

# Package ‘fastrerandomize’

January 6, 2025

**Title** FastRerandomize: An R Package for Hardware-accelerated Rerandomization for Improved Balance

**Version** 0.1

**Description** Provides hardware-accelerated tools for performing rerandomization and randomization testing in experimental research. Using a JAX backend, the package enables exact rerandomization inference even for large experiments with hundreds of millions of possible randomizations. Key functionalities include generating pools of acceptable rerandomizations based on covariate balance, conducting exact randomization tests, and performing pre-analysis evaluations to determine optimal rerandomization acceptance thresholds. The package supports various hardware acceleration frameworks including CPU, CUDA, and METAL, making it versatile across computing environments. This allows researchers to efficiently implement rerandomization designs and conduct valid inference even with large sample sizes.

**URL** <https://github.com/cjerzak/fastrerandomize-software>

**BugReports** <https://github.com/cjerzak/fastrerandomize-software/issues>

**Depends** R (>= 3.3.3)

**License** GPL-3

**Encoding** UTF-8

**LazyData** false

**Imports** reticulate,  
assertthat

**Suggests** knitr,  
rmarkdown

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

## Contents

build_backend . . . . .	2
fastrerandomize_class . . . . .	3
fastrerandomize_test . . . . .	3
generate_randomizations . . . . .	4
generate_randomizations_exact . . . . .	5
generate_randomizations_mc . . . . .	7

plot.fastrerandomize_randomizations . . . . .	9
plot.fastrerandomize_test . . . . .	9
print.fastrerandomize_randomizations . . . . .	10
print.fastrerandomize_test . . . . .	10
print2 . . . . .	11
QJEDData . . . . .	12
randomization_test . . . . .	12
summary.fastrerandomize_randomizations . . . . .	14
summary.fastrerandomize_test . . . . .	14
YOPData . . . . .	15

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

---

build_backend	<i>Build the environment for fastrerandomize. Builds a conda environment in which jax and np are installed.</i>
---------------	---

---

## Description

Build the environment for fastrerandomize. Builds a conda environment in which jax and np are installed.

## Usage

```
build_backend(conda_env = "fastrerandomize", conda = "auto", tryMetal = T)
```

## Arguments

conda_env	(default = "fastrerandomize") Name of the conda environment in which to place the backends.
conda	(default = auto) The path to a conda executable. Using "auto" allows reticulate to attempt to automatically find an appropriate conda binary.

## Value

Builds the computational environment for fastrerandomize. This function requires an Internet connection. You may find out a list of conda Python paths via: `system("which python")`

## Examples

```
# For a tutorial, see
# github.com/cjerkzak/fastrerandomize-software/
```

---

`fastrerandomize_class` *Constructor for fastrerandomize randomizations*

---

### Description

Create an S3 object of class `fastrerandomize_randomizations` that stores the randomizations (and optionally balance statistics) generated by functions such as [generate\\_randomizations](#).

### Usage

```
fastrerandomize_class(randomizations, balance = NULL, call = NULL)
```

### Arguments

<code>randomizations</code>	A matrix or array where each row (or slice) represents one randomization.
<code>balance</code>	A numeric vector or similar object holding balance statistics for each randomization, or NULL if not applicable.
<code>call</code>	The function call, if you wish to store it for reference (optional).

### Value

An object of class `fastrerandomize_randomizations`.

---

`fastrerandomize_test` *Constructor for fastrerandomize randomization test objects*

---

### Description

Constructor for fastrerandomize randomization test objects

### Usage

```
fastrerandomize_test(p_value, FI, tau_obs, call = NULL, ...)
```

### Arguments

<code>p_value</code>	A numeric value representing the p-value of the test.
<code>FI</code>	A numeric vector (length 2) representing the fiducial interval, or NULL if not requested.
<code>tau_obs</code>	A numeric value (or vector) representing the estimated treatment effect.
<code>call</code>	An optional function call, stored for reference.
<code>...</code>	Other slots you may want to store (e.g. additional diagnostics).

### Value

An object of class `fastrerandomize_test`.

---

generate\_randomizations

*Generate randomizations for experimental design*


---

## Description

This function generates randomizations for experimental design using either exact enumeration or Monte Carlo sampling methods. It provides a unified interface to both approaches while handling memory and computational constraints appropriately.

## Usage

```
generate_randomizations(
  n_units,
  n_treated,
  X = NULL,
  randomization_accept_prob,
  threshold_func = NULL,
  max_draws = 10^6,
  batch_size = 10^5,
  randomization_type = "monte_carlo",
  approximate_inv = TRUE,
  seed = NULL,
  verbose = TRUE,
  file = NULL,
  return_type = "R",
  conda_env = "fastrerandomize",
  conda_env_required = T
)
```

## Arguments

n_units	An integer specifying the total number of experimental units
n_treated	An integer specifying the number of units to be assigned to treatment
X	A numeric matrix of covariates used for balance checking. Cannot be NULL.
randomization_accept_prob	A numeric value between 0 and 1 specifying the probability threshold for accepting randomizations based on balance
threshold_func	A JAX function that computes a balance measure for each randomization. Only used for Monte Carlo sampling.
max_draws	An integer specifying the maximum number of randomizations to draw in Monte Carlo sampling
batch_size	An integer specifying batch size for Monte Carlo processing
randomization_type	A string specifying the type of randomization: either "exact" or "monte_carlo"
seed	An integer seed for random number generation in Monte Carlo sampling
verbose	A logical value indicating whether to print progress information. Default is TRUE
file	A string specifying where to save candidate randomizations (if saving, not returning)

## Details

The function supports two methods of generating randomizations:

1. Exact enumeration: Generates all possible randomizations (memory intensive but exact)
2. Monte Carlo sampling: Generates randomizations through sampling (more memory efficient)

For large problems (e.g.,  $X$  with  $>20$  columns), Monte Carlo sampling is recommended.

## Value

A JAX array containing the accepted randomizations, where each row represents one possible treatment assignment vector

## See Also

[GenerateRandomizations\\_Exact](#) for the exact enumeration method [GenerateRandomizations\\_MonteCarlo](#) for the Monte Carlo sampling method

## Examples

```
# Generate randomizations using exact enumeration
X <- matrix(rnorm(100*5), 100, 5)
RandomizationSet_Exact <- GenerateRandomizations(100, 50, X,
  randomization_accept_prob=0.1,
  randomization_type="exact")

# Generate randomizations using Monte Carlo sampling
RandomizationSet_MC <- GenerateRandomizations(
  n_units = 100,
  n_treated = 50,
  X = X,
  randomization_accept_prob=0.1,
  randomization_type="monte_carlo",
  max_draws=1000)
```

---

generate\_randomizations\_exact

*Generate Complete Randomizations with Optional Balance Constraints*

---

## Description

Generates all possible treatment assignments for a completely randomized experiment, optionally filtering them based on covariate balance criteria. The function can generate either all possible randomizations or a subset that meets specified balance thresholds using Hotelling's  $T^2$  statistic.

**Usage**

```
generate_randomizations_exact(
  n_units,
  n_treated,
  X = NULL,
  randomization_accept_prob = 1,
  approximate_inv = TRUE,
  threshold_func = VectorizedFastHotel2T2,
  seed = NULL,
  file = NULL,
  conda_env = "fastrerandomize",
  conda_env_required = T
)
```

**Arguments**

<code>n_units</code>	An integer specifying the total number of experimental units
<code>n_treated</code>	An integer specifying the number of units to be assigned to treatment
<code>X</code>	A numeric matrix of covariates where rows represent units and columns represent different covariates. Default is <code>NULL</code> , in which case all possible randomizations are returned without balance filtering.
<code>randomization_accept_prob</code>	A numeric value between 0 and 1 specifying the quantile threshold for accepting randomizations based on balance statistics. Default is 1 (accept all randomizations).
<code>threshold_func</code>	A function that calculates balance statistics for candidate randomizations. Default is <code>VectorizedFastHotel2T2</code> which computes Hotelling's $T^2$ statistic.

**Details**

The function works in two main steps: 1. Generates all possible combinations of treatment assignments given `n_units` and `n_treated` 2. If covariates (`X`) are provided, filters these combinations based on balance criteria using the specified threshold function

The balance filtering process uses Hotelling's T-squared statistic by default to measure multivariate balance between treatment and control groups. Randomizations are accepted if their balance measure is below the specified quantile threshold.

**Value**

A JAX NumPy array where each row represents a valid treatment assignment vector (binary: 1 for treated, 0 for control) that meets the balance criteria if specified.

**Note**

This function requires JAX and NumPy to be installed and accessible through the reticulate package. The function assumes the existence of helper functions `InsertOnesVectorized` and `VectorizedFastHotel2T2`.

**References**

Hotelling, H. (1931). The generalization of Student's ratio. The Annals of Mathematical Statistics, 2(3), 360-378.

**See Also**

[VectorizedFastHotel2T2](#) for details on the balance statistic calculation [InsertOnesVectorized](#) for the treatment assignment generation

**Examples**

```
# Generate all possible randomizations for 6 units with 3 treated
rand <- GenerateRandomizations(n_units = 6, n_treated = 3)

# Generate balanced randomizations with covariates
X <- matrix(rnorm(60), nrow = 10) # 10 units, 6 covariates
BalancedRandomizations <- GenerateRandomizations(
  n_units = 10,
  n_treated = 5,
  X = X,
  randomization_accept_prob = 0.25 # Keep top 25% most balanced
)
```

---

```
generate_randomizations_mc
```

*Draws a random sample of acceptable randomizations from all possible complete randomizations using Monte Carlo sampling*

---

**Description**

This function performs sampling with replacement to generate randomizations in a memory-efficient way. It processes randomizations in batches to avoid memory issues and filters them based on covariate balance. The function uses JAX for fast computation and memory management.

**Usage**

```
generate_randomizations_mc(
  n_units,
  n_treated,
  X,
  randomization_accept_prob = 1,
  threshold_func = VectorizedFastHotel2T2,
  max_draws = 1e+05,
  batch_size = 1000,
  seed = NULL,
  approximate_inv = TRUE,
  verbose = FALSE,
  file = NULL,
  conda_env = "fastrerandomize",
  conda_env_required = T
)
```

## Arguments

<code>n_units</code>	An integer specifying the total number of experimental units
<code>n_treated</code>	An integer specifying the number of units to be assigned to treatment
<code>X</code>	A numeric matrix of covariates used for balance checking. Cannot be NULL.
<code>randomization_accept_prob</code>	A numeric value between 0 and 1 specifying the probability threshold for accepting randomizations based on balance. Default is 1
<code>threshold_func</code>	A JAX function that computes a balance measure for each randomization. Must be vectorized using <code>jax.vmap</code> with <code>in_axes = list(NULL, 0L, NULL, NULL)</code> , and inputs <code>covariates</code> (matrix of X), <code>treatment_assignment</code> (vector of 0s and 1s), <code>n0</code> (scalar), <code>n1</code> (scalar). Default is <code>VectorizedFastHotel2T2</code> which uses Hotelling's $T^2$ statistic
<code>max_draws</code>	An integer specifying the maximum number of randomizations to draw. Default is 100000
<code>batch_size</code>	An integer specifying how many randomizations to process at once. Default is 10000. Lower values use less memory but may be slower
<code>seed</code>	An integer seed for random number generation. Default is 42
<code>verbose</code>	A logical value indicating whether to print detailed information about batch processing progress, and GPU memory usage. Default is FALSE

## Details

The function works by:

1. Generating batches of random permutations using JAX's random permutation functionality
2. Computing balance measures for each permutation using the provided threshold function
3. Keeping only the top permutations that meet the acceptance probability threshold
4. Managing memory by clearing unused objects and JAX caches between batches

The function uses smaller data types (int8, float16) where possible to reduce memory usage. It also includes assertions to verify array shapes and dimensions throughout.

## Value

A JAX array containing the accepted randomizations, where each row represents one possible treatment assignment vector

## See Also

[GenerateRandomizations](#) for the non-Monte Carlo version [VectorizedFastHotel2T2](#) for the default threshold function

## Examples

```
# Generate 1000 randomizations for 100 units with 50 treated
X <- matrix(rnorm(100*5), 100, 5) # 5 covariates
rand <- GenerateRandomizations_MonteCarlo(100, 50, X, max_draws=1000)

# Use a stricter balance criterion
rand_strict <- GenerateRandomizations_MonteCarlo(
  n_units = 100,
```



```
n_treated = 50,
X = X,
randomization_accept_prob=0.1,
max_draws=1000)
```

---

plot.fastrerandomize\_randomizations

*Plot method for fastrerandomize\_randomizations objects*


---

### Description

Plot method for fastrerandomize\_randomizations objects

### Usage

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

### Arguments

x	An object of class fastrerandomize_randomizations.
...	Further arguments passed to base <a href="#">plot</a> functions.

---

plot.fastrerandomize\_test

*Plot method for fastrerandomize\_test objects*


---

### Description

Plots a simple visualization of the observed effect and the fiducial interval (if present) on a horizontal axis.

### Usage

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

### Arguments

x	An object of class fastrerandomize_test.
...	Further graphical parameters passed to <a href="#">plot</a> .

---

```
print.fastrerandomize_randomizations
```

*Print method for fastrerandomize\_randomizations objects*

---

### Description

Print method for fastrerandomize\_randomizations objects

### Usage

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

### Arguments

x	An object of class fastrerandomize_instance.
...	Further arguments passed to or from other methods.

---

```
print.fastrerandomize_test
```

*Print method for fastrerandomize\_test objects*

---

### Description

Print method for fastrerandomize\_test objects

### Usage

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

### Arguments

x	An object of class fastrerandomize_test.
...	Further arguments passed to or from other methods.

---

print2	<i>Print timestamped messages with optional quieting</i>
--------	--

---

### Description

This function prints messages prefixed with the current timestamp in a standardized format. Messages can be suppressed using the quiet parameter.

### Usage

```
print2(text, quiet = F)
```

### Arguments

text	A character string containing the message to be printed
quiet	A logical value indicating whether to suppress output. Default is FALSE

### Details

The function prepends the current timestamp in "YYYY-MM-DD HH:MM:SS" format to the provided message.

### Value

No return value, called for side effect of printing

### See Also

`Sys.time` for the underlying timestamp functionality

### Examples

```
# Print a basic message with timestamp
print2("Processing started")

# Suppress output
print2("This won't show", quiet = TRUE)

# Use in a loop
for(i in 1:3) {
  print2(sprintf("Processing item %d", i))
}
```

---

 QJEData

*QJEData: Agricultural Treatment Experiment Data*


---

### Description

Data from a field experiment studying moral hazard in tenancy contracts in agriculture.

### Usage

```
data(QJEData)
```

### Format

A data frame with 968 rows and 2161 column.

**rows** Indicate units

**columns** Indicate RCT data

### Source

Burchardi, K.B., Ghatak, M., & Johanssen, A. (2019). Moral hazard: Experimental evidence from tenancy contracts. *The Quarterly Journal of Economics*, 134(1), 281-347.

---

 randomization\_test

*Fast randomization test*


---

### Description

Fast randomization test

### Usage

```
randomization_test(
  obsW = NULL,
  obsY = NULL,
  X = NULL,
  alpha = 0.05,
  candidate_randomizations = NULL,
  candidate_randomizations_array = NULL,
  n0_array = NULL,
  n1_array = NULL,
  randomization_accept_prob = 1,
  findFI = F,
  c_initial = 2,
  max_draws = 10^6,
  batch_size = 10^5,
  randomization_type = "monte_carlo",
  approximate_inv = TRUE,
  file = NULL,
  conda_env = "fastrerandomize",
  conda_env_required = T
)
```

**Arguments**

obsW	A numeric vector where 0's correspond to control units and 1's to treated units.
obsY	An optional numeric vector of observed outcomes. If not provided, the function assumes a NULL value.
X	A numeric matrix of covariates.
alpha	The significance level for the test. Default is 0.05.
candidate_randomizations	A numeric matrix of candidate randomizations.
candidate_randomizations_array	An optional JAX array of candidate randomizations. If not provided, the function coerces candidate_randomizations into a JAX array.
n0_array	An optional array specifying the number of control units.
n1_array	An optional array specifying the number of treated units.
randomization_accept_prob	An numeric scalar or vector of probabilities for accepting each randomization.
findFI	A logical value indicating whether to find the fiducial interval. Default is FALSE.
c_initial	A numeric value representing the initial criterion for the randomization. Default is 2.
true_treatment_effect	An optional numeric value specifying the true treatment effect. Default is NULL.

**Value**

A list consisting of

- p\_value A numeric value or vector representing the p-value of the test (or the expected p-value under the prior structure specified in the function inputs).
- FI A numeric vector representing the fiducial interval if findFI=T.
- tau\_obs A numeric value or vector representing the estimated treatment effect(s)

**References**

- 

**Examples**

```
# For a tutorial, see
# github.com/cjerkzak/fastrerandomization-software
```

---

```
summary.fastrerandomize_randomizations
```

*Summary method for fastrerandomize\_randomizations objects*

---

### Description

Summary method for fastrerandomize\_randomizations objects

### Usage

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

### Arguments

object	An object of class fastrerandomize_randomizations.
...	Further arguments passed to or from other methods.

### Value

A list with summary statistics, printed by default.

---

```
summary.fastrerandomize_test
```

*Summary method for fastrerandomize\_test objects*

---

### Description

Summary method for fastrerandomize\_test objects

### Usage

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

### Arguments

object	An object of class fastrerandomize_test.
...	Further arguments passed to or from other methods.

### Value

Returns an (invisible) list with detailed test results.

---

YOPData

*YOPData*

---

### Description

Data from a re-analysis of the Youth Opportunities Program anti-poverty RCT in Uganda, with satellite imagery neural representations linked to treatment units.

### Usage

```
data(YOPData)
```

### Format

A list containing two data frames:

**RCTData** Treatment, outcome, and geolocation information

**ImageEmbeddings** CLIP-RSICD neural embeddings of satellite imagery

### Source

- Blattman, C., Fiala, N. and Martinez, S. (2020). The Long-term Impacts of Grants on Poverty: Nine-year Evidence from Uganda's Youth Opportunities Program. *American Economic Review: Insights*, 2(3), 287-304.
- Jerzak, C.T., Johansson, F.D. and Daoud, A. (2023). Image-based Treatment Effect Heterogeneity. *Conference on Causal Learning and Reasoning*, 531-552. *PMLR*.

# Index

build\_backend, [2](#)

fastrerandomize\_class, [3](#)  
fastrerandomize\_test, [3](#)

generate\_randomizations, [3](#), [4](#)  
generate\_randomizations\_exact, [5](#)  
generate\_randomizations\_mc, [7](#)  
GenerateRandomizations, [8](#)  
GenerateRandomizations\_Exact, [5](#)  
GenerateRandomizations\_MonteCarlo, [5](#)

InsertOnesVectorized, [7](#)

plot, [9](#)  
plot.fastrerandomize\_randomizations, [9](#)  
plot.fastrerandomize\_test, [9](#)  
print.fastrerandomize\_randomizations,  
    [10](#)  
print.fastrerandomize\_test, [10](#)  
print2, [11](#)

QJEData, [12](#)

randomization\_test, [12](#)

summary.fastrerandomize\_randomizations,  
    [14](#)  
summary.fastrerandomize\_test, [14](#)

VectorizedFastHotel2T2, [7](#), [8](#)

YOPData, [15](#)