Package: ldamatch (via r-universe)

September 12, 2024

Title Selection of Statistically Similar Research Groups

Version 1.0.3

Description Select statistically similar research groups by backward
selection using various robust algorithms, including a
heuristic based on linear discriminant analysis, multiple
heuristics based on the test statistic, and parallelized
exhaustive search.

Depends R (>= 3.0.0)

License MIT + file LICENSE

VignetteBuilder knitr

Suggests knitr, markdown, rmarkdown, testthat, roxygen2, doParallel

Imports RUnit, data.table, entropy, foreach, iterators, iterpc,
kSamples, stats, car, gmp, utils, methods

RoxygenNote 7.3.1

Encoding UTF-8

NeedsCompilation no

Author Kyle Gorman [aut, cre], Géza Kiss [aut]

Maintainer Kyle Gorman <kylebgorman@gmail.com>

Date/Publication 2024-04-14 17:50:05 UTC

Repository https://kylebgorman.r-universe.dev

RemoteUrl https://github.com/cran/ldamatch

RemoteRef HEAD

RemoteSha 99a67f6817d56dfcc9a78bdfbf21d75a838436d8

Contents

`.get_if_args_are_missing`

Determines which arguments for a function, which is its caller by default.

Description

Determines which arguments for a function, which is its caller by default.

Usage

```
.get_if_args_are_missing(fun = sys.function(-1), ncall = 3)
```

Arguments

`fun`	A function; default: the caller.
`ncall`	The parent frame index; default: 3 (the great-grandparent).

Value

A named boolean vector that contains whether each argument is missing.

`ad_halt` *A univariate halting test using the Anderson-Darling test.*

Description

A univariate halting test using the Anderson-Darling test.

Usage

```
ad_halt(condition, covariates, thresh)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.

Value

The ratio of the p-value and the threshold, or 0 if the p-value is less than the threshold.

`calc_metrics` *Calculates basic metrics about ldamatch search result.*

Description

Calculates basic metrics about ldamatch search result.

Usage

```
calc_metrics(
  is.in,
  condition,
  covariates,
  halting_test,
  props = prop.table(table(condition)),
  tiebreaker = NULL
)
```

Arguments

`is.in`	The output of `match_groups()`: either a logical vector, or a list of those.
`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to ‘covariates‘ (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used. This is preferred among configurations with the same taken into account by the other methods to some extent. For example, c(A = 0.4, B = 0.4, C = 0.2) means that we would like the number of subjects in groups A, B, and C to be around 40%, 40%, and 20% of the total number of subjects, respectively. Whereas c("A", "B", "C") means that if possible, we would like to keep all subjects in group A, and prefer keeping subjects in B, even if it results in losing more subjects from C.
`tiebreaker`	NULL, or a function similar to halting_test, used to decide between cases for which halting_test yields equal values.

Value

A list containing:

all.is.in all results as a list;

is.in simply the first item in all.is.in or the error contained in is.in if there was an error running `match_groups`;

num_excluded the number of excluded subjects;

p_matched the test statistic from halting_test for the matched groups;

p_tiebreaker the test statistic from tiebreaker for the matched groups; and

balance_divergence a value characterizing the deviation from the expected group size proportions specified in props.

If the value for a field cannot be calculated, it will still be present with a value of NA.

`calc_p_value` *Calculates p-value using specified halting test.*

Description

Calculates p-value using specified halting test.

Usage

```
calc_p_value(condition, covariates, halting_test)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to ‘covariates‘ (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.

Value

The p-value.

`compare_ldamatch_outputs` *Compares outputs of ldamatch runs.*

Description

It favors, in decreasing order of priority, fewer excluded subjects, better balance (i.e. subsamples that diverge less from the expected proportions, which are by default the proportions of the input groups), and better (i.e. larger) test statistic for the matched groups. The preference order for the last two items can be reversed by specifying prefer_test = TRUE.

Usage

```
compare_ldamatch_outputs(
  is.in1,
  is.in2,
  condition,
  covariates = matrix(),
  halting_test = NA,
  props = prop.table(table(condition)),
  prefer_test = is.null(props),
  tiebreaker = NULL
)
```

Arguments

`is.in1`	A logical vector for output 1, TRUE iff row is in the match.
`is.in2`	A logical vector for output 2, TRUE iff row is in the match.
`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to ‘covariates‘ (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used. This is preferred among configurations with the same taken into account by the other methods to some extent. For example, c(A = 0.4, B = 0.4, C = 0.2) means that we would like the number of subjects in groups A, B, and C to be around 40%, 40%, and 20% of the total number of subjects, respectively. Whereas c("A", "B", "C") means that if possible, we would like to keep all subjects in group A, and prefer keeping subjects in B, even if it results in losing more subjects from C.
`prefer_test`	If TRUE, it prioritizes the test statistic more than the group size proportion.
`tiebreaker`	NULL, or a function similar to halting_test, used to decide between cases for which halting_test yields equal values.

Value

A number that is > 0 if is.in1 is a better solution than is.in2, < 0 if is.in1 is a worse solution than is.in2, or 0 if the two solutions are equivalent (not necessarily identical).

`create_halting_test` *Creates halting test from multiple tests.*

Description

The created halting test function returns the smallest p-value-to-threshold ratio of the values produced by the supplied tests, or zero if any of the p-values does not exceed the threshold. The resulting function expects one threshold per halting test in a vector or it recycles the given value(s) to get a threshold for each one.

Usage

```
create_halting_test(halting_tests)
```

Arguments

`halting_tests`	Either a vector of halting test functions (or function names) with the signature halting_test(condition, covariates, thresh) (for the meaning of the parameters see `match_groups`); or it may be a list of list(test = halting_test, cond = subset_of_conditions, cov = variable_selector, thresh) fields. All fields can be left out except test, and test need not be named if it is the first item in the list. The subset_of_conditions can be names of the conditions to match (a character vector or a factor). The variable_selector can be a logical vector with as many items as there will be columns in covariates (recommended), or a vector of integer covariate column indices. Each halting_test is then only applied to the specified subset of conditions and variables of the covariate matrix, with the specified threshold; when a value is not specified the defaults are used. Note that ordering the functions does not change the behavior, but can make the execution of the combined function faster, as the later ones are often evaluated only if the criteria for the earlier ones are met.

Value

A function that returns the minimum of all halting test values; the threshold value supplied to it is recycled for the individual functions.

`estimate_exhaustive` *Estimates the maximum number of cases to be checked during exhaustive search.*

Description

Estimates the maximum number of cases to be checked during exhaustive search.

Usage

```
estimate_exhaustive(
  min_preserved = sum(group_sizes),
  condition,
  cases_per_second = 100,
  print_info = TRUE,
  max_removed_per_cond = NULL,
  group_sizes = NULL,
  props = prop.table(table(condition)),
  max_cases = Inf
)
```

Arguments

`min_preserved`	Assumes that at least a total of this many subjects will be preserved.
`condition`	A factor vector containing condition labels.

`cases_per_second`	Assumes that this number of cases are checked out per second, for estimating the time it takes to run the exhaustive search; default: 100.
`print_info`	If TRUE, prints partial calculations as well for the number of cases and estimated time when removing 1, 2, ... subjects.
`max_removed_per_cond`	A named integer vector, containing the maximum number of subjects that can be removed from each group. Specify 0 for groups if you want to preserve all of their subjects. If you do not specify a value for a group, it defaults to 2 less than the group size. Values outside the valid range of 0..(N-1) (where N is the number of subjects in the group) are corrected without a warning.
`group_sizes`	A particular set of group sizes that we know a matched solution for; min_preserved need not be specified if this one is.
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used.
`max_cases`	Once it is certain that the number of cases is definitely above this number, calculation stops. In this case, the returned number is guaranteed to be larger than max_cases, but it is not the exact number of exhaustive cases. Default is infimum, i.e. the exact number of cases is calculated.

Value

The maximum number of cases: an integer if not greater than the maximum integer size (.Machine$integer.max), otherwise a Big Integer (see the gmp package).

Examples

```
estimate_exhaustive(58, as.factor(c(rep("ALN", 25), rep("TD", 44))))
estimate_exhaustive(84, as.factor(c(rep("ASD", 51), rep("TD", 44))))
```

f_halt — *A univariate halting test using Fisher's exact test.*

Description

A univariate halting test using Fisher's exact test.

Usage

```
f_halt(condition, covariates, thresh)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.

Value

The ratio of the p-value and the threshold, or 0 if the p-value is less than the threshold.

`get_param` *Gets value for ldamatch global parameter.*

Description

Gets value for ldamatch global parameter.

Usage

```
get_param(name)
```

Arguments

`name`	The name of the global parameter.

Value

The value of the global parameter.

See Also

`set_param` for parameter names.

`ks_halt` *A univariate halting test using the Kolmogorov-Smirnov Test, which must be satisfied for all condition pairs.*

Description

The condition must have two levels.

Usage

```
ks_halt(condition, covariates, thresh)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.

Details

Note that unlike many tests, the null hypothesis is that the two samples are are drawn from the same distribution.

Warnings such as "cannot compute exact p-value with ties" are suppressed.

Value

The ratio of the p-value and the threshold, or 0 if the p-value is less than the threshold. If there are more than two conditions, it returns the smallest value found for any condition pair.

`l_halt` *A univariate halting test using Levene's test.*

Description

Warnings such as "ANOVA F-tests on an essentially perfect fit are unreliable" are suppressed.

Usage

```
l_halt(condition, covariates, thresh)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.

Value

The ratio of the p-value and the threshold, or 0 if the p-value is less than the threshold.

`matching_methods` *The available methods for matching.*

Description

The available methods for matching.

Usage

```
matching_methods
```

Format

An object of class `character` of length 5.

`match_groups` *Creates a matched group via backward selection.*

Description

Creates a matched group via backward selection.

Usage

```
match_groups(
  condition,
  covariates,
  halting_test,
  thresh = 0.2,
  method = ldamatch::matching_methods,
  props = prop.table(table(condition)),
  replicates = get("RND_DEFAULT_REPLICATES", .ldamatch_globals),
  min_preserved = length(levels(condition)),
  print_info = get("PRINT_INFO", .ldamatch_globals),
  max_removed_per_cond = NULL,
  tiebreaker = NULL,
  lookahead = 2,
  all_results = FALSE,
  prefer_test = TRUE,
  max_removed_per_step = 1,
  max_removed_percent_per_step = 0.5,
  ratio_for_slowdown = 0.5
)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to 'covariates' (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.
`method`	The choice of search method, one of "random", You can get more information about each method on the help page for "search_<method_name>" (e.g. "`search_exhaustive`").
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used. This is preferred among configurations with the same taken into account by the other methods to some extent. For example, c(A = 0.4, B = 0.4, C = 0.2) means that we would like the number of subjects in groups A, B, and C to be around 40%, 40%, and 20% of the total number of subjects, respectively. Whereas c("A", "B", "C") means that if possible, we would like to keep all subjects in group A, and prefer keeping subjects in B, even if it results in losing more subjects from C.
`replicates`	The maximum number of random replications to be performed. This is only used for the "random" method.
`min_preserved`	The minimum number of preserved subjects. It can be used to ensure that the search will not take forever to run, but instead fail when a solution is not found when preserving this number of subjects.
`print_info`	If TRUE, prints summary information on the input and the results, as well as progress information for the exhaustive search and random algorithms. Default: TRUE; can be changed using `set_param("PRINT_INFO", FALSE)`.
`max_removed_per_cond`	A named integer vector, containing the maximum number of subjects that can be removed from each group. Specify 0 for groups if you want to preserve all of their subjects. If you do not specify a value for a group, it defaults to 2 less than the group size. Values outside the valid range of 0..(N-1) (where N is the number of subjects in the group) are corrected without a warning.
`tiebreaker`	NULL, or a function similar to halting_test, used to decide between cases for which halting_test yields equal values.
`lookahead`	The lookahead to use: a positive integer. It is used by the heuristic3 and heuristic4 algorithms, with a default of 2. The running time is O(N ^ lookahead), wheren N is the number of subjects.
`all_results`	If TRUE, returns all results found by method in a list. (A list is returned even if there is only one result.) If FALSE (the default), it returns the first result (a logical vector).

`prefer_test` If TRUE, prefers higher test statistic more than the expected group size proportion; default is TRUE. Used by all algorithms except exhaustive, which always

`max_removed_per_step`
The number of equivalent subjects that can be removed in each step. (The actual allowed number may be less depending on the p-value / theshold ratio.) This parameters is used by the heuristic3 and heuristic4 algorithms, with a default value of 1.

`max_removed_percent_per_step`
The percentage of remaining subjects that can be removed in each step. Used when max_removed_per_step > 1, with a default value of 0.5.

`ratio_for_slowdown`
The p-value / threshold ratio at which it starts removing subjects one by one. Used when max_removed_per_step > 1, with a default value of 0.5.

Details

The exhaustive, heuristic3, and heuristic4 search methods use the foreach package to parallelize computation. To take advantage of this, you must register a cluster. For example, to use all but one of the CPU cores, run: `doParallel::registerDoParallel(cores = max(1, parallel::detectCores() - 1))` To use sequential processing without getting a warning, run: `foreach::registerDoSEQ()`

Value

A logical vector that contains TRUE for the conditions that are in the matched groups; or if all_results = TRUE, a list of such vectors.

See Also

`calc_p_value` for calculating the test statistic for a group setup.

`calc_metrics` for calculating multiple metrics about the goodness of the result.

`compare_ldamatch_outputs` for comparing multiple different results from this function.

`search_heuristic2`, `search_heuristic3`, `search_heuristic4`, `search_random`, `search_exhaustive` for

`nondeterministic_matching_methods`
The available nondeterministic methods for matching.

Description

The available nondeterministic methods for matching.

Usage

```
nondeterministic_matching_methods
```

Format

An object of class `character` of length 3.

`parallelized_matching_methods` *The available parallelized methods for matching.*

Description

The available parallelized methods for matching.

Usage

```
parallelized_matching_methods
```

Format

An object of class `character` of length 3.

`search_exhaustive` *Searches the space backwards, prefering more subjects and certain group size proportions.*

Description

Searches the space backwards, prefering more subjects and certain group size proportions.

Usage

```
search_exhaustive(
  condition,
  covariates,
  halting_test,
  thresh,
  props,
  max_removed_per_cond,
  tiebreaker = NULL,
  min_preserved = length(levels(condition)),
  print_info = TRUE,
  given_args = NULL,
  ...
)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to 'covariates' (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used. This is preferred among configurations with the same taken into account by the other methods to some extent. For example, c(A = 0.4, B = 0.4, C = 0.2) means that we would like the number of subjects in groups A, B, and C to be around 40%, 40%, and 20% of the total number of subjects, respectively. Whereas c("A", "B", "C") means that if possible, we would like to keep all subjects in group A, and prefer keeping subjects in B, even if it results in losing more subjects from C.
`max_removed_per_cond`	The maximum number of subjects that can be removed from each group. It must have a valid number for each group.
`tiebreaker`	NULL, or a function similar to halting_test, used to decide between cases for which halting_test yields equal values.
`min_preserved`	The minimum number of preserved subjects. It can be used to ensure that the search will not take forever to run, but instead fail when a solution is not found when preserving this number of subjects.
`print_info`	If TRUE, prints summary information on the input and the results, as well as progress information for the exhaustive search and random algorithms. Default: TRUE; can be changed using `set_param("PRINT_INFO", FALSE)`.
`given_args`	The names of arguments given to the search function.
`...`	Consumes extra parameters that are not used by the search algorithm at hand; this function gives a warning about the ones whose value is not NULL that their value is not used.

Details

While the search is done in parallel, the search space is enormous and so it can be very slow in the worst case. It is perhaps most useful as a tool to study other matching procedures.

You can calculate the maximum possible number of cases to evaluate by calling estimate_exhaustive().

Value

All results found by search method in a list. It raises a "Convergence failure" error if it cannot find a matched set.

search_heuristic2 *OBSOLETE: Finds matching using depth-first search recursively.*

Description

Please use the heuristic3 search algorithm with lookahead=1 instead for nearly equivalent results. Note that heuristic3 is parallelized, more memory efficient, and chooses subject to remove randomly from among equivalent choices instead of choosing the first one deterministically. This function is implemented recursively, so may run out of memory when applied to many subjects.

Usage

```
search_heuristic2(
  condition,
  covariates,
  halting_test,
  thresh,
  props,
  max_removed_per_cond,
  tiebreaker = NULL,
  prefer_test = TRUE,
  print_info = TRUE,
  given_args = NULL,
  ...
)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to ‘covariates‘ (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used. This is preferred among configurations with the same taken into account by the other methods to some extent. For example, c(A = 0.4, B = 0.4, C = 0.2) means that we would like the number of subjects in groups A, B, and C to be around 40%, 40%, and 20% of the total number of subjects, respectively. Whereas c("A", "B", "C") means that if possible, we would like to keep all subjects in group A, and prefer keeping subjects in B, even if it results in losing more subjects from C.

`max_removed_per_cond`	The maximum number of subjects that can be removed from each group. It must have a valid number for each group.
`tiebreaker`	NULL, or a function similar to halting_test, used to decide between cases for which halting_test yields equal values.
`prefer_test`	If TRUE, prefers higher test statistic more than the expected group size proportion; default is TRUE. Used by all algorithms except exhaustive, which always
`print_info`	If TRUE, prints summary information on the input and the results, as well as progress information for the exhaustive search and random algorithms. Default: TRUE; can be changed using `set_param("PRINT_INFO", FALSE)`.
`given_args`	The names of arguments given to the search function.
`...`	Consumes extra parameters that are not used by the search algorithm at hand; this function gives a warning about the ones whose value is not NULL that their value is not used.

Details

In each step, it removes one subject from the set of subjects with the smallest p-value recursively.

Value

All results found by search method in a list. It raises a "Convergence failure" error if it cannot find a matched set.

`search_heuristic3` *Finds matching using depth-first search, looking ahead n steps.*

Description

In each step, it removes one subject from the set of subjects with the smallest associated p-value after "lookahead" steps.

Usage

```
search_heuristic3(
  condition,
  covariates,
  halting_test,
  thresh,
  props,
  max_removed_per_cond,
  tiebreaker = NULL,
  min_preserved = length(levels(condition)),
  lookahead = 2,
  prefer_test = TRUE,
  print_info = TRUE,
    max_removed_per_step = 1,
    max_removed_percent_per_step = 0.5,
    ratio_for_slowdown = 0.5,
    given_args = NULL,
    ...
  )
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to ‘covariates‘ (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used. This is preferred among configurations with the same taken into account by the other methods to some extent. For example, c(A = 0.4, B = 0.4, C = 0.2) means that we would like the number of subjects in groups A, B, and C to be around 40%, 40%, and 20% of the total number of subjects, respectively. Whereas c("A", "B", "C") means that if possible, we would like to keep all subjects in group A, and prefer keeping subjects in B, even if it results in losing more subjects from C.
`max_removed_per_cond`	The maximum number of subjects that can be removed from each group. It must have a valid number for each group.
`tiebreaker`	NULL, or a function similar to halting_test, used to decide between cases for which halting_test yields equal values.
`min_preserved`	The minimum number of preserved subjects. It can be used to ensure that the search will not take forever to run, but instead fail when a solution is not found when preserving this number of subjects.
`lookahead`	The lookahead to use: a positive integer. It is used by the heuristic3 and heuristic4 algorithms, with a default of 2. The running time is O(N ^ lookahead), wheren N is the number of subjects.
`prefer_test`	If TRUE, prefers higher test statistic more than the expected group size proportion; default is TRUE. Used by all algorithms except exhaustive, which always
`print_info`	If TRUE, prints summary information on the input and the results, as well as progress information for the exhaustive search and random algorithms. Default: TRUE; can be changed using `set_param("PRINT_INFO", FALSE)`.

`max_removed_per_step`	The number of equivalent subjects that can be removed in each step. (The actual allowed number may be less depending on the p-value / theshold ratio.) This parameters is used by the heuristic3 and heuristic4 algorithms, with a default value of 1.
`max_removed_percent_per_step`	The percentage of remaining subjects that can be removed in each step. Used when max_removed_per_step > 1, with a default value of 0.5.
`ratio_for_slowdown`	The p-value / threshold ratio at which it starts removing subjects one by one. Used when max_removed_per_step > 1, with a default value of 0.5.
`given_args`	The names of arguments given to the search function.
`...`	Consumes extra parameters that are not used by the search algorithm at hand; this function gives a warning about the ones whose value is not NULL that their value is not used.

Details

Note that this algorithm is not deterministic, as it chooses one possible path randomly when there are multiple apparently equivalent ones. In practice this means that it may return different results on different runs (including the case that it fails to converge to a solution in one run, but converges in another run). If print_info = TRUE (the default), you will see a message about "Random choices" if the algorithm needed to make random path choices.

Value

All results found by search method in a list. It raises a "Convergence failure" error if it cannot find a matched set.

`search_heuristic4` *Finds matching using depth-first search, looking ahead n steps.*

Description

In each step, it removes one subject from the set of subjects that were removed on most paths after "lookahead" steps, preferring one with the smallest associate p-value.

Usage

```
search_heuristic4(
  condition,
  covariates,
  halting_test,
  thresh,
  props,
  max_removed_per_cond,
  tiebreaker = NULL,
    min_preserved = length(levels(condition)),
    lookahead = 2,
    prefer_test = TRUE,
    print_info = TRUE,
    max_removed_per_step = 1,
    max_removed_percent_per_step = 0.5,
    ratio_for_slowdown = 0.5,
    given_args = NULL,
    ...
)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to ‘covariates‘ (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used. This is preferred among configurations with the same taken into account by the other methods to some extent. For example, c(A = 0.4, B = 0.4, C = 0.2) means that we would like the number of subjects in groups A, B, and C to be around 40%, 40%, and 20% of the total number of subjects, respectively. Whereas c("A", "B", "C") means that if possible, we would like to keep all subjects in group A, and prefer keeping subjects in B, even if it results in losing more subjects from C.
`max_removed_per_cond`	The maximum number of subjects that can be removed from each group. It must have a valid number for each group.
`tiebreaker`	NULL, or a function similar to halting_test, used to decide between cases for which halting_test yields equal values.
`min_preserved`	The minimum number of preserved subjects. It can be used to ensure that the search will not take forever to run, but instead fail when a solution is not found when preserving this number of subjects.
`lookahead`	The lookahead to use: a positive integer. It is used by the heuristic3 and heuristic4 algorithms, with a default of 2. The running time is O(N ^ lookahead), wheren N is the number of subjects.
`prefer_test`	If TRUE, prefers higher test statistic more than the expected group size proportion; default is TRUE. Used by all algorithms except exhaustive, which always

`print_info`	If TRUE, prints summary information on the input and the results, as well as progress information for the exhaustive search and random algorithms. Default: TRUE; can be changed using `set_param("PRINT_INFO", FALSE)`.
`max_removed_per_step`	The number of equivalent subjects that can be removed in each step. (The actual allowed number may be less depending on the p-value / theshold ratio.) This parameters is used by the heuristic3 and heuristic4 algorithms, with a default value of 1.
`max_removed_percent_per_step`	The percentage of remaining subjects that can be removed in each step. Used when max_removed_per_step > 1, with a default value of 0.5.
`ratio_for_slowdown`	The p-value / threshold ratio at which it starts removing subjects one by one. Used when max_removed_per_step > 1, with a default value of 0.5.
`given_args`	The names of arguments given to the search function.
`...`	Consumes extra parameters that are not used by the search algorithm at hand; this function gives a warning about the ones whose value is not NULL that their value is not used.

Details

Note that this algorithm is not deterministic, as it chooses one possible subject for removal randomly when there are multiple apparently equivalent ones. In practice it means that it may return different results on different runs (including the case that it fails to converge to a solution in one run, but converges in another run). If print_info = TRUE (the default), you will see a message about "Random choices" if the algorithm needed to make such random decisions.

Value

All results found by search method in a list. It raises a "Convergence failure" error if it cannot find a matched set.

`search_random` *Searches by randomly selecting subspaces with decreasing expected size.*

Description

Searches by randomly selecting subspaces with decreasing expected size.

Usage

```
search_random(
  condition,
  covariates,
  halting_test,
    thresh,
    props,
    max_removed_per_cond,
    tiebreaker = NULL,
    replicates,
    prefer_test = TRUE,
    print_info = TRUE,
    given_args = NULL,
    ...
  )
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`halting_test`	A function to apply to 'covariates' (in matrix form) which is TRUE iff the conditions are matched. Signature: halting_test(condition, covariates, thresh). The following halting tests are part of this package: `t_halt`, `U_halt`, `l_halt`, `ad_halt`, `ks_halt`, `wilks_halt`, `f_halt`. You can create the intersection of two or more halting tests using `create_halting_test`.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.
`props`	Either the desired proportions (percentage) of the sample for each condition as a named vector, or the names of the conditions for which we prefer to preserve the subjects, in decreasing order of preference. If not specified, the (full) sample proportions are used. This is preferred among configurations with the same taken into account by the other methods to some extent. For example, c(A = 0.4, B = 0.4, C = 0.2) means that we would like the number of subjects in groups A, B, and C to be around 40%, 40%, and 20% of the total number of subjects, respectively. Whereas c("A", "B", "C") means that if possible, we would like to keep all subjects in group A, and prefer keeping subjects in B, even if it results in losing more subjects from C.
`max_removed_per_cond`	The maximum number of subjects that can be removed from each group. It must have a valid number for each group.
`tiebreaker`	NULL, or a function similar to halting_test, used to decide between cases for which halting_test yields equal values.
`replicates`	The maximum number of random replications to be performed. This is only used for the "random" method.
`prefer_test`	If TRUE, prefers higher test statistic more than the expected group size proportion; default is TRUE. Used by all algorithms except exhaustive, which always
`print_info`	If TRUE, prints summary information on the input and the results, as well as progress information for the exhaustive search and random algorithms. Default: TRUE; can be changed using `set_param("PRINT_INFO", FALSE)`.
`given_args`	The names of arguments given to the search function.

`...`	Consumes extra parameters that are not used by the search algorithm at hand; this function gives a warning about the ones whose value is not NULL that their value is not used.

Value

All results found by search method in a list. It raises a

`set_param` *Sets value for ldamatch global parameter.*

Description

Sets value for ldamatch global parameter.

Usage

```
set_param(name, value)
```

Arguments

`name`	The name of the global parameter.
`value`	The new value of the global parameter.

Details

The names of the available parameters:

RND_DEFAULT_REPLICATES random search: default number of replicates

Anderson-Darling test parameters; see kSamples::ad.test for explanation **AD_METHOD** the method parameter for ad.test; default: asymptotic

- **AD_NSIM** the Nsim parameter for ad.test, used when AD_METHOD is 'simulated'; default: 10000
- **AD_VERSION** 1 or 2 for the two versions of the test statistic; default: 1

PRINT_INFO print summary information, and progress information for the exhaustive search algorithm

PRINT_PROGRESS whether to print progress information about parallel processing of cases

PROCESSED_CHUNK_SIZE the number of cases to be retrieved at a time from iterators for parallel processing

Value

The previous value of the global parameter.

See Also

`get_param` for retrieving the current value of a parameter.

`t_halt` *A univariate halting test using the t-test, which must be satisfied for all condition pairs.*

Description

A univariate halting test using the t-test, which must be satisfied for all condition pairs.

Usage

```
t_halt(condition, covariates, thresh)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.

Value

The ratio of the p-value and the threshold, or 0 if the p-value is less than the threshold. If there are more than two conditions, it returns the smallest value found for any condition pair.

`U_halt` *A univariate halting test using the Wilcoxon test, which must be satisfied for all condition pairs.*

Description

A univariate halting test using the Wilcoxon test, which must be satisfied for all condition pairs.

Usage

```
U_halt(condition, covariates, thresh)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.

Value

The ratio of the p-value and the threshold, or 0 if the p-value is less than the threshold. If there are more than two conditions, it returns the smallest value found for any condition pair.

`wilks_halt` *A multivariate halting test appropriate for more than two condition levels.*

Description

A multivariate halting test appropriate for more than two condition levels.

Usage

```
wilks_halt(condition, covariates, thresh)
```

Arguments

`condition`	A factor vector containing condition labels.
`covariates`	A columnwise matrix containing covariates to match the conditions on.
`thresh`	The return value of halting_test has to be greater than or equal to thresh for the matched groups.

Value

The ratio of the p-value and the threshold, or 0 if the p-value is less than the threshold.

Index