Package 'SeqExpMatch'

June 1, 2021

Type Package

Title Sequential Experimental Design via Matching on-the-Fly

Version 0.1.0

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>.

License GPL-3

Encoding UTF-8

Depends R6, checkmate, doParallel, R (>= 3.6.3)

Imports stats

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

RoxygenNote 7.1.1

NeedsCompilation no

Author Adam Kapelner [aut, cre] (<https://orcid.org/0000-0001-5985-6792>),
Abba Krieger [aut]

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

Repository CRAN

Date/Publication 2021-06-01 07:50:01 UTC

R topics documented:

`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 type of sequential experimental design (see constructor's documentation).

`X` A numeric matrix 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 matrix is filled in sequentially 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.

`y` A numeric vector of subject responses with number of entries n (the number of subjects). During the KK21 designs this must be filled in sequentially (similar to X) and will have data present for entries 1...t (i.e. the number of subjects in the experiment currently) but otherwise will be missing. For non-KK21 designs, this vector can be set at anytime (but must be set before inference is desired).

`w` A binary vector of subject assignments with number of entries n (the number of subjects). This vector is filled in sequentially (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.

`verbose` A flag that indicates whether messages should be displayed to the user

Methods

Public methods:

- `SeqDesign$new()`
- `SeqDesign$add_subject_to_experiment()`
- `SeqDesign$print_current_subject_assignment()`

- SeqDesign$add_current_subject_response()
- SeqDesign$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, p, design, verbose = TRUE, ...)
```

Arguments:

`n` Number of subjects fixed beforehand. A future version of this software will allow for sequential stopping and thus n will not need to be prespecified.

`p` Number of characteristics measured for each subject. If measurement j are categorical with L_j levels, you must select a reference level and convert this information to L_j-1 dummies. Thus p := # of numeric variables + sum_j (L_j - 1).

`design` The type of sequential experimental design. This must be one of the following "CRD" for the completely randomized design / Bernoulli design, "BCRD" for the balanaced completely randomized design with 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

`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 "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.

Returns: A new 'SeqDesign' object.

Examples:

```
seq_des = SeqDesign$new(n = 100, p = 10, design = "KK21stepwise")
```

Method `add_subject_to_experiment()`**:** Add subject-specific measurements for the next subject entrant

Usage:

```
SeqDesign$add_subject_to_experiment(x_vec)
```

Arguments:

`x_vec` A p-length numeric vector

Examples:

```
seq_des = SeqDesign$new(n = 100, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
```

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

Usage:

```
SeqDesign$print_current_subject_assignment()
```

Examples:

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

seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$print_current_subject_assignment()
```

Method `add_current_subject_response()`**:** For CARA designs, add subject response for the current subject entrant

Usage:

```
SeqDesign$add_current_subject_response(y)
```

Arguments:

`y` The response as a numeric scalar

Examples:

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

seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))

seq_des$add_current_subject_response(4.71)
```

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

Usage:

```
SeqDesign$add_all_subject_responses(y)
```

Arguments:

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

Examples:

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

seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))

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")

seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))

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$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))

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

seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))

#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$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))

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

seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))

#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(n = 100, p = 10, design = "KK21stepwise")


## ------------------------------------------------
## Method `SeqDesign$add_subject_to_experiment`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 100, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
```

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

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

seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$print_current_subject_assignment()


## ------------------------------------------------
## Method `SeqDesign$add_current_subject_response`
## ------------------------------------------------

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

seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))

seq_des$add_current_subject_response(4.71)


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

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

seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))

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")

seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))

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$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))

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

seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))

#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$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))

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

seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))

#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 (difference-in-means or OLS) and the type of sampling assumption (i.e. the superpopulation assumption leading to normal-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 type of estimate to compute (either "difference-in-means" or "OLS").

`test_type` The type of test to run (either "normal-based" or "randomization-exact").

`num_cores` The number of CPU cores to employr during sampling within randomization inference

`verbose` A flag that indicates whether messages should be displayed to the user

Methods

Public methods:

- `SeqDesignInference$new()`
- `SeqDesignInference$compute_treatment_estimate()`
- `SeqDesignInference$compute_pval_for_no_treatment_effect()`
- `SeqDesignInference$randomization_inference_samples_for_no_treatment_effect()`
- `SeqDesignInference$compute_confidence_interval()`
- `SeqDesignInference$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 = "OLS",
  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 (either "difference-in-means" or "OLS"). Default is "OLS" as this provided higher power in our simulations.

`test_type` The type of test to run (either "normal-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 = "normal-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(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$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 either the classic different-in-means estimate of the additive treatment effect, i.e. ybar_T - ybar_C or the OLS estimate of the additive treatment effect linearly i.e. the treatment different adjusted linearly for the p covariates.

Usage:

```
SeqDesignInference$compute_treatment_estimate()
```

Returns: The numeric estimate of the treatment effect

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$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_pval_for_no_treatment_effect()`**:** Computes either the classic different-in-means estimate of the additive treatment effect, i.e. ybar_T - ybar_C or the OLS estimate of the additive treatment effect linearly i.e. the treatment different adjusted linearly for the p covariates.

Usage:

```
SeqDesignInference$compute_pval_for_no_treatment_effect(nsim_exact_test = 501)
```

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.

Returns: The frequentist p-val for the test of nonzero treatment effect

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$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_pval_for_no_treatment_effect()
```

Method `randomization_inference_samples_for_no_treatment_effect()`**:** Computes many randomization samples of either the classic different-in-means estimate of the additive treatment effect, i.e. ybar_T - ybar_C or the OLS estimate of the additive treatment effect linearly i.e. the treatment different adjusted linearly for the p covariates. This function is useful if you wish to run your own, custom hypothesis tests.

Usage:

```
SeqDesignInference$randomization_inference_samples_for_no_treatment_effect(
  nsim_exact_test = 501
)
```

Arguments:

`nsim_exact_test` The number of randomization vectors. The default is 501 providing pvalue resolution to a fifth of a percent.

Returns: The `nsim_exact_test` samples of the treatment effect under the null hypothesis of no treatment effect where each sample is estimated from a different assignment vector for the prespecified design

Examples:

```
seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))
```

```
seq_des_inf = SeqDesignInference$new(seq_des)
samps = seq_des_inf$randomization_inference_samples_for_no_treatment_effect()
summary(samps)
```

Method `compute_confidence_interval()`**:** Computes either a:

1. classic frequentist confidence interval (CI) of the additive treatment effect employing the normal theory approximation for both the (a) difference in means estimator i.e. [ybar_T - ybar_C +/- t_alpha/2, n_T + n_C - 2 s_ybar_T - ybar_C] or (b) the OLS estimator i.e. [beta_hat_T +/- t_alpha/2, n + p - 2 s_beta_hat_T] where the z approximation is employed in lieu of the t is the design is a KK design or

2. a randomization-based CI of an additive shift effect of the potential outcomes under treatment and control by an inversion of the randomization test at level alpha (this feature is incomplete).

Usage:

```
SeqDesignInference$compute_confidence_interval(
  alpha = 0.05,
  nsim_exact_test = 501
)
```

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. The default is 1000 providing good resolutions to confidence intervals.

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(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des, test_type = "normal-based")
seq_des_inf$compute_confidence_interval()
```

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(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$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$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$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_pval_for_no_treatment_effect`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$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_pval_for_no_treatment_effect()
```

```
## ------------------------------------------------
## Method `SeqDesignInference$randomization_inference_samples_for_no_treatment_effect`
## ------------------------------------------------

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des)
samps = seq_des_inf$randomization_inference_samples_for_no_treatment_effect()
summary(samps)


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

seq_des = SeqDesign$new(n = 6, p = 10, design = "CRD")
seq_des$add_subject_to_experiment(c(1, 38, 142, 71, 5.3, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(0, 27, 127, 60, 5.5, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 42, 169, 74, 5.1, 0, 1, 0, 0, 0))
seq_des$add_subject_to_experiment(c(0, 59, 105, 62, 5.9, 0, 0, 0, 1, 0))
seq_des$add_subject_to_experiment(c(1, 32, 186, 66, 5.6, 1, 0, 0, 0, 0))
seq_des$add_subject_to_experiment(c(1, 37, 178, 75, 6.5, 0, 0, 0, 0, 1))
seq_des$add_all_subject_responses(c(4.71, 1.23, 4.78, 6.11, 5.95, 8.43))

seq_des_inf = SeqDesignInference$new(seq_des, test_type = "normal-based")
seq_des_inf$compute_confidence_interval()
```

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

Index