

# Package: OptimalRerandExpDesigns (via r-universe)

October 21, 2024

**Type** Package

**Title** Optimal Rerandomization Experimental Designs

**Version** 1.1

**Date** 2021-01-25

**Author** Adam Kapelner, Michael Sklar, Abba M. Krieger and David Azriel

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

**Description** This is a tool to find the optimal rerandomization threshold in non-sequential experiments. We offer three procedures based on assumptions made on the residuals distribution: (1) normality assumed (2) excess kurtosis assumed (3) entire distribution assumed. Illustrations are included. Also included is a routine to unbiasedly estimate Frobenius norms of variance-covariance matrices. Details of the method can be found in ``Optimal Rerandomization via a Criterion that Provides Insurance Against Failed Experiments'' Adam Kapelner, Abba M. Krieger, Michael Sklar and David Azriel (2020) <[arXiv:1905.03337](https://arxiv.org/abs/1905.03337)>.

**License** GPL-3

**Depends** R (>= 3.2.0), ggplot2 (>= 3.0), momentchi2 (>= 0.1.5), GreedyExperimentalDesign (>= 1.3)

**Imports** stats

**RoxygenNote** 7.1.0

**URL** <https://github.com/kapelner/OptimalRerandExpDesigns>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2021-01-28 12:50:06 UTC

## Contents

complete_randomization_plus_one_min_one . . . . .	2
complete_randomization_with_forced_balance_plus_one_min_one . . . . .	3
compute_objective_val_plus_one_min_one_enc . . . . .	3
frob_norm_sq . . . . .	4
frob_norm_sq_debiased . . . . .	5
frob_norm_sq_debiased_times_matrix . . . . .	5
generate_W_base_and_sort . . . . .	6
OptimalRerandExpDesigns . . . . .	7
optimal_rerandomization_exact . . . . .	8
optimal_rerandomization_normality_assumed . . . . .	9
optimal_rerandomization_tail_approx . . . . .	10
plot.optimal_rerandomization_obj . . . . .	12
plot.W_base_object . . . . .	12
print.optimal_rerandomization_obj . . . . .	13
print.W_base_object . . . . .	13
summary.optimal_rerandomization_obj . . . . .	14
summary.W_base_object . . . . .	15

<b>Index</b>	<b>16</b>
--------------	-----------

---

complete\_randomization\_plus\_one\_min\_one  
*Implements the complete randomization design (CRD) AKA Bernoulli Trial*

---

### Description

Implements the complete randomization design (CRD) AKA Bernoulli Trial

### Usage

```
complete_randomization_plus_one_min_one(n, r)
```

### Arguments

n	number of observations
r	number of randomized designs you would like

### Value

a matrix where each column is one of the r designs

### Author(s)

Adam Kapelner

---

`complete_randomization_with_forced_balance_plus_one_min_one`*Implements the balanced complete randomization design (BCRD)*

---

**Description**

Implements the balanced complete randomization design (BCRD)

**Usage**

```
complete_randomization_with_forced_balance_plus_one_min_one(n, r)
```

**Arguments**

n	number of observations
r	number of randomized designs you would like

**Value**

a matrix where each column is one of the r designs

**Author(s)**

Adam Kapelner

---

`compute_objective_val_plus_one_min_one_enc`*Returns the objective value given a design vector as well as an objective function. This is code duplication since this is implemented within Java. This is only to be run if...*

---

**Description**

Returns the objective value given a design vector as well as an objective function. This is code duplication since this is implemented within Java. This is only to be run if...

**Usage**

```
compute_objective_val_plus_one_min_one_enc(  
  X,  
  indic_T,  
  objective = "abs_sum_diff",  
  inv_cov_X = NULL  
)
```

**Arguments**

<code>X</code>	The $n \times p$ design matrix
<code>indic_T</code>	The $n$ -length binary allocation vector
<code>objective</code>	The objective function to use. Default is <code>abs_sum_diff</code> .
<code>inv_cov_X</code>	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.

**Value**

A vector of computed objective values.

**Author(s)**

Adam Kapelner

---

<code>frob_norm_sq</code>	<i>Naive Frobenius Norm Squared</i>
---------------------------	-------------------------------------

---

**Description**

Compute naive / vanilla squared Frobenius Norm of matrix  $A$

**Usage**

```
frob_norm_sq(A)
```

**Arguments**

<code>A</code>	The matrix of interest
----------------	------------------------

**Value**

The Frobenius Norm of  $A$  squared.

**Author(s)**

Adam Kapelner

---

frob\_norm\_sq\_debiased *Debiased Frobenius Norm Squared Var-Cov matrix*

---

### Description

Compute debiased Frobenius Norm of matrix  $\hat{\Sigma}$  (Appendix 5.8). Note that for  $S \leq 2$ , it returns the naive estimate.

### Usage

```
frob_norm_sq_debiased(
  Sigmahat,
  s,
  n,
  frob_norm_sq_bias_correction_min_samples = 10
)
```

### Arguments

Sigmahat	The var-cov matrix of interest
s	The number of vectors Sigmahat was generated from
n	The length of each vector
frob_norm_sq_bias_correction_min_samples	This estimate suffers from high variance when there are not enough samples. Thus, we only implement the correction beginning at this number of samples otherwise we return the naive estimate. Default is 10.

### Value

The unbiased estimate of the Frobenius Norm of a variance-covariance matrix squared.

### Author(s)

Adam Kapelner

---

frob\_norm\_sq\_debiased\_times\_matrix  
*Debiased Frobenius Norm Squared Constant Times Var-Cov matrix*

---

### Description

Compute debiased Frobenius Norm of matrix  $P$  times  $\hat{\Sigma}$  (Appendix 5.9). Note that for  $S \leq 2$ , it returns the naive estimate.

**Usage**

```
frob_norm_sq_debiased_times_matrix(  
  Sigmahat,  
  A,  
  s,  
  n,  
  frob_norm_sq_bias_correction_min_samples = 10  
)
```

**Arguments**

Sigmahat	The var-cov matrix of interest
A	The matrix that multiplies Sigmahat
s	The number of vectors Sigmahat was generated from
n	The length of each vector
frob_norm_sq_bias_correction_min_samples	This estimate suffers from high variance when there are not enough samples. Thus, we only implement the correction beginning at this number of samples otherwise we return the naive estimate. Default is 10.

**Value**

The unbiased estimate of the Frobenius Norm of A times a variance-covariance matrix quantity squared.

**Author(s)**

Adam Kapelner

---

generate\_W\_base\_and\_sort

*Generate Base Assignments and Sorts*

---

**Description**

Generates the base vectors to be used when locating the optimal rerandomization threshold

**Usage**

```
generate_W_base_and_sort(  
  X,  
  max_designs = 25000,  
  imbalance_function = "mahal_dist",  
  r = 0,  
  max_max_iters = 5  
)
```

**Arguments**

<code>X</code>	The data as an $n \times p$ matrix.
<code>max_designs</code>	The maximum number of designs. Default is 25,000.
<code>imbalance_function</code>	A string indicating the imbalance function. Currently, "abs_sum_difference" and "mahal_dist" are the options with the latter being the default.
<code>r</code>	An experimental feature that adds lower imbalance vectors to the base set using the GreedyExperimentalDesign package. This controls the number of vectors to search through on each iteration.
<code>max_max_iters</code>	An experimental feature that adds lower imbalance vectors to the base set using the GreedyExperimentalDesign package. The maximum number of iterations to use for the greedy search.

**Value**

A list including all arguments plus a matrix `W_base_sorted` whose `max_designs` rows are  $n$ -length allocation vectors and the allocation vectors are in

**Author(s)**

Adam Kapelner

**Examples**

```
n = 100
p = 10
X = matrix(rnorm(n * p), nrow = n, ncol = p)
X = apply(X, 2, function(xj){(xj - mean(xj)) / sd(xj)})
S = 1000

W_base_obj = generate_W_base_and_sort(X, max_designs = S)
W_base_obj
```

---

OptimalRerandExpDesigns

*Optimal Rerandomization Threshold Search for Experimental Design*

---

**Description**

A tool to find the optimal rerandomization threshold in non-sequential experiments

**Author(s)**

Adam Kapelner <kapelner@qc.cuny.edu>

**References**

Kapelner, A

---

 optimal\_rerandomization\_exact

*Find the Optimal Rerandomization Design Exactly*


---

### Description

Finds the optimal rerandomization threshold based on a user-defined quantile and a function that generates the non-linear component of the response

### Usage

```
optimal_rerandomization_exact(
  W_base_object,
  estimator = "linear",
  q = 0.95,
  skip_search_length = 1,
  smoothing_degree = 1,
  smoothing_span = 0.1,
  z_sim_fun,
  N_z = 1000,
  dot_every_x_iters = 100
)
```

### Arguments

<code>W_base_object</code>	An object that contains the assignments to begin with sorted by
<code>estimator</code>	"linear" for the covariate-adjusted linear regression estimator (default).
<code>q</code>	The tail criterion's quantile of MSE over $z$ 's. The default is 95%.
<code>skip_search_length</code>	In the exhaustive search, how many designs are skipped? Default is 1 for full exhaustive search through all assignments provided for in <code>W_base_object</code> .
<code>smoothing_degree</code>	The smoothing degree passed to loess.
<code>smoothing_span</code>	The smoothing span passed to loess.
<code>z_sim_fun</code>	This function returns vectors of numeric values of size $n$ . No default is provided.
<code>N_z</code>	The number of times to simulate $z$ 's within each strategy.
<code>dot_every_x_iters</code>	Print out a dot every this many iterations. The default is 100. Set to NULL for no printout.

### Value

A list containing the optimal design threshold, strategy, and other information.



**Author(s)**

Adam Kapelner

**Examples**

```

n = 100
p = 10
X = matrix(rnorm(n * p), nrow = n, ncol = p)
X = apply(X, 2, function(xj){(xj - mean(xj)) / sd(xj)})
S = 25000

W_base_obj = generate_W_base_and_sort(X, max_designs = S)
design = optimal_rerandomization_exact(W_base_obj,
  z_sim_fun = function(){rnorm(n)},
  skip_search_length = 10)
design

```

---

optimal\_rerandomization\_normality\_assumed

*Find the Optimal Rerandomization Design Under the Gaussian Approximation*

---

**Description**

Finds the optimal rerandomization threshold based on a user-defined quantile and a function that generates the non-linear component of the response

**Usage**

```

optimal_rerandomization_normality_assumed(
  W_base_object,
  estimator = "linear",
  q = 0.95,
  skip_search_length = 1,
  dot_every_x_iters = 100
)

```

**Arguments**

**W\_base\_object** An object that contains the assignments to begin with sorted by

**estimator** "linear" for the covariate-adjusted linear regression estimator (default).

**q** The tail criterion's quantile of MSE over z's. The default is 95%.

**skip\_search\_length**  
In the exhaustive search, how many designs are skipped? Default is 1 for full exhaustive search through all assignments provided for in W\_base\_object.

**dot\_every\_x\_iters**  
Print out a dot every this many iterations. The default is 100. Set to NULL for no printout.

**Value**

A list containing the optimal design threshold, strategy, and other information.

**Author(s)**

Adam Kapelner

**Examples**

```
n = 100
p = 10
X = matrix(rnorm(n * p), nrow = n, ncol = p)
X = apply(X, 2, function(xj){(xj - mean(xj)) / sd(xj)})
S = 25000

W_base_obj = generate_W_base_and_sort(X, max_designs = S)
design = optimal_rerandomization_normality_assumed(W_base_obj,
  skip_search_length = 10)
design
```

---

optimal\_rerandomization\_tail\_approx

*Find the Optimal Rerandomization Design Under the Tail and Kurtosis Approximation*

---

**Description**

Finds the optimal rerandomization threshold based on a user-defined quantile and kurtosis based on an approximation of tail standard errors

**Usage**

```
optimal_rerandomization_tail_approx(
  W_base_object,
  estimator = "linear",
  q = 0.95,
  c_val = NULL,
  skip_search_length = 1,
  binary_search = FALSE,
  excess_kurtosis_z = 0,
  use_frob_norm_sq_unbiased_estimator = TRUE,
  frob_norm_sq_bias_correction_min_samples = 10,
  smoothing_degree = 1,
  smoothing_span = 0.1,
  dot_every_x_iters = 100
)
```

**Arguments**

<code>W_base_object</code>	An object that contains the assignments to begin with sorted by imbalance.
<code>estimator</code>	"linear" for the covariate-adjusted linear regression estimator (default).
<code>q</code>	The tail criterion's quantile of MSE over $z$ 's. The default is 95%.
<code>c_val</code>	The $c$ value used (see Equation 8 in the paper). The default is NULL corresponding to $qnorm(q)$ .
<code>skip_search_length</code>	In the exhaustive search, how many designs are skipped? Default is 1 for full exhaustive search through all assignments provided for in <code>W_base_object</code> .
<code>binary_search</code>	If TRUE, a binary search is employed to find the optimal threshold instead of an exhaustive search. Default is FALSE.
<code>excess_kurtosis_z</code>	An estimate of the excess kurtosis in the measure on $z$ . Default is 0.
<code>use_frob_norm_sq_unbiased_estimator</code>	If TRUE, this would use the debiased Frobenius norm estimator instead of the naive. Default is TRUE.
<code>frob_norm_sq_bias_correction_min_samples</code>	The bias-corrected estimate suffers from high variance when there are not enough samples. Thus, we only implement the correction beginning at this number of vectors. Default is 10 and this parameter is only applicable if <code>use_frob_norm_sq_unbiased_estimator</code> is TRUE.
<code>smoothing_degree</code>	The smoothing degree passed to loess.
<code>smoothing_span</code>	The smoothing span passed to loess.
<code>dot_every_x_iters</code>	Print out a dot every this many iterations. The default is 100. Set to NULL for no printout.

**Value**

A list containing the optimal design threshold, strategy, and other information.

**Author(s)**

Adam Kapelner

**Examples**

```
n = 100
p = 10
X = matrix(rnorm(n * p), nrow = n, ncol = p)
X = apply(X, 2, function(xj){(xj - mean(xj)) / sd(xj)})
S = 25000

W_base_obj = generate_W_base_and_sort(X, max_designs = S)
design = optimal_rerandomization_tail_approx(W_base_obj,
  skip_search_length = 10)
design
```

---

plot.optimal\_rerandomization\_obj

*Plots a summary of a optimal\_rerandomization\_obj object*

---

### Description

Plots a summary of a optimal\_rerandomization\_obj object

### Usage

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

### Arguments

x                    The optimal\_rerandomization\_obj object to be summarized in the plot

...                   The option advanced = TRUE can be passed here for optimal rerandomization results from algorithm type "approx" to see how all the terms in the criterion behave. You can pass s\_min which controls the minimum number of vectors the plot begins at. Below a certain number, the criterion is unstable. Also, title, subtitle, xlab and ylab can be passed here.

### Value

No return value, called for side effects

### Author(s)

Adam Kapelner

---

plot.W\_base\_object

*Plots a summary of the imbalances in a W\_base\_object object*

---

### Description

Plots a summary of the imbalances in a W\_base\_object object

### Usage

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

### Arguments

x                    The W\_base\_object object to be summarized in the plot

...                   title, subtitle, xlab, bins can be specified here to be passed to the ggplot plotting function. Also log10 can be set to FALSE to not log the x-axis.

**Value**

No return value, called for side effects

**Author(s)**

Adam Kapelner

---

`print.optimal_rerandomization_obj`  
*Prints a summary of a `optimal_rerandomization_obj` object*

---

**Description**

Prints a summary of a `optimal_rerandomization_obj` object

**Usage**

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

**Arguments**

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

**Value**

No return value, called for side effects

**Author(s)**

Adam Kapelner

---

`print.W_base_object`    *Prints a summary of a `W_base_object` object*

---

**Description**

Prints a summary of a `W_base_object` object

**Usage**

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

**Arguments**

x                    The W\_base\_object object to be summarized in the console  
...                   Other parameters to pass to the default print function

**Value**

No return value, called for side effects

**Author(s)**

Adam Kapelner

---

summary.optimal\_rerandomization\_obj

*Prints a summary of a optimal\_rerandomization\_obj object*

---

**Description**

Prints a summary of a optimal\_rerandomization\_obj object

**Usage**

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

**Arguments**

object                The optimal\_rerandomization\_obj object to be summarized in the console  
...                    Other parameters to pass to the default summary function

**Author(s)**

Adam Kapelner

---

summary.W\_base\_object *Prints a summary of a W\_base\_object object*

---

### **Description**

Prints a summary of a W\_base\_object object

### **Usage**

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

### **Arguments**

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

### **Author(s)**

Adam Kapelner

# Index

`complete_randomization_plus_one_min_one,`  
    [2](#)  
`complete_randomization_with_forced_balance_plus_one_min_one,`  
    [3](#)  
`compute_objective_val_plus_one_min_one_enc,`  
    [3](#)

`frob_norm_sq,` [4](#)  
`frob_norm_sq_debiased,` [5](#)  
`frob_norm_sq_debiased_times_matrix,` [5](#)

`generate_W_base_and_sort,` [6](#)

`optimal_rerandomization_exact,` [8](#)  
`optimal_rerandomization_normality_assumed,`  
    [9](#)  
`optimal_rerandomization_tail_approx,`  
    [10](#)  
`OptimalRerandExpDesigns,` [7](#)

`plot.optimal_rerandomization_obj,` [12](#)  
`plot.W_base_object,` [12](#)  
`print.optimal_rerandomization_obj,` [13](#)  
`print.W_base_object,` [13](#)

`summary.optimal_rerandomization_obj,`  
    [14](#)  
`summary.W_base_object,` [15](#)