Package ‘optrefine’

April 18, 2023

Title Optimally Refine Strata

Version 1.1.0

Description Splits initial strata into refined strata that optimize covariate balance. For more information, please email the author for a copy of the accompanying manuscript. 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/optrefine, https://kkbrum.github.io/optrefine/, https://www.gurobi.com/documentation/9.1/refman/ins_the_r_package.html

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

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 2.10), MASS, Rglpk, sampling, ggplot2

Suggests covr, gurobi, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Katherine Brumberg [aut, cre] (<https://orcid.org/0000-0002-5193-6250>)

Maintainer Katherine Brumberg <kbrum@wharton.upenn.edu>

Repository CRAN

Date/Publication 2023-04-18 19:20:09 UTC

R topics documented:

best_split *Find the best split for a stratum*

Description

Runs `split_stratum()` many times and selects the best result.

Usage

```
best_split(
  z,
  X,
  strata,
  ist,
  nc_list,
  nt_list,
  wMax = 5,
  wEach = 1,
  solver = "Rglpk",
  integer = FALSE,
  min_split = 10,
  threads = threads
)
```

Arguments

`z`	Vector of treatment assignment
`X`	Covariate matrix or data.frame
`strata`	vector of initial strata assignments; only used if `object` is not supplied. Can be `NULL`, in which case an initial stratification using the quintiles of the propensity score is generated using `prop_strat()` and the generated propensity score is also added to the X matrix as an extra covariate
`ist`	the stratum to be split

`nc_list`	a list of choices for the `nc` parameter in `split_stratum()`. Each element is a vector with entries corresponding to the number of control units that should be placed in each new stratum
`nt_list`	a list of choices for the `nt` parameter in `split_stratum()`. Each element is a vector with entries corresponding to the number of treated units that should be placed in each new stratum
`wMax`	the weight the objective places on the maximum epsilon
`wEach`	the weight the objective places on each epsilon
`solver`	character specifying the optimization software to use. Options are "Rglpk" or "gurobi". The default is "gurobi"
`integer`	boolean whether to use integer programming instead of randomized rounding. Default is `FALSE`. It is not recommended to set this to `TRUE` as the problem may never finish
`min_split`	a numeric specifying the minimum number of each control and treated units to be tolerated in a stratum. Any combination of elements from `nc_list` and `nt_list` that violate this are skipped
`threads`	how many threads to use in the optimization if using "gurobi" as the solver. Default will use all available threads

Value

A list containing the following elements:

- valuesIP, valuesLP: matrices containing integer and linear programming scaled objective values for each sample size tried, with rows corresponding to the elements of `nc_list` and columns corresponding to the elements of `nt_list`
- besti, bestj: indices of the best sample sizes in `nc_list` and in `nt_list`, respectively
- n_smds: number of standardized mean differences contributing to the objective values (multiply the scaled objective values by this number to get the true objective values)
- n_fracs: number of units with fractional LP solutions in the best split
- rand_c_prop, rand_t_prop: proportions of the control and treated units in each stratum that were selected with randomness for the best split
- pr: linear programming solution for the best split, with rows corresponding to the strata and columns to the units
- selection: vector of selected strata for each unit in the initial stratum to be split for the best split

Examples

```
# Generate a small data set
set.seed(25)
samp <- sample(1:nrow(rhc_X), 1000)
cov_samp <- sample(1:26, 10)

# Create some strata
```

```
ps <- prop_strat(z = rhc_X[samp, "z"],
                 X = rhc_X[samp, cov_samp], nstrata = 5)

# Save the sample sizes
tab <- table(ps$z, ps$base_strata)

# Choose the best sample sizes among the options provided
best_split(z = ps$z, X = ps$X, strata = ps$base_strata, ist = 1,
           nc_list = list(c(floor(tab[1, 1] * 0.25), ceiling(tab[1, 1] * 0.75)),
                          c(floor(tab[1, 1] * 0.4), ceiling(tab[1, 1] * 0.6))),
           nt_list = list(c(floor(tab[2, 1] * 0.3), ceiling(tab[2, 1] * 0.7))),
           min_split = 5)
```

calc_smds *Calculate standardized mean differences for initial and refined strata*

Description

Summarizes initial and/or refined strata in terms of standardized mean differences (SMDs).

Usage

```
calc_smds(
  object = NULL,
  z = NULL,
  X = NULL,
  base_strata = NULL,
  refined_strata = NULL,
  abs = TRUE
)
```

Arguments

`object`	an optional object of class `strat`, typically created using `strat()` or as a result of a call to `prop_strat()`. If not provided, `z` and `X` must be specified
`z`	vector of treatment assignment; only used if `object` is not supplied
`X`	covariate matrix/data.frame; only used if `object` is not supplied
`base_strata`	optional initial stratification for which to calculate SMDs; only used if `object` is not supplied
`refined_strata`	optional refined stratification for which to calculate SMDs; only used if `object` is not supplied
`abs`	boolean whether to return absolute standardized mean differences or raw values. Default is TRUE for absolute values

Value

List with two elements, "base" and "refined", each containing a matrix of standardized mean differences for each stratum (row) and covariate (column).

Examples

```
# Choose 500 patients and 5 covariates to work with for the example
set.seed(15)
samp <- sample(1:nrow(rhc_X), 500)
cov_samp <- sample(1:26, 5)

# Let it create propensity score strata for you and then refine them
ref <- refine(X = rhc_X[samp, cov_samp], z = rhc_X[samp, "z"])

# Look at covariate balance for propensity score and refined strata
calc_smds(object = ref)
```

`new_strat` *Constructor for object of class "strat"*

Description

Creates an object of S3 class "strat"

Usage

```
new_strat(z, X, base_strata = NULL, refined_strata = NULL, details = NULL)
```

Arguments

`z`	Vector of treatment assignment
`X`	Covariate matrix or data.frame
`base_strata`	Original strata, if they exist
`refined_strata`	Refined strata, if they exist
`details`	A list of details from the linear program. Also includes `X_std` if calculated

---

`plot.strat` *Plot diagnostics for a "strat" object*

Description

Plots the standardized mean differences for `strat` objects in the format of Love (2002).

Usage

```
## S3 method for class 'strat'
plot(
  x,
  incl_none = TRUE,
  incl_base = TRUE,
  by_strata = FALSE,
  weighted_avg = FALSE,
  legend = c("No strata", "Base strata", "Refined strata"),
  ...
)
```

Arguments

`x`	object of S3 class 'strat'
`incl_none`	whether to plot imbalances before any stratification
`incl_base`	whether to plot imbalances for the base stratification (if one exists)
`by_strata`	whether to generate a list of plots, one for each base stratum if `incl_base` is `TRUE`, or one for each refined stratum if `incl_base` is `FALSE`. Not used if `incl_none` is `TRUE`
`weighted_avg`	whether to take the weighted average instead of the straight average when collapsing standardized mean differences across strata. Default is `FALSE`
`legend`	a vector of labels to use for the three stratifications on the plot. The corresponding label for any stratification that is not to be plotted must still be provided but will be ignored and can be set to `NA`
`...`	further arguments passed to or from other methods

Value

Either a ggplot object for the Love plot of standardized mean differences or a list of such ggplot objects if `by_strata` is `TRUE`

References

Love, T. E. (2002), "Displaying covariate balance after adjustment for selection bias", Joint Statistical Meetings, yumpu.com/en/document/read/41664623.

Examples

```
# Choose 800 patients and 5 covariates to work with for the example
set.seed(28)
samp <- sample(1:nrow(rhc_X), 800)
cov_samp <- sample(1:26, 5)
ref <- refine(X = rhc_X[samp, cov_samp], z = rhc_X[samp, "z"])
plot(ref)
```

print.strat *Print stratification object*

Description

Print method for class "strat". Prints tables of numbers of control and treated units without strata and in initial and/or improved strata. Also displays average and maximum standardized mean difference for each stratification.

Usage

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

Arguments

x	object of S3 class 'strat'
...	further arguments passed to or from other methods

Value

Returns x back invisibly and prints tables and statistics to the console

Examples

```
# Choose 750 patients and 5 covariates to work with for the example
set.seed(21)
samp <- sample(1:nrow(rhc_X), 750)
cov_samp <- sample(1:26, 5)
ref <- refine(X = rhc_X[samp, cov_samp], z = rhc_X[samp, "z"])
print(ref)
```

prop_strat *Form propensity score strata*

Description

Form initial propensity score strata to be improved upon by refine().

Usage

```
prop_strat(z, X, nstrata = 5)
```

Arguments

`z`	Vector of treatment assignment
`X`	Covariate matrix or data.frame
`nstrata`	The number of strata to form

Value

Object of class "strat", which is a list containing `z`, `X` with the propensity score as an additional column, `base_strata` (a factor of the resulting propensity score strata), and `details`, (a list containing `X_std`, which is the standardized version of the new `X`)

Examples

```
 ps <- prop_strat(z = rhc_X[, "z"],
                  X = rhc_X[,  !(colnames(rhc_X) %in% c("pr", "z"))])
table(rhc_X[, "z"], ps$base_strata)
```

`rand_pvals` *Generate P-values using empirical randomization null distribution*

Description

Randomize the treatment assignment within strata to generate the randomization distribution of covariate balance given the strata and observed covariate values. Compare the observed covariate balance to this null distribution to calculate P-values.

Usage

```
rand_pvals(
  object = NULL,
  z = NULL,
  X = NULL,
  base_strata = NULL,
  refined_strata = NULL,
  options = list()
)
```

Arguments

`object`	an optional object of class `strat`, typically created using `strat()` or as a result of a call to `prop_strat()` or `refine()`. If not provided, `z` and `X` must be specified
`z`	vector of treatment assignment; only used if `object` is not supplied
`X`	covariate matrix/data.frame; only used if `object` is not supplied
`base_strata`	optional initial stratification for which to calculate the empirical randomization null distribution; only used if `object` is not supplied

`refined_strata`	optional refined stratification for which to calculate the empirical randomization null distribution; only used if `object` is not supplied
`options`	list of additional options, listed in the `details` below

Details

The literature on multivariate matching has recently developed a new way of evaluating covariate imbalances, comparing the imbalances found in an observational matched sample to the imbalances that would have been produced in the same data by randomization (Pimentel et al. 2015, Yu 2021). We modify that approach for use with strata, randomizing patients within strata. For a given stratification, we create a large number of stratified randomized experiments, taking the actual patients in their actual strata, and randomizing them to treatment or control with fixed within-stratum sample sizes.

To investigate how the actual observational imbalance in covariates compares to covariate imbalance in the randomized experiments built from the same strata, patients and covariates, we look at 4 metrics– the scaled objective value, which is a weighted combination of the maximum and the sum of all SMDs, depending on the `criterion` argument, the maximum and average SMDs across covariates and strata, and the average SMD across strata for each covariate individually. For each of these metrics, we record the observational value, the median and minimum of the randomized values, and the proportion of randomized values more imbalanced than the observational value (the P-value).

The `options` list argument can contain any of the following elements:

- nrand: how many times to randomize the treatment assignment when forming the null distribution. Default is 10000
- criterion: which optimization criterion to use when calculating the objective value. Options are "max", "sum", or "combo", referring to whether to include the maximum standardized mean difference (SMD), the sum of all SMDs, or a combination of the maximum and the sum. The default is "combo"
- wMax: how much to weight the maximum standardized mean difference compared to the sum. Only used if criterion is set to "combo". Default is 5
- incl_base: whether to include columns for the initial stratification in the table. Default is `TRUE` if a base stratification is provided

Value

List with three components:

- pvals: list containing `base` and `refined` elements, each of which is a list with randomization P-values for the objective value (`NULL` for the base stratification), the maximum standardized mean difference (SMD), the average SMD across covariates and strata, and for the average SMD across strata for each covariate (this element is a vector)
- obs_details: list containing `base` and `refined` elements, each of which is a list with the observed values for the objective value (`NULL` for the base stratification), the maximum standardized mean difference (SMD), and for the average SMD across strata for each covariate (this element is a vector)

- rand_details: list containing `base` and `refined` elements, each of which is a list with a vector of `nrand` randomized values for the objective value (`NULL` for the base stratification), the maximum standardized mean difference (SMD), and for the average SMD across strata for each covariate (this element is a matrix with `nrand` rows and a column for each covariate)

Examples

```
# Choose 500 patients and 5 covariates to work with for the example
set.seed(15)
samp <- sample(1:nrow(rhc_X), 500)
cov_samp <- sample(1:26, 5)

# Let it create propensity score strata for you and then refine them
ref <- refine(X = rhc_X[samp, cov_samp], z = rhc_X[samp, "z"])

# Calculate info for covariate balance randomization distribution
rpvals <- rand_pvals(object = ref, options = list(nrand = 100))

# Look at pvals before and after
rpvals$pvals
```

`refine` *Refine initial stratification*

Description

Refine an initial stratification by splitting each stratum or specified subset of strata into two refined strata. If no initial stratification is provided, one is first generated using `prop_strat()`.

Usage

```
refine(object = NULL, z = NULL, X = NULL, strata = NULL, options = list())
```

Arguments

`object`	an optional object of class `strat`, typically created using `strat()` or as a result of a call to `prop_strat()`. If not provided, `z` and `X` must be specified
`z`	vector of treatment assignment; only used if `object` is not supplied
`X`	covariate matrix/data.frame; only used if `object` is not supplied
`strata`	vector of initial strata assignments; only used if `object` is not supplied. Can be `NULL`, in which case an initial stratification using the quintiles of the propensity score is generated using `prop_strat()` and the generated propensity score is also added to the X matrix as an extra covariate
`options`	list containing various options described in the `Details` below

Details

The `options` argument can contain any of the following elements:

- solver: character specifying the optimization software to use. Options are "Rglpk" or "gurobi". The default is "Rglpk" unless a gurobi installation is detected, in which case it is set to "gurobi". It is recommended to use "gurobi" if available.
- standardize: boolean whether or not to standardize the covariates in X. Default is `TRUE`
- criterion: which optimization criterion to use. Options are "max", "sum", or "combo", referring to whether to optimize the maximum standardized mean difference (SMD), the sum of all SMDs, or a combination of the maximum and the sum. The default is "combo"
- integer: boolean whether to use integer programming as opposed to randomized rounding of linear programs. Note that setting this to `TRUE` may cause this function to never finish depending on the size of the data and is not recommended except for tiny data sets
- wMax: how much to weight the maximum standardized mean difference compared to the sum. Only used if criterion is set to "combo". Default is 5
- ist: which strata to split. Should be a level from the specified `strata` or a vector of multiple levels. Default is to split all strata
- minsplit: The minimum number of treated and control units to allow in a refined stratum. Default is 10
- threads: How many threads you'd like the optimization to use if using the "gurobi" solver. Uses all available threads by default

Note that setting a seed before using this function will ensure that the results are reproducible on the same machine, but results may vary across machines due to how the optimization solvers work.

Value

Object of class "strat", which is a list object with the following components:

- z: treatment vector
- X: covariate matrix
- base_strata: initial stratification
- refined_strata: refined_stratification
- details: various details about the optimization that can be ignored in practice, but may be interesting:
  - valueIP, valueLP: integer (determined via randomized rounding, unless `integer` option set to true) and linear programming scaled objective values
  - n_fracs: number of units with fractional LP solutions
  - rand_c_prop, rand_t_prop: proportions of the control and treated units in each stratum that were selected with randomness
  - pr: linear programming solution, with rows corresponding to the strata and columns to the units
  - criterion: criterion used in the optimization (see the `details` about the `options` for the optimization)
  - wMax: weight placed on the maximum standardized mean difference in the optimization (see the `details` about the `options` for the optimization)
  - X_std: standardized version of `X`

Examples

```
# Choose 400 patients and 4 covariates to work with for the example
set.seed(15)
samp <- sample(1:nrow(rhc_X), 400)
cov_samp <- sample(1:26, 4)

# Let it create propensity score strata for you and then refine them
ref <- refine(X = rhc_X[samp, cov_samp], z = rhc_X[samp, "z"])

# Or, specify your own initial strata
ps <- prop_strat(z = rhc_X[samp, "z"],
                 X = rhc_X[samp, cov_samp], nstrata = 3)
ref <- refine(X = ps$X, z = ps$z, strata = ps$base_strata)

# Can just input the output of prop_strat() directly
ref <- refine(object = ps)
```

`rhc_X` *Right Heart Catheterization Data*

Description

The data in the example are from Frank Harrell's `Hmisc` package. The data there are very similar to the data in Connors et al. (1996), but do not exactly reproduce analyses from that article. So, we employ the version of that analysis in the documentation for Ruoqi Yu's `RBestMatch` package, which attempts to be close to the analysis in Connors et al. In Yu's version, the propensity score (her `pr`) is built using 76 covariates, and the focus of attention is on 26 "priority" covariates (her `X`) and the propensity score that were emphasized in the Connors et al. article, including those in that article's Table 3.

Usage

```
rhc_X
```

Format

Matrix with 5,735 rows and 28 columns:

aps1 APACHE score

surv2md1 Support model estimate of the prob. of surviving 2 months

age Age

NumComorbid Number of comorbidities

adld3p_impute ADL with missing data imputed

adld3p_na ADL missing

das2d3pc DASI (Duke Activity Status Index)

temp1 Temperature

hrt1 Heart rate

meanbp1 Mean blood pressure

resp1 Respiratory rate

wblc1 WBC

pafi1 PaO2/FIO2 ratio

paco21 PaCo2

ph1 PH

crea1 Creatinine

alb1 Albumin

scoma1 Glasgow Coma Score

cat1_copd Primary disease category COPD

cat1_mosfsep Primary disease category MOSF w sepsis

cat1_mosfmal Primary disease category MOSF w malignancy

cat1_chf Primary disease category CHF

cat1_coma Primary disease category coma

cat1_cirr Primary disease category cirrhosis

cat1_lung Primary disease category lung cancer

cat1_colon Primary disease category colon cancer

pr Propensity score using 76 covariates

z Treatment indicator

References

Connors et al. (1996): The effectiveness of RHC in the initial care of critically ill patients. J American Medical Association 276:889-897.

https://hbiostat.org/data/.

`split_stratum` *Split one stratum into multiple strata*

Description

Split one stratum into multiple with specified sample sizes.

Usage

```
split_stratum(
  z,
  X,
  strata,
  ist,
  nc,
  nt,
  wMax = 5,
  wEach = 1,
  solver = "Rglpk",
  integer = FALSE,
  threads = NULL
)
```

Arguments

`z`	Vector of treatment assignment
`X`	Covariate matrix or data.frame
`strata`	vector of initial strata assignments; only used if `object` is not supplied. Can be `NULL`, in which case an initial stratification using the quintiles of the propensity score is generated using `prop_strat()` and the generated propensity score is also added to the X matrix as an extra covariate
`ist`	the stratum to be split
`nc`	a vector stating how many control units to place in each of the new split strata. The sum must be the total number of controls in the stratum to be split
`nt`	a vector stating how many treated units to place in each of the new split strata. The sum must be the total number of treated units in the stratum to be split
`wMax`	the weight the objective places on the maximum epsilon
`wEach`	the weight the objective places on each epsilon
`solver`	character specifying the optimization software to use. Options are "Rglpk" or "gurobi". The default is "gurobi"
`integer`	boolean whether to use integer programming instead of randomized rounding. Default is `FALSE`. It is not recommended to set this to `TRUE` as the problem may never finish
`threads`	how many threads to use in the optimization if using "gurobi" as the solver. Default will use all available threads

Value

A list containing the following elements:

- valueIP, valueLP: integer and linear programming scaled objective values
- n_smds: number of standardized mean differences contributing to the objective values (multiply the scaled objective values by this number to get the true objective values)

- n_fracs: the number of units with fractional linear programming solutions
- rand_c_prop, rand_t_prop: proportions of the control and treated units in each stratum that were selected with randomness
- pr: linear programming solution, with rows corresponding to the strata and columns to the units
- selection: vector of selected strata for each unit in the initial stratum to be split

Examples

```
# Generate a small data set
set.seed(25)
samp <- sample(1:nrow(rhc_X), 1000)
cov_samp <- sample(1:26, 10)

# Create some strata
ps <- prop_strat(z = rhc_X[samp, "z"],
                 X = rhc_X[samp, cov_samp], nstrata = 5)

# Save the sample sizes
tab <- table(ps$z, ps$base_strata)

# Choose the best sample sizes among the options provided
split_stratum(z = ps$z, X = ps$X, strata = ps$base_strata, ist = 1,
           nc = c(floor(tab[1, 1] * 0.25), ceiling(tab[1, 1] * 0.75)),
           nt = c(floor(tab[2, 1] * 0.3), ceiling(tab[2, 1] * 0.7)))
```

`strat` *Helper for object of class "strat"*

Description

Creates an object of S3 class "strat"

Usage

```
strat(z, X, base_strata = NULL, refined_strata = NULL, details = NULL)
```

Arguments

`z`	Vector of treatment assignment
`X`	Covariate matrix or data.frame
`base_strata`	Original strata, if they exist
`refined_strata`	Refined strata, if they exist
`details`	A list of details from the linear program. Include X_std if calculated

Value

Object of class `strat` if valid

Examples

```
# Don't need to include any stratification
strat_object <- strat(z = rhc_X[, "z"], X = rhc_X[, !(colnames(rhc_X) %in% "z")])

# Can include base and/or refined stratification if desired
strat_object <- strat(z = rhc_X[, "z"], X = rhc_X[, !(colnames(rhc_X) %in% "z")],
                      base_strata = rep(1, nrow(rhc_X)),
                      refined_strata = NULL)
```

`table_rand_pvals` *Generate a covariate balance table from the empirical randomization null distribution*

Description

Generate a table using the information collected in `rand_pvals()`. See `rand_pvals()` for more details about the methods used.

Usage

```
table_rand_pvals(
  object = NULL,
  z = NULL,
  X = NULL,
  base_strata = NULL,
  refined_strata = NULL,
  options = list()
)
```

Arguments

`object`	an optional object of class `strat`, typically created using `strat()` or as a result of a call to `prop_strat()` or `refine()`. If not provided, `z` and `X` must be specified
`z`	vector of treatment assignment; only used if `object` is not supplied
`X`	covariate matrix/data.frame; only used if `object` is not supplied
`base_strata`	optional initial stratification for which to calculate the empirical randomization null distribution; only used if `object` is not supplied
`refined_strata`	optional refined stratification for which to calculate the empirical randomization null distribution; only used if `object` is not supplied
`options`	list of additional options, listed in the `details` below

Details

The `options` list argument can contain any of the following elements:

- nrand: how many times to randomize the treatment assignment when forming the null distribution. Default is 10000
- criterion: which optimization criterion to use when calculating the objective value. Options are "max", "sum", or "combo", referring to whether to include the maximum standardized mean difference (SMD), the sum of all SMDs, or a combination of the maximum and the sum. The default is "combo"
- wMax: how much to weight the maximum standardized mean difference compared to the sum. Only used if criterion is set to "combo". Default is 5
- incl_base: whether to include columns for the initial stratification in the table. Default is `TRUE` if a base stratification is provided
- rand_pvals: if already calculated, the returned list of information from `rand_pvals()`. If `NULL`, this will be calculated

Value

Matrix with 4 or 8 columns, depending whether one or both of base and refined strata are provided and the `incl_base` option. The columns give the observed standardized mean difference or objective value, the median and maximum across `nrand` null simulations, and the P-value which is the proportion of the null simulations that have worse covariate balance than the observed value. The top three rows give the scaled objective value and the average and maximum standardized mean differences across all strata and covariates. The following rows, one for each covariate, give the standardized mean difference for that covariate, averaged across strata. The first row for the scaled objective value is `NULL` for the base stratification, if included, as the base stratification does not generally minimize a mathematical objective function.

`validate_strat` *Validator for object of class "strat"*

Description

Checks validity of an object of S3 class "strat"

Usage

```
validate_strat(object)
```

Arguments

`object`	An object of class `strat`

Value

Error or object of class `strat` if valid

Index