Package: SeqExpMatch (via r-universe)

August 28, 2024

Type Package

Title Sequential Experimental Design via Matching on-the-Fly

Version 1.0.0

Description Generates the following sequential two-arm experimental designs: (1) completely randomized (Bernoulli) (2) balanced completely randomized (3) Efron's (1971) Biased Coin (4) Atkinson's (1982) Covariate-Adjusted Biased Coin (5) Kapelner and Krieger's (2014) Covariate-Adjusted Matching on the Fly (6) Kapelner and Krieger's (2021) CARA Matching on the Fly with Differential Covariate Weights (Naive) (7) Kapelner and Krieger's (2021) CARA Matching on the Fly with Differential Covariate Weights (Stepwise) and also provides the following types of inference: (1) estimation (with both Z-style estimators and OLS estimators), (2) frequentist testing (via asymptotic distribution results and via employing the nonparametric randomization test) and (3) frequentist confidence intervals (only under the superpopulation sampling assumption currently). Details can be found in our publication: Kapelner and Krieger ``A Matching Procedure for Sequential Experiments that Iteratively Learns which Covariates Improve Power'' (2020) <arXiv:2010.05980>. We now offer support for incidence, count, proportion and survival (with censoring) outcome types. We also have support for adding responses whenever they become available, and we can impute missing data in the subjects' covariate records (where each covariate record can thereby have different information). On the inference side, there is built-in support for many types of parametric models such as random effects for incidence outcomes and count outcomes. There is Kaplan-Meier estimation, weibull and coxph models for survival outcomes.

License GPL-3

Encoding UTF-8

Depends R6, checkmate, Matrix, data.table, survival, controlTest, betareg, statmod, numDeriv, lme4, lmerTest, coxme, missRanger, missForest, doParallel

Imports stats, checkmate, doParallel, R6

URL `https://github.com/kapelner/matching_on_the_fly_designs_R_package_and_paper_repr`

RoxygenNote 7.3.2

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

RemoteUrl https://github.com/kapelner/matching_on_the_fly_designs_r_package_and_paper_repr

RemoteRef HEAD

RemoteSha 901173847b8e05e29e78508e771067ae68376848

Contents

`SeqDesign` *A Sequential Design*

Description

An R6 Class encapsulating the data and functionality for a sequential experimental design. This class takes care of data intialization and sequential assignments. The class object should be saved securely after each assignment e.g. on an encrypted cloud server.

Public fields

`t` The current number of subjects in this sequential experiment (begins at zero).

`design` The experimenter-specified type of sequential experimental design (see constructor's documentation).

`Xraw` A data frame (data.table object) of subject data with number of rows n (the number of subjects) and number of columns p (the number of characteristics measured for each subject). This data frame is filled in sequentially by the experimenter and thus will have data present for rows 1...t (i.e. the number of subjects in the experiment currently) but otherwise will be missing.

`Ximp` Same as `Xraw` except with imputations for missing values (if necessary) and deletions of linearly dependent columns

`X` Same as `Ximp` except turned into a model matrix (i.e. all numeric with factors dummified) with no linearly dependent columns (and it is also a matrix object, not a data.table object)

`y` A numeric vector of subject responses with number of entries n (the number of subjects). During the KK21 designs the experimenter fills these values in when they are measured. For non-KK21 designs, this vector can be set at anytime (but must be set before inference is desired).

`dead` A binary vector of whether the subject is dead with number of entries `n` (the number of subjects). This vector is filled in only for `response_type` values "survival". The value of 1 indicates uncensored (as the subject died) and a value 0 indicates the real survival value is censored as the subject is still alive at the time of measurement. This follows the same convention as the `event` argument in the canonical `survival` package in the constructor `survival::Surv`. During the KK21 designs the experimenter fills these values in when they are measured. For non-KK21 designs, this vector can be set at anytime (but must be set before inference is desired).

`prob_T` The experimenter-specified probability a subject becomes wtated to the treatment arm.

`w` A binary vector of subject assignments with number of entries `n` (the number of subjects). This vector is filled in sequentially by this package (similar to X) and will have assignments present for entries 1...t (i.e. the number of subjects in the experiment currently) but otherwise will be missing.

`response_type` This is the experimenter-specified type of response value which is one of the following: "continuous", "incidence", "proportion", "count", "survival"

`covariate_weights` The running values of the weights for each covariate

Methods

Public methods:

- `SeqDesign$new()`
- `SeqDesign$add_subject_to_experiment_and_assign()`
- `SeqDesign$print_current_subject_assignment()`
- `SeqDesign$add_subject_response()`
- `SeqDesign$add_all_subject_responses()`
- `SeqDesign$matching_statistics()`
- `SeqDesign$assert_experiment_completed()`
- `SeqDesign$check_experiment_completed()`
- `SeqDesign$clone()`

Method `new()`**:** Initialize a sequential experimental design

Usage:

```
SeqDesign$new(
  n,
  design,
  response_type,
  prob_T = 0.5,
  include_is_missing_as_a_new_feature = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments:

`n` Number of subjects fixed beforehand.

`design` The type of sequential experimental design. This must be one of the following "CRD" for the completely randomized design / Bernoulli design, "iBCRD" for the incomplete / balanaced completely randomized design with appropriate permuted blocks based on `prob_T` (e.g., if `prob_T = 2`, then this design would enforce n/2 T's and n/2 C's), "Efron" for Efron's (1971) Biased Coin Design "Atkinson" for Atkinson's (1982) Covariate-Adjusted Biased Coin Design "KK14" for Kapelner and Krieger's (2014) Covariate-Adjusted Matching on the Fly Design "KK21" for Kapelner and Krieger's (2021) CARA Matching on the Fly with Differential Covariate Weights Design "KK21stepwise" for Kapelner and Krieger's (2021) CARA Matching on the Fly with Differential Covariate Weights Stepwise Design

`response_type` The data type of response values which must be one of the following: "continuous", "incidence", "proportion", "count", "survival". This package will enforce that all added responses via add_subject_response will be of the appropriate type.

`prob_T` The probability of the treatment assignment. This defaults to `0.5`.

`include_is_missing_as_a_new_feature` If missing data is present in a variable, should we include another dummy variable for its missingness in addition to imputing its value? If the feature is type factor, instead of creating a new column, we allow missingness to be its own level. The default is `TRUE`.

`verbose` A flag indicating whether messages should be displayed to the user. Default is `TRUE`.

`...` Design-specific parameters: "Efron" requires "weighted_coin_prob" which is the probability of the weighted coin for assignment. If unspecified, default is 2/3. All "KK" designs require "lambda", the quantile cutoff of the subject distance distribution for determining matches. If unspecified, default is 10 All "KK" designs require "t_0_pct", the percentage of total sample size n where matching begins. If unspecified, default is 35 All "KK" designs have optional flag KK_verbose with default `FALSE` which prints out debug messages about how the matching-on-the-fly is working. All "KK21" designs further require "num_boot" which is the number of bootstrap samples taken to approximate the subject-distance distribution. If unspecified, default is 500. There is an optional flag "proportion_use_speedup = TRUE" which uses a continuous regression on log(y/(1-y)) instead of a beta regression each time to generate the weights in KK21 designs. The default is this flag is on.

Returns: A new ‘SeqDesign‘ object.

Examples:

```
seq_des = SeqDesign$new(design = "KK21stepwise", response_type = "continuous")
```

Method `add_subject_to_experiment_and_assign()`**:** Add subject-specific measurements for the next subject entrant and return this new subject's treatment assignment

Usage:

```
SeqDesign$add_subject_to_experiment_and_assign(x_new, allow_new_cols = TRUE)
```

Arguments:

`x_new` A row of the data frame corresponding to the new subject to be added (must be type data.table).

`allow_new_cols` Should we allow new/different features than previously seen in previous subjects in the new subject's covariates? Default is `TRUE`.

`KK_verbose` If `TRUE`, we will print out messages about the KK assignment. This is useful for understanding how the KK assignment is working

Examples:

```
seq_des = SeqDesign$new(n = 100, p = 10, design = "CRD", response_type = "continuous")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
```

Method `print_current_subject_assignment()`**:** Prints the current assignment to screen. Should be called after `add_subject_to_experiment_and_assign`.

Usage:

```
SeqDesign$print_current_subject_assignment()
```

Examples:

```
seq_des = SeqDesign$new(n = 100, p = 10, design = "CRD", response_type = "continuous")

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$print_current_subject_assignment()
```

Method `add_subject_response()`**:** For CARA designs, add subject response for the a subject

Usage:

```
SeqDesign$add_subject_response(t, y, dead = 1)
```

Arguments:

`t` The subject index for which to attach a response (beginning with 1, ending with n). You cannot add responses for subjects that have not yet been added to the experiment via add_subject_to_experiment_and_ass

`y` The response value which must be appropriate for the response_type.

`dead` If the response is censored, enter 0 for this value. This is only necessary to specify for response type "survival" otherwise do not specify this argument (as it will default to 1).

Examples:

```
seq_des = SeqDesign$new(n = 100, p = 10, design = "KK21", response_type = "continuous")

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])

seq_des$add_subject_response(4.71, 1)
#works
seq_des$add_subject_response(4.71, 2)
#fails
```

Method `add_all_subject_responses()`**:** For non-CARA designs, add all subject responses

Usage:

```
SeqDesign$add_all_subject_responses(ys, deads = NULL)
```

Arguments:

`ys` The responses as a numeric vector of length n

`deads` The binary vector of length n where 1 indicates the the subject is dead (survival value is uncensored) and 0 indicates the subject is alive (survival value is censored). This is only necessary for response type "survival" otherwise do not specify and the value will default to 1.

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD", response_type = "continuous")

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])

seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))
```

Method `matching_statistics()`**:** For KK designs only, this returns a list with useful matching statistics.

Usage:

```
SeqDesign$matching_statistics()
```

Returns: A list with the following data: `num_matches`, `prop_subjects_matched`, `num_subjects_remaining_in_reser`
`prop_subjects_remaining_in_reservoir`.

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "KK14", response_type = "continuous")

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])

seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des$matching_statistics()
```

Method `assert_experiment_completed()`**:** Asserts if the experiment is completed (all n assignments are assigned in the w vector and all n responses in the y vector are recorded), i.e. throws descriptive error if the experiment is incomplete.

Usage:

```
SeqDesign$assert_experiment_completed()
```

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD", response_type = "continuous")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])

#if run, it would throw an error since all of the covariate vectors are not yet recorded
#seq_des$assert_experiment_completed()
```

```
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])

#if run, it would throw an error since the responses are not yet recorded
#seq_des$assert_experiment_completed()

seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des$assert_experiment_completed() #no response means the assert is true
```

Method `check_experiment_completed()`**:** Checks if the experiment is completed (all `n` assignments are assigned in the w vector and all n responses in the y vector are recorded).

Usage:

```
SeqDesign$check_experiment_completed()
```

Returns: `TRUE` if experiment is complete, `FALSE` otherwise.

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD", response_type = "continuous")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])

#returns FALSE since all of the covariate vectors are not yet recorded
seq_des$check_experiment_completed()

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])

#returns FALSE since the responses are not yet recorded
seq_des$check_experiment_completed()

seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des$check_experiment_completed() #returns TRUE
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SeqDesign$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `SeqDesign$new`
## ------------------------------------------------

seq_des = SeqDesign$new(design = "KK21stepwise", response_type = "continuous")


## ------------------------------------------------
## Method `SeqDesign$add_subject_to_experiment_and_assign`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 100, p = 10, design = "CRD", response_type = "continuous")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])


## ------------------------------------------------
## Method `SeqDesign$print_current_subject_assignment`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 100, p = 10, design = "CRD", response_type = "continuous")

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$print_current_subject_assignment()


## ------------------------------------------------
## Method `SeqDesign$add_subject_response`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 100, p = 10, design = "KK21", response_type = "continuous")

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])

seq_des$add_subject_response(4.71, 1)
#works
seq_des$add_subject_response(4.71, 2)
#fails


## ------------------------------------------------
## Method `SeqDesign$add_all_subject_responses`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD", response_type = "continuous")

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
```

```
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))


## ------------------------------------------------
## Method `SeqDesign$matching_statistics`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "KK14", response_type = "continuous")

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])

seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des$matching_statistics()


## ------------------------------------------------
## Method `SeqDesign$assert_experiment_completed`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD", response_type = "continuous")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])

#if run, it would throw an error since all of the covariate vectors are not yet recorded
#seq_des$assert_experiment_completed()

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])

#if run, it would throw an error since the responses are not yet recorded
#seq_des$assert_experiment_completed()

seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des$assert_experiment_completed() #no response means the assert is true

## ------------------------------------------------
## Method `SeqDesign$check_experiment_completed`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD", response_type = "continuous")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])

#returns FALSE since all of the covariate vectors are not yet recorded
```

```
seq_des$check_experiment_completed()

seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])

#returns FALSE since the responses are not yet recorded
seq_des$check_experiment_completed()

seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des$check_experiment_completed() #returns TRUE
```

SeqDesignInference *Inference for A Sequential Design*

Description

An R6 Class that estimates, tests and provides intervals for a treatment effect in a sequential design. This class takes a `SeqDesign` object as an input where this object contains data for a fully completed sequential experiment (i.e. all treatment assignments were allocated and all responses were collected). Then the user specifies the type of estimation (mean_difference-or-medians or default_regression) and the type of sampling assumption (i.e. the superpopulation assumption leading to MLE-or-KM-based inference or the finite population assumption implying randomization-exact-based inference) and then can query the estimate and pval for the test. If the test is normal-theory based it is testing the population H_0: beta_T = 0 and if the test is a randomization test, it is testing the sharp null that H_0: Y_T_i = Y_C_i for all subjects. Confidence interval construction is available for normal-theory based test type as well.

Public fields

`estimate_type` The estimate type (see initializer documentation).

`test_type` The type of test to run (see initializer documentation).

Methods

Public methods:

- `SeqDesignInference$new()`
- `SeqDesignInference$compute_treatment_estimate()`
- `SeqDesignInference$compute_confidence_interval()`
- `SeqDesignInference$compute_two_sided_pval_for_treatment_effect()`
- `SeqDesignInference$clone()`

Method `new()`**:** Initialize a sequential experimental design estimation and test object after the sequential design is completed.

Usage:

```
SeqDesignInference$new(
  seq_des_obj,
  estimate_type,
  test_type = "randomization-exact",
  num_cores = 1,
  verbose = TRUE
)
```

Arguments:

`seq_des_obj` A SeqDesign object whose entire n subjects are assigned and response y is recorded within.

`estimate_type` The type of estimate to compute of which there are many and identified by the response type as its first word. If the string "KK" appears after the first word, then this estimate type is only applicable to KK14, KK21, KK21stepwise designs. * "continuous_simple_mean_difference" assumes the treatment effect parameter is an additive treatment effect and estimates via the simple average difference * "continuous_regression_with_covariates" assumes the treatment effect parameter is an additive treatment effect and the presence of linear additive covariates and estimates via OLS * "continuous_KK_compound_mean_difference" assumes the treatment effect parameter is an additive treatment effect and estimates via combining a simple average difference estimator for both the matches and the reservoir * "continuous_KK_compound_multivariate_regression" assumes the treatment effect parameter is an additive treatment effect and estimates via combining an OLS estimator for bothe ther matches and the reservoir * "continuous_KK_regression_with_covariates_with_matching_dummies" assumes the treatment effect parameter is an additive treatment effect and the presence of linear additive covariates treating the match ID as a factor and estimates via OLS (not recommended) * "continuous_KK_regression_with_covariates_with_random_intercepts" assumes the treatment effect parameter is an additive treatment effect and the presence of linear additive covariates and random intercepts on the match ID and estimates via restricted maximum likelihood * "incidence_simple_mean_difference" assumes the treatment effect parameter is an additive probability difference and estimates via the simple average difference * "incidence_simple_log_odds" assumes the treatment effect parameter is additive in the log odds probability of the positive class and estimates via maximum likelihood * "incidence_logistic_regression" assumes the treatment effect parameter is additive in the log odds probability of the positive class and the presence of linear additive covariates also in the log odds probability of the positive class and estimates via maximum likelihood * "incidence_KK_compound_multivariate_logistic_regression" assumes the treatment effect parameter is additive in the log odds probability of the positive class and the presence of linear additive covariates treating the match ID as a factor also in the log odds probability of the positive class and estimates via maximum likelihood * "incidence_KK_multivariate_logistic_regression_with_matching_dummies" assumes the treatment effect parameter is additive in the log odds probability of the positive class and the presence of linear additive covariates treating the match ID as a factor also in the log odds probability of the positive class and estimates via maximum likelihood * "incidence_KK_compound_multivariate_log assumes the treatment effect parameter is additive in the log odds probability of the positive class and the presence of linear additive covariates and random intercepts on the match ID also in units of log odds probability of the positive class and estimates via restricted maximum likelihood * "proportion_simple_mean_difference" assumes the treatment effect parameter is an additive proportion difference and estimates via the simple

average difference * "proportion_simple_logodds_regression" assumes the treatment effect parameter is additive in the log odds proportion and estimates via beta regression * "proportion_beta_regression" assumes the treatment effect parameter is additive in the log odds proportion and the presence of linear additive covariates and estimates via beta regression * "proportion_KK_compound_univariate_beta_regression" assumes the treatment effect parameter is an additive treatment effect in log odds of proportion and the presence of linear additive covariates also in the log odds of proportion and estimates via combining a simple average difference estimator for both the matches and the reservoir * "proportion_KK_compound_multivariate_beta_regression" assumes the treatment effect parameter is an additive treatment effect in log odds and estimates via combining a simple average difference estimator for both the matches and the reservoir * "proportion_KK_multivariate_beta_regression_with_match
assumes the treatment effect parameter is additive in the log odds proportion and the presence of linear additive covariates and estimates via beta regression * "count_simple_mean_difference" assumes the treatment effect parameter is an additive mean count difference and estimates via the simple average difference * "count_univariate_negative_binomial_regression" assumes the treatment effect parameter is additive in the log count and estimates via negative binomial regression * "count_multivariate_negative_binomial_regression" assumes the treatment effect parameter is additive in the log count and the presence of linear additive covariates and estimates via negative binomial regression * "count_KK_compound_univariate_negative_binomial_regre
assumes the treatment effect parameter is additive in the log count and treating the match ID as a factor and estimates via maximum likelihood * "count_KK_multivariate_negative_binomial_regression_with_ma
assumes the treatment effect parameter is additive in the log count and the presence of linear additive covariates and treating the match ID as a factor and estimates via maximum likelihood * "count_KK_multivariate_negative_binomial_regression_with_random_intercepts_for_matches" assumes the treatment effect parameter is additive in the log count and the presence of linear additive covariates in units of log count and random intercepts on the match ID in the log count and estimates via maximum likelihood * "survival_simple_median_difference" assumes the treatment effect parameter is the difference in survival medians and estimates via Kaplan-Meier * "survival_simple_restricted_mean_difference" assumes the treatment effect parameter is the difference in survival means and estimates via restricted means (assuming the largest survival time is the absolute limit) * "survival_univariate_weibull_regression" assumes the treatment effect parameter is the additive mean survival difference and estimates via Weibull regression * "survival_multivariate_weibull_regression" assumes the treatment effect parameter is the additive mean survival difference and the presence of linear additive covariates and estimates via Weibull regression * "survival_KK_multivariate_weibull_regression_with_matc
assumes the treatment effect parameter is the additive mean survival difference and the presence of linear additive covariates and treating the match ID as a factor and estimates via Weibull regression * "survival_univariate_coxph_regression" assumes the treatment effect is a log difference in hazard which is constant conditional on covariate values and estimates via maximum likelihood * "survival_multivariate_coxph_regression" assumes the treatment effect is a log difference in hazard which is constant conditional on covariate values and the presence of linear additive covariates in log hazard and estimates via maximum likelihood * "survival_KK_multivariate_coxph_regression_with_matching_dummies" assumes the treatment effect is a log difference in hazard which is constant conditional on covariate values and the presence of linear additive covariates in log hazard and treating the match ID as a factor and estimates via maximum likelihood * "survival_KK_multivariate_coxph_regression_with_random_inte
assumes the treatment effect is a log difference in hazard which is constant conditional on covariate values and the presence of linear additive covariates in log hazard and random

intercepts on the match ID in units of log hazard and estimates via maximum likelihood

`test_type` The type of test to run (either "MLE-or-KM-based" implying your subject entrant sampling assumption is from a superpopulation or "randomization-exact" implying a finite sampling assumption). The default option is "randomization-exact" as it provided properly-sized tests in our simulations.

`num_cores` The number of CPU cores to use to parallelize the sampling during randomization-based inference (which is very slow). The default is 1 for serial computation. This parameter is ignored for `test_type = "MLE-or-KM-based"`.

`verbose` A flag indicating whether messages should be displayed to the user. Default is `TRUE`

Returns: A new ‘SeqDesignTest‘ object.

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des)
```

Method `compute_treatment_estimate()`: Computes for estimate type "mean_difference-or-medians" either (1a) for incidence outcomes, the additive log odds treatment effect using logistic regression (1b) for survival outcomes, the median difference for suvival using the Kaplan-Meier estimates for both arms (1c) for count outcomes, the additive treatment effect on log count using negative binomial regression (1d) for proportion and continous outcomes (where the latter is not under an equal allocation KK design), the classic mean_difference estimate of the additive treatment effect, (1e) for continuous outcome, equal allocation to arms and KK designs, there's a special match-reservoir weighted classic mean_difference estimate

Computes for estimte type "default_regression" either (2a) for incidence outcomes, the additive log odds treatment effect using logistic regression controlled for all other covariates (2b) for survival outcomes, the additive treatment effect on log suvival using Weibull regression controlled for all other covariates (2c) for count outcomes, the additive treatment effect on log count using negative binomial regression controlled for all other covariates (2d) for proportion outcome, the additive treatment effect on proportion using beta regression controlled for all other covariates (2e) for continous outcomes but not under an equal allocation KK design, the additive treatment effect using OLS regression controlled for all other covariates (2f) for continuous outcome, equal allocation to arms and KK designs, there's a special match-reservoir weighted OLS regression controlled for all other covariates

Usage:

```
SeqDesignInference$compute_treatment_estimate()
```

Returns: The setting-appropriate (see description) numeric estimate of the treatment effect

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD", response_type = "continuous")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des)
seq_des_inf$compute_treatment_estimate()
```

Method `compute_confidence_interval()`**:** Computes a 1-alpha level frequentist confidence interval differently for all response types, estimate types and test types.

For "mean_difference" it computes (1a) for incidence outcomes (ignoring the KK design structure), the p-value for the test of the additive log odds treatment effect being zero using logistic regression's MLE normal approximation (1b) for survival outcomes (ignoring the KK design structure), the median difference for survival using the Kaplan-Meier estimates for both arms (1c) for count, proportion and continous outcomes (all ignoring the KK design structure), the classic mean_difference estimate of the additive treatment effect, (1d) for continuous outcome, equal allocation to arms and KK designs, there's a special match-reservoir weighted classic mean_difference estimate

For "medial_difference" it computes only (2) for survival outcomes (ignoring the KK design structure), the difference of medians of the two arms

Computes for estimte type "default_regression" either (3a) for incidence outcomes, the additive log odds treatment effect using logistic regression controlled for all other covariates (3b) for survival outcomes, the additive treatment effect on log suvival using Weibull regression controlled for all other covariates (3c) for count outcomes, the additive treatment effect on log count using negative binomial regression controlled for all other covariates (3d) for proportion outcome, the additive treatment effect on proportion using beta regression controlled for all other covariates (3e) for continous outcomes but not under an equal allocation KK design, the additive treatment effect using OLS regression controlled for all other covariates (3f) for continuous outcome, equal allocation to arms and KK designs, there's a special match-reservoir weighted OLS regression controlled for all other covariates

The confidence interval is computed differently for [I] test type "MLE-or-KM-based" Here we use the theory that MLE's computed for GLM's are asymptotically normal (except in the case of estimat_type "median difference" where a nonparametric bootstrap confidence interval (see the `controlTest::quantileControlTest` method) is employed. Hence these confidence intervals are asymptotically valid and thus approximate for any sample size.

[II] test type "randomization-exact" Here we invert the randomization test that tests the strong null H_0: y_T_i - y_C_i = delta <=> (y_T_i - delta) - y_C_i = 0 so we adjust the treatment responses downward by delta. We then find the set of all delta values that is above 1 - alpha/2 (i.e. two-sided) This is accomplished via a bisection algorithm (algorithm 1 of Glazer and Stark, 2025 available at https://arxiv.org/abs/2405.05238). These confidence intervals are exact to within tolerance `pval_epsilon`.

Usage:

```
SeqDesignInference$compute_confidence_interval(
  alpha = 0.05,
  nsim_exact_test = 501,
  pval_epsilon = 0.001,
  B = NULL
)
```

Arguments:

`alpha` The confidence level in the computed confidence interval is 1 - `alpha`. The default is 0.05.

`nsim_exact_test` The number of randomization vectors (applicable for test type "randomization-exact" only). The default is 1000 providing good resolutions to confidence intervals.

`pval_epsilon` The bisection algorithm tolerance for the test inversion (applicable for test type "randomization-exact" only). The default is to find a CI accurate to within a tenth of a percent.

`B` Number of bootstrap samples for the survival response where `estimate_type` is "median_difference" (see the `controlTest::quantileControlTest` method). The default is NULL which corresponds to B=501 providing pvalue resolution to a fifth of a percent.

Returns: A 1 - alpha sized frequentist confidence interval for the treatment effect

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des, test_type = "MLE-or-KM-based")
seq_des_inf$compute_confidence_interval()
```

Method `compute_two_sided_pval_for_treatment_effect()`**:** Computes a 2-sided p-value for all types of inferential settings written about in the initializer (1) estimate type "mean_difference-or-medians" and test type "MLE-or-KM-based" This implies the classic mean_difference estimator which means that (a) For continous and proportion outcomes, H_0: E[Y_T] - E[Y_C] = delta, (b) For incidence outcomes, H_0: log(Odds(P(Y_T = 1)) - log(Odds(P(Y_C = 1) = delta, (c) For count outcomes, H_0: E[ln(Y_T)] - E[ln(Y_C)] = delta or (d) For survival outcomes, H_0: MED[Y_T] - MED[Y_C] = delta (2) Fisher's randomization test which means that H_0: y_i_T - y_i_C = delta for all subjects either the classic different-in-means estimate of the additive treatment effect, i.e. ybar_T - ybar_C or the default_regression estimate of the additive treatment effect linearly i.e. the treatment different adjusted linearly for the p covariates.

Usage:

```
SeqDesignInference$compute_two_sided_pval_for_treatment_effect(
  nsim_exact_test = 501,
  B = NULL,
  delta = 0
)
```

Arguments:

`nsim_exact_test` The number of randomization vectors to use in the randomization test (ignored if `test_type` is not "randomization-exact"). The default is 501 providing pvalue resolution to a fifth of a percent.

`B` Number of bootstrap samples for the survival response where `estimate_type` is "median_difference" (see the `controlTest::quantileControlTest` method). The default is 501 providing pvalue resolution to a fifth of a percent.

`delta` The null difference to test against. For any treatment effect at all this is set to zero (the default).

Returns: The approximate frequentist p-value

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des)
seq_des_inf$compute_two_sided_pval_for_treatment_effect()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SeqDesignInference$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `SeqDesignInference$new`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des)
```

```
## ------------------------------------------------
## Method `SeqDesignInference$compute_treatment_estimate`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD", response_type = "continuous")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des)
seq_des_inf$compute_treatment_estimate()


## ------------------------------------------------
## Method `SeqDesignInference$compute_confidence_interval`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des, test_type = "MLE-or-KM-based")
seq_des_inf$compute_confidence_interval()


## ------------------------------------------------
## Method `SeqDesignInference$compute_two_sided_pval_for_treatment_effect`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[1, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[2, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[3, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[4, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[5, 2 : 10])
seq_des$add_subject_to_experiment_and_assign(MASS::biopsy[6, 2 : 10])
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des)
seq_des_inf$compute_two_sided_pval_for_treatment_effect()
```

SeqExpMatch — *Sequential Experimental Designs via Matching On-the-Fly*

Description

SeqExpMatch

Details

Generates the following sequential two-arm experimental designs (1) completely randomized (Bernoulli) (2) balanced completely randomized (3) Efron's (1971) Biased Coin (4) Atkinson's (1982) Covariate-Adjusted Biased Coin (5) Kapelner and Krieger's (2014) Covariate-Adjusted Matching on the Fly (6) Kapelner and Krieger's (2021) CARA Matching on the Fly with Weighted Covariates (7) Kapelner and Krieger's (2021) CARA Matching on the Fly with Weighted Covariates Stepwise

Author(s)

Adam Kapelner `<kapelner@qc.cuny.edu>`

References

Adam Kapelner and Abba Krieger A Matching Procedure for Sequential Experiments that Iteratively Learns which Covariates Improve Power, Arxiv 2010.05980

See Also

Useful links:

- `https://github.com/kapelner/matching_on_the_fly_designs_R_package_and_paper_repr`

Index