bad_geos` Logical. Option to have the function skip any geolocations that are not present in the census data, returning a partial data set. Default is set to `FALSE`, in which case it will break and provide error message with a list of offending geolocations.

`use.counties` A logical, defaulting to FALSE. Should census data be filtered by counties available in *census.data*?

`model` Character string: "BISG" (default), "fBISG" (fully-Bayesian with error correction), or "eBISG" (embedding-supplemented BISG, which uses text embeddings to predict race probabilities for names not found in Census surname lists). The eBISG model requires Python with sentence-transformers and torch; run `setup_ebisg` to configure.

`race.init` Vector of initial race for each observation in voter.file. Must be an integer vector, with 1=white, 2=black, 3=hispanic, 4=asian, and 5=other. Defaults to values obtained using `model="BISG_surname"`.

`name.dictionaries`
Optional named list of `data.frame`'s containing counts of names by race. Any of the following named elements are allowed: "surname", "first", "middle". When present, the objects must follow the same structure as `last_c`, `first_c`, `mid_c`, respectively.

`names.to.use` One of 'surname', 'surname, first', or 'surname, first, middle'. Defaults to 'surname'.

`control` List of control arguments only used when `model="fBISG"`, including

iter Number of MCMC iterations. Defaults to 1000.

burnin Number of iterations discarded as burnin. Defaults to half of `iter`.

verbose Print progress information. Defaults to `TRUE`.

me.correct Boolean. Should the model correct measurement error for `races|geo`? Defaults to `TRUE`.

seed RNG seed. If `NULL`, a seed is generated and returned as an attribute for reproducibility.

`ebisg.model` Character string (HuggingFace model ID) or named list specifying which embedding model to use when `model = "eBISG"`. The only built-in option is `"intfloat/multilingual-e5-` (default; 1024-dim), which is keyed by its full HuggingFace ID for consistency with the Python side. To use a different sentence-transformer, pass a named list with elements `transformer` (HuggingFace model ID), `dim` (embedding dimension), and `surname_mlp`, `firstname_mlp` (paths to custom .pt checkpoints).

Details

This function implements the Bayesian race prediction methods outlined in Imai and Khanna (2015). The function produces probabilistic estimates of individual-level race/ethnicity, based on surname, geolocation, and party.

Value

Output will be an object of class `data.frame`. It will consist of the original user-input `voter.file` with additional columns with predicted probabilities for each of the five major racial categories: *pred.whi* for White, *pred.bla* for Black, *pred.his* for Hispanic/Latino, *pred.asi* for Asian/Pacific Islander, and *pred.oth* for Other/Mixed.

Examples

```
#' data(voters)
try(predict_race(voter.file = voters, surname.only = TRUE))
## Not run:
try(predict_race(voter.file = voters, census.geo = "tract"))

## End(Not run)
## Not run:
try(predict_race(
  voter.file = voters, census.geo = "place", year = "2020"))

## End(Not run)
## Not run:
CensusObj <- try(get_census_data(state = c("NY", "DC", "NJ")))
try(predict_race(
  voter.file = voters, census.geo = "tract", census.data = CensusObj, party = "PID")
  )

## End(Not run)
## Not run:
CensusObj2 <- try(get_census_data(state = c("NY", "DC", "NJ"), age = T, sex = T))
try(predict_race(
  voter.file = voters, census.geo = "tract", census.data = CensusObj2, age = T, sex = T))

## End(Not run)
## Not run:
CensusObj3 <- try(get_census_data(state = c("NY", "DC", "NJ"), census.geo = "place"))
try(predict_race(voter.file = voters, census.geo = "place", census.data = CensusObj3))

## End(Not run)
```

`setup_ebisg` *Set up Python environment for eBISG predictions*

Description

Installs the required Python packages (sentence-transformers and torch) and downloads pre-trained MLP model weights. Call this once before using `predict_race(..., model = "eBISG")`.

Usage

```
setup_ebisg(
  envname = "r-ebisg",
  ebisg.model = "intfloat/multilingual-e5-large"
)
```

Arguments

envname	Name of the Python virtual environment to use. Defaults to "r-ebisg".
ebisg.model	Which embedding model to set up. Defaults to "intfloat/multilingual-e5-large".

---

state_fips *Dataset with FIPS codes for US states*

Description

Dataset including FIPS codes and postal abbreviations for each U.S. state, district, and territory.

Usage

```
state_fips
```

Format

A tibble with 57 rows and 3 columns:

`state` Two-letter postal abbreviation

`state_code` Two-digit FIPS code

`state_name` English name

Source

Derived from `tidycensus::fips_codes()`

surnames2000 *Census Surname List (2000).*

Description

Census Surname List from 2000 with race/ethnicity probabilities by surname.

Usage

```
surnames2000
```

Format

A data frame with 157,728 rows and 6 variables:

surname Surname
p_whi Pr(White | Surname)
p_bla Pr(Black | Surname)
p_his Pr(Hispanic/Latino | Surname)
p_asi Pr(Asian/Pacific Islander | Surname)
p_oth Pr(Other | Surname) #'

Examples

```
data(surnames2000)
```

surnames2010 *Census Surname List (2010).*

Description

Census Surname List from 2010 with race/ethnicity probabilities by surname.

Usage

```
surnames2010
```

Format

A data frame with 167,613 rows and 6 variables:

surname Surname
p_whi Pr(White | Surname)
p_bla Pr(Black | Surname)
p_his Pr(Hispanic/Latino | Surname)
p_asi Pr(Asian/Pacific Islander | Surname)
p_oth Pr(Other | Surname) #'

Examples

```
data(surnames)
```

`voters` *Example voter file.*

Description

An example dataset containing voter file information.

Usage

```
voters
```

Format

A data frame with 10 rows and 12 variables:

VoterID Voter identifier (numeric)

surname Surname

state State of residence

CD Congressional district

county Census county (three-digit code)

first First name

last Last name or surname

tract Census tract (six-digit code)

block Census block (four-digit code)

precinct Voting precinct

place Voting place

age Age in years

sex 0=male, 1=female

party Party registration (character)

PID Party registration (numeric) #'

Examples

```
data(voters)
```

`wru_data_preflight` *Preflight for name data*

Description

Checks if namedata is available in the current working directory, if not downloads it from github using piggyback. By default, wru will download the data to a temporary directory that lasts as long as your session does. However, you may wish to set the `wru_data_wd` option to save the downloaded data to your current working directory for more permanence.

Usage

```
wru_data_preflight()
```

Index