Package ‘AMDconfigurations’

May 6, 2026

Type Package

Title Geometric Analysis of Configurations in High-Dimensional Spaces

Version 0.1.0

Description Tools for analysing the geometry of configurations in high-dimensional spaces using the Average Membership Degree (AMD) framework and synthetic configuration generation. The package supports a domain-agnostic approach to studying the shape, dispersion, and internal structure of point clouds, with applications across biological and ecological datasets, including those derived from deep-time records. The AMD framework builds on the idea that strongly coupled systems may occupy a limited set of recurrent regimes in state space, producing high-occupancy regions separated by sparsely populated transitional configurations. The package focuses on detecting these concentration patterns and quantifying their geometric definition without assuming any underlying dynamical model. It provides AMD curve computation, cluster assignment, and sigma-equivalent estimation, together with S3 methods for plotting, printing, and summarising AMD and sigma-equivalent objects. Mendoza (2025) <`https://mmendoza1967.github.io/AMDconfigurations/`>.

License MIT + file LICENSE

URL `https://github.com/mmendoza1967/AMDconfigurations`,
`https://mmendoza1967.github.io/AMDconfigurations/`

BugReports `https://github.com/mmendoza1967/AMDconfigurations/issues`

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1)

Imports e1071, stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown, pkgdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Manuel Mendoza [aut, cre] (ORCID: <https://orcid.org/0000-0002-2143-8138>)

Maintainer Manuel Mendoza <mmendoza@mncn.csic.es>

Repository CRAN

Date/Publication 2026-03-17 18:10:03 UTC

Contents

`assign_configurations` *Fuzzy c-means clustering with multiple random initializations*

Description

This function performs fuzzy c-means clustering on a dataset using multiple random initializations. For each initialization, the algorithm is run using `e1071::cmeans()`, and the solution with the smallest within-cluster objective function is retained.

Usage

```
assign_configurations(
  data,
  c,
  its = 50,
  iter_max = 100,
  m = 2,
  scale_data = FALSE,
  verbose = TRUE
)
```

Arguments

`data`	A numeric matrix or data frame. Rows are samples and columns are features.
`c`	Integer. Number of clusters.
`its`	Integer. Number of random initializations (default: 50).
`iter_max`	Integer. Maximum number of iterations for the c-means algorithm (default: 100).
`m`	Numeric. Fuzziness parameter (default: 2).
`scale_data`	Logical. If TRUE, the data matrix is scaled before clustering.
`verbose`	Logical. If TRUE, progress messages are printed.

Details

The function is independent from the AMD workflow. It simply provides a robust fuzzy clustering procedure by selecting the best solution across several random seeds.

Value

A list of class `"amd_assignment"` containing:

c_opt The number of clusters used.

cluster A vector of hard cluster assignments (1..c).

membership The fuzzy membership matrix.

centers Cluster centers from the best solution.

objective The minimum within-cluster objective value.

Examples

```
X <- matrix(rnorm(2000), ncol = 10)
res <- assign_configurations(X, c = 4)
table(res$cluster)
```

`compute_amd_curve` *Compute the Average Membership Degree (AMD) curve*

Description

This function computes the Average Membership Degree (AMD) curve for fuzzy c-means clustering across a sequence of cluster numbers. For each value of `k`, the algorithm is run multiple times and AMD summarises how sharply samples are assigned to clusters across iterations. The function returns the AMD values (raw, mean, max) and the estimated optimal number of configurations copt.

Usage

```
compute_amd_curve(
  data,
  its,
  nin,
  nsp,
  seeds = NULL,
  verbose = TRUE,
  plot_curve = FALSE,
  open_device = TRUE,
  scale_data = FALSE,
  iter_max = 100,
  m = 2,
  preselect_top_sd = NULL
)
```

Arguments

`data`	A numeric matrix or data frame. Non-numeric columns should be removed beforehand.
`its`	Number of AMD iterations to compute.
`nin`	Minimum number of clusters to evaluate.
`nsp`	Maximum number of clusters to evaluate.
`seeds`	Optional vector of random seeds for reproducibility. If `NULL`, seeds are generated internally.
`verbose`	Logical; if `TRUE`, print progress information.
`plot_curve`	Logical; if `TRUE`, plot the AMD curve.
`open_device`	Logical; if `TRUE`, open a new graphics device. computing AMD.
`scale_data`	Logical; if `TRUE`, standardize the data before computing AMD.
`iter_max`	Maximum number of iterations for the fuzzy c-means algorithm.
`m`	Fuzziness parameter for fuzzy c-means (typically 1.5–2.5).
`preselect_top_sd`	Optional integer. If not `NULL`, restricts AMD computation to the variables with highest standard deviation.

Details

AMD is computed as the mean maximum membership across samples minus $1/k$, which corrects for the expected value under random assignment.

Value

An object of class `"amd_curve"` with components:

k_opt Estimated optimal number of configurations copt.

max Vector of maximum AMD values across iterations for each `k`.

mean Vector of mean AMD values across iterations for each `k`.

raw Matrix of AMD values (iterations × `k`).

coordinates Final membership matrix for `k` = copt.

`estimate_sigma_equivalent`

Estimate the sigma-equivalent value from an AMD curve

Description

This function computes the sigma-equivalent value associated with a given AMD curve. It generates a synthetic sigma-AMDmax sweep, interpolates the relationship between sigma and AMDmax, and finds the sigma value whose AMDmax matches the observed peak in the real data.

Usage

```
estimate_sigma_equivalent(
  real_data,
  its = 10,
  nin = 2,
  nsp = 12,
  c_opt = NULL,
  c_synth = NULL,
  sigmas,
  iter_max = 100,
  make_plot = TRUE,
  return_plot = TRUE,
  quiet = FALSE
)
```

Arguments

`real_data`	A list returned by `compute_amd_curve()`, containing at least `AMDmax` and `sigma_grid`.
`its`	Integer. Number of synthetic AMD curves to generate.
`nin`	Integer. Number of inner iterations used in the synthetic AMD computation.
`nsp`	Integer. Number of spline points used in the synthetic AMD computation.
`c_opt`	Optional integer. Number of configurations used in the real AMD curve.
`c_synth`	Optional integer. Number of configurations used in the synthetic AMD curves.
`sigmas`	Numeric vector of sigma values used to generate the synthetic sweep.
`iter_max`	Integer. Maximum number of iterations for the synthetic AMD computation.
`make_plot`	Logical. If TRUE, a plot of the sigma-AMDmax sweep is generated.
`return_plot`	Logical. If TRUE, the plot object is returned.
`quiet`	Logical. If TRUE, suppresses progress messages.

Value

A list containing:

sigma_equivalent Numeric value of the estimated sigma-equivalent.

sweep Data frame with sigma values and corresponding AMDmax values.

plot A ggplot object, returned only if `return_plot = TRUE`.

Examples

```
# Synthetic dataset
X <- matrix(rnorm(2000), ncol = 10)

# Compute AMD curve with small, fast settings
res <- compute_amd_curve(
  data = X,
  its  = 5,
  nin  = 2,
  nsp  = 5
)

# Sigma-equivalent estimation
estimate_sigma_equivalent(
  real_data = X,
  its       = 5,
  nin       = 2,
  nsp       = 5,
  c_opt     = res$c_opt,
  sigmas    = seq(0.1, 2, length.out = 10),
  iter_max  = 10,
  quiet     = TRUE
)
```

`Fulldata` *Ecological dataset used in the example*

Description

The dataset `Fulldata` contains the ecological data used in the example script `inst/examples/example_ecology.R`. It is included in the package to provide a complete reproducible workflow for the ecological case study.

Usage

```
data("Fulldata")
```

Format

A data frame with multiple variables (see example script for details).

Source

Internal dataset prepared by the authors.

`plot.amd_curve` *Plot method for AMD curve objects*

Description

This method provides a convenient interface to plot AMD curves returned by `compute_amd_curve()`. It forwards the call to `plot_AMD()`, allowing users to simply call `plot(res)`.

Usage

```
## S3 method for class 'amd_curve'
plot(x, type = c("mean", "max"), open_device = TRUE, ...)
```

Arguments

`x`	An object of class `"amd_curve"`.
`type`	Character string indicating which AMD curve to plot: `"mean"` (default) or `"max"`.
`open_device`	Logical; if `TRUE`, open a new graphics device.
`...`	Additional arguments passed to `plot_AMD()`.

Value

Invisibly returns `NULL`.

`plot.amd_sigma` *Plot method for sigma-equivalent objects*

Description

Plots the relationship between tested sigma values and the corresponding synthetic AMD peaks, marking the real AMD peak and the estimated sigma-equivalent value σ_{eq}.

Usage

```
## S3 method for class 'amd_sigma'
plot(x, open_device = TRUE, ...)
```

Arguments

`x`	An object of class `"amd_sigma"` returned by `estimate_sigma_equivalent()`.
`open_device`	Logical; if `TRUE`, open a new graphics device.
`...`	Additional graphical parameters passed to `plot()`.

Details

This method is automatically invoked when calling `plot(x)` on an object returned by `estimate_sigma_equivalent()`.

Value

Invisibly returns `NULL`.

`plot_AMD` *Plot the AMD curve*

Description

This function plots the Average Membership Degree (AMD) curve from an object returned by `compute_amd_curve()`. It visualises either the mean AMD values or the maximum AMD values across iterations, and marks the optimal number of configurations copt.

Usage

```
plot_AMD(amd_object, type = c("mean", "max"), open_device = TRUE, ...)
```

Arguments

`amd_object`	A list returned by `compute_amd_curve()`.
`type`	Character string indicating which AMD curve to plot: • `"mean"` (default): plot mean AMD across iterations. • `"max"`: plot maximum AMD across iterations.
`open_device`	Logical; if `TRUE`, open a new graphics device.
`...`	Additional graphical parameters passed to `plot()`.

Value

Invisibly returns `NULL`. The function is called for its side effect of producing a plot.

Examples

```
X <- matrix(rnorm(2000), ncol = 10)
res <- compute_amd_curve(X, its = 10, nin = 2, nsp = 8)
plot_AMD(res)
plot_AMD(res, type = "max")
```

`print.amd_curve` *Print method for AMD curve objects*

Description

Provides a concise summary of an AMD curve object returned by `compute_amd_curve()`. Displays the optimal number of configurations copt, the AMD peak, and the evaluated range of `k`.

Usage

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

Arguments

`x`	An object of class `"amd_curve"`.
`...`	Unused.

Value

Invisibly returns `x`.

`print.amd_sigma` *Print method for sigma-equivalent objects*

Description

Provides a concise summary of an object returned by `estimate_sigma_equivalent()`, including the real AMD peak, the estimated sigma-equivalent value sigma_eq, and the tested sigma range.

Usage

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

Arguments

`x`	An object of class `"amd_sigma"`.
`...`	Unused.

Value

Invisibly returns `x`.

`summary.amd_curve` *Summary method for AMD curve objects*

Description

Provides a detailed summary of an AMD curve object, including mean and maximum AMD values, the optimal number of configurations, and the evaluated range of `k`.

Usage

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

Arguments

`object`	An object of class "`amd_curve`".
`...`	Unused.

Value

Invisibly returns `object`.

`summary.amd_sigma` *Summary method for sigma-equivalent objects*

Description

Provides a detailed summary of an object returned by `estimate_sigma_equivalent()`, including the real AMD peak, the tested sigma values, the corresponding synthetic AMD peaks, and the estimated sigma-equivalent value σ_{eq}.

Usage

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

Arguments

`object`	An object of class "`amd_sigma`".
`...`	Unused.

Value

Invisibly returns `object`.

`summary.sigma_equivalent` *Summary method for sigma_equivalent objects*

Description

Summary method for sigma_equivalent objects

Usage

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

Arguments

`object`	An object of class "sigma_equivalent".
`...`	Additional arguments (ignored).

Value

Invisibly returns the object.

`Transcdata_small` *Reduced transcriptomic dataset (1000 variables)*

Description

This dataset contains the 1000 variables with highest variance from the original transcriptomic dataset used in the AMDconfigurations package.

Usage

```
data("Transcdata_small")
```

Format

A numeric matrix with 933 rows and 1000 columns.

Details

The full cleaned dataset (~58,000 variables) is available at Zenodo (DOI: https://doi.org/10.5281/zenodo.18604443).

Source

Original dataset cleaned and processed by the authors.

Index