# Package ‘fastrerandomize’

November 16, 2024

**Title** fastrerandomize: R Package for Ultra-fast Re-randomization Using a JAX Backend

**Version** 0.1

**Author** 'Connor Jerzak <connor.jerzak@gmail.com> [aut, cre], Rebecca Goldstein <rgoldstein@berkeley.edu> [aut]'

**Description** An R Package for Ultra-fast Re-randomization Using a JAX Backend

**Depends** R (>= 3.3.3)

**License**
Creative Commons Attribution-Noncommercial-No Derivative Works 4.0, for academic use only.

**Encoding** UTF-8

**LazyData** false

**Maintainer** 'Connor Jerzak' <connor.jerzak@gmail.com>

**Imports** reticulate

**RoxygenNote** 7.3.2

## Contents

## `GenerateCausalData` *This function generates simulated causal data based on specified parameters. The functional form of the outcome models is:*

$$Y\_0 = X\beta\_0 + \epsilon\_0$$

$$Y\_1 = X\beta\_1 + \tau + \epsilon\_1$$

*where $\tau$ is the treatment effect, which is drawn from a normal distribution with mean* `treatment_effect_mean` *and standard deviation* `treatment_effect_SD`*. The dimension of $\beta\_0$ and $\beta\_1$ is* `k_covars`*. The correlation coefficient of the covariates is* `rho`*. Y0_coefficients and Y1_coefficients are optional arguments that can be provided to specify the coefficients for the control and treated outcome models, and they determine $\beta\_0$ and $\beta\_1$. If they are not provided, the function assumes a NULL value, and the coefficients are drawn from a normal distribution with decreasing variance. Example usage:*

```
GenerateCausalData(n_units = 100, proportion_treated =
0.5, k_covars = 3, rho = 0.5, SD_inherent = 1,
treatment_effect_mean = 0, treatment_effect_SD = 1,
covariates_SD = 1)
```

### Description

This function generates simulated causal data based on specified parameters. The functional form of the outcome models is:

$$Y_0 = X\beta_0 + \epsilon_0$$
$$Y_1 = X\beta_1 + \tau + \epsilon_1$$

where $\tau$ is the treatment effect, which is drawn from a normal distribution with mean `treatment_effect_mean` and standard deviation `treatment_effect_SD`. The dimension of $\beta_0$ and $\beta_1$ is `k_covars`. The correlation coefficient of the covariates is `rho`. Y0_coefficients and Y1_coefficients are optional arguments that can be provided to specify the coefficients for the control and treated outcome models, and they determine $\beta_0$ and $\beta_1$. If they are not provided, the function assumes a NULL value, and the coefficients are drawn from a normal distribution with decreasing variance. Example usage:

```
GenerateCausalData(n_units = 100, proportion_treated = 0.5, k_covars = 3, rho = 0.5, SD_inherent = 1
```

### Usage

```
GenerateCausalData(
  n_units,
  proportion_treated,
  k_covars,
  rho,
  SD_inherent,
  treatment_effect_mean,
  treatment_effect_SD,
  covariates_SD,
  Y0_coefficients = NULL,
  Y1_coefficients = NULL
)
```

## Arguments

`n_units` A numeric value specifying the total number of units in the sample.

`proportion_treated`
A numeric value between 0 and 1 indicating the proportion of units that receive treatment.

`k_covars` A numeric value indicating the number of covariates to be generated.

`rho` A numeric value representing the correlation coefficient of the covariates.

`SD_inherent` A numeric value indicating the standard deviation inherent to the data.

`treatment_effect_mean`
A numeric value representing the mean of the treatment effect.

`treatment_effect_SD`
A numeric value indicating the standard deviation of the treatment effect.

`covariates_SD` A numeric value or vector specifying the standard deviation of the covariates.

`Y0_coefficients`
An optional numeric vector specifying the coefficients for the control outcome model. If not provided, the function assumes a NULL value, and the coefficients are drawn from a normal distribution with decreasing variance.

`Y1_coefficients`
An optional numeric vector specifying the coefficients for the treated outcome model. If not provided, the function assumes a NULL value, and the coefficients are drawn from a normal distribution with decreasing variance.

## Value

A list consisting of

- `data_matrix` A data frame containing the simulated covariates and outcomes for both control (Y0) and treatment (Y1) groups. Access them through `data_matrix$Y0` and `data_matrix$Y1`.
- `Y0_coefficients` A numeric vector representing the coefficients used for the control outcome model.
- `Y1_coefficients` A numeric vector representing the coefficients used for the treated outcome model.

## Examples

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

---

`GenerateRandomizations_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 T² statistic.

### Usage

```
GenerateRandomizations_Exact(
  n_units,
  n_treated,
  X = NULL,
  randomization_accept_prob = 1,
  threshold_func = VectorizedFastHotel2T2
)
```

### 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). |
| threshold_func | A function that calculates balance statistics for candidate randomizations. Default is VectorizedFastHotel2T2 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 $T^2$ 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
balanced_rand <- GenerateRandomizations(
  n_units = 10,
  n_treated = 5,
  X = X,
  randomization_accept_prob = 0.25  # Keep top 25% most balanced
)
```

---

## `GenerateRandomizations_MonteCarlo`

*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

```
GenerateRandomizations_MonteCarlo(
  n_units,
  n_treated,
  X,
  randomization_accept_prob = 1,
  threshold_func = VectorizedFastHotel2T2,
  max_draws = 1e+05,
  seed = 42,
  batch_size = 10000
)
```

### 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

`seed` An integer seed for random number generation. Default is 42

`batch_size` An integer specifying how many randomizations to process at once. Default is 10000. Lower values use less memory but may be slower

### 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(100, 50, X,
               randomization_accept_prob=0.1, max_draws=1000)
```

## `InitializeJAX` *Initialize JAX*

### Description

Initialize JAX

### Usage

```
InitializeJAX(conda_env = NULL, conda_env_required = T)
```

### Arguments

`conda_env` An optioanl character string representing the conda environment to activate. A version of JAX should live in that environment. If NULL, we look in the default Python environment for JAX.

`conda_env_required` A logical representing whether to force use the specified conda environment. Used only if `conda_env` specified.

### Value

This function initializes a JAX-containing conda environment as specified by `conda_env`. This function must be run before any others in `fastrerandomize`.

### Examples

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

## `QJEData` *QJEData*

### Description

The dataset originates from the study "Moral hazard: Experimental evidence from tenancy contracts" by Burchardi, Konrad B et al., published in "The Quarterly Journal of Economics" in 2019 (Volume 134, Issue 1, Pages 281-347).

### Usage

```
QJEData
```

### Format

A data frame with 968 rows and many columns containing treatment data for a Quarterly Journal of Economics experiment on agriculture.

### Source

Burchardi, Konrad B et al. (2019). "Moral hazard: Experimental evidence from tenancy contracts." In: The Quarterly Journal of Economics 134.1, pp. 281–347

RandomizationTest *Fast randomization test*

## Description

Fast randomization test

## Usage

```
RandomizationTest(
  obsW = NULL,
  obsY = NULL,
  X = NULL,
  alpha = 0.05,
  candidate_randomizations = NULL,
  candidate_randomizations_array = NULL,
  n0_array = NULL,
  n1_array = NULL,
  prior_treatment_effect_mean = NULL,
  prior_treatment_effect_SD = NULL,
  true_treatment_effect = NULL,
  simulate = F,
  coef_prior = NULL,
  nSimulate_obsW = 50L,
  nSimulate_obsY = 50L,
  randomization_accept_prob = 1,
  findFI = F,
  c_initial = 2
)
```

## 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.

`prior_treatment_effect_mean` An optional numeric value for the prior mean of the treatment effect. Default is NULL.

`prior_treatment_effect_SD` An optional numeric value for the prior standard deviation of the treatment effect. Default is NULL.

`true_treatment_effect`
An optional numeric value specifying the true treatment effect. Default is NULL.

`simulate` A logical value indicating whether to run `RandomizationTest` in simulation mode. Default is FALSE.

`coef_prior` An optional function generating coefficients on values of `X` for predicting `Y(0)`.

`nSimulate_obsW` A numeric value specifying the number of simulated values for obsW. Default is `50L`.

`nSimulate_obsY` A numeric value specifying the number of simulated values for obsY. Default is `50L`.

`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.

## 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/cjerzak/fastrerandomization-software
```

---

`sanity_check_synthetic_data`

## *Perform sanity checks on synthetic data*

---

## Description

This function performs several sanity checks on synthetic data to ensure the quality of the generated dataset and the strength of relationships between variables.

## Usage

```
sanity_check_synthetic_data(
  synthetic_data,
  InSampleR_threshold = 0.01,
  OOS_R_threshold = 0.01,
  treatment_pval_threshold = 0.05
)
```

## Arguments

`synthetic_data` A list containing:

- data_matrix - Matrix containing the synthetic data
- Y0_coefficients - Coefficients for potential outcome Y0
- Y1_coefficients - Coefficients for potential outcome Y1

`InSampleR_threshold`
A numeric value indicating the threshold for in-sample R-squared.

`OOS_R_threshold`
A numeric value indicating the threshold for out-of-sample R-squared.

`treatment_pval_threshold`
A numeric value indicating the threshold for treatment effect p-value.

## Details

The function performs the following checks:

1. Verifies R-squared > InSampleR_threshold for Y0 and Y1 regressed on X
2. Checks out-of-sample R-squared > OOS_R_threshold for Y0 and Y1 predictions
3. Confirms treatment effect is statistically significant (p < treatment_pval_threshold)

## Value

A list of 4 linear models:

- lm_model_Y0 - Linear model for Y0 ~ X
- lm_model_Y1 - Linear model for Y1 ~ X
- lm_model_obsY - Linear model for observed Y ~ X
- lm_model_obsY_obsW - Linear model for treatment effect

## Examples

```
## Not run:
synthetic_data <- generate_synthetic_data()
models <- sanity_check_synthetic_data(synthetic_data)

## End(Not run)
```

# Index