Package ‘GreedyExperimentalDesign’

July 12, 2023

**Type** Package

**Title** Greedy Experimental Design Construction

**Version** 1.5.6.1

**Date** 2023-07-11

**Description** Computes experimental designs for a two-arm experiment with covariates via a number of methods:

1. Complete randomization and randomization with forced-balance,
2. Greedily optimizing a balance objective function via pairwise switching. This optimization provides lower variance for the treatment effect estimator (and higher power) while preserving a design that is close to complete randomization. We return all iterations of the designs for use in a permutation test,
3. The second is via numerical optimization (via 'gurobi' which must be installed, see <https://www.gurobi.com/documentation/9.1/quickstart_windows/r_ins_the_r_package.html>) a la Bertsimas and Kallus,
4. Karp’s method for one covariate,
5. Exhaustive enumeration to find the optimal solution (only for small sample sizes),
6. Binary pair matching using the 'nbpMatching' library,
7. Binary pair matching plus design number (1) to further optimize balance,
8. Binary pair matching plus design number (3) to further optimize balance,
9. Hadamard designs,
10. Simultaneous Multiple Kernels.

License  GPL-3
Encoding  UTF-8
Depends  R (>= 4.1.0), rJava (>= 0.9-6)
SystemRequirements  Java (>= 7.0)
LinkingTo  Rcpp
Imports  Rcpp, checkmate, npMatching, survey, rlist, stringr, stringi,
         kernlab, graphics, grDevices, stats
URL  https://github.com/kapelner/GreedyExperimentalDesign
RoxygenNote  7.2.3
NeedsCompilation  yes
Author  Adam Kapelner [aut, cre] (<https://orcid.org/0000-0001-5985-6792>),
        David Azriel [aut],
        Abba Krieger [aut]
Maintainer  Adam Kapelner <kapelner@qc.cuny.edu>
Repository  CRAN
Date/Publication  2023-07-12 18:30:28 UTC

R topics documented:

automobile ................................................. 3
complete_randomization .................................. 4
complete_randomization_with_forced_balanced ............ 5
computeBinaryMatchStructure ............................. 5
compute_gram_matrix ...................................... 6
compute_objective_val .................................... 7
compute_randomization_metrics ........................... 7
generate_standardized_design_matrix ..................... 8
GreedyExperimentalDesign ................................ 9
greedy_orthogonalization_curation ....................... 9
greedy_orthogonalization_curation2 ..................... 10
hadamardExperimentalDesign ............................. 10
imbalanced_block_designs ................................ 11
imbalanced_complete_randomization ....................... 12
initBinaryMatchExperimentalDesignSearch ................. 12
initBinaryMatchFollowedByGreedyExperimentalDesignSearch
-initBinaryMatchFollowedByRerandomizationDesignSearch 13
initGreedyExperimentalDesignObject ..................... 14
initGreedyMultipleKernelExperimentalDesignObject .... 15
initKarpExperimentalDesignObject ....................... 16
initOptimalExperimentalDesignObject ................... 17
initRerandomizationExperimentalDesignObject ........... 18
optimize_asymmetric_treatment_assignment ................ 19
plot.greedy_experimental_design_search ................ 20
plot.greedy_multiple_kernel_experimental_design ....... 21
plot.greedy_randomization_metrics ...................... 22
Data concerning automobile prices.

Description

The automobile data frame has 201 rows and 25 columns and concerns automobiles in the 1985 Auto Imports Database. The response variable, price, is the log selling price of the automobile. There are 7 categorical predictors and 17 continuous/integer predictors which are features of the automobiles. 41 automobiles have missing data in one or more of the feature entries. This dataset is true to the original except with a few of the predictors dropped.
Usage

data(automobile)

Source


---

complete_randomization

*Implements complete randomization (without forced balance)*

---

Description

For debugging, you can use `set.seed` to be assured of deterministic output.

Usage

```r
complete_randomization(n, r, form = "one_zero")
```

Arguments

- `n`: number of observations
- `r`: number of randomized designs you would like
- `form`: Which form should it be in? The default is `one_zero` for 1/0’s or `pos_one_min_one` for +1/-1’s.

Value

- a matrix where each column is one of the `r` designs

Author(s)

Adam Kapelner
complete_randomization_with_forced_balanced

Implements forced balanced randomization

Description
For debugging, you can use set.seed to be assured of deterministic output.

Usage
complete_randomization_with_forced_balanced(n, r, form = "one_zero")

Arguments
n
number of observations
r
number of randomized designs you would like
form
Which form should it be in? The default is one_zero for 1/0’s or pos_one_min_one for +1/-1’s.

Value
a matrix where each column is one of the r designs

Author(s)
Adam Kapelner

computeBinaryMatchStructure

Compute Binary Matching Structure

Description
This method creates an object of type binary_match_structure and will compute pairs. You can then use the functions initBinaryMatchExperimentalDesignSearch and resultsBinaryMatchSearch to create randomized allocation vectors. For one column in X, we just sort to find the pairs trivially.

Usage
computeBinaryMatchStructure(
  X,
  mahal_match = FALSE,
  compute_dist_matrix = NULL,
  D = NULL
)
compute_gram_matrix

**Arguments**

- **X**
  
  The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). This is the design matrix you wish to search for a more optimal design.

- **mahal_match**
  
  Match using Mahalanobis distance. Default is FALSE.

- **compute_dist_matrix**
  
  The function that computes the distance matrix between every two observations in $X$, its only argument. The default is NULL signifying euclidean squared distance optimized in C++.

- **D**
  
  A distance matrix precomputed. The default is NULL indicating the distance matrix should be computed.

**Value**

An object of type `binary_experimental_design` which can be further operated upon.

**Author(s)**

Adam Kapelner

---

**compute_gram_matrix**  
**Gram Matrix Computation**

**Description**

Computes the Gram Matrix for a user-specified kernel using the library `kernlab`. Note that this function automatically standardizes the columns of the data entered.

**Usage**

```r
compute_gram_matrix(X, kernel_type, params = c())
```

**Arguments**

- **X**
  
  The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). This is the design matrix you wish to search for a more optimal design.

- **kernel_type**
  
  One of the following: "vanilla", "rbf", "poly", "tanh", "bessel", "laplace", "anova" or "spline".

- **params**
  
  A vector of numeric parameters. Each `kernel_type` has different numbers of parameters required. For more information see documentation for the `kernlab` library.

**Value**

The $n \times n$ gram matrix for the given kernel on the given data.
compute_objective_val

Author(s)
Adam Kapelner

compute_objective_val  Computes Objective Value From Allocation Vector

Description
Returns the objective value given a design vector as well as an objective function. This is sometimes duplicated in Java. However, within Java, tricks are played to make optimization go faster so Java’s objective values may not always be the same as the true objective function (e.g. logs or constants dropped).

Usage
compute_objective_val(X, indic_T, objective = "abs_sum_diff", inv_cov_X = NULL)

Arguments
X  The n x p design matrix
indic_T  The n-length binary allocation vector
objective  The objective function to use. Default is abs_sum_diff and the other option is mahal_dist.
inv_cov_X  Optional: the inverse sample variance covariance matrix. Use this argument if you will be doing many calculations since passing this in will cache this data.

Author(s)
Adam Kapelner

compute_randomization_metrics

Computes Randomization Metrics (explained in paper) about a design algorithm

Description
Computes Randomization Metrics (explained in paper) about a design algorithm

Usage
compute_randomization_metrics(designs)
Arguments
designs A matrix where each column is one design.

Value
A list of resulting data: the probability estimates for each pair in the design of randomness where estimates close to ~0.5 represent random assignment, then the entropy metric the distance metric, the maximum eigenvalue of the allocation var-cov matrix (operator norm) and the squared Frobenius norm (the sum of the squared eigenvalues)

Author(s)
Adam Kapelner
GreedyExperimentalDesign

Greedy Experimental Design Search

Description

A tool to find many types of a priori experimental designs

Author(s)

Adam Kapelner &lt;kapelner@qc.cuny.edu&gt;

References

Kapelner, A

greedy_orthogonalization_curation

Curate More Orthogonal Vectors Greedily

Description

This function takes a set of allocation vectors and pares them down one-by-one by eliminating the vector that can result in the largest reduction in \( \text{Avg} \{ |r_{ij}| \} \). It is recommended to begin with a set of unmirrored vectors for speed. Then add the mirrors later for whichever subset you wish.

Usage

\texttt{greedy_orthogonalization\_curation(W, Rmin = 2, verbose = FALSE)}

Arguments

\begin{itemize}
  \item \textit{W} \hspace{1cm} A matrix in \(-1, 1^R \times n\) which have \( R \) allocation vectors for an experiment of sample size \( n \).
  \item \textit{Rmin} \hspace{1cm} The minimum number of vectors to consider in a design. The default is the true bottom, two.
  \item \textit{verbose} \hspace{1cm} Default is FALSE but if not, it will print out a message for each iteration.
\end{itemize}

Value

A list with two elements: (1) \texttt{avg\_abs\_rij\_by\_R} which is a data frame with \( R - \text{Rmin} + 1 \) rows and columns \( R \) and average absolute \( r_{ij} \) and (2) \texttt{Wsorted} which provides the collection of vectors in sorted by best average absolute \( r_{ij} \) in row order from best to worst.

Author(s)

Adam Kapelner
**greedy_orthogonalization_curation2**

*Curate More Orthogonal Vectors Greedily*

**Description**

This function takes a set of allocation vectors and pare them down one-by-one by eliminating the vector that can result in the largest reduction in Avg[|r_ij|]. It is recommended to begin with a set of unmirrored vectors for speed. Then add the mirrors later for whichever subset you wish.

**Usage**

```r
greedy_orthogonalization_curation2(W, R0 = 100, verbose = FALSE)
```

**Arguments**

- `W`: A matrix in $-1, 1^R x n$ which have R allocation vectors for an experiment of sample size n.
- `R0`: The minimum number of vectors to consider in a design. The default is the true bottom, two.
- `verbose`: Default is `FALSE` but if not, it will print out a message for each iteration.

**Value**

A list with two elements: (1) `avg_abs_rij_by_R` which is a data frame with $R - R_{min} + 1$ rows and columns R and average absolute r_ij and (2) `Wsorted` which provides the collection of vectors in sorted by best average absolute r_ij in row order from best to worst.

**Author(s)**

Adam Kapelner

---

**hadamardExperimentalDesign**

*Create a Hadamard Design*

**Description**

This method returns unique designs according to a Hadamard matrix. For debugging, you can use `set.seed` to be assured of deterministic output.

**Usage**

```r
hadamardExperimentalDesign(X, strict = TRUE, form = "zero_one")
```
imbalanced_block_designs

Arguments

- **X**
  The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). The measurements aren’t used to compute the Hadamard designs, only the number of rows.
- **strict**
  Hadamard matrices are not available for all $n$.
- **form**
  Which form should it be in? The default is one_zero for 1/0’s or pos_one_min_one for +1/-1’s.

Value

An matrix of dimension $R \times n$ where $R$ is the number of Hadamard allocations.

Author(s)

Adam Kapelner

---

**imbalanced_block_designs**

*Implements unequally allocated block designs*

Description

For debugging, you can use `set.seed` to be assured of deterministic output. The following quantities in this design must be integer valued or an error will be thrown: $n_B := n / B$ and $n_B \times prop_T$

Usage

```
imbalanced_block_designs(n, prop_T, B, r, form = "one_zero")
```

Arguments

- **n**
  number of observations
- **prop_T**
  the proportion of treatments needed
- **B**
  the number of blocks
- **r**
  number of randomized designs you would like
- **form**
  Which form should it be in? The default is one_zero for 1/0’s or pos_one_min_one for +1/-1’s.

Value

A matrix where each column is one of the $r$ designs

Author(s)

Adam Kapelner
imbalanced_complete_randomization

*Implements unequally allocated complete randomization*

**Description**

For debugging, you can use `set.seed` to be assured of deterministic output.

**Usage**

```r
imbalanced_complete_randomization(n, prop_T, r, form = "one_zero")
```

**Arguments**

- `n`: number of observations
- `prop_T`: the proportion of treatments needed
- `r`: number of randomized designs you would like
- `form`: Which form should it be in? The default is `one_zero` for 1/0’s or `pos_one_min_one` for +1/-1’s.

**Value**

A matrix where each column is one of the `r` designs

**Author(s)**

Adam Kapelner

---

`initBinaryMatchExperimentalDesignSearch`

*Begin a Binary Match Search*

**Description**

This method creates an object of type pairwise_matching_experimental_design_search and will immediately initiate a search through $1_T$ space for pairwise match designs based on the structure computed in the function `computeBinaryMatchStructure`. For debugging, you can use set the `seed` parameter and `num_cores = 1` to be assured of deterministic output.
initBinaryMatchFollowedByGreedyExperimentalDesignSearch

Begin a Search for Binary Matching Followed by Greedy Switch Designs

Description

This method creates an object of type binary_then_greedy_experimental_design and will find optimal matched pairs which are then greedily switched in order to further minimize a balance metric. You can then use the function resultsBinaryMatchThenGreedySearch to obtain the randomized allocation vectors. For one column in X, the matching just sorts the values to find the pairs trivially.

Usage

initBinaryMatchExperimentalDesignSearch(
  binary_match_structure,
  max_designs = 1000,
  wait = FALSE,
  start = TRUE,
  num_cores = 1,
  seed = NULL,
  prop_flips = 1
)

Arguments

  binary_match_structure
    The binary_experimental_design object where the pairs are computed.
  max_designs
    How many random allocation vectors you wish to return. The default is 1000.
  wait
    Should the R terminal hang until all max_designs vectors are found? The default is FALSE.
  start
    Should we start searching immediately (default is TRUE).
  num_cores
    The number of CPU cores you wish to use during the search. The default is 1.
  seed
    The set to set for deterministic output. This should only be set if num_cores = 1 otherwise the output will not be deterministic. Default is NULL for no seed set.
  prop_flips
    Proportion of flips. Default is all. Lower for more correlated assignments (useful for research only).

Author(s)

Adam Kapelner
Usage

initBinaryMatchFollowedByRerandomizationDesignSearch(
    X,
    diff_method = FALSE,
    compute_dist_matrix = NULL,
    ...
)

Arguments

X
The design matrix with \(n\) rows (one for each subject) and \(p\) columns (one for each measurement on the subject). This is the design matrix you wish to search for a more optimal design.

diff_method
Once the subjects (i.e. row vectors) are paired, do we create a set of \(n/2\) difference vectors and feed that into greedy? If TRUE, this technically breaks the objective function, but it is shown to have better performance. The default is thus FALSE.

compute_dist_matrix
The function that computes the distance matrix between every two observations in \(X\), its only argument. The default is NULL signifying euclidean squared distance optimized in C++.

...
Arguments passed to initGreedyExperimentalDesignObject. It is recommended to set max_designs otherwise it will default to 10,000.

Value

An object of type binary_experimental_design which can be further operated upon.

Author(s)

Adam Kapelner

Description

This method creates an object of type binary_then_rerandomization_experimental_design and will find optimal matched pairs which are then rerandomized in order to further minimize a balance metric. You can then use the function resultsBinaryMatchThenRerandomizationDesignSearch to obtain the randomized allocation vectors. For one column in \(X\), the matching just sorts the values to find the pairs trivially.
**Usage**

```r
initBinaryMatchFollowedByRerandomizationDesignSearch(
  X,
  compute_dist_matrix = NULL,
  ...
)
```

**Arguments**

- `X` The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). This is the design matrix you wish to search for a more optimal design.
- `compute_dist_matrix` The function that computes the distance matrix between every two observations in `X`, its only argument. The default is `NULL` signifying euclidean squared distance optimized in C++.
- `...` Arguments passed to `initGreedyExperimentalDesignObject`. It is recommended to set `max_designs` otherwise it will default to 10,000.

**Value**

An object of type `binary_experimental_design` which can be further operated upon.

**Author(s)**

Adam Kapelner

---

**initGreedyExperimentalDesignObject**

*Begin A Greedy Pair Switching Search*

**Description**

This method creates an object of type `greedy_experimental_design` and will immediately initiate a search through $1_T$ space for forced balance designs. For debugging, you can use set the `seed` parameter and `num_cores = 1` to be assured of deterministic output.

**Usage**

```r
initGreedyExperimentalDesignObject(
  X = NULL,
  nT = NULL,
  max_designs = 10000,
  objective = "mahal_dist",
  indicies_pairs = NULL,
  Kgram = NULL,
)```
wait = FALSE,
start = TRUE,
max_iters = Inf,
semigreedy = FALSE,
diagnostics = FALSE,
num_cores = 1,
seed = NULL
)

Arguments

X
The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). This is the design matrix you wish to search for a more optimal design. This parameter must be specified unless you choose objective type "kernel" in which case, the Kgram parameter must be specified.

nT
The number of treatments to assign. Default is NULL which is for forced balance allocation i.e. $nT = nC = n / 2$ where $n$ is the number of rows in $X$ (or Kgram if $X$ is unspecified).

max_designs
The maximum number of designs to be returned. Default is 10,000. Make this large so you can search however long you wish as the search can be stopped at any time by using the stopSearch method.

objective
The objective function to use when searching design space. This is a string with valid values "mahal_dist" (the default), "abs_sum_diff" or "kernel".

indicies_pairs
A matrix of size $n/2$ times 2 whose rows are indicies pairs. The values of the entire matrix must enumerate all indicies $1, ..., n$. The default is NULL meaning to use all possible pairs.

Kgram
If the objective = kernel, this argument is required to be an $n x n$ matrix whose entries are the evaluation of the kernel function between subject $i$ and subject $j$. Default is NULL.

wait
Should the R terminal hang until all max_designs vectors are found? The default is FALSE.

start
Should we start searching immediately (default is TRUE).

max_iters
Should we impose a maximum number of greedy switches? The default is Inf which a flag for "no limit."

semigreedy
Should we use a fully greedy approach or the quicker semi-greedy approach? The default is FALSE corresponding to the fully greedy approach.

diagnostics
Returns diagnostic information about the iterations including (a) the initial starting vectors, (b) the switches at every iteration and (c) information about the objective function at every iteration (default is FALSE to decrease the algorithm’s run time).

num_cores
The number of CPU cores you wish to use during the search. The default is 1.

seed
The set to set for deterministic output. This should only be set if num_cores = 1 otherwise the output will not be deterministic. Default is NULL for no seed set.
Value

An object of type greedy_experimental_design_search which can be further operated upon

Author(s)

Adam Kapelner

Examples

```r
## Not run:
library(MASS)
data(Boston)
#pretend the Boston data was an experiment setting
#first pull out the covariates
X = Boston[, 1:13]
#begin the greedy design search
ged = initGreedyExperimentalDesignObject(X,
  max_designs = 1000, num_cores = 3, objective = "abs_sum_diff")
#wait
ged
## End(Not run)
```

Description

This method creates an object of type greedy_multiple_kernel_experimental_design and will immediately initiate a search through $T_1$ space for forced balance designs. For debugging, you can use set the seed parameter and num_cores = 1 to be assured of deterministic output.

Usage

```r
initGreedyMultipleKernelExperimentalDesignObject(
  X = NULL,
  max_designs = 10000,
  objective = "added_pct_reduction",
  kernel_pre_num_designs = 2000,
  kernel_names = NULL,
  Kgrams = NULL,
  maximum_gain_scaling = 1.1,
  kernel_weights = NULL,
  wait = FALSE,
  start = TRUE,
  max_iters = Inf,
  semigreedy = FALSE,
)```
diagnostics = FALSE,
num_cores = 1,
seed = NULL
)

Arguments

X
The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). This is the design matrix you wish to search for a more optimal design. We will standardize this matrix by column internally.

max_designs
The maximum number of designs to be returned. Default is 10,000. Make this large so you can search however long you wish as the search can be stopped at any time by using the `stopSearch` method.

objective
The method used to aggregate the kernel objective functions together. Default is "added_pct_reduction".

kernel_pre_num_designs
How many designs per kernel to run to explore the space of kernel objective values. Default is 2000.

kernel_names
An array with the kernels to compute with default parameters. Must have elements in the following set: "mahalanobis", "poly_s" where the "s" is a natural number 1 or greater, "exponential", "laplacian", "inv_mult_quad", "gaussian". Default is NULL to indicate the kernels are specified manually using the Kgrams parameter.

Kgrams
A list of M >= 1 elements where each is a $n \times n$ matrix whose entries are the evaluation of the kernel function between subject i and subject j. Default is NULL to indicate this was specified using the convenience parameter kernel_names.

maximum_gain_scaling
This controls how much the percentage of possible improvement on a kernel objective function should be scaled by. The minimum is 1 which allows for designs that could potentially have >=100 improvement over original. We recommend 1.1 which means that a design that was found to be the best of the kernel_pre_num_designs still has 1/1.1 = 9% room to grow making it highly unlikely that any design could be >= 100%.

kernel_weights
A vector with positive weights (need not be normalized) where each element represents the weight of each kernel. The default is NULL for uniform weighting.

wait
Should the R terminal hang until all max_designs vectors are found? The default is FALSE.

start
Should we start searching immediately (default is TRUE).

max_iters
Should we impose a maximum number of greedy switches? The default is Inf which a flag for "no limit."

semigreedy
Should we use a fully greedy approach or the quicker semi-greedy approach? The default is FALSE corresponding to the fully greedy approach.

diagnostics
Returns diagnostic information about the iterations including (a) the initial starting vectors, (b) the switches at every iteration and (c) information about the objective function at every iteration (default is FALSE to decrease the algorithm’s run time).
The number of CPU cores you wish to use during the search. The default is 1.

The set to set for deterministic output. This should only be set if `num_cores = 1` otherwise the output will not be deterministic. Default is NULL for no seed set.

Value

An object of type `greedy_experimental_design_search` which can be further operated upon

Author(s)

Adam Kapelner

Examples

```r
## Not run:
library(MASS)
data(Boston)
#pretend the Boston data was an experiment setting
#first pull out the covariates
X = Boston[, 1:13]
#begin the greedy design search
ged = initGreedyMultipleKernelExperimentalDesignObject(X,
  max_designs = 1000, num_cores = 3, kernel_names = c("mahalanobis", "gaussian"))
#wait
ged

## End(Not run)
```
Arguments

X

The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). This is the design matrix you wish to search for a more karp design.

wait

Should the R terminal hang until all `max_designs` vectors are found? The default is `FALSE`.

balanced

Should the final vector be balanced? Default and recommended is `TRUE`.

start

Should we start searching immediately (default is `TRUE`).

Value

An object of type `karp_experimental_design_search` which can be further operated upon.

Author(s)

Adam Kapelner

Description

This method creates an object of type `optimal_experimental_design` and will immediately initiate a search through $1_T$ space. Since this search takes exponential time, for most machines, this method is futile beyond 28 samples. You've been warned! For debugging, you can use set `num_cores = 1` to be assured of deterministic output.

Usage

```r
initOptimalExperimentalDesignObject(
  X = NULL,
  objective = "mahal_dist",
  Kgram = NULL,
  wait = FALSE,
  start = TRUE,
  num_cores = 1
)
```

Arguments

X

The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). This is the design matrix you wish to search for a more optimal design.

objective

The objective function to use when searching design space. This is a string with valid values "mahal_dist" (the default), "abs_sum_diff" or "kernel".
Kgram

If the objective = kernel, this argument is required to be an $n \times n$ matrix whose entries are the evaluation of the kernel function between subject $i$ and subject $j$. Default is NULL.

wait

Should the R terminal hang until all max_designs vectors are found? The default is FALSE.

start

Should we start searching immediately (default is TRUE).

num_cores

The number of CPU cores you wish to use during the search. The default is 1.

Value

An object of type optimal_experimental_design_search which can be further operated upon

Author(s)

Adam Kapelner

initRerandomizationExperimentalDesignObject

Begin a Rerandomization Search

Description

This method creates an object of type rerandomization_experimental_design and will immediately initiate a search through $1_T$ space for forced-balance designs. For debugging, you can use set the seed parameter and num_cores = 1 to be assured of deterministic output.

Usage

initRerandomizationExperimentalDesignObject(
    X = NULL,
    obj_val_cutoff_to_include,
    max_designs = 1000,
    objective = "mahal_dist",
    Kgram = NULL,
    wait = FALSE,
    start = TRUE,
    num_cores = 1,
    seed = NULL
)

Arguments

X

The design matrix with $n$ rows (one for each subject) and $p$ columns (one for each measurement on the subject). This is the design matrix you wish to search for a more optimal design.
optimize_asymmetric_treatment_assignment

obj_val_cutoff_to_include
   Only allocation vectors with objective values lower than this threshold will be returned. If the
cutoff is infinity, you are doing BCRD and you should use the
complete_randomization_with_forced_balanced function instead.

max_designs
   The maximum number of designs to be returned. Default is 10,000. Make this
large so you can search however long you wish as the search can be stopped at
any time by using the stopSearch method.

objective
   The objective function to use when searching design space. This is a string with
valid values "mahal_dist" (the default), "abs_sum_diff" or "kernel".

Kgram
   If the objective = kernel, this argument is required to be an n x n matrix
whose entries are the evaluation of the kernel function between subject i and
subject j. Default is NULL.

wait
   Should the R terminal hang until all max_designs vectors are found? The default
is FALSE.

start
   Should we start searching immediately (default is TRUE).

num_cores
   The number of CPU cores you wish to use during the search. The default is 1.

seed
   The set to set for deterministic output. This should only be set if num_cores = 1
otherwise the output will not be deterministic. Default is NULL for no seed set.

Value

An object of type rerandomization_experimental_design_search which can be further oper-
ated upon.

Author(s)

Adam Kapelner

optimize_asymmetric_treatment_assignment

Compute Optimal Number of Treatments/Controls

Description

Given a total budget and asymmetric treatment and control costs, calculate the number of treatments
and controls that optimize the variance of the estimator. The number of treatments is rounded up by
default.

Usage

optimize_asymmetric_treatment_assignment(
   c_treatment = NULL,
   c_control = NULL,
   c_total_max = NULL,
   n = NULL
)


plot.greedy_experimental_design_search

Arguments

c_treatment    The cost of a treatment assignment. Default is NULL for symmetric costs.
c_control     The cost of a control assignment. Default is NULL for symmetric costs.
c_total_max   The total cost constraint of any allocation. Either this or \( n \) must be specified. Default is NULL.
\( n \)         The total cost constraint as specified by the total number of subjects. Either this or \( c_{\text{total max}} \) must be specified. Default is NULL.

Value

A list with three keys: \( n \), \( n_T \), \( n_C \) plus specified arguments

Author(s)

Adam Kapelner

Examples

## Not run:
optimize_asymmetric_treatment_assignment(n = 100)
# nT = nC = 50
optimize_asymmetric_treatment_assignment(n = 100, c_treatment = 2, c_control = 1)
# nT = 66, nC = 34
optimize_asymmetric_treatment_assignment(c_total_max = 50, c_treatment = 2, c_control = 1)

## End(Not run)

plot.greedy_experimental_design_search

Plots a summary of a greedy search object object

Description

Plots a summary of a greedy search object object

Usage

## S3 method for class 'greedy_experimental_design_search'
plot(x, ...)

Arguments

\( x \)            The greedy search object object to be summarized in the plot
\ldots             Other parameters to pass to the default plot function

Value

An array of order statistics from \texttt{plot_obj_val_order_statistic} as a list element
Author(s)

Adam Kapelner

---

plot.greedy_multiple_kernel_experimental_design

*Plots a summary of a greedy_multiple_kernel_experimental_design object*

---

Description

Plots a summary of a greedy_multiple_kernel_experimental_design object

Usage

```r
## S3 method for class 'greedy_multiple_kernel_experimental_design'
plot(x, ...)
```

Arguments

- `x` The greedy_multiple_kernel_experimental_design object to be summarized in the plot
- `...` Other parameters to pass to the default plot function

Value

An array of order statistics from `plot_obj_val_order_statistic` as a list element

Author(s)

Adam Kapelner

---

plot_obj_val_by_iter

*Plots the objective value by iteration*

---

Description

Plots the objective value by iteration

Usage

```r
plot_obj_val_by_iter(res, runs = NULL)
```
Arguments

- **res**
  Results from a greedy search object

- **runs**
  A vector of run indices you would like to see plotted (default is to plot the first up to 9)

Author(s)

Adam Kapelner

---

**plot_obj_val_order_statistic**

*Plots an order statistic of the object value as a function of number of searches*

Description

Plots an order statistic of the object value as a function of number of searches

Usage

```r
plot_obj_val_order_statistic(
  obj,
  order_stat = 1,
  skip_every = 5,
  type = "o",
  ...
)
```

Arguments

- **obj**
  The greedy search object object whose search history is to be visualized

- **order_stat**
  The order statistic that you wish to plot. The default is 1 for the minimum.

- **skip_every**
  Plot every nth point. This makes the plot generate much more quickly. The default is 5.

- **type**
  The type parameter for plot.

- **...**
  Other arguments to be passed to the plot function.

Value

An array of order statistics as a list element

Author(s)

Adam Kapelner
print.binary_match_structure

Prints a summary of a binary_match_structure object

Description

Prints a summary of a binary_match_structure object

Usage

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

Arguments

x The binary_match_structure object to be summarized in the console
...

Other parameters to pass to the default print function

Author(s)

Adam Kapelner

print.binary_then_greedy_experimental_design

Prints a summary of a binary_then_greedy_experimental_design object

Description

Prints a summary of a binary_then_greedy_experimental_design object

Usage

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

Arguments

x The binary_then_greedy_experimental_design object to be summarized in the console
...

Other parameters to pass to the default print function

Author(s)

Adam Kapelner
print.binary_then_rerandomization_experimental_design

Prints a summary of a binary_then_rerandomization_experimental_design object

Description
Prints a summary of a binary_then_rerandomization_experimental_design object

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

Arguments
x The binary_then_rerandomization_experimental_design object to be summarized in the console
... Other parameters to pass to the default print function

Author(s)
Adam Kapelner

print.greedy_experimental_design_search

Prints a summary of a greedy_experimental_design_search object

Description
Prints a summary of a greedy_experimental_design_search object

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

Arguments
x The greedy_experimental_design_search object to be summarized in the console
... Other parameters to pass to the default print function

Author(s)
Adam Kapelner
print.greedy_multiple_kernel_experimental_design

*Prints a summary of a greedy_multiple_kernel_experimental_design object*

Description

Prints a summary of a greedy_multiple_kernel_experimental_design object

Usage

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

Arguments

- `x` The greedy_multiple_kernel_experimental_design object to be summarized in the console
- `...` Other parameters to pass to the default print function

Author(s)

Adam Kapelner

print.karp_experimental_design_search

*Prints a summary of a karp_experimental_design_search object*

Description

Prints a summary of a karp_experimental_design_search object

Usage

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

Arguments

- `x` The karp_experimental_design_search object to be summarized in the console
- `...` Other parameters to pass to the default print function

Author(s)

Adam Kapelner
print.optimal_experimental_design_search

Prints a summary of a optimal_experimental_design_search object

Description
Prints a summary of a optimal_experimental_design_search object

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

Arguments
x The optimal_experimental_design_search object to be summarized in the console

... Other parameters to pass to the default print function

Author(s)
Adam Kapelner

print.pairwise_matching_experimental_design_search

Prints a summary of a pairwise_matching_experimental_design_search object

Description
Prints a summary of a pairwise_matching_experimental_design_search object

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

Arguments
x The pairwise_matching_experimental_design_search object to be summarized in the console

... Other parameters to pass to the default print function

Author(s)
Adam Kapelner
print.rerandomization_experimental_design_search

Prints a summary of a rerandomization_experimental_design_search object

Description

Prints a summary of a rerandomization_experimental_design_search object

Usage

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

Arguments

x
The rerandomization_experimental_design_search object to be summarized in the console

...
Other parameters to pass to the default print function

Author(s)

Adam Kapelner

resultsBinaryMatchSearch

Binary Pair Match Search

Description

Returns the results (thus far) of the binary pair match design search

Usage

resultsBinaryMatchSearch(obj, form = "one_zero")

Arguments

obj
The pairwise_matching_experimental_design_search object that is currently running the search

form
Which form should the assignments be in? The default is one_zero for 1/0’s or pos_one_min_one for +1/-1’s.

Author(s)

Adam Kapelner
resultsBinaryMatchThenGreedySearch

Returns unique allocation vectors that are binary matched

Description

Returns unique allocation vectors that are binary matched

Usage

resultsBinaryMatchThenGreedySearch(
    obj,
    num_vectors = NULL,
    compute_obj_vals = FALSE,
    form = "zero_one"
)

Arguments

obj The binary_then_greedy_experimental_design object where the pairs are computed.
num_vectors How many random allocation vectors you wish to return. The default is NULL indicating you want all of them.
compute_obj_vals Should we compute all the objective values for each allocation? Default is FALSE.
form Which form should it be in? The default is one_zero for 1/0’s or pos_one_min_one for +1/-1’s.

Author(s)

Adam Kapelner

resultsBinaryMatchThenRerandomizationSearch

Returns unique allocation vectors that are binary matched

Description

Returns unique allocation vectors that are binary matched
resultsGreedySearch

Usage

resultsBinaryMatchThenRerandomizationSearch(
    obj,
    num_vectors = NULL,
    compute_obj_vals = FALSE,
    form = "zero_one"
)

Arguments

obj

The binary_then_greedy_experimental_design object where the pairs are computed.

num_vectors

How many random allocation vectors you wish to return. The default is NULL indicating you want all of them.

compute_obj_vals

Should we compute all the objective values for each allocation? Default is FALSE.

form

Which form should it be in? The default is one_zero for 1/0’s or pos_one_min_one for +1/-1’s.

Author(s)

Adam Kapelner

resultsGreedySearch  Returns the results (thus far) of the greedy design search

Description

Returns the results (thus far) of the greedy design search

Usage

resultsGreedySearch(obj, max_vectors = 9, form = "one_zero")

Arguments

obj

The greedy_experimental_design object that is currently running the search

max_vectors

The number of design vectors you wish to return. NULL returns all of them. This is not recommended as returning over 1,000 vectors is time-intensive. The default is 9.

form

Which form should it be in? The default is one_zero for 1/0’s or pos_one_min_one for +1/-1’s.

Author(s)

Adam Kapelner
Examples

```r
## Not run:
library(MASS)
data(Boston)
# pretend the Boston data was an experiment setting
# first pull out the covariates
X = Boston[, 1:13]
# begin the greedy design search
ged = initGreedyExperimentalDesignObject(X,
  max_designs = 1000, num_cores = 2, objective = "abs_sum_diff")
# wait
res = resultsGreedySearch(ged, max_vectors = 2)
design = res$ending_indicTs[, 1] # ordered already by best-->worst
design
# what is the balance on this vector?
res$obj_vals[1]
# compute balance explicitly in R to double check
compute_objective_val(X, design) # same as above
# how far have we come?
ged
# we can cut it here
stopSearch(ged)
```

## End(Not run)

---

resultsKarpSearch  
*Returns the results (thus far) of the karp design search*

Description

Returns the results (thus far) of the karp design search

Usage

```r
resultsKarpSearch(obj)
```

Arguments

- **obj**  
The `karp_experimental_design` object that is currently running the search

Author(s)

Adam Kapelner
resultsMultipleKernelGreedySearch

Returns the results (thus far) of the greedy design search for multiple kernels

Description

Returns the results (thus far) of the greedy design search for multiple kernels

Usage

resultsMultipleKernelGreedySearch(obj, max_vectors = 9, form = "one_zero")

Arguments

obj
The greedy_multiple_kernel_experimental_design object that is currently running the search

max_vectors
The number of design vectors you wish to return. NULL returns all of them. This is not recommended as returning over 1,000 vectors is time-intensive. The default is 9.

form
Which form should it be in? The default is one_zero for 1/0’s or pos_one_min_one for +1/-1’s.

Author(s)

Adam Kapelner

Examples

## Not run:
library(MASS)
data(Boston)
#pretend the Boston data was an experiment setting
#first pull out the covariates
X = Boston[, 1 : 13]
#begin the greedy design search
ged = initGreedyMultipleKernelExperimentalDesignObject(X,
max_designs = 1000, num_cores = 3, kernel_names = c("mahalanobis", "gaussian"))
#wait
res = resultsMultipleKernelGreedySearch(ged, max_vectors = 2)
design = res$ending_indicTs[, 1] #ordered already by best-->worst
design
#how far have we come of the 1000 we set out to do?
ged
#we can cut it here
stopSearch(ged)

## End(Not run)
resultsOptimalSearch  
*Returns the results (thus far) of the optimal design search*

**Description**

Returns the results (thus far) of the optimal design search

**Usage**

```r
resultsOptimalSearch(obj, num_vectors = 2, form = "one_zero")
```

**Arguments**

- `obj`  
The optimal_experimental_design object that is currently running the search
- `num_vectors`  
How many allocation vectors you wish to return. The default is 1 meaning the best vector. If `Inf`, it means all vectors.
- `form`  
Which form should it be in? The default is `one_zero` for 1/0's or `pos_one_min_one` for +1/-1's.

**Author(s)**

Adam Kapelner

---

resultsRerandomizationSearch  
*Returns the results (thus far) of the rerandomization design search*

**Description**

Returns the results (thus far) of the rerandomization design search

**Usage**

```r
resultsRerandomizationSearch(
  obj,
  include_assignments = FALSE,
  form = "one_zero"
)
```

**Arguments**

- `obj`  
The rerandomization_experimental_design object that is currently running the search
- `include_assignments`  
Do we include the assignments (takes time) and default is `FALSE`.
- `form`  
Which form should the assignments be in? The default is `one_zero` for 1/0's or `pos_one_min_one` for +1/-1's.
searchTimeElapsed  
*Returns the amount of time elapsed*

**Description**

Returns the amount of time elapsed

**Usage**

searchTimeElapsed(obj)

**Arguments**

- **obj**  
The experimental_design object that is currently running the search

**Author(s)**

Adam Kapelner

---

`standardize_data_matrix`

*Standardizes the columns of a data matrix.*

**Description**

Standardizes the columns of a data matrix.

**Usage**

`standardize_data_matrix(X)`

**Arguments**

- **X**  
The n x p design matrix

**Value**

The n x p design matrix with columns standardized

**Author(s)**

Adam Kapelner
**startSearch**

*Starts the parallelized greedy design search.*

**Description**

Once begun, this function cannot be run again.

**Usage**

```
startSearch(obj)
```

**Arguments**

- `obj` The `experimental_design` object that will be running the search

**Author(s)**

Adam Kapelner

---

**stopSearch**

*Stops the parallelized greedy design search.*

**Description**

Once stopped, it cannot be restarted.

**Usage**

```
stopSearch(obj)
```

**Arguments**

- `obj` The `experimental_design` object that is currently running the search

**Author(s)**

Adam Kapelner
### summary.binary_match_structure

*Prints a summary of a binary_match_structure object*

**Description**

Prints a summary of a binary_match_structure object

**Usage**

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

**Arguments**

- `object` The binary_match_structure object to be summarized in the console
- `...` Other parameters to pass to the default summary function

**Author(s)**

Adam Kapelner

---

### summary.binary_then_greedy_experimental_design

*Prints a summary of a binary_then_greedy_experimental_design object*

**Description**

Prints a summary of a binary_then_greedy_experimental_design object

**Usage**

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

**Arguments**

- `object` The binary_then_greedy_experimental_design object to be summarized in the console
- `...` Other parameters to pass to the default summary function

**Author(s)**

Adam Kapelner
**Summary.binary_then_rerandomization_experimental_design**

Prints a summary of a binary_then_rerandomization_experimental_design object

**Description**

Prints a summary of a binary_then_rerandomization_experimental_design object

**Usage**

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

**Arguments**

- `object` The binary_then_rerandomization_experimental_design object to be summarized in the console
- `...` Other parameters to pass to the default summary function

**Author(s)**

Adam Kapelner

---

**Summary.greedy_experimental_design_search**

Prints a summary of a greedy_experimental_design_search object

**Description**

Prints a summary of a greedy_experimental_design_search object

**Usage**

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

**Arguments**

- `object` The greedy_experimental_design_search object to be summarized in the console
- `...` Other parameters to pass to the default summary function

**Author(s)**

Adam Kapelner
summary.greedy_multiple_kernel_experimental_design

Prints a summary of a greedy_multiple_kernel_experimental_design object

Description

Prints a summary of a greedy_multiple_kernel_experimental_design object

Usage

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

Arguments

object The greedy_multiple_kernel_experimental_design object to be summarized in the console
...

Other parameters to pass to the default summary function

Author(s)

Adam Kapelner

summary.karp_experimental_design_search

Prints a summary of a karp_experimental_design_search object

Description

Prints a summary of a karp_experimental_design_search object

Usage

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

Arguments

object The karp_experimental_design_search object to be summarized in the console
...

Other parameters to pass to the default summary function

Author(s)

Adam Kapelner
**summary.optimal_experimental_design_search**

*Prints a summary of a optimal_experimental_design_search object*

**Description**

Prints a summary of a optimal_experimental_design_search object

**Usage**

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

**Arguments**

- `object` The optimal_experimental_design_search object to be summarized in the console
- `...` Other parameters to pass to the default summary function

**Author(s)**

Adam Kapelner

---

**summary.pairwise_matching_experimental_design_search**

*Prints a summary of a pairwise_matching_experimental_design_search object*

**Description**

Prints a summary of a pairwise_matching_experimental_design_search object

**Usage**

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

**Arguments**

- `object` The pairwise_matching_experimental_design_search object to be summarized in the console
- `...` Other parameters to pass to the default summary function

**Author(s)**

Adam Kapelner
summary.rerandomization_experimental_design_search

Prints a summary of a rerandomization_experimental_design_search object

Description

Prints a summary of a rerandomization_experimental_design_search object

Usage

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

Arguments

object The rerandomization_experimental_design_search object to be summarized in the console
...
Other parameters to pass to the default summary function

Author(s)

Adam Kapelner
Index

* datasets
  automobile, 3
* design
  GreedyExperimentalDesign, 9
* optimize
  GreedyExperimentalDesign, 9
automobile, 3
complete_randomization, 4
complete_randomization_with_forced_balanced, 5
compute_gram_matrix, 6
compute_objective_val, 7
compute_randomization_metrics, 7
computeBinaryMatchStructure, 5
generate_stdzied_design_matrix, 8
greedy_orthogonalization_curation, 9
greedy_orthogonalization_curation2, 10
GreedyExperimentalDesign, 9
hadamardExperimentalDesign, 10
imbalanced_block_designs, 11
imbalanced_complete_randomization, 12
initBinaryMatchExperimentalDesignSearch, 12
initBinaryMatchFollowedByGreedyExperimentalDesignSearch, 13
initBinaryMatchFollowedByRerandomizationDesignSearch, 14
initGreedyExperimentalDesignObject, 15
initGreedyMultipleKernelExperimentalDesignObject, 17
initKarpExperimentalDesignObject, 19
initOptimalExperimentalDesignObject, 20
initRerandomizationExperimentalDesignObject, 21
optimize_asymmetric_treatment_assignment, 22
plot.greedy_experimental_design_search, 23
plot.greedy_multiple_kernel_experimental_design_search, 24
plot_obj_val_by_iter, 24
plot_obj_val_order_statistic, 23, 24, 25
print.binary_match_structure, 26
print.binary_then_greedy_experimental_design_search, 26
print.binary_then_rerandomization_experimental_design_search, 27
print.greedy_experimental_design_search, 27
print.greedy_multiple_kernel_experimental_design_search, 28
print.karp_experimental_design_search, 28
print.optimal_experimental_design_search, 29
print.pairwise_matching_experimental_design_search, 29
print.rerandomization_experimental_design_search, 30
resultsBinaryMatchSearch, 30
resultsBinaryMatchThenGreedySearch, 31
resultsBinaryMatchThenRerandomizationSearch, 31
resultsGreedySearch, 32
resultsKarpSearch, 33
resultsMultipleKernelGreedySearch, 34
resultsOptimalSearch, 35
resultsRerandomizationSearch, 35
searchTimeElapsed, 36
standardize_data_matrix, 36
startSearch, 37
stopSearch, 16, 18, 22, 37
summary.binary_match_structure, 38
summary.binary_then_greedy_experimental_design,
38
summary.binary_then_rerandomization_experimental_design,
39
summary.greedy_experimental_design_search,
39
summary.greedy_multiple_kernel_experimental_design,
40
summary.karp_experimental_design_search,
40
summary.optimal_experimental_design_search,
41
summary.pairwise_matching_experimental_design_search,
41
summary.rerandomization_experimental_design_search,
42