Package ‘natstrat’

February 4, 2026

Type Package

Title Obtain Unweighted Natural Strata that Balance Many Covariates

Version 2.0.1

Description Natural strata can be used in observational studies to balance the distributions of many covariates across any number of treatment groups and any number of comparisons. These strata have proportional amounts of units within each stratum across the treatments, allowing for simple interpretation and aggregation across strata. Within each stratum, the units are chosen using randomized rounding of a linear program that balances many covariates. For more details, see Brumberg et al. (2022) <doi:10.1111/rssa.12848> and Brumberg et al.(2023) <doi:10.1093/jrsssc/qlad010>.
To solve the linear program, the 'Gurobi' commercial optimization software is recommended, but not required. The 'gurobi' R package can be installed by following the instructions at <https://docs.gurobi.com/projects/optimizer/en/current/reference/r/setup.html> after claiming your free academic license at <https://www.gurobi.com/academia/academic-program-and-licenses/>.

URL https://github.com/kkbrum/natstrat,
https://kkbrum.github.io/natstrat/,
https://docs.gurobi.com/projects/optimizer/en/current/reference/r/setup.html

BugReports https://github.com/kkbrum/natstrat/issues

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Imports Rglpk, stats, plyr, pps, sampling, ggplot2, rlang, ramify, slam

Depends R (>= 2.10), caret

Suggests knitr, rmarkdown, markdown, testthat (>= 3.0.0), DT, stringr, covr, gurobi

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Katherine Brumberg [aut, cre]

Maintainer Katherine Brumberg <kbrum@umich.edu>

Repository CRAN

Date/Publication 2026-02-03 23:00:22 UTC

Contents

`check_balance` *Check covariate balance of the control and treated groups*

Description

Reports standardized differences in means between the treated and control group before and after choosing a subset of controls. These differences are reported both across strata and within strata. This function can also generate love plots of the same quantities.

Usage

```
check_balance(
  z,
  X,
  st,
  selected,
  treated = 1,
  control = 0,
  denom_variance = "treated",
  plot = FALSE,
  message = TRUE
)
```

Arguments

`z` a factor with the `i`th entry equal to the treatment of unit `i`.

`X` a data frame containing the covariates in the columns over which balance is desired. The number of rows should equal the length of `z`.

`st` a stratum vector with the `i`th entry equal to the stratum of unit `i`. This should have the same order of units and length as `z`.

`selected` a boolean vector including whether each unit was selected as part of the treated and control groups for analysis. Should be the same length as `z` and typically comes from the results of `optimize_controls()`.

`treated` which treatment value should be considered the treated units. This must be one of the values of `z`.

`control` which treatment value should be considered the control units. This must be one of the values of `z`.

`denom_variance` character stating what variance to use in the standardization: either the default "treated", meaning the standardization will use the treated variance (across all strata), where the treated group is declared in the `treated` argument, or "pooled", meaning the standardization will use the average of the variances of each treatment group.

`plot` a boolean denoting whether to generate love plots for the standardized differences.

`message` a boolean denoting whether to print a message about the level of balance achieved

Value

List containing:

sd_across matrix with one row per covariate and two columns: one for the standardized difference before a subset of controls were selected and one for after.

sd_strata matrix similar to `sd_across`, but with separate standardized differences for each stratum for each covariate.

sd_strata_avg matrix similar to `sd_across`, but taking the average of the standardized differences within the strata, weighted by stratum size.

plot_across ggplot object plotting `sd_across`, only exists if `plot = TRUE`.

plot_strata a named list of ggplot objects plotting `sd_strata`, one for each stratum, only exists if `plot = TRUE`.

plot_strata_avg ggplot object plotting `sd_strata_avg`, only exists if `plot = TRUE`.

plot_pair ggplot object with two facets displaying `sd_across` and `sd_strata_avg` with one y-axis, only exists if `plot = TRUE`.

Examples

```
data('nh0506')

# Create strata
age_cat <- cut(nh0506$age,
               breaks = c(19, 39, 50, 85),
               labels = c('< 40 years', '40 - 50 years', '> 50 years'))
strata <- age_cat : nh0506$sex

# Balance age, race, education, poverty ratio, and bmi both across and within the levels of strata
constraints <- generate_constraints(
             balance_formulas = list(age + race + education + povertyr + bmi ~ 1 + strata),
                 z = nh0506$z,
                 data = nh0506)

# Choose one control for every treated unit in each stratum,
# balancing the covariates as described by the constraints
results <- optimize_controls(z = nh0506$z,
                             X = constraints$X,
                             st = strata,
                             importances = constraints$importances,
                             ratio = 1)

cov_data <- nh0506[, c('sex', 'age', 'race', 'education', 'povertyr', 'bmi')]

# Check balance
stand_diffs <- check_balance(z = nh0506$z,
                             X = cov_data,
                             st = strata,
                             selected = results$selected,
                             plot = TRUE)
```

create_dist_matrix *Create matrix of distances between strata*

Description

Create a distance matrix between strata levels created from the interactions of factors. Used as input to generate_qs().

Usage

```
create_dist_matrix(...)
```

Arguments

...	any number of matrices that contain the distances between levels of a single stratifying factor. These should have both column and row names which correspond to the levels of the stratifying factor.

Value

Matrix containing the distances between all levels of the factor of all interactions between the inputted factors. The row and column names correspond to the levels of the strata, formed by combining the level name of each stratifying factor separated with ‘:‘.

Examples

```
data('nh0506')

age_cat <- cut(nh0506$age,
               breaks = c(19, 39, 50, 85),
               labels = c('< 40 years', '40 - 50 years', '> 50 years'))

age_dist <- matrix(data = c(0, 1, 2, 1, 0, 1, 2, 1, 0),
                   nrow = 3,
                   byrow = TRUE,
                   dimnames = list(levels(age_cat), levels(age_cat)))

sex_dist <- matrix(data = c(0, 1, 1, 0),
                   nrow = 2,
                   dimnames = list(levels(nh0506$sex), levels(nh0506$sex)))

strata_dist <- create_dist_matrix(age_dist, sex_dist)
```

generate_constraints *Generate constraints to encourage covariate balance*

Description

This function generates constraints that encourage covariate balance as specified. The main inputs are formula like objects, where the left hand side indicates the covariate to be balanced and the right hand side indicates the groups within which to balance. The constraints are weighted and standardized by stand() to be used in optimize_controls(). Missingness indicators can also be added and weighted for any covariate that has NA values.

Usage

```
generate_constraints(
  balance_formulas,
  z,
  data,
  default_rhs = NULL,
  weight_by_size = 0,
  denom_variance = "treated",
  treated = 1,
  autogen_missing = 50
)
```

Arguments

balance_formulas
: a list of formulas where the left hand side represents the covariate to be balanced, and the terms on the right hand side represent the populations within which the covariate should be balanced. More information can be found in the details below.

`z` a factor with the `ith` entry equal to the treatment of unit `i`.

`data` a data frame containing the relevant covariates in the columns. The number of rows should equal the length of `treated`.

`default_rhs` the list of `balance_formulas` can also contain entries that are just the character corresponding to a covariate to balance. If so, the covariate will be balanced according to `default_rhs`.

`weight_by_size` numeric between 0 and 1 stating how to adjust constraints for the size of the population they represent. Default is 0, meaning imbalance within populations is viewed in absolute terms, not relative to the population size. The program may thus prioritize balancing the covariate in larger populations compared to smaller populations. A value of 1 means that imbalance will be measured relative to the population's size, not in absolute terms, implying that it is equally important to balance in every population.

`denom_variance` character stating what variance to use in the standardization: either the default "treated", meaning the standardization will use the treated variance (across all strata), where the treated group is declared in the `treated` argument, or "pooled", meaning the standardization will use the average of the variances of each treatment group.

`treated` which treatment value should be considered the treated group. This must be one of the values of `z`. This is used if `denom_variance = "treated"` for calculating the variance to use in the standardization or if `weight_by_size > 0` to determine which treatment group to use to calculate population sizes.

`autogen_missing`

whether to automatically generate missingness constraints and how heavily to prioritize them. Should be a numeric or `NULL`. `NULL` indicates that constraints to balance the rate of missingness (denoted by `NA`s in `data`) should not be automatically generated. Note that this is not recommended unless the user has already accounted for missing values. If not `NULL`, `autogen_missing` should be a numeric stating how heavily to prioritize generated missingness constraints over covariate constraints. The default is 50.

Value

A list with two named components:

`X` a matrix with constraints as columns and the same number of rows as the inputs. The column names provide information about the constraints, including the covariate names and the factor and level to which it pertains.

`importances` a named vector with names corresponding to the constraint names and values corresponding to how heavily that constraint should be prioritized, based on the information provided through `balance_formulas`, `weight_by_size`, and `autogen_missing`.

Details

The `balance_formulas` argument can include formulas beyond those interpreted by `R` to be `formulas`. Their interpretation is also different, as explained below:

Left hand side The left hand side of the formula contains the covariate to be balanced. It can also be the sum of multiple covariates, in which case each term will be balanced individually according to the right hand side. Additionally, '.' on the left hand side will designate that all covariates in `data` should be balanced according to the designated or default right hand side (as usual, terms may be subtracted to remove them).

Right hand side The right hand side should be the sum of factor, character, or boolean variables. The covariate of the left hand side will be balanced within each level of each term on the right hand side. The right hand side can also contain '.', meaning the covariate will be balanced across all levels of all categorical, character, or boolean variables found in `data` (as usual, terms may be subtracted to remove them). In the most common case, the user will have one term on the right hand side consisting of the strata within which balance in desired.

Coefficients The formulas can contain coefficients specifying how much to weight a certain set of constraints. Coefficients of the left hand side terms will weight all constraints generated for that covariate, and coefficients of the right hand side will weight the constraints generated for each level of that term.

Intercept The intercept term, 1, is automatically included on the right hand side of the formula, and designates that the covariate of the left hand side will be balanced across all control units. You may enter a different numeric > 0 that will signify how much to weight the constraint, or you may enter "- 1" or "+ 0" to remove the intercept and its associated constraint, as per usual.

Examples

```
data('nh0506')

# Create strata
age_cat <- cut(nh0506$age,
               breaks = c(19, 39, 50, 85),
               labels = c('< 40 years', '40 - 50 years', '> 50 years'))
strata <- age_cat : nh0506$sex

# Balance age, race, education, poverty ratio, and bmi both across and within the levels of strata
constraints <- generate_constraints(
             balance_formulas = list(age + race + education + povertyr + bmi ~ 1 + strata),
                 z = nh0506$z,
                 data = nh0506)

# Balance age and race both across and within the levels of strata,
# with balance for race being prioritized twice as much as for age,
# and balance across strata prioritized twice as much as within.
# Balance education across and within strata,
# with balance within strata prioritized twice as much as across.
# Balance poverty ratio and bmi only within the levels of strata,
# as specified in the default_rhs argument
constraints <- generate_constraints(
                 balance_formulas = list(age + 2 * race ~ 2 + strata,
                                         education ~ 1 + 2 * strata,
                                         'povertyr',
                                         'bmi'),
                 z = nh0506$z,
                 data = nh0506,
                    default_rhs = '0 + strata')
```

generate_qs — *Calculate desired number of controls per stratum*

Description

Figure out how many units to take from each stratum when some strata are deficient. The result should be used as an input to optimize_controls().

Usage

```
generate_qs(
  z,
  st,
  ratio,
  treated = 1,
  max_ratio = NULL,
  max_extra_s = 5,
  strata_dist = NULL
)
```

Arguments

`z` a factor with the `i`th entry equal to the treatment of unit `i`.

`st` a stratum vector with the `i`th entry equal to the stratum of unit `i`. This should have the same order of units and length as `z`.

`ratio` a numeric or vector specifying the desired ratio of controls to ‘treated‘ in each stratum. If there is one control group and all treated units should be included, this can be a numeric. Otherwise, this should be a vector with one entry per treatment group, in the same order as the levels of `z`, including the treated level. If `NULL`, `q_s` should be specified.

`treated` which treatment value should be considered the treated units. This must be one of the values of `z`.

`max_ratio` a numeric or vector specifying the maximum ratio to allow in a stratum to achieve the overall `ratio` specified. If `NULL`, it is set by default to 1.1 times the desired `ratio`. To have no maximum ratio, set this to `Inf`.

`max_extra_s` single numeric or named vector or matrix with values corresponding to the maximum desired number of extra controls to be chosen from each stratum to achieve the overall `ratio` specified. If this is a vector, the names should correspond to the stratum values from `st`. If there are more than two treatment levels, this should be a matrix with one row per treatment level, in the same order as the levels of `z`. The default is 5 for each stratum in each treatment group. To have no maximum, set this to `Inf`. If both `max_ratio` and `max_s` are specified, the maximum of the two will be used for each stratum.

`strata_dist` matrix with both row and column names with names corresponding to the stratum values from `st` and entries corresponding to the distance associated with taking a control from the stratum associated with the row when the desired stratum is the one associated with the column. Lower distance values are more desirable replacements. Typically the diagonal should be 0, meaning there is no penalty for choosing a unit from the correct stratum.

Value

A named vector stating how many controls to take from each stratum.

`nh0506` *Homocysteine and smoking example data*

Description

NHANES 2005-2006 data on smoking and homocysteine levels in adults.

Usage

```
nh0506
```

Format

A data frame with 2928 rows and 11 variables:

SEQN NHANES identification number.

z smoking status treatment indicator: 1 = daily smoker, 0 = never smoker.

sex factor with levels "Male" and "Female".

age age in years, 20-85, with 85 recorded for everyone >= 85 years.

race factor with levels "Mexican American", "Other Hispanic", "Non-Hispanic White", "Non-Hispanic Black", and "Other Race - Including Multi-Racial".

education factor with levels "< Grade 9", "9-11th grade", "High school grad/GED", "Some college or AA degree", "College graduate or above".

povertyr ratio of family income to the poverty level, capped at 5 times poverty, has missing entries.

bmi BMI (body mass index), has missing entries.

cigsperday30 cigarettes smoked per day, 0 for never smokers.

cotinine blood cotinine level, a biomarker of recent exposure to tobacco.

homocysteine homocysteine level.

Details

The code used to generate this data is documented in the source version of this package under ‘data-raw/‘. This data is composed of adults aged at least 20 years. Individuals who have smoked at least 100 cigarettes but do not now smoke at least 10 cigarettes daily are excluded. Individuals with missing homocysteine values, cotinine values, or smoking information are excluded. After filtering for all these criteria, one individual with unknown education remains and is also excluded. Missing values remain in the poverty ratio and bmi covariates.

Source

https://wwwn.cdc.gov/nchs/nhanes/ContinuousNhanes/Default.aspx?BeginYear=2005

Examples

```
data('nh0506')
```

nh0506_3groups *Homocysteine and smoking example data with multiple control groups*

Description

NHANES 2005-2006 data on smoking and homocysteine levels in adults, comparing daily smokers to never smokers and occasional smokers.

Usage

```
nh0506_3groups
```

Format

A data frame with 4457 rows and 11 variables:

SEQN NHANES identification number.

z smoking status treatment factor: 0 = never smoker, 1 = some smoking, 2 = daily smoker.

sex factor with levels "Male" and "Female".

age age in years, 20-85, with 85 recorded for everyone >= 85 years.

race factor with levels "Mexican American", "Other Hispanic", "Non-Hispanic White", "Non-Hispanic Black", and "Other Race - Including Multi-Racial".

education factor with levels "< Grade 9", "9-11th grade", "High school grad/GED", "Some college or AA degree", "College graduate or above".

povertyr ratio of family income to the poverty level, capped at 5 times poverty, has missing entries.

bmi BMI (body mass index), has missing entries.

cigsperday30 cigarettes smoked per day, 0 for never smokers.

cotinine blood cotinine level, a biomarker of recent exposure to tobacco.

homocysteine homocysteine level.

Details

The code used to generate this data is documented in the source version of this package under ‘data-raw/‘. This data is composed of adults aged at least 20 years. Individuals who have smoked at least 100 cigarettes but do not now smoke at least 10 cigarettes daily are excluded. Individuals with missing homocysteine values, cotinine values, or smoking information are excluded. After filtering for all these criteria, five individuals with unknown education remain and are also excluded. Missing values remain in the poverty ratio and bmi covariates.

Source

https://wwwn.cdc.gov/nchs/nhanes/ContinuousNhanes/Default.aspx?BeginYear=2005

Examples

```
data('nh0506_3groups')
```

---

optimize_controls *Select control units that optimize covariate balance*

Description

Select control units within strata that optimize covariate balance. Uses randomized rounding of a linear program or a mixed integer linear program.

Usage

```
optimize_controls(
  z,
  X,
  st,
  importances = NULL,
  treated = 1,
  ratio = NULL,
  q_s = NULL,
  treated_star = NULL,
  q_star_s = NULL,
  weight_star = 1,
  integer = FALSE,
  solver = "Rglpk",
  seed = NULL,
  runs = 1,
  time_limit = Inf,
  threads = 1,
  correct_sizes = TRUE,
  low_memory = FALSE
)
```

Arguments

`z` a factor with the `i`th entry equal to the treatment of unit `i`.

`X` a matrix or data frame containing constraints in the columns. The number of rows should equal the length of `z`. Balance is achieved when a constraint sums to 0, such that numbers closer to 0 are better. When a constraint does not apply to a particular unit, the entry should be `NA`. This should typically be generated using `generate_constraints()`.

`st` a stratum vector with the `i`th entry equal to the stratum of unit `i`. This should have the same order of units and length as `z`.

`importances` a vector with length equal to the number of constraints or columns in `X`. This can be generated using `generate_constraints()` and each nonnegative value denotes how much to prioritize each constraint, with the default being 1 for all constraints.

`treated` which treatment value should be considered the treated units. This must be one of the values of `z`.

`ratio` a numeric or vector specifying the desired ratio of controls to ‘treated‘ in each stratum. If there is one control group and all treated units should be included, this can be a numeric. Otherwise, this should be a vector with one entry per treatment group, in the same order as the levels of `z`, including the treated level. If `NULL`, `q_s` should be specified.

`q_s` a named vector or matrix indicating how many units are to be selected from each stratum. If there is one control group and all treated units are desired, this can be a vector; otherwise, this should have one row per treatment group, where the order of the rows matches the order of the levels of `z`, including the treated level. If `NULL`, `ratio` should be specified. If both are specified, `q_s` will take priority. Typically, if the desired ratio is not feasible for every stratum, `q_s` should be generated using `generate_qs()`.

`treated_star` which treatment value should be considered the treated units for the supplemental comparison. This must be one of the values of `z`. If multiple supplemental comparisons are desired, this should be a vector with one entry per supplemental comparison.

`q_star_s` a named vector or matrix, indicating how many supplemental units are to be selected from each stratum. The matrix should have one row per treatment group, where the order of the rows matches the order of the levels of `z`, including the treated level. If multiple supplemental comparisons are desired, this should be a list with one entry per supplemental comparison.

`weight_star` a numeric stating how much to prioritize balance between the supplemental units as compared to balance between the main units. If multiple supplemental comparisons are desired, this should be a vector with one entry per supplemental comparison.

`integer` a logical stating whether to use a mixed integer programming solver instead of randomized rounding. Default is `FALSE`.

`solver` a character stating which solver to use to run the linear program. Options are "Rglpk" (default) or "gurobi". You must have the ’gurobi’ package installed to use the "gurobi" option. If available, this is the recommended solver.

`seed` the seed to use when doing the randomized rounding of the linear program. This will allow results to be reproduced if desired. The default is `NULL`, which will choose a random seed to use and return.

`runs` the number of times to run randomized rounding of the linear solution. The objective values of all runs will be reported, but the detailed results will only be reported for the run with the lowest objective value. The default is 1.

`time_limit` numeric stating maximum amount of seconds for which the program is allowed to run before aborting. Default is `Inf` for no time limit.

threads The maximum number of threads that should be used. This is only applicable if solver = 'gurobi'.

correct_sizes boolean stating whether the desired sample sizes should be exactly correct (if correct_sizes = TRUE) or only need to be correct in expectation. For multiple comparisons, sample sizes may only be correct in expectation.

low_memory boolean stating whether some outputs should not be included due to the scale of the problem being too large compared to memory space. If TRUE, eps and eps_star will not be reported. Imbalances can be computed post hoc using the check_balance() instead.

Value

List containing:

objective objective value of the randomized rounding or mixed integer linear program solution.

objective_wo_importances objective value of the randomized rounding or mixed integer linear program solution not weighted by the variable importances.

eps the amount of imbalance obtained in each constraint from the linear program. The row names specify the covariate, the population of interest, and, if there are more than two comparison groups, which groups are being compared.

eps_star same as eps but for the supplemental units instead of the units in the main comparison. If there are multiple supplemental comparisons, this is a list. If there are none, this is NULL.

importances the importance of each on the balance constraints.

weight_star the importance of balancing in the supplemental comparison relative to the main comparison. If there are multiple supplemental comparisons, this is a vector. If there are none, this is NULL.

selected whether each unit was selected for the main comparison.

selected_star whether each unit was selected for the supplement. If there are multiple supplemental comparisons, this is a list. If there are none, this is NULL.

pr the linear program weight assigned to each unit for the main comparison.

pr_star the linear program weight assigned to each unit for the supplement. If there are multiple supplemental comparisons, this is a list. If there are none, this is NULL.

rrdetails A list containing:

- seed the seed used before commencing the random sampling.
- run_objectives the objective values for each run of randomized rounding.
- run_objectives_wo_importances the objective values for each run of randomized rounding, not scaled by constraint importances.

lpdetails the full return of the function Rglpk_solve_LP() or gurobi() plus information about the epsilons and objective values for the linear program solution.

Examples

```
data('nh0506')

# Create strata
```

```
age_cat <- cut(nh0506$age,
               breaks = c(19, 39, 50, 85),
               labels = c('< 40 years', '40 - 50 years', '> 50 years'))
strata <- age_cat : nh0506$sex

# Balance age, race, education, poverty ratio, and bmi both across and within the levels of strata
constraints <- generate_constraints(
             balance_formulas = list(age + race + education + povertyr + bmi ~ 1 + strata),
                 z = nh0506$z,
                 data = nh0506)

# Choose one control for every treated unit in each stratum,
# balancing the covariates as described by the constraints
results <- optimize_controls(z = nh0506$z,
                             X = constraints$X,
                             st = strata,
                             importances = constraints$importances,
                             ratio = 1)

# If you want to use a ratio that's not feasible,
# you can supply a vector of the desired number of controls per stratum, q_s,
# typically generated by creating a distance matrix between strata and using
# generate_qs():

## Not run:
age_dist <- matrix(data = c(0, 1, 2, 1, 0, 1, 2, 1, 0),
                   nrow = 3,
                   byrow = TRUE,
                   dimnames = list(levels(age_cat), levels(age_cat)))

sex_dist <- matrix(data = c(0, 1, 1, 0),
                   nrow = 2,
                   dimnames = list(levels(nh0506$sex), levels(nh0506$sex)))

strata_dist <- create_dist_matrix(age_dist, sex_dist)

qs <- generate_qs(z = nh0506$z,
                  st = strata,
                  ratio = 2.5,
                  max_ratio = 2.6,
                  max_extra_s = 0,
                  strata_dist = strata_dist)

results <- optimize_controls(z = nh0506$z,
                             X = constraints$X,
                             st = strata,
                             importances = constraints$importances,
                             q_s = qs)


## End(Not run)

# We can also have multiple treatment and control groups,
# as well as multiple simultaneous comparisons:

## Not run:
data('nh0506_3groups')
strata2 <- cut(nh0506_3groups$age, breaks = c(19, 39, 50, 85),
              labels = c('< 40 years', '40 - 50 years', '> 50 years'))
constraints2 <- generate_constraints(
  balance_formulas = list(age + race + education + povertyr + bmi + sex ~ 1 + strata2),
  z = nh0506_3groups$z,
  data = nh0506_3groups,
  treated = 'daily smoker')
q_star_s <- matrix(c(rep(table(nh0506_3groups$z, strata2)['some smoking', ] -
                           table(nh0506_3groups$z, strata2)['daily smoker', ], 2),
                     rep(0, 3)), byrow = TRUE, nrow = 3,
                   dimnames = list(levels(nh0506_3groups$z), levels(strata2)))

results <- optimize_controls(z = nh0506_3groups$z,
                             X = constraints2$X,
                             importances = constraints2$importances,
                             st = strata2,
                             ratio = 1,
                             treated = 'daily smoker',
                             treated_star = 'some smoking',
                             q_star_s = q_star_s,
                             correct_sizes = FALSE)


## End(Not run)
```

stand *Standardize covariate vector for balance constraint*

Description

This function is used by `generate_constraints()` to standardize covariate vectors to become balance constraints. The function divides the covariate values by the treated or average group standard deviation (across strata).

Usage

```
stand(z, x, denom_variance = "treated", treated = 1, autogen_missing = 50)
```

Arguments

z a factor with the ith entry equal to the treatment of unit i.

x a covariate vector with ith entry equal to the covariate value of unit i. This should have the same order of units and length as z.

`denom_variance` character stating what variance to use in the standardization: either the default "treated", meaning the standardization will use the treated variance (across all strata), where the treated group is declared in the `treated` argument, or "pooled", meaning the standardization will use the average of the variances of each treatment group.

`treated` which treatment value should be considered the treated units. This must be one of the values of `z`.

`autogen_missing` whether to automatically generate missingness constraint and how heavily to prioritize it. Should be a numeric or `NULL` value. `NULL` indicates that a constraint to balance the rate of missingness (denoted by `NA` in `x`) should not be automatically generated. Note that this is not recommended unless the user has already accounted for missing values. If not `NULL`, `autogen_missing` should be a numeric stating how heavily to prioritize generated missingness constraints over covariate constraint. The default is 50.

Value

A list with two components:

covariate a balance constraint for the standardized covariate values of all unites.

missingness a corresponding balance constraint for the rate of missingness if `autogen_missing` not `NULL`, otherwise `NULL`.

Index