Package ‘natstrat’

May 26, 2021

Type    Package
Title   Obtain Unweighted Natural Strata that Balance Many Covariates
Version 1.0.0

Description Natural strata fix a constant ratio of controls to treated units within each stratum. This ratio need not be an integer. The control units are chosen using randomized rounding of a linear program that balances many covariates.
To solve the linear program, the 'Gurobi' commercial optimization software is recommended, but not required. The 'gurobi' R package can be installed following the instructions at <https://www.gurobi.com/documentation/9.1/refman/ins_the_r_package.html>.

URL https://github.com/kkbrum/natstrat,
      https://kkbrum.github.io/natstrat/,
      https://www.gurobi.com/documentation/9.1/refman/ins_the_r_package.html

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

License GPL-3
Encoding UTF-8
LazyData true

RoxygenNote 7.1.1
Imports Rglpk, stats, plyr, pps, sampling, ggplot2, rlang
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@wharton.upenn.edu>

Repository CRAN
Date/Publication 2021-05-26 08:10:02 UTC
**check_balance**

**R topics documented:**

- check_balance ............................ 2  
- create_dist_matrix .......................... 4  
- generate_constraints ...................... 5  
- generate_qs ............................... 7  
- nh0506 ................................... 8  
- optimize_controls ......................... 9  
- stand ..................................... 12

**Index**

| 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**

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

**Arguments**

- `z`  
  a treatment indicator vector with ith entry equal to 0 if unit i is a control and equal to 1 if unit i is treated.

- `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 ith 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()`.

- ` 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), or "pooled", meaning the standardization will use the average of the treated and control variances.
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.

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

ds_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 + poverty + bmi ~ 1 + strata),
  x = 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', 'poverty', '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)
**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**

```r
generate_constraints(
  balance_formulas, 
  z, 
  data, 
  default_rhs = NULL, 
  weight_by_size = 0, 
  denom_variance = "treated", 
  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 groups within which the covariate should be balanced. More information can be found in the details below.

- `z`  
  a treatment indicator vector with i\textsuperscript{th} entry equal to 0 if unit i is a control and equal to 1 if unit i is treated.

- `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 groups they represent. Default is 0, meaning imbalance within groups is viewed in absolute terms, not relative to the group size. The program may thus prioritize balancing the covariate in larger groups compared to smaller groups. A value of 1 means that imbalance will be measured relative to the group's size, not in absolute terms, implying that it is equally important to balance in every group.

- `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), or "pooled", meaning the standardization will use the average of the treated and control variances.

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 NAs 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 + povverty + 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,
                            'povverty',
                            '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 = NULL, 
   max_ratio = NULL, 
   max_extra_s = 5, 
   strata_dist = NULL)
Arguments

z  a treatment indicator vector with ith entry equal to 0 if unit i is a control and equal to 1 if unit i is treated.

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

ratio  a numeric specifying the desired ratio of controls to treated in each stratum.

max_ratio  a numeric 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 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. The default is 5 for each stratum. 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.

Value

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.
**optimize_controls**

- **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".
- **poverty** 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**


**Examples**

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

---

### 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**

```r
optimize_controls(
  z,
  X,
  st,
  importances = NULL,
  ratio = NULL,
  q_s = NULL,
  integer = FALSE,
  solver = "Rglpk",
)```
```r
seed = NULL,
runs = 1,
time_limit = Inf
)
```

**Arguments**

- `z` a treatment indicator vector with \(i\)th entry equal to 0 if unit \(i\) is a control and equal to 1 if unit \(i\) is treated.
- `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.
- `ratio` a numeric specifying the desired ratio of controls to treated in each stratum. If `NULL`, `q_s` should be specified.
- `q_s` a named vector indicating how many control units are to be selected from each stratum. 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()`.
- `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.

**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.

**importances**  the importance of each on the balance constraints.

**selected**  the selected treated and control units.

**controls**  the linear program weight assigned to each control and whether it was selected by randomized rounding.

**rrdetails**  A list containing:

- **seed**  the seed used before commencing the random sampling.
- **raw_objective**  objective value of the randomized rounding or mixed integer linear program solution before the denominator has been corrected for the number of units chosen with missing covariate values.
- **raw_objective_wo_importances**  objective value of the randomized rounding or mixed integer linear program solution not weighted by the variable importances before the denominator has been corrected for the number of units chosen with missing covariate values.
- **raw_eps**  the amount of imbalance obtained in each constraint from the linear program, before the denominators have been corrected for the number of units chosen with missing covariate values.
- **run_raw_objectives**  the objective values for each run of randomized rounding, before denominators have been corrected for missingness.
- **run_raw_objectives_wo_importances**  the objective values for each run of randomized rounding, before denominators have been corrected for missingness, not scaled by constraint importances.
- **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

```r
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():

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)

---

**stand**  
*Standardize covariate vector for balance constraint*

**Description**

This function is used by **generate_constraints**() to standardize covariate vectors to become balance constraints. This standardization is done such that the balance constraint will be minimized when the treated and control groups (within or across strata) have equal means. The function subtracts the treated mean (within or across strata) and divides by the treated or pooled standard deviation (across strata).

**Usage**

stand(z, x, st, ist = NULL, denom_variance = "treated", autogen_missing = 50)
**Arguments**

- **z**: a treatment indicator vector with $i$th entry equal to 0 if unit $i$ is a control and equal to 1 if unit $i$ is treated.
- **x**: a covariate vector with $i$th entry equal to the covariate value of unit $i$. This should have the same order of units and length as $z$.
- **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 $z$.
- **ist**: an optional specification of the target stratum within which the balance constraint is desired. Must be one of the values within st. By default, this is NULL, meaning the generated constraint balances across strata.
- **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), or "pooled", meaning the standardization will use the average of the treated and control variances.
- **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 either all treated and control units, or just those in stratum ist.
- **missingness**: a corresponding balance constraint for the rate of missingness if autogen_missing not NULL, otherwise NULL.
Index

* datasets
  nh0506, 8

check_balance, 2
create_dist_matrix, 4

generate_constraints, 5, 10, 12
generate_qs, 4, 7, 10

nh0506, 8

optimize_controls, 2, 5, 7, 9

Rglpk_solve_LP, 11

stand, 5, 12