Package: EDOtrans (via r-universe)

October 19, 2026

Type Package

Title Euclidean Distance-Optimized Data Transformation

Version 0.3.5

Maintainer Jorn Lotsch <j.lotsch@em.uni-frankfurt.de>

Description A data transformation method which takes into account the special property of scale non-invariance with a breakpoint at 1 of the Euclidean distance.

License GPL-3

Encoding UTF-8

LazyData true

LinkingTo Rcpp

Imports methods, stats, utils, cABCanalysis, opGMMassessment, Rcpp (>= 1.0.0)

URL https://github.com/JornLotsch/EDOtrans

Depends R (>= 3.5.0)

Config/pak/sysreqs cmake libgmp3-dev make libicu-dev libuv1-dev libssl-dev zlib1g-dev

Repository https://jornlotsch.r-universe.dev

Date/Publication 2026-05-21 08:20:26 UTC

RemoteUrl https://github.com/jornlotsch/edotrans

RemoteRef HEAD

RemoteSha a96c549c13d9837bad9cd3f1639f79d8683f205c

Contents

EDOtrans 2
FACSdata 3
get_seed 4
GMMartificialData 6

Index **7**

EDOtrans *Euclidean distance-optimized data transformation*

Description

The package provides the necessary functions for performing the EDO data transformation.

Usage

```
EDOtrans(Data, Cls, PlotIt = FALSE, FitAlg = "normalmixEM", Criterion = "LR",
              MaxModes = 8, MaxCores = getOption("mc.cores", 2L), Seed = "simple")
```

Arguments

Data	the data as a numerical vector (1D).
Cls	the class information, if any, as a vector of similar length as instances in the data.
PlotIt	whether to plot the fit directly.
FitAlg	which fit algorithm to use: "ClusterRGMM" = GMM from ClusterR, "densityMclust" from mclust, "DO" from DistributionOptimization (slow), "MCMC" = NMixMCMC from mixAK, or "normalmixEM" from mixtools.
Criterion	which criterion should be used to establish the number of modes from the best GMM fit: "AIC", "BIC", "FM", "GAP", "LR" (likelihood ratio test), "NbClust" (from NbClust), "SI" (Silverman).
MaxModes	for automated GMM assessment: the maximum number of modes to be tried.
MaxCores	for automated GMM assessment: the maximum number of processor cores used under Unix.
Seed	Seed value for reproducibility. Options: `"auto"` (complex seed recovery, slow), `"simple"` (fast reproducible seed, default), or integer (exact control, recommended). Use integers for systematic testing with different seeds, `"simple"` for general use, or `"auto"` when you need to maintain exact RNG state continuity. This parameter is needed because the opGMMassessment function, which is called within EDOtrans, requires a "Seed" parameter for reproducible results.

Value

Returns a list of transformed data and class assignments.

DataEDO	the EDO transformed data.
EDOfactor	the factor by which each data value has been divided.
Cls	the class information for each data instance.

Author(s)

Jorn Lotsch and Alfred Ultsch

References

Ultsch A, Lotsch J. Euclidean distance-optimized data transformation for cluster analysis in biomedical data (EDOtrans). BMC Bioinformatics. 2022 Jun 16;23(1):233, https://doi.org/10.1186/s12859-022-04769-w

Examples

```
## example 1
data(iris)
IrisEDOdata <- EDOtrans(Data = as.vector(iris[,1]), Cls = iris$Species, Seed = 42)

## example 2
data(iris)
IrisEDOdata <-
 EDOtrans(Data = as.vector(iris[,1]), Cls = as.integer(iris$Species), MaxModes = 1, Seed = 42)

## example 3
data(iris)
set.seed(42)
IrisEDOdata <-
  EDOtrans(Data = as.vector(iris[,1]), Cls = iris$Species)
```

FACSdata *Example data of hematologic marker expression.*

Description

Data set of 4 flow cytometry-based lymphoma makers from 1559 cells from healthy subjects (class 1) and 1441 cells from lymphoma patients (class 2).

Usage

```
data("FACSdata")
```

Details

Size 3000 x 4 , stored in `FACSdata$[FS,CDa,CDb,CDd]` Original classes 2, stored in `FACSdata$Cls`

Examples

```
data(FACSdata)
str(FACSdata)
```

get_seed *Random Seed Recovery from RNG State*

Description

Functions for recovering the original seed value that produced the current random number generator state. Provides both R and C++ implementations with the C++ version offering significantly improved performance for large search spaces.

Usage

```
get_seed(range = NULL, fallback_seed = 42, max_search = 2147483647,
  step_size = 50000, use_cpp = TRUE, ...)

get_seed_cpp(range = NULL, fallback_seed = 42, max_search = 2147483647,
  step_size = 50000, batch_size = 10000, verbose = TRUE)
```

Arguments

range	Optional integer vector of specific seed values to search. If provided, only these seeds will be tested instead of systematic range searching.
fallback_seed	Integer seed value to return if no matching seed is found during the search process (default: 42).
max_search	Maximum seed value to search up to when performing systematic range searching. Must be a positive integer within the valid range for R's random number generator (default: 2147483647).
step_size	Step size for systematic range searching when no specific range is provided. Larger values speed up search but may miss the target seed if it falls between steps (default: 50000).
use_cpp	Logical; if TRUE, uses the fast C++ implementation via get_seed_cpp(). If FALSE, falls back to the slower R implementation with a warning (default: TRUE).
batch_size	Integer specifying the number of seeds to process in each C++ batch operation. Larger batches are more memory efficient but require more RAM. Only used in get_seed_cpp() (default: 10000).
verbose	Logical; if TRUE, prints progress information during the search process including batch progress and timing information. Only used in get_seed_cpp() (default: TRUE).
...	Additional arguments passed to get_seed_cpp() when use_cpp = TRUE.

Details

The functions work by systematically testing seed values to find one that reproduces the current RNG state stored in .Random.seed. The search process:

- Tests each candidate seed by setting it and comparing the resulting RNG state

- Uses efficient C++ implementation for faster processing of large search spaces
- Supports both targeted searching (via `range` parameter) and systematic range searching
- Employs batched processing to optimize memory usage and performance

Performance Considerations:

The C++ implementation (`get_seed_cpp()`) provides significant performance improvements:

- Batch processing reduces overhead for large search spaces
- Optimized memory management prevents excessive RAM usage
- Native C++ random number generation matching R's implementation
- Progress reporting for long-running searches

Search Strategy:

- If `range` is provided: Tests only the specified seed values
- If `range` is NULL: Performs systematic search from 1 to `max_search` in steps of `step_size`
- Search terminates immediately when a matching seed is found
- Returns `fallback_seed` if no match is found within the search parameters

Memory Management:

The C++ implementation uses batched processing controlled by `batch_size` to:

- Process large search ranges without excessive memory allocation
- Provide regular progress updates during long searches
- Allow interruption of long-running operations

Value

Returns an integer representing the seed value that reproduces the current random number generator state. If no matching seed is found within the search parameters, returns the `fallback_seed` value.

Note

- Requires an active RNG state (i.e., `.Random.seed` must exist)
- Large search ranges may take considerable time even with C++ optimization
- The search is deterministic but computationally intensive
- Consider using smaller `step_size` values if the initial search fails

See Also

`set.seed`, `.Random.seed`

Examples

```
## Basic seed recovery after generating random numbers
set.seed(123)
recovered_seed <- get_seed()
print(recovered_seed)
```

GMMartificialData *Example data an artificial Gaussioan mixture.*

Description

Dataset of 3000 instances with 3 variables that are Gaussian mixtures and belong to classes Cls = 1, 2, or 3, with different means and standard deviations and equal weights of 0.7, 0.3, and 0.1, respectively.

Usage

```
data("GMMartificialData")
```

Details

Size 3000 x 3, stored in `GMMartificialData$[Var1,Var2,Var3]`

Classes 3, stored in `GMMartificialData$Cls`

Examples

```
data(GMMartificialData)
str(GMMartificialData)
```

Index

∗ **EDOtrans**
EDOtrans, 2
∗ **RNG**
get_seed, 4
∗ **data transformation**
EDOtrans, 2
∗ **random**
get_seed, 4
∗ **reproducibility**
get_seed, 4
∗ **seed**
get_seed, 4
∗ **state recovery**
get_seed, 4
.Random.seed, *5*

EDOtrans, 2

FACSdata, 3

get_seed, 4
get_seed_cpp *(*get_seed*)*, 4
GMMartificialData, 6

set.seed, *5*