Package ‘birdie’

June 11, 2024

Title Bayesian Instrumental Regression for Disparity Estimation

Version 0.6.1

Description Bayesian models for accurately estimating conditional distributions by race, using Bayesian Improved Surname Geocoding (BISG) probability estimates of individual race. Implements the methods described in McCartan, Fisher, Goldin, Ho and Imai (2024) <doi:10.3386/w32373>.

Depends R (>= 3.5.0)

Imports rlang (>= 0.1.2), Rcpp (>= 0.12.0), cli, vctrs, generics, dplyr, methods, stringi, stringr, RcppParallel (>= 5.0.1), SQUAREM

Suggests daarem, easycensus, wru, knitr, roxygen2, rmarkdown, testthat (>= 3.0.0)

LinkingTo Rcpp (>= 0.12.0), cli, BH (>= 1.66.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), RcppThread, StanHeaders (>= 2.18.0)

License GPL (>= 3)

Encoding UTF-8

VignetteBuilder knitr

Config/testthat/edition 3

SystemRequirements GNU make, C++17

Biarch true

LazyData true

URL `https://github.com/CoryMcCartan/birdie`,
`https://corymccartan.com/birdie/`

BugReports `https://github.com/CoryMcCartan/birdie/issues`

RoxygenNote 7.3.1

NeedsCompilation yes

Author Cory McCartan [aut, cre],
Kosuke Imai [ctb],
Daniel Ho [ctb],
Jacob Goldin [ctb],
Robin Fisher [ctb],
The Stan Development Team [cph] (include/rstan)

Maintainer Cory McCartan <mccartan@psu.edu>

Repository CRAN

Date/Publication 2024-06-11 12:10:08 UTC

Contents

---

birdie *Fit BIRDiE Models*

Description

Fits one of three possible Bayesian Instrumental Regression for Disparity Estimation (BIRDiE) models to BISG probabilities and covariates. The simplest Categorical-Dirichlet model ([cat_dir()]) is appropriate when there are no covariates or when all covariates are discrete and fully interacted with another. The more general Categorical mixed-effects model ([cat_mixed()]) is a supports any number of fixed effects and up to one random intercept. For continuous outcomes a Normal linear model is available ([gaussian()]).

Usage

```
birdie(
  r_probs,
  formula,
  data,
  family = cat_dir(),
  prior = NULL,
  weights = NULL,
  algorithm = c("em", "gibbs", "em_boot"),
  iter = 400,
  warmup = 50,
  prefix = "pr_",
  ctrl = birdie.ctrl()
)
```

Arguments

`r_probs` A data frame or matrix of BISG probabilities, with one row per individual. The output of [bisg()] can be used directly here.

`formula` A two-sided formula object describing the model structure. The left-hand side is the outcome variable, which must be discrete. A single random intercept term, denoted with a vertical bar (‘"(1 | <term>)"‘), is supported on the right-hand side.

`data` An optional data frame containing the variables named in ‘formula‘.

`family` A description of the complete-data model type to fit. Options are:

- [cat_dir()]: Categorical-Dirichlet model. All covariates must be fully interacted. - [cat_mixed()]: Categorical mixed-effects model. Up to one random effect is supported. - [gaussian()]: Linear model.

See the Details section below for more information on the various models.

`prior` A list with entries specifying the model prior.

- For the ‘cat_dir‘ model, the only entry is ‘alpha‘, which should be a matrix of Dirichlet hyperparameters. The matrix should have one row for every level of the outcome variable and one column for every racial group. The default prior (used when ‘prior=NULL‘) is an empirical Bayes prior equal to the weighted-mean estimate of the outcome-race table. A fully noninformative prior with all entries set to ϵ can be obtained by setting ‘prior=NA‘. When ‘prior=NULL‘ and ‘algorithm="em"‘ or ‘"em_boot"‘, 1 is added to the prior so that the posterior mode, rather than the mean, is shrunk toward these values. - For the ‘cat_mixed‘ model, the ‘prior‘ list should contain three scalar entries: ‘scale_int‘, the standard deviation on the Normal prior for the intercepts (which control the global estimates of ‘Y|R‘), ‘scale_beta‘, the standard deviation on the Normal prior for the fixed effects, and ‘scale_sigma‘, the prior mean of the standard deviation of the random intercepts. These can be a single scalar or a vector with an entry for each racial group. - For the ‘gaussian‘ model, the ‘prior‘ list should contain two entries: ‘scale_int‘, controlling, the standard deviation on the Normal prior for the intercepts (which control the global estimates of ‘Y|R‘), and ‘scale_beta‘, controlling the standard deviation on the Normal prior for the fixed effects. These must be a single scalar. Each is expressed in terms of the estimated residual standard deviation (i.e., they are multiplied together to form the "true" prior).

The prior is stored after model fitting in the ‘$prior‘ element of the fitted model object.

`weights` An optional numeric vector specifying likelihood weights.

algorithm The inference algorithm to use. One of 3 options:

- ‘"em"‘: An expectation-maximization algorithm which will perform inference for the maximum a posteriori (MAP) parameter values. Computationally efficient and supported by all the model families. No uncertainty quantification. - ‘"gibbs"‘: A Gibbs sampler for performing full Bayesian inference. Generally more computationally demanding than the EM algorithm, but provides uncertainty quantification. Currently supported for ‘cat_dir()‘ and ‘gaussian()‘ model families. Computation-reliability tradeoff can be controlled with ‘iter‘ argument. - ‘"em_boot"‘: Bootstrapped version of EM algorithm. Number of bootstrap replicates controlled by ‘iter‘ parameter. Provides approximate uncertainty quantification. Currently supported for ‘cat_dir()‘ and ‘gaussian()‘ model families.

iter The number of post-warmup Gibbs samples, or the number of bootstrap replicates to use to compute approximate standard errors for the main model estimates. Only available when ‘family=cat_dir()‘ or ‘gaussian()‘. Ignored if ‘algorithm="em"‘.

For bootstrapping, when there are fewer than 1,000 individuals or 100 or fewer replicates, a Bayesian bootstrap is used instead (i.e., weights are drawn from a Dirichlet$(1, 1, ..., 1)$ distribution, which produces more reliable estimates.

warmup Number of warmup iterations for Gibbs sampling. Ignored unless ‘algorithm="gibbs"‘.

prefix If ‘r_probs‘ is a data frame, the columns containing racial probabilities will be selected as those with names starting with ‘prefix‘. The default will work with the output of [bisg()].

ctrl A list containing control parameters for the EM algorithm and optimization routines. A list in the proper format can be made using [birdie.ctrl()].

Details

By default, ‘birdie()‘ uses an expectation-maximization (EM) routine to find the maximum *a posteriori* (MAP) estimate for the specified model. Asymptotic variance-covariance matrices for the MAP estimate are available for the Categorical-Dirichlet and Normal linear models via bootstrapping. Full Bayesian inference is supported via Gibbs sampling for the Categorical-Dirichlet and Normal linear models as well.

Whatever model or method is used, a finite-population estimate of the outcome-given-race distribution for the entire observed sample is always calculated and stored as ‘$est‘ in the returned object, which can be accessed with [coef.birdie()] as well.

The Categorical-Dirichlet model is specified as follows:

$$Y_i \mid R_i, X_i, \Theta \sim \text{Categorical}(\theta_{R_i X_i}) \theta_{rx} \sim \text{Dirichlet}(\alpha_r),$$

where Y is the outcome variable, R is race, X are covariates (fixed effects), and θ_{rx} and α_r are vectors with length matching the number of levels of the outcome variable. There is one vector θ_{rx} for every combination of race and covariates, hence the need for ‘formula‘ to either have no covariates or a fully interacted structure.

The Categorical mixed-effects model is specified as follows:

$$Y_i \mid R_i, X_i, \Theta \sim \text{Categorical}(g^{-1}(\mu_{R_i X_i}))\mu_{rxy} = W\beta_{ry} + Zu_{ry}u_r \mid \vec{\sigma}_r, L_r \sim \mathcal{N}(0, \text{diag}(\vec{\sigma}_r)C_r\text{diag}(\vec{\sigma}_r))\beta_{ry} \sim \mathcal{N}(0, s^2_{r\beta})$$

where β_{ry} are the fixed effects, u_{ry} is the random intercept, and g is a softmax link function. Estimates for β_{ry} and σ_{ry} are stored in the ‘$beta‘ and ‘$sigma‘ elements of the fitted model object.

The Normal linear model is specified as follows:

$$Y_i \mid R_i, \vec{X}_i, \Theta \sim \mathcal{N}(\vec{X}_i^\top \vec{\theta}, \sigma^2)\sigma^2 \sim \text{Inv-Gamma}(n_\sigma/2, l_\sigma^2 n_\sigma/2)\beta_{\text{intercept}} \sim \mathcal{N}(0, s_{\text{int}}^2)\beta_k \sim \mathcal{N}(0, s_\beta^2),$$

where $\vec{\theta}$ is a vector of linear model coefficients. Estimates for θ and σ are stored in the ‘$beta‘ and ‘$sigma‘ elements of the fitted model object.

More details on the models and their properties may be found in the paper referenced below.

Value

An object of class [‘birdie‘][birdie::birdie-class], for which many methods are available. The model estimates may be accessed with [coef.birdie()], and updated BISG probabilities (conditioning on the outcome) may be accessed with [fitted.birdie()]. Uncertainty estimates, if available, can be accessed with ‘$se‘ and [vcov.birdie()].

References

McCartan, C., Fisher, R., Goldin, J., Ho, D.E., & Imai, K. (2024). Estimating Racial Disparities when Race is Not Observed. Available at https://www.nber.org/papers/w32373.

Examples

```
data(pseudo_vf)

r_probs = bisg(~ nm(last_name) + zip(zip), data=pseudo_vf)

# Process zip codes to remove missing values
pseudo_vf$zip = proc_zip(pseudo_vf$zip)

fit = birdie(r_probs, turnout ~ 1, data=pseudo_vf)
print(fit)
fit$se # uncertainty quantification

fit = birdie(r_probs, turnout ~ zip, data=pseudo_vf, algorithm="gibbs")

fit = birdie(r_probs, turnout ~ (1 | zip), data=pseudo_vf,
             family=cat_mixed(), ctrl=birdie.ctrl(abstol=1e-3))

summary(fit)
coef(fit)
fitted(fit)
```

birdie-class *Class "birdie" of BIRDiE Models*

Description

The output of [birdie()] is an object of class ‘birdie‘, which supports many generic functions. Notably ‘coef.birdie()‘ returns the main model estimates of outcome given race, and ‘fitted.birdie()‘ returns a table analogous to the output of [bisg()] with updated race probabilities.

Usage

```
## S3 method for class 'birdie'
coef(object, subgroup = FALSE, ...)

## S3 method for class 'birdie'
fitted(object, ...)

## S3 method for class 'birdie'
residuals(object, x_only = FALSE, ...)

## S3 method for class 'birdie'
predict(object, adj = NULL, ...)

## S3 method for class 'birdie'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'birdie'
plot(x, log = FALSE, ...)

## S3 method for class 'birdie'
tidy(x, subgroup = FALSE, ...)

## S3 method for class 'birdie'
glance(x, ...)

## S3 method for class 'birdie'
augment(x, data, ...)

## S3 method for class 'birdie'
formula(x, ...)

## S3 method for class 'birdie'
family(object, ...)

## S3 method for class 'birdie'
nobs(object, ...)
```

```
## S3 method for class 'birdie'
vcov(object, ...)

## S3 method for class 'birdie'
print(x, ...)

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

Arguments

`object, x` A ‘birdie‘ model object

`subgroup` If ‘TRUE‘, return subgroup-level (rather than marginal) coefficient estimates as a 3D array.

`...` Potentially further arguments passed from other methods

`x_only` if ‘TRUE‘, calculate fitted values using covariates only (i.e., without using surnames).

`adj` A point in the simplex that describes how BISG probabilities will be thresholded to produce point predictions. The probabilities are divided by ‘adj‘, then the racial category with the highest probability is predicted. Can be used to trade off types of prediction error. Must be nonnegative but will be normalized to sum to 1. The default is to make no adjustment.

`nsim` The number of vectors to simulate. Defaults to 1.

`seed` Used to seed the random number generator. See [stats::simulate()].

`log` If ‘TRUE‘, plot estimated probabilities on a log scale.

`data` A data frame to augment with ‘Pr(R | Y, X, S)‘ probabilities

Details

The internal structure of ‘birdie‘ objects is not designed to be accessed directly. The generics listed here should be used instead.

Value

Varies, depending on the method. See generic functions’ documentation for details.

Functions

- `coef(birdie)`: Return estimated outcome-given-race distributions. When ‘subgroup=FALSE‘ this always returns a finite-population estimate of the outcome-given-race distribution for the observed sample.
- `fitted(birdie)`: Return an updated race probability table. [bisg()] estimates ‘Pr(R | G, X, S)‘; this table is ‘Pr(R | Y, G, X, S, Theta-hat)‘.
- `residuals(birdie)`: Return the residuals for the outcome variable as a matrix. Useful in sensitivity analyses and to get an idea of how well race, location, names, etc. predict the outcome.

- predict(birdie): Create point predictions of individual race. Returns factor vector of individual race labels. Strongly not recommended for any kind of inferential purpose, as biases may be extreme and in unpredictable directions.
- simulate(birdie): Simulate race from the posterior distribution ‘Pr(R | Y, G, X, S, Theta-hat)‘. Does not account for uncertainty in model parameters.
- plot(birdie): Visualize the estimated conditional distributions for a BIRDiE model. If available, marginal standard error estimates (‘$se‘) will be visualized with 95
- tidy(birdie): Put BIRDiE model coefficients in a tidy format.
- glance(birdie): Glance at a BIRDiE model.
- augment(birdie): Augment data with individual race predictions from a BIRDiE model.
- formula(birdie): Extract the formula used to specify a BIRDiE model.
- family(birdie): Return the BIRDiE complete-data model family.
- nobs(birdie): Return the number of observations used to fit a BIRDiE model.
- vcov(birdie): Return the estimated variance-covariance matrix for the BIRDiE model estimates, if available.
- print(birdie): Print a summary of the model fit.
- summary(birdie): Print a more detailed summary of the model fit.

Examples

```
methods(class="birdie")
```

birdie-family *BIRDiE Complete-Data Model Families*

Description

BIRDiE supports a number of complete-data outcome models, including categorical regression models. Models specific to BIRDiE are listed here. See the Details section of [birdie()] for more information about each model.

Usage

```
cat_dir(link = "identity")

cat_mixed(link = "softmax")
```

Arguments

link	The link function. Only one option available for categorical regression models.

Value

A list of class ‘family‘ containing the specification.

Examples

```
cat_dir()
cat_mixed()
```

`birdie.ctrl` *Control of BIRDiE Model Fitting*

Description

Constructs control parameters for BIRDiE model fitting. All arguments have defaults.

Usage

```
birdie.ctrl(
  abstol = 1e-06,
  reltol = 1e-06,
  max_iter = 1000,
  fix_sigma = FALSE,
  accel = c("squarem", "anderson", "daarem", "none"),
 order = switch(match.arg(accel), none = 0L, anderson = -1L, daarem = -1L, squarem = 1L),
  anderson_restart = TRUE
)
```

Arguments

`abstol` The absolute tolerance used in checking convergence or in estimating linear model coefficients.

`reltol` The relative tolerance used in checking convergence. Ignored if ‘accel = "squarem"‘ or ‘"daarem"‘.

`max_iter` The maximum number of EM iterations.

`fix_sigma` If ‘TRUE‘ when ‘model=gaussian()‘, fix sigma to an initial estimate, in order to avoid estimation collapse when the outcomes are discrete.

`accel` The acceleration algorithm to use in doing EM. The default ‘"squarem"‘ is good for most purposes, though ‘"anderson"‘ may be faster when there are few parameters or very tight tolerances. ‘"daarem"‘ is an excellent choice as well that works across a range of problems, though it requires installing the small ‘daarem‘ package. ‘"none"‘ is not recommended unless other algorithms are running into numerical issues. See the references below for details on these schemes.

`order` The order to use in the acceleration algorithm. Interpretation varies by algorithm. Can range from 1 (default) to 3 for SQUAREM and from 1 to the number of parameters for Anderson and DAAREM (default -1 allows the order to be determined by problem size).

`anderson_restart` Whether to use restarts in Anderson acceleration.

Value

A list containing the control parameters.

References

Varadhan, R., & Roland, C. (2004). Squared extrapolation methods (SQUAREM): A new class of simple and efficient numerical schemes for accelerating the convergence of the EM algorithm.

Walker, H. F., & Ni, P. (2011). Anderson acceleration for fixed-point iterations. SIAM Journal on Numerical Analysis, 49(4), 1715-1735.

Henderson, N. C., & Varadhan, R. (2019). Damped Anderson acceleration with restarts and monotonicity control for accelerating EM and EM-like algorithms. Journal of Computational and Graphical Statistics, 28(4), 834-846.

Examples

```
str(birdie.ctrl(max_iter=100))
```

---

bisg *Bayesian Improved Surname Geocoding (BISG)*

Description

Calculates individual probabilities of belonging to racial groups given last name, location, and other covariates (optional). The standard function ‘bisg()‘ treats the input tables as fixed. An alternative function ‘bisg_me()‘, assumes that the input tables are subject to measurement error, and uses a Gibbs sampler to impute the individual race probabilities, using the model of Imai et al. (2022).

Usage

```
bisg(
  formula,
  data = NULL,
  p_r = p_r_natl(),
  p_rgx = NULL,
  p_rs = NULL,
  save_rgx = TRUE
)

bisg_me(
  formula,
  data = NULL,
  p_r = p_r_natl(),
  p_rgx = NULL,
  p_rs = NULL,
  iter = 1000,
  warmup = 100,
  cores = 1L
)

## S3 method for class 'bisg'
summary(object, p_r = NULL, ...)

## S3 method for class 'bisg'
predict(object, adj = NULL, ...)

## S3 method for class 'bisg'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

`formula` A formula specifying the BISG model. Must include the special term ‘nm()‘ to identify the surname variable. Certain geographic variables can be identified similarly: ‘zip()‘ for ZIP codes, and ‘state()‘ for states. If no other predictor variables are provided, then ‘bisg()‘ will automatically be able to build a table of census data to use in inference. If other predictor variables are included, or if other geographic identifiers are used, then the user must specify the ‘p_rgx‘ argument below. The left-hand side of the formula is ignored. See the examples section below for sample formulas.

`data` The data frame containing the variables in ‘formula‘.

`p_r` The prior distribution of race in the sample, as a numeric vector. Defaults to U.S. demographics as provided by [p_r_natl()]. Can also set ‘p_r="est"‘ or ‘"estimate"‘ to estimate this from the geographic distribution. Since the prior distribution on race strongly affects the calibration of the BISG probabilities and thus the accuracy of downstream estimates, users are encouraged to think carefully about an appropriate value for ‘p_r‘. If no prior information on the racial makeup of the sample is available, and yet the sample is very different from the overall U.S. population, then ‘p_r="estimate"‘ will likely produce superior results.

`p_rgx` The distribution of race given location (G) and other covariates (X) specified in ‘formula‘. Should be provided as a data frame, with columns matching the predictors in ‘formula‘, and additional columns for each racial group containing the conditional probability for that racial group given the predictors. For example, if Census tracts are the only predictors, ‘p_rgx‘ should be a data frame with a tract column and columns ‘white‘, ‘black‘, etc. containing the racial distribution of each tract. If ‘formula‘ contains only labeled terms (like ‘zip()‘), then by default ‘p_rgx‘ will be constructed automatically from the most recent Census data. This table will be normalized by row, so it can be provided as population counts as well. Counts are required for ‘bisg_me()‘. The [census_race_geo_table()] function can be helpful to prepare tables, as can be the ‘build_dec()‘ and ‘build_acs()‘ functions in the ‘censable‘ package.

`p_rs` The distribution of race given last name. As with ‘p_rgx‘, should be provided as a data frame, with a column of names and additional columns for each racial group. Users should not have to specify this argument in most cases, as the table will be built from published Census surname tables automatically. Counts are required for ‘bisg_me()‘.

`save_rgx` If ‘TRUE‘, save the ‘p_rgx‘ table (matched to each individual) as the ‘"p_rgx"‘ and ‘"gx"‘ attributes of the output. Necessary for some sensitivity analyses.

`iter` How many sampling iterations in the Gibbs sampler

`warmup` How many burn-in iterations in the Gibbs sampler

`cores` How many parallel cores to use in computation. Around 4 seems to be optimal, even if more are available.

`object` An object of class ‘bisg‘, the result of running [bisg()].

`...` Additional arguments to generic methods (ignored).

`adj` A point in the simplex that describes how BISG probabilities will be thresholded to produce point predictions. The probabilities are divided by ‘adj‘, then the racial category with the highest probability is predicted. Can be used to trade off types of prediction error. Must be nonnegative but will be normalized to sum to 1. The default is to make no adjustment.

`nsim` The number of vectors to simulate. Defaults to 1.

`seed` Used to seed the random number generator. See [stats::simulate()].

Value

An object of class ‘bisg‘, which is just a data frame with some additional attributes. The data frame has rows matching the input data and columns for the race probabilities.

Methods (by generic)

- `summary(bisg)`: Summarize predicted race probabilities. Returns vector of individual entropies.
- `predict(bisg)`: Create point predictions of individual race. Returns factor vector of individual race labels. Strongly not recommended for any kind of inferential purpose, as biases may be extreme and in unpredictable directions.
- `simulate(bisg)`: Simulate race from the ‘Pr(R | G, X, S)‘ distribution.

Functions

- `bisg()`: The standard BISG model.
- `bisg_me()`: The measurement error BISG model.

References

Elliott, M. N., Fremont, A., Morrison, P. A., Pantoja, P., and Lurie, N. (2008). A new method for estimating race/ethnicity and associated disparities where administrative records lack self-reported race/ethnicity. *Health Services Research*, 43(5p1):1722–1736.

Fiscella, K. and Fremont, A. M. (2006). Use of geocoding and surname analysis to estimate race and ethnicity. *Health Services Research*, 41(4p1):1482–1500.

Imai, K., Olivella, S., & Rosenman, E. T. (2022). Addressing census data problems in race imputation via fully Bayesian Improved Surname Geocoding and name supplements. *Science Advances*, 8(49), eadc9824.

Examples

```
data(pseudo_vf)
bisg(~ nm(last_name), data=pseudo_vf)

r_probs = bisg(~ nm(last_name) + zip(zip), data=pseudo_vf)
summary(r_probs)
head(predict(r_probs))

data(pseudo_vf)
bisg_me(~ nm(last_name) + zip(zip), data=pseudo_vf)
```

`census_race_geo_table` *Download Census Race Data*

Description

Downloads and prepares race-by-geography tables from U.S. census data, using the [‘easycensus‘][easycensus::easycensus] package. Requires that an api key be set up through [easycensus::cens_auth()] in that package, usually by storing it in the ‘CENSUS_API_KEY‘ environment variable. Supports data from the decennial census and the American Community Survey at a variety of levels of geographic detail. The output of this function can be used directly in [bisg()].

Usage

```
census_race_geo_table(
  geo = c("us", "state", "county", "zcta", "tract"),
  ...,
  year = 2010,
  survey = c("dec", "acs1", "acs5"),
  GEOIDs = TRUE,
  counts = TRUE
)
```

Arguments

`geo` The geographic level to return. Common options are listed in the function signature, but any of the geographies listed at [easycensus::cens_geo()] may be used.

`...` Further subgeographies to return, as in [easycensus::cens_geo()].

`year` The year for the data

`survey` The data product to use: either the decennial census (‘"dec"‘), or the the 1-year or 5-year ACS.

`GEOIDs` If ‘TRUE‘, return the ‘GEOID‘ column as the unique geographic identifier; if ‘FALSE‘, return a human-readable name. For example, with ‘geo="state"‘, setting ‘GEOIDs=FALSE‘ will return a column named ‘state‘ with entries like ‘"Massachusetts"‘.

`counts` If ‘TRUE‘, return the table as actual population counts; if ‘FALSE‘, return table as percentages within each geography.

Value

A data frame with geographic identifier column(s) and six columns ‘white‘, ‘black‘, etc. containing the counts or proportion of residents in each racial group.

Examples

```
census_race_geo_table("zcta", year=2010)
## Not run:
# Census API key required
census_race_geo_table("us", year=2010)
census_race_geo_table("state", year=2021, survey="acs1")
census_race_geo_table("state", year=2021, survey="acs1", GEOIDs=FALSE)
## End(Not run)
```

---

disparities *Compute Racial Disparities from Model Estimates*

Description

This function lets you easily compute differences in conditional expectations between all pairs of specified racial groups.

Usage

```
disparities(x, subgroup = FALSE, races = TRUE)
```

Arguments

x	A ‘birdie‘ model object.
subgroup	If ‘TRUE‘, return subgroup-level (rather than marginal) disparity estimates.
races	A character vector of racial groups to compute disparities for. The special value ‘TRUE‘, the default, computes disparities for all racial groups.

Value

A data frame containing a row with every possible disparity for the specified ‘races‘, which are identified by columns ‘race_1‘ and ‘race_2‘. The reported disparity is ‘estimate_1 - estimate_2‘.

Examples

```
data(pseudo_vf)
r_probs = bisg(~ nm(last_name) + zip(zip), data=pseudo_vf)
fit = birdie(r_probs, turnout ~ 1, data=pseudo_vf)

disparities(fit)
disparities(fit, races=c("white", "black"))
```

est_weighted *Calculate Weighted Estimate of (Discrete) Outcomes By Race*

Description

Calculates the "standard" weighted estimator of conditional distributions of an outcome variable Y by race R, using BISG probabilities. This estimator, while commonly used, is only appropriate if $Y \perp R \mid X, S$, where S and X are the last names and covariates (possibly including geography) used in making the BISG probabilities. In most cases this assumption is not plausible and [birdie()] should be used instead. See the references below for more discussion as to selecting the right estimator.

Up to Monte Carlo error, the weighted estimate is equivalent to performing multiple imputations of the race vector from the BISG probabilities and then using them inside a weighted average or linear regression.

Usage

```
est_weighted(
  r_probs,
  formula,
  data = NULL,
  weights = NULL,
  prefix = "pr_",
  se_boot = 0
)

## S3 method for class 'est_weighted'
print(x, ...)

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

Arguments

`r_probs` A data frame or matrix of BISG probabilities, with one row per individual. The output of [bisg()] can be used directly here.

`formula` A two-sided formula object describing the estimator structure. The left-hand side is the outcome variable, which must be discrete. Subgroups for which to calculate estimates may be specified by adding covariates on the right-hand side. Subgroup estimates are available with ‘coef(..., subgroup=TRUE)‘ and ‘tidy(..., subgroup=TRUE)‘.

`data` An optional data frame containing the variables named in ‘formula‘.

`weights` An optional numeric vector specifying weights.

`prefix` If ‘r_probs‘ is a data frame, the columns containing racial probabilities will be selected as those with names starting with ‘prefix‘. The default will work with the output of [bisg()].

`se_boot` The number of bootstrap replicates to use to compute an approximate covariance matrix for the estimator. If no bootstrapping is used, an analytical estimate of standard errors will be returned as ‘$se‘. For bootstrapping, when there are fewer than 1,000 individuals or 100 or fewer replicates, a Bayesian bootstrap is used instead (i.e., weights are drawn from a Dirichlet$(1, 1, ..., 1)$ distribution, which produces more reliable estimates.

`...` Additional arguments to generic methods (ignored).

`object, x` An object of class ‘est_weighted‘.

Value

An object of class ‘est_weighted‘, inheriting from [‘birdie‘][birdie::birdie-class], for which many methods are available. The model estimates may be accessed with ‘coef()‘. Uncertainty estimates, if available, can be accessed with ‘$se‘ and [vcov.birdie()].

Methods (by generic)

- `print(est_weighted)`: Print a summary of the model fit.
- `summary(est_weighted)`: Print a more detailed summary of the model fit.

References

McCartan, C., Fisher, R., Goldin, J., Ho, D.E., & Imai, K. (2024). Estimating Racial Disparities when Race is Not Observed. Available at https://www.nber.org/papers/w32373.

Examples

```
data(pseudo_vf)

r_probs = bisg(~ nm(last_name) + zip(zip), data=pseudo_vf)

# Process zip codes to remove missing values
pseudo_vf$zip = proc_zip(pseudo_vf$zip)

est_weighted(r_probs, turnout ~ 1, data=pseudo_vf)

est = est_weighted(r_probs, turnout ~ zip, data=pseudo_vf)
tidy(est, subgroup=TRUE)
```

`preproc` *Preprocess Last Names and Geographic Identifiers*

Description

These functions are called automatically by [bisg()] but may be useful, especially when geographic variables are included in a [birdie()] model. ‘proc_zip()‘ and ‘proc_state()‘ preprocess their corresponding geographic identifiers. States are partially matched to state names and abbreviations and are returned as FIPS codes. ZIP codes are crosswalked to Census ZCTAs. Missing identifiers are replaced with ‘"<none>"‘. ‘proc_name()‘ processes last names in accordance with Census processing rules (<https://www2.census.gov/topics/genealogy/2010surnames/surnames.pdf>). Names are converted to Latin characters, capitalized, stripped of prefixes and suffixes, and otherwise standardized.

Usage

```
proc_zip(x)

proc_state(x)

proc_name(x, to_latin = TRUE)
```

Arguments

`x`	A character vector of names or geographic identifiers to process
`to_latin`	If ‘TRUE‘, convert names to Latin characters only. Strongly recommended if non-Latin characters are present, since these will not match Census tables. However, the conversion is slightly time-consuming and so can be disabled with this flag.

Value

A processed character vector

Functions

- `proc_zip()`: Match ZIP codes to ZCTAs and fill in missing values.
- `proc_state()`: Match state names and abbreviations and fill in missing values.
- `proc_name()`: Process names to a Census-standardized format.

Examples

```
proc_name("Smith Jr.")
proc_zip("00501")
proc_state("Washington")
```

pseudo_vf *A pseudo-voterfile*

Description

A dataset containing 5,000 fake voter records. Created by randomizing a subset of the North Carolina voter file. Turnout records are completely randomly generated.

Usage

```
pseudo_vf
```

Format

A data frame with 5,000 rows and 4 records:

last_name Voter's last name

zip 5-digit ZIP code. May be NA

race One of "white", "black", "hisp", "asian", "aian", or "other"

turnout 1 if the voter voted in the most recent election, 0 otherwise

Source

https://www.ncsbe.gov/results-data/voter-registration-data

Examples

```
data(pseudo_vf)
print(pseudo_vf)
```

p_r_natl *National Racial Demographics*

Description

Returns the proportion of the U.S. population in six racial groups in a given year. Group definitions necessarily follow those used by the Census Bureau in its surname tables: * ‘white‘: Non-Hispanic White alone * ‘black‘: Non-Hispanic Black alone * ‘hisp‘: Hispanic, any race * ‘asian‘: Non-Hispanic Asian, Native Hawaiian, or Pacific Islander alone * ‘aian‘: Non-Hispanic American Indian/Alaska Native * ‘other‘: Non-Hispanic, two or more races, or other race

Usage

```
p_r_natl(year = 2021, vap = FALSE)
```

Arguments

`year`	The year to return demographics for.
`vap`	If ‘TRUE‘, return statistics for the voting-age population (18+) rather than the full U.S. population.

Value

A named numeric vector of length 6.

Examples

```
p_r_natl(year=2010)
```

Index