

Package ‘ResistantProcrustes’

October 10, 2026

Title Resistant Procrustes Methods for Multivariate Data

Version 0.1.1

Imports gllvm, fBasics, gtools

Description Provides resistant alternatives to the standard Procrustes analysis for comparing multivariate datasets, including seven resistant Procrustean methods and their associated permutation tests, as well as the standard Procrustes analysis and its permutation tests. Methods are based on Tang and Jackson (2026) ``Improving and evaluating resistant alternatives to Procrustes analysis for multivariate dataset matching" <[doi:10.1002/env.70073](https://doi.org/10.1002/env.70073)>.

License MIT + file LICENSE

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Xiaozhuo Tang [aut, cre]

Maintainer Xiaozhuo Tang <xiaozhuot@mun.ca>

Repository CRAN

Date/Publication 2026-10-10 11:30:19 UTC

Contents

Biweight_permute	2
Huber_permute	3
ILMS_permute	3
LMS_estimator	5
LMS_LTS_estimator	6
LMS_permute	7
LTS_estimator	8
LTS_permute	9
majorization	9
majorization_tuned	10
MM2_estimator_no_reflection	11
MM2_estimator_with_reflection	11
MM2_permute	12

Procrustes_fun	13
Procrustes_permute	14
S_estimator	15
S_permute	16
Index	18

Biweight_permute	<i>Biweight Method Permutation Test Performs a permutation test for the Biweight method.</i>
------------------	--

Description

Biweight Method Permutation Test Performs a permutation test for the Biweight method.

Usage

Biweight_permute(P_mat, Q_mat, P_N, S0, verbose = FALSE)

Arguments

P_mat	A matrix representing the configuration to be transformed to best align with Q_mat.
Q_mat	A matrix representing the fixed configuration that remains unchanged during the transformation.
P_N	Number of permutations.
S0	Observed objective function value from the Biweight method.
verbose	Logical; if TRUE, displays progress messages during iterations. Defaults to FALSE.

Value

A list containing:

count_n	The number of permuted objective function values less than or equal to the observed value.
p_value	The permutation test p-value.
S	A vector of objective function values obtained from the permutations.

Huber_permute	<i>Huber Method Permutation Test Performs a permutation test for the Huber method.</i>
---------------	--

Description

Huber Method Permutation Test Performs a permutation test for the Huber method.

Usage

```
Huber_permute(P_mat, Q_mat, P_N, S0, verbose = FALSE)
```

Arguments

P_mat	A matrix representing the configuration to be transformed to best align with Q_mat.
Q_mat	A matrix representing the fixed configuration that remains unchanged during the transformation.
P_N	Number of permutations.
S0	Observed objective function value from the Huber method.
verbose	Logical; if TRUE, displays progress messages during iterations. Defaults to FALSE.

Value

A list containing:

count_n	The number of permuted objective function values less than or equal to the observed value.
p_value	The permutation test p-value.
S	A vector of objective function values obtained from the permutations.

ILMS_permute	<i>The Improved Least Median of Squares (ILMS) Permutation Test Performs a permutation test for the ILMS method.</i>
--------------	--

Description

The Improved Least Median of Squares (ILMS) Permutation Test Performs a permutation test for the ILMS method.

Usage

```
ILMS_permute(
  P_mat,
  Q_mat,
  P_N,
  S0,
  nR = 0,
  s_l_LMS = 1e-09,
  s_u_LMS = 10,
  verbose = FALSE
)
```

Arguments

P_mat	A matrix representing the configuration to be transformed to best align with Q_mat.
Q_mat	A matrix representing the fixed configuration that remains unchanged during the transformation.
P_N	Number of permutations.
S0	Observed objective function value from the ILMS method.
nR	Number of randomly sampled subsets. If nR <= 0, all possible subsets are used.
s_l_LMS	Lower bound for the scale s.
s_u_LMS	Upper bound for the scale s.
verbose	Logical; if TRUE, displays progress messages during iterations. Defaults to FALSE.

Value

A list containing:

count_n	The number of permuted objective function values less than or equal to the observed value.
p_value	The permutation test p-value.
S	A vector of objective function values obtained from the permutations.

Examples

```
P_mat <- matrix(c(
  0.3068822, -1.24752606,
  -2.1150143, 4.24520347,
  0.5271354, 1.64468795,
  -0.1264126, -0.08113392,
  -1.4749896, -2.26016700,
  -0.2347269, -1.27047483,
  -0.4833630, 1.43158341,
  -2.3499156, 0.85892382,
  -2.1021129, 0.40267502,
```

```

0.5812934, -0.57210742,
0.7397990, 1.54898604,
-1.2173015, -0.65486036,
-0.6093984, -1.16937177,
0.3833907, -0.72611343,
-0.8423056, 1.22427834
), ncol = 2, byrow = TRUE)

Q_mat <- matrix(c(
  1.96990726, -1.67860936,
  1.16952104, -3.19996198,
  -1.64680983, 3.40009837,
  1.51340633, 0.90762220,
  6.02678651, -1.29463718,
  2.56525182, -0.70441947,
  0.08388971, 3.87254957,
  4.11361309, 4.77780158,
  -0.91329668, 0.01870814,
  0.73714253, -0.50391759,
  -1.60842305, 3.07780435,
  3.90039741, 1.29926458,
  3.31127451, -0.28114418,
  0.85909201, -0.86365691,
  1.52112556, 4.02668249
), ncol = 2, byrow = TRUE)
result_ILMS <- LMS_LTS_estimator(P_mat,Q_mat,nR=0,s_l_LMS=0.000000001,s_u_LMS=10)
S0 <- result_ILMS$s
P_N <- 10
permute_fit <- ILMS_permute(P_mat,Q_mat,P_N,S0,nR=0,s_l_LMS=0.000000001,s_u_LMS=10)

```

LMS_estimator	<i>Least median of squares (LMS) method Align two matrices using the LMS.</i>
---------------	---

Description

Least median of squares (LMS) method Align two matrices using the LMS.

Usage

```
LMS_estimator(X1, X2, nR, s_l_LMS = 1e-09, s_u_LMS = 10)
```

Arguments

X1	A matrix representing the configuration to be transformed to best align with X2.
X2	A matrix representing the fixed configuration that remains unchanged during the transformation.
nR	Number of randomly sampled subsets.
s_l_LMS	Lower bound for the scale s.
s_u_LMS	Upper bound for the scale s.

Value

A list containing the estimated rotation matrix, scaling factor, translation vectors for X1, and scale s.

LMS_LTS_estimator	<i>Improved Least median of squares (ILMS) method Align two matrices using the ILMS.</i>
-------------------	--

Description

Improved Least median of squares (ILMS) method Align two matrices using the ILMS.

Usage

```
LMS_LTS_estimator(X1, X2, nR = 0, s_l_LMS = 1e-09, s_u_LMS = 10)
```

Arguments

X1	A matrix representing the configuration to be transformed to best align with X2.
X2	A matrix representing the fixed configuration that remains unchanged during the transformation.
nR	Number of randomly sampled subsets.
s_l_LMS	Lower bound for the scale s.
s_u_LMS	Upper bound for the scale s.

Value

A list containing the estimated rotation matrix, scaling factor, translation vectors for X1, and scale s.

Examples

```
X1 <- matrix(c(
  0.3068822, -1.24752606,
  -2.1150143, 4.24520347,
  0.5271354, 1.64468795,
  -0.1264126, -0.08113392,
  -1.4749896, -2.26016700,
  -0.2347269, -1.27047483,
  -0.4833630, 1.43158341,
  -2.3499156, 0.85892382,
  -2.1021129, 0.40267502,
  0.5812934, -0.57210742,
  0.7397990, 1.54898604,
  -1.2173015, -0.65486036,
  -0.6093984, -1.16937177,
  0.3833907, -0.72611343,
  -0.8423056, 1.22427834
), ncol = 2, byrow = TRUE)
```

```

X2 <- matrix(c(
  1.96990726, -1.67860936,
  1.16952104, -3.19996198,
  -1.64680983, 3.40009837,
  1.51340633, 0.90762220,
  6.02678651, -1.29463718,
  2.56525182, -0.70441947,
  0.08388971, 3.87254957,
  4.11361309, 4.77780158,
  -0.91329668, 0.01870814,
  0.73714253, -0.50391759,
  -1.60842305, 3.07780435,
  3.90039741, 1.29926458,
  3.31127451, -0.28114418,
  0.85909201, -0.86365691,
  1.52112556, 4.02668249
), ncol = 2, byrow = TRUE)
fit <- LMS_LTS_estimator(X1,X2,nR=0,s_l_LMS=0.000000001,s_u_LMS=10)

```

LMS_permute

The Least Median of Squares (LMS) Permutation Test Performs a permutation test for the LMS method.

Description

The Least Median of Squares (LMS) Permutation Test Performs a permutation test for the LMS method.

Usage

```

LMS_permute(
  P_mat,
  Q_mat,
  P_N,
  S0,
  nR = 0,
  s_l_LMS = 1e-09,
  s_u_LMS = 10,
  verbose = FALSE
)

```

Arguments

P_mat	A matrix representing the configuration to be transformed to best align with Q_mat.
Q_mat	A matrix representing the fixed configuration that remains unchanged during the transformation.

P_N	Number of permutations.
S0	Observed objective function value from the LMS method.
nR	Number of randomly sampled subsets. If nR <= 0, all possible subsets are used.
s_l_LMS	Lower bound for the scale s.
s_u_LMS	Upper bound for the scale s.
verbose	Logical; if TRUE, displays progress messages during iterations. Defaults to FALSE.

Value

A list containing:	
count_n	The number of permuted objective function values less than or equal to the observed value.
p_value	The permutation test p-value.
S	A vector of objective function values obtained from the permutations.

LTS_estimator	<i>Least trimmed squares (LTS) method Align two matrices using the ILMS.</i>
---------------	--

Description

Least trimmed squares (LTS) method Align two matrices using the ILMS.

Usage

LTS_estimator(X1, X2, nR)

Arguments

X1	A matrix representing the configuration to be transformed to best align with X2.
X2	A matrix representing the fixed configuration that remains unchanged during the transformation.
nR	Number of randomly sampled subsets.

Value

A list containing the estimated rotation matrix, scaling factor,translation vectors for X1, and least trimmed squares.

LTS_permute	<i>The Least Trimmed Squares (LTS) Permutation Test Performs a permutation test for the LTS method.</i>
-------------	---

Description

The Least Trimmed Squares (LTS) Permutation Test Performs a permutation test for the LTS method.

Usage

```
LTS_permute(P_mat, Q_mat, P_N, S0, nR = 0, verbose = FALSE)
```

Arguments

P_mat	A matrix representing the configuration to be transformed to best align with Q_mat.
Q_mat	A matrix representing the fixed configuration that remains unchanged during the transformation.
P_N	Number of permutations.
S0	Observed objective function value from the LTS method.
nR	Number of randomly sampled subsets. If nR <= 0, all possible subsets are used.
verbose	Logical; if TRUE, displays progress messages during iterations. Defaults to FALSE.

Value

A list containing:

count_n	The number of permuted objective function values less than or equal to the observed value.
p_value	The permutation test p-value.
S	A vector of objective function values obtained from the permutations.

majorization	<i>Huber and Biweight methods Align two matrices using the Huber and Biweight loss functions via iterative majorization.</i>
--------------	--

Description

Huber and Biweight methods Align two matrices using the Huber and Biweight loss functions via iterative majorization.

Usage

```
majorization(X1, X2, tuning_con, loss)
```

Arguments

X1	A matrix representing the configuration to be transformed to best align with X2.
X2	A matrix representing the fixed configuration that remains unchanged during the transformation.
tuning_con	Tuning constant beyond which the resistant loss functions become flat.
loss	Specifies the loss function to use: "Huber" or "Biweight".

Value

A list containing the estimated rotation matrix, scaling factor, translation vectors for X1, weight matrix, objective function value, and convergence.

majorization_tuned	<i>Huber and Biweight Methods with Automatic Tuning Constant Selection Align two matrices using Huber or biweight loss functions via iterative majorization, with automatic selection of the tuning constant.</i>
--------------------	---

Description

Huber and Biweight Methods with Automatic Tuning Constant Selection Align two matrices using Huber or biweight loss functions via iterative majorization, with automatic selection of the tuning constant.

Usage

```
majorization_tuned(X1, X2, loss)
```

Arguments

X1	A matrix representing the configuration to be transformed to best align with X2.
X2	A matrix representing the fixed configuration that remains unchanged during the transformation.
loss	Specifies the loss function to use: "Huber" or "Biweight".

Value

A list containing the estimated rotation matrix, scaling factor, translation vectors for X1, weight matrix, objective function value, and convergence.

MM2_estimator_no_reflection

Repeated Median Method without reflection Align two matrices using repeated median method that cannot deal with reflection

Description

Repeated Median Method without reflection Align two matrices using repeated median method that cannot deal with reflection

Usage

MM2_estimator_no_reflection(X1, X2)

Arguments

X1	This matrix is fixed during the transformation
X2	This matrix is transformed to fit the other matrix

Value

A list containing rotation matrix, scaling, translation of matrix X1, translation of matrix X2, objective function, and sum of square value

MM2_estimator_with_reflection

Repeated Median Method with reflection Align two matrices using the repeated median method while allowing for reflection. This method is recommended for practical applications where reflected configurations may occur.

Description

Repeated Median Method with reflection Align two matrices using the repeated median method while allowing for reflection. This method is recommended for practical applications where reflected configurations may occur.

Usage

MM2_estimator_with_reflection(X1, X2)

Arguments

X1	A matrix representing the fixed configuration that remains unchanged during the transformation.
X2	A matrix representing the configuration to be transformed to best align with X1.

Value

A list containing the estimated rotation matrix, scaling factor,translation vectors for X1 and X2, objective function value, and the sum of squared errors.

MM2_permute	<i>Repeated Median Method Permutation Test Performs a permutation test for the repeated median method.</i>
-------------	--

Description

Repeated Median Method Permutation Test Performs a permutation test for the repeated median method.

Usage

```
MM2_permute(P_mat, Q_mat, P_N, S0, verbose = FALSE)
```

Arguments

P_mat	A matrix representing the fixed configuration that remains unchanged during the transformation.
Q_mat	A matrix representing the configuration to be transformed to best align with P_mat.
P_N	Number of permutations.
S0	Observed objective function value from the repeated median method.
verbose	Logical; if TRUE, displays progress messages during iterations. Defaults to FALSE.

Value

A list containing:

count_n	The number of permuted objective function values less than or equal to the observed value.
p_value	The permutation test p-value.
S	A vector of objective function values obtained from the permutations.

Procrustes_fun	<i>The standard Procrustes analysis Align two matrices using Procrustes analysis</i>
----------------	--

Description

The standard Procrustes analysis Align two matrices using Procrustes analysis

Usage

```
Procrustes_fun(X1, X2)
```

Arguments

X1	A matrix representing the configuration to be transformed to best align with X2.
X2	A matrix representing the fixed configuration that remains unchanged during the transformation.

Value

A list containing the estimated rotation matrix, scaling factor, translation vectors for X1, and the sum of square values.

Examples

```
X1 <- matrix(c(
  0.3068822, -1.24752606,
  -2.1150143, 4.24520347,
  0.5271354, 1.64468795,
  -0.1264126, -0.08113392,
  -1.4749896, -2.26016700,
  -0.2347269, -1.27047483,
  -0.4833630, 1.43158341,
  -2.3499156, 0.85892382,
  -2.1021129, 0.40267502,
  0.5812934, -0.57210742,
  0.7397990, 1.54898604,
  -1.2173015, -0.65486036,
  -0.6093984, -1.16937177,
  0.3833907, -0.72611343,
  -0.8423056, 1.22427834
), ncol = 2, byrow = TRUE)

X2 <- matrix(c(
  1.96990726, -1.67860936,
  1.16952104, -3.19996198,
  -1.64680983, 3.40009837,
  1.51340633, 0.90762220,
  6.02678651, -1.29463718,
```

```

2.56525182, -0.70441947,
0.08388971, 3.87254957,
4.11361309, 4.77780158,
-0.91329668, 0.01870814,
0.73714253, -0.50391759,
-1.60842305, 3.07780435,
3.90039741, 1.29926458,
3.31127451, -0.28114418,
0.85909201, -0.86365691,
1.52112556, 4.02668249
), ncol = 2, byrow = TRUE)
fit <- Procrustes_fun(X1,X2)

```

Procrustes_permute	<i>Standard Procrustes Analysis Permutation Test Performs a permutation test for the standard Procrustes analysis.</i>
--------------------	--

Description

Standard Procrustes Analysis Permutation Test Performs a permutation test for the standard Procrustes analysis.

Usage

```
Procrustes_permute(P_mat, Q_mat, P_N, S0, verbose = FALSE)
```

Arguments

P_mat	A matrix representing the configuration to be transformed to best align with Q_mat.
Q_mat	A matrix representing the fixed configuration that remains unchanged during the transformation.
P_N	Number of permutations.
S0	Observed objective function value from the Procrustes analysis.
verbose	Logical; if TRUE, displays progress messages during iterations. Defaults to FALSE.

Value

A list containing:

count_n	The number of permuted objective function values less than or equal to the observed value.
p_value	The permutation test p-value.
S	A vector of objective function values obtained from the permutations.

Examples

```

P_mat <- matrix(c(
  0.3068822, -1.24752606,
  -2.1150143, 4.24520347,
  0.5271354, 1.64468795,
  -0.1264126, -0.08113392,
  -1.4749896, -2.26016700,
  -0.2347269, -1.27047483,
  -0.4833630, 1.43158341,
  -2.3499156, 0.85892382,
  -2.1021129, 0.40267502,
  0.5812934, -0.57210742,
  0.7397990, 1.54898604,
  -1.2173015, -0.65486036,
  -0.6093984, -1.16937177,
  0.3833907, -0.72611343,
  -0.8423056, 1.22427834
), ncol = 2, byrow = TRUE)

Q_mat <- matrix(c(
  1.96990726, -1.67860936,
  1.16952104, -3.19996198,
  -1.64680983, 3.40009837,
  1.51340633, 0.90762220,
  6.02678651, -1.29463718,
  2.56525182, -0.70441947,
  0.08388971, 3.87254957,
  4.11361309, 4.77780158,
  -0.91329668, 0.01870814,
  0.73714253, -0.50391759,
  -1.60842305, 3.07780435,
  3.90039741, 1.29926458,
  3.31127451, -0.28114418,
  0.85909201, -0.86365691,
  1.52112556, 4.02668249
), ncol = 2, byrow = TRUE)
result_Procrustes <- Procrustes_fun(P_mat,Q_mat)
S0 <- result_Procrustes$s
P_N <- 10
permute_fit <- Procrustes_permute(P_mat,Q_mat,P_N,S0)

```

S_estimator

*S-estimator method Align two matrices using the S-estimator.***Description**

S-estimator method Align two matrices using the S-estimator.

Usage

```
S_estimator(X1, X2, c_value, A_value, nR, s_l = 1e-09, s_u = 10)
```

Arguments

X1	A matrix representing the configuration to be transformed to best align with X2.
X2	A matrix representing the fixed configuration that remains unchanged during the transformation.
c_value	It controls where the loss function begins to flatten.
A_value	It is a normalization constant used in the S-estimator defining equation.
nR	Number of randomly sampled subsets.
s_l	Lower bound for the scale s.
s_u	Upper bound for the scale s.

Value

A list containing the estimated rotation matrix, scaling factor,translation vectors for X1,and scale s.

S_permute	<i>S-estimator Permutation Test Performs a permutation test for the S-estimator method.</i>
-----------	---

Description

S-estimator Permutation Test Performs a permutation test for the S-estimator method.

Usage

```
S_permute(  
  P_mat,  
  Q_mat,  
  P_N,  
  S0,  
  nR = 0,  
  s_l = 1e-09,  
  s_u = 10,  
  verbose = FALSE  
)
```

Arguments

P_mat	A matrix representing the configuration to be transformed to best align with Q_mat.
Q_mat	A matrix representing the fixed configuration that remains unchanged during the transformation.
P_N	Number of permutations.
S0	Observed objective function value from the S-estimator method.
nR	Number of randomly sampled subsets. If nR <= 0, all possible subsets are used.

<code>s_l</code>	Lower bound for the scale <code>s</code> .
<code>s_u</code>	Upper bound for the scale <code>s</code> .
<code>verbose</code>	Logical; if TRUE, displays progress messages during iterations. Defaults to FALSE.

Value

A list containing:

<code>count_n</code>	The number of permuted objective function values less than or equal to the observed value.
<code>p_value</code>	The permutation test p-value.
<code>S</code>	A vector of objective function values obtained from the permutations.

Index

Biweight_permute, [2](#)

Huber_permute, [3](#)

ILMS_permute, [3](#)

LMS_estimator, [5](#)

LMS_LTS_estimator, [6](#)

LMS_permute, [7](#)

LTS_estimator, [8](#)

LTS_permute, [9](#)

majorization, [9](#)

majorization_tuned, [10](#)

MM2_estimator_no_reflection, [11](#)

MM2_estimator_with_reflection, [11](#)

MM2_permute, [12](#)

Procrustes_fun, [13](#)

Procrustes_permute, [14](#)

S_estimator, [15](#)

S_permute, [16](#)