Package ‘fastrerandomize’

January 8, 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 billions 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 accelerated computing environments. This allows researchers to efficiently implement stringent 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,
utils,
assertthat

Suggests knitr,
rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.2

Contents

`build_backend` *An optional function to build the environment for fastrerandomize. Builds a conda environment in which jax and np are installed. Users can also create a conda environment where jax and np are installed themselves.*

Description

An optional function to build the environment for fastrerandomize. Builds a conda environment in which jax and np are installed. Users can also create a conda environment where jax and np are installed themselves.

Usage

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

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: `Sys.which("python")`

Examples

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

`check_jax_availability` *Check if Python and JAX are available*

Description

This function checks if Python and JAX can be accessed via ‘reticulate‘. If not, it returns ‘NULL‘ and prints a message suggesting to run ‘build_backend()‘.

Usage

```
check_jax_availability(conda_env = "fastrerandomize", conda = "auto")
```

Arguments

`conda_env`	A character string specifying the name of the conda environment. Default is ‘"fastrerandomize"‘.
`conda`	The path to a conda executable, or ‘"auto"‘. Default is ‘"auto"‘.

Value

Returns ‘TRUE‘ (invisibly) if both Python and JAX are available; otherwise returns ‘NULL‘.

Examples

```
## Not run:
  check_jax_availability()

## End(Not run)
```

`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

`randomizations`	A matrix or array where each row (or slice) represents one randomization.
`balance`	A numeric vector or similar object holding balance statistics for each randomization, or `NULL` if not applicable.
`call`	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

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

Value

An object of class `fastrerandomize_test`.

`generate_randomizations` *Generate randomizations for a rerandomization-based 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 = TRUE
)
```

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"
`approximate_inv`	A logical value indicating whether to use an approximate inverse (diagonal of the covariance matrix) instead of the full matrix inverse when computing balance metrics. This can speed up computations for high-dimensional covariates. Default is `TRUE`.
`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)
`return_type`	A string specifying the format of the returned randomizations and balance measures. Allowed values are "R" for base R objects (e.g., `matrix`, `numeric`) or "jax" for JAX/NumPy arrays. Default is "R".
`conda_env`	A character string specifying the name of the conda environment to use via `reticulate`. Default is "fastrerandomize".
`conda_env_required`	A logical indicating whether the specified conda environment must be strictly used. If `TRUE`, an error is thrown if the environment is not found. Default is TRUE.

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

`generate_randomizations_exact` for the exact enumeration method `generate_randomizations_mc` for the Monte Carlo sampling method

Examples

```
## Not run:
# Generate synthetic data
X <- matrix(rnorm(20*5), 20, 5)

# Generate randomizations using exact enumeration
RandomizationSet_Exact <- generate_randomizations(
               n_units = nrow(X),
               n_treated = round(nrow(X)/2),
               X = X,
               randomization_accept_prob=0.1,
               randomization_type="exact")

# Generate randomizations using Monte Carlo sampling
RandomizationSet_MC <- generate_randomizations(
               n_units = nrow(X),
               n_treated = round(nrow(X)/2),
               X = X,
               randomization_accept_prob = 0.1,
               randomization_type = "monte_carlo",
               max_draws = 100000,
               batch_size = 1000)

## End(Not run)
```

`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 = NULL,
  seed = NULL,
  file = NULL,
  conda_env = "fastrerandomize",
  conda_env_required = TRUE
)
```

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 where rows represent units and columns represent different covariates. Default is NULL, in which case all possible randomizations are returned without balance filtering.
`randomization_accept_prob`	A numeric value between 0 and 1 specifying the quantile threshold for accepting randomizations based on balance statistics. Default is 1 (accept all randomizations).
`approximate_inv`	A logical value indicating whether to use an approximate inverse (diagonal of the covariance matrix) instead of the full matrix inverse when computing balance metrics. This can speed up computations for high-dimensional covariates. Default is TRUE.
`threshold_func`	A function that calculates balance statistics for candidate randomizations. Default is VectorizedFastHotel2T2 which computes Hotelling's T^2 statistic.
`seed`	An integer seed for random number generation, used when enumerating or filtering exact randomizations with potentially randomized steps (e.g., random draws in thresholding). Default is NULL (no fixed seed).
`file`	A character string specifying the path (including filename) where candidate randomizations will be saved. If `NULL`, the function returns the randomizations in memory. Default is NULL.
`conda_env`	A character string specifying the name of the conda environment to use via `reticulate`. Default is "fastrerandomize".
`conda_env_required`	A logical indicating whether the specified conda environment must be strictly used. If `TRUE`, an error is thrown if the environment is not found. Default is TRUE.

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.

References

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

See Also

`generate_randomizations` for full randomization generation function. `generate_randomizations_mc` for the Monte Carlo version.

Examples

```
## Not run:
# Generate synthetic data
X <- matrix(rnorm(60), nrow = 10)  # 10 units, 6 covariates

# Generate balanced randomizations with covariates
BalancedRandomizations <- generate_randomizations_exact(
  n_units = 10,
  n_treated = 5,
  X = X,
  randomization_accept_prob = 0.25  # Keep top 25% most balanced
)

## End(Not run)
```

`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 = NULL,
  max_draws = 1e+05,
  batch_size = 1000,
  seed = NULL,
  approximate_inv = TRUE,
  verbose = FALSE,
  file = NULL,
  conda_env = "fastrerandomize",
  conda_env_required = TRUE
)
```

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. Default is 1
`threshold_func`	A JAX function that computes a balance measure for each randomization. Must be vectorized using jax$vmap with in_axes = list(NULL, 0L, NULL, NULL), and inputs covariates (matrix of X), treatment_assignment (vector of 0s and 1s), n0 (scalar), n1 (scalar). Default is VectorizedFastHotel2T2 which uses Hotelling's T^2 statistic
`max_draws`	An integer specifying the maximum number of randomizations to draw. Default is 100000
`batch_size`	An integer specifying how many randomizations to process at once. Default is 10000. Lower values use less memory but may be slower
`seed`	An integer seed for random number generation. Default is 42
`approximate_inv`	A logical value indicating whether to use an approximate inverse (diagonal of the covariance matrix) instead of the full matrix inverse when computing balance metrics. This can speed up computations for high-dimensional covariates. Default is `TRUE`.
`verbose`	A logical value indicating whether to print detailed information about batch processing progress, and GPU memory usage. Default is FALSE
`file`	A character string specifying the path (including filename) where candidate randomizations will be saved. If `NULL`, the function returns the randomizations in memory. Default is `NULL`.
`conda_env`	A character string specifying the name of the conda environment to use via `reticulate`. Default is "fastrerandomize".
`conda_env_required`	A logical indicating whether the specified conda environment must be strictly used. If `TRUE`, an error is thrown if the environment is not found. Default is `TRUE`.

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

`generate_randomizations` for full randomization generation function. `generate_randomizations_exact` for the exact version.

Examples

```
## Not run:
# Generate synthetic data
X <- matrix(rnorm(100*5), 100, 5) # 5 covariates

# Generate 1000 randomizations for 100 units with 50 treated
rand_less_strict <- generate_randomizations_mc(
               n_units = 100,
               n_treated = 50,
               X = X,
               randomization_accept_prob=0.01,
               max_draws = 100000,
               batch_size = 1000)

# Use a stricter balance criterion
rand_more_strict <- generate_randomizations_mc(
               n_units = 100,
               n_treated = 50,
               X = X,
               randomization_accept_prob=0.001,
               max_draws = 1000000,
               batch_size = 1000)

## End(Not run)
```

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 `plot` 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 `plot`.

`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 *Print timestamped messages with optional quieting*

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 = FALSE)
```

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.

After subsetting, this dataset includes observations on 968 experimental units with the following variables of interest: household composition, treatment assignment, and agricultural outcomes.

Usage

```
data(QJEData)
```

Format

A data frame with 968 rows and 7 columns:

children Numeric (integer). Number of children in the household. Larger numbers may reflect increased household labor needs and different investment or effort incentives.

married Numeric/binary. Whether the household head is currently married (1) or not (0). Marital status may influence decision-making and risk preferences in farming.

hh_size Numeric (integer). Household size. Differences in family labor availability or consumption needs can influence effort levels and thus relate to moral hazard in production decisions.

hh_sexrat Numeric. The ratio of adult men to adult women in the household. Imbalances in the male–female ratio can affect labor division and investment decisions.

treat1 Numeric/binary. Primary treatment indicator (e.g., whether a farmer is offered a specific tenancy contract or cost-sharing arrangement).

R_yield_ELA_sqm Numeric. Crop yield per square meter (e.g., kilograms of output per square meter). This is a principal outcome measure for evaluating productivity and treatment impact on farm performance.

ELA_Fertil_D Numeric/binary. Indicator for whether fertilizer was used (1) or not (0). This measures input investment—a key mechanism in moral hazard models (farmers may alter input use under different contracts).

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 = FALSE,
  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 = TRUE
)
```

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.

`max_draws`	An integer specifying the maximum number of candidate randomizations to generate (or to consider) for the test when `randomization_type = "monte_carlo"`. Default is `1e6`.
`batch_size`	An integer specifying the batch size for Monte Carlo sampling. Batches are processed one at a time for memory efficiency. Default is `1e5`.
`randomization_type`	A string specifying the type of randomization for the test. Allowed values are "exact" or "monte_carlo". Default is "monte_carlo".
`approximate_inv`	A logical value indicating whether to use an approximate inverse (diagonal of the covariance matrix) instead of the full matrix inverse when computing balance metrics. This can speed up computations for high-dimensional covariates. Default is `TRUE`.
`file`	A character string specifying the path (including filename) where candidate randomizations will be saved or loaded from. If `NULL`, randomizations remain in memory. Default is NULL.
`conda_env`	A character string specifying the name of the conda environment to use via `reticulate`. Default is "fastrerandomize".
`conda_env_required`	A logical indicating whether the specified conda environment must be strictly used. If `TRUE`, an error is thrown if the environment is not found. Default is `TRUE`.

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=TRUE`.
- `tau_obs` A numeric value or vector representing the estimated treatment effect(s)

References

- Zhang, Y. and Zhao, Q., 2023. What is a randomization test?. Journal of the American Statistical Association, 118(544), pp.2928-2942.

See Also

`generate_randomizations` for randomization generation function.

Examples

```
## Not run:
# A small synthetic demonstration with 6 units, 3 treated and 3 controls:

# Seed for reproducability
set.seed(12345)

# Generate pre-treatment covariates
X <- matrix(rnorm(24*2), ncol = 2)

# Generate candidate randomizations
```

```
RandomizationSet_MC <- generate_randomizations(
  n_units = nrow(X),
  n_treated = round(nrow(X)/2),
  X = X,
  randomization_accept_prob = 0.1,
  randomization_type = "monte_carlo",
  max_draws = 100000,
  batch_size = 1000
)

# Generate outcome
W <- RandomizationSet_MC$randomizations[1,]
obsY <- rnorm(nrow(X), mean = 2 * W)

# Perform randomization test
results_base <- randomization_test(
  obsW = W,
  obsY = obsY,
  X = X,
  candidate_randomizations = RandomizationSet_MC$randomizations,
)
print(results_base)

# Perform randomization test
result_fi <- randomization_test(
  obsW = W,
  obsY = obsY,
  X = X,
  candidate_randomizations = RandomizationSet_MC$randomizations,
  findFI = TRUE
)
print(result_fi)

## End(Not run)
```

`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 RCT 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