Package: manifoldsR (via r-universe)

November 17, 2026

Title Fast Low-Dimensional Embeddings and Clustering via 'Rust'

Version 0.5.0

Description R interface to the Rust crates 'manifolds-rs', 'ann-search-rs' and 'evoc-rs'. Builds 2D (or higher) embeddings with UMAP (McInnes et al., 2018) <doi:10.48550/arXiv.1802.03426>, t-SNE (van der Maaten and Hinton, 2008) <https://jmlr.org/papers/v9/vandermaaten08a.html>, densMAP and den-SNE (Narayan et al., 2021) <doi:10.1038/s41587-020-00801-7>, PHATE (Moon et al., 2019) <doi:10.1038/s41587-019-0336-3>, PaCMAP (Wang et al., 2021) <https://jmlr.org/papers/v22/20-1061.html>, ForceAtlas2 (Jacomy et al., 2014) <doi:10.1371/journal.pone.0098679>, DREAMS and diffusion maps. Also generates approximate nearest neighbour graphs that every method can reuse, plus EVoC and k-means clustering. Multi-threaded via 'Rayon', with f32 or f64 precision.

License MIT + file LICENSE

Copyright Rust crate authors and licences are listed in inst/AUTHORS.

Encoding UTF-8

Imports checkmate, lifecycle, purrr, stats

Config/rextendr/version 0.5.0

SystemRequirements Cargo (Rust's package manager), rustc >= 1.90, xz

Depends R (>= 4.2)

Suggests igraph, tinytest, zeallot

Config/Needs/website data.table, ggplot2, gridExtra, knitr, magrittr, microbenchmark, phateR, quarto, reticulate, Rtsne, umap, uwot

URL https://gregorlueg.github.io/manifoldsR/, https://github.com/GregorLueg/manifoldsR

BugReports https://github.com/GregorLueg/manifoldsR/issues

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Gregor Lueg [aut, cre], Grant Neilson [aut], Authors of the dependency Rust crates [ctb, cph] (see inst/AUTHORS file)

Maintainer Gregor Lueg <gregorlueg@me.com>

Config/pak/sysreqs xz-utils libclang-dev

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

Date/Publication 2026-11-17 12:20:02 UTC

RemoteUrl https://github.com/cran/manifoldsR

RemoteRef HEAD

RemoteSha 737a49d92568a8a3d9beec72dbb2a1a19a4fbb86

Contents

best_membership . . . 4
calc_ari . . . 4
calc_inertia . . . 5
calc_silhouette_score . . . 6
densmap . . . 6
densne . . . 8
diffusion_maps . . . 10
dreams . . . 11
evoc . . . 13
forceatlas2 . . . 14
forceatlas2_from_graph . . . 15
generate_knn_graph . . . 16
get_centroids . . . 18
get_dist_flat . . . 18
get_dist_mat . . . 19
get_idx_flat . . . 20
get_idx_mat . . . 20
get_layer . . . 21
get_nearest_neighbours . . . 22
kmeans_cluster . . . 22
manifold_synthetic_data . . . 23
membership . . . 24
new_nearest_neighbour . . . 25
pacmap . . . 26
params_clusters . . . 27
params_densmap . . . 28
params_densne . . . 29
params_diffusion_maps . . . 30
params_dreams . . . 31
params_evoc . . . 32
params_fa2 . . . 34
params_hierarchical . . . 36

params_kmeans . . . 37
params_nn . . . 38
params_pacmap . . . 40
params_phate . . . 42
params_swiss_roll . . . 43
params_swiss_roll_biased . . . 44
params_trajectory . . . 45
params_tsne . . . 46
params_umap . . . 47
phate . . . 49
rs_approx_nearest_neighbours . . . 50
rs_ari . . . 51
rs_data_biased_swiss_roll . . . 51
rs_data_clusters . . . 52
rs_data_hierarchical . . . 53
rs_data_swiss_roll . . . 54
rs_data_trajectory . . . 54
rs_densmap . . . 55
rs_densmap_from_knn . . . 56
rs_densne . . . 57
rs_densne_from_knn . . . 58
rs_diffusion_maps . . . 59
rs_diffusion_maps_from_knn . . . 60
rs_dreams . . . 61
rs_dreams_from_knn . . . 62
rs_evoc . . . 63
rs_evoc_from_knn . . . 64
rs_forceatlas2 . . . 65
rs_forceatlas2_from_graph . . . 65
rs_forceatlas2_from_knn . . . 66
rs_inertia . . . 67
rs_k_means . . . 68
rs_k_means_mini_batch . . . 68
rs_pacmap . . . 69
rs_pacmap_from_knn . . . 70
rs_phate . . . 71
rs_phate_from_knn . . . 71
rs_silhouette_score . . . 72
rs_tsne . . . 73
rs_tsne_from_knn . . . 74
rs_umap . . . 75
rs_umap_from_knn . . . 76
tsne . . . 77
umap . . . 78

Index **80**

best_membership *Get the cluster membership at the best persistence score*

Description

Get the cluster membership at the best persistence score

Usage

```
best_membership(x)

## S3 method for class 'Evoc'
best_membership(x)
```

Arguments

x	An Evoc object.

Value

A named list with elements `labels` (integer vector of cluster assignments, `-1` for noise), `membership` (numeric vector of strengths), `layer` (which layer index was selected), and `persistence` (the score).

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
res <- evoc(d$data, .verbose = FALSE)
table(best_membership(res)$cluster)
```

calc_ari *Adjusted Rand index calculation*

Description

Calculates the adjusted Rand index between two sets of cluster assignments.

Usage

```
calc_ari(members_a, members_b)
```

Arguments

members_a	Integer vector. Membership/assignment of the first clustering algorithm.
members_b	Integer vector. Membership/assignment of the second clustering algorithm.

Value

Adjusted Rand index

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
km <- kmeans_cluster(d$data, k = 4L, .verbose = FALSE)
calc_ari(membership(km), d$membership)
```

`calc_inertia` *Calculate the cluster inertia*

Description

Calculate the cluster inertia

Usage

```
calc_inertia(x, data)

## S3 method for class 'KMeansCluster'
calc_inertia(x, data)
```

Arguments

`x`	`KMeansCluster` class.
`data`	Numerical matrix. The original data used to generate the k-means clusters. Shape of samples x features

Value

The cluster inertia

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
km <- kmeans_cluster(d$data, k = 4L, .verbose = FALSE)
calc_inertia(km, d$data)
```

calc_silhouette_score *Calculates the Silhouette score of a given clustering*

Description

If the membership has only one group, the function will return `NaN` for `mean_silhouette` and `silhouette_scores`.

Usage

```
calc_silhouette_score(data, membership)
```

Arguments

`data`	Numerical matrix. Samples x features. The matrix that was used to generate the clustering.
`membership`	Integer vector. Membership/assignment of the clustering algorithm.

Value

A list with the following items:

- mean_silhouette - The average Silhouette scores across all the data points.
- silhouette_scores - The individual Silhouette scores of all the data points.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
km <- kmeans_cluster(d$data, k = 4L, .verbose = FALSE)
res <- calc_silhouette_score(d$data, membership(km))
res$mean_silhouette
```

densmap *Rust-based densMAP*

Description

Performs densMAP dimensionality reduction on the input data. densMAP is UMAP with an added density-preserving term, so a tight cluster stays tight and a diffuse one stays diffuse. Plain UMAP gives you no such guarantee: relative sizes in the embedding mean nothing. This function provides a user-friendly interface with input validation before calling the Rust implementation.

Usage

```
densmap(
  data,
  knn = NULL,
  n_dim = 2L,
  k = 15L,
  min_dist = 0.5,
  spread = 1,
 knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  umap_params = params_umap(),
  dens_params = params_densmap(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

data	Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.
knn	Optional NearestNeighbours class. If provided, densMAP will skip the k-nearest neighbour graph generation and use this one. Defaults to NULL.
n_dim	Integer. Number of dimensions in the embedding space. Defaults to 2L.
k	Integer. Number of nearest neighbours to consider for manifold approximation. Larger values result in more global structure being preserved. Defaults to 15L.
min_dist	Numeric. Minimum distance between points in the embedding. Controls how tightly points are packed. Smaller values result in more clustered embeddings. Must be >= 0. Defaults to 0.5. If you use SGD, consider reducing this!
spread	Numeric. Effective scale of embedded points. Determines the scale at which embedded points will be spread out. Defaults to 1.0.
knn_method	Character. (Approximate) Nearest neighbour method to use. One of "kmknn", "hnsw", "annoy", "nndescent", "balltree", "ivf" or "exhaustive". Defaults to "kmknn".
nn_params	Named list. Nearest neighbour search parameters, see params_nn().
umap_params	Named list. UMAP algorithm parameters, see params_umap().
dens_params	Named list. Density-preservation parameters, see params_densmap().
seed	Integer. Random seed for reproducibility. Defaults to 42L.
use_high_precision	Optional boolean. Gives fine-grained control over fp32 vs fp64 usage.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Details

Setting lambda to 0 in params_densmap() recovers plain umap() exactly. The density term is only active over the final frac of the epochs, which is why it costs comparatively little.

Value

A numerical matrix with dimensions samples x n_dim containing the densMAP embedding.

References

Narayan, Berger & Cho, Nat. Biotechnol., 2021

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- densmap(d$data, .verbose = FALSE)
dim(emb)
```

---

densne *Rust-based den-SNE*

Description

Performs den-SNE dimensionality reduction on the input data. den-SNE is t-SNE with an added density-preserving term, so a tight cluster stays tight and a diffuse one stays diffuse. Plain t-SNE gives you no such guarantee: relative sizes in the embedding mean nothing. This function provides a user-friendly interface with input validation before calling the Rust implementation.

Usage

```
densne(
  data,
  knn = NULL,
  n_dim = 2L,
  perplexity = 20,
  approx_type = c("bh", "bh_qd", "fft", "fft_3k"),
 knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  tsne_params = params_tsne(),
  dens_params = params_densne(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

data	Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.
knn	Optional NearestNeighbours class. If provided, den-SNE will skip the k-nearest neighbour graph generation and use this one. Defaults to NULL.

n_dim	Integer. Number of dimensions in the embedding space. Currently only 2L is supported. Defaults to 2L.
perplexity	Numeric. Perplexity parameter, related to the number of nearest neighbours used in manifold learning. Typical values are between 5 and 50. Defaults to 20.0.
approx_type	Character. Approximation method for computing repulsive forces. One of "bh" for Barnes-Hut, "bh_qd" for the quick-and-dirty Barnes-Hut of qdtsne (tree depth capped at max_depth in params_tsne(), repulsion computed once per leaf), "fft" for FFT-accelerated interpolation or "fft_3k" for the three-kernel variant of the latter (one forward and three inverse FFTs per epoch instead of four each). Defaults to "bh".
knn_method	Character. (Approximate) Nearest neighbour method to use. One of "kmknn", "hnsw", "annoy", "nndescent", "balltree", "ivf" or "exhaustive". Defaults to "kmknn".
nn_params	Named list. Nearest neighbour search parameters, see params_nn().
tsne_params	Named list. t-SNE algorithm parameters, see params_tsne().
dens_params	Named list. Density-preservation parameters, see params_densne().
seed	Integer. Random seed for reproducibility.
use_high_precision	Optional boolean. Gives fine-grained control over fp32 vs fp64 usage.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Details

The number of neighbours will be 3 * perplexity, as this is a usual default in tSNE. Setting lambda to 0 in params_densne() recovers plain tsne() exactly. The default lambda is twenty times smaller than the densMAP one, matching the reference implementations.

Value

A numerical matrix with dimensions samples x n_dim containing the den-SNE embedding.

References

Narayan, Berger & Cho, Nat. Biotechnol., 2021

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- densne(d$data, .verbose = FALSE)
dim(emb)
```

`diffusion_maps` *Rust-based diffusion maps*

Description

Performs diffusion maps dimensionality reduction on the input data. This function provides a user-friendly interface with input validation before calling the Rust implementation.

Usage

```
diffusion_maps(
  data,
  knn = NULL,
  n_dim = 2L,
  k = 5L,
 knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  dm_params = params_diffusion_maps(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

`data`	Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.
`knn`	Optional `NearestNeighbours` class. If provided, diffusion maps will skip the k-nearest neighbour graph generation and use this one. Defaults to `NULL`.
`n_dim`	Integer. Number of dimensions in the embedding space. Defaults to `2L`.
`k`	Integer. Number of nearest neighbours to consider for the affinity graph. Defaults to `5L`.
`knn_method`	Character. (Approximate) Nearest neighbour method to use. One of `"kmknn"`, `"hnsw"`, `"annoy"`, `"nndescent"`, `"balltree"`, `"ivf"` or `"exhaustive"`. Defaults to `"kmknn"`.
`nn_params`	Named list. Nearest neighbour search parameters, see `params_nn()`.
`dm_params`	Named list. Diffusion maps algorithm parameters, see `params_diffusion_maps()`.
`seed`	Integer. Random seed for reproducibility. Defaults to `42L`.
`use_high_precision`	Optional boolean. Gives fine-grained control over `fp32` vs `fp64` usage.
`.verbose`	Logical. Controls verbosity. Defaults to `TRUE`.

Value

A numerical matrix with dimensions samples x n_dim containing the diffusion maps embedding.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- diffusion_maps(d$data, .verbose = FALSE)
dim(emb)
```

`dreams` *Rust-based DREAMS*

Description

Performs DREAMS dimensionality reduction on the input data. DREAMS is t-SNE with a pull towards a global reference embedding, PCA by default. You get the local cluster structure of t-SNE, and the relative positions of the clusters still mean something. Plain t-SNE makes no such promise. This function provides a user-friendly interface with input validation before calling the Rust implementation.

Usage

```
dreams(
  data,
  knn = NULL,
  reference = NULL,
  n_dim = 2L,
  perplexity = 20,
  approx_type = c("bh", "bh_qd", "fft", "fft_3k"),
 knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  tsne_params = params_tsne(),
  dreams_params = params_dreams(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

`data`	Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.
`knn`	Optional `NearestNeighbours` class. If provided, DREAMS will skip the k-nearest neighbour graph generation and use this one. Defaults to `NULL`.
`reference`	Optional numerical matrix or data frame of shape samples x 2. Global reference embedding to regularise towards, e.g. a PHATE or MDS result. If `NULL`, the first two principal components are used. Defaults to `NULL`.
`n_dim`	Integer. Number of dimensions in the embedding space. Currently only `2L` is supported. Defaults to `2L`.

`perplexity` Numeric. Perplexity parameter, related to the number of nearest neighbours used in manifold learning. Typical values are between 5 and 50. Defaults to `20.0`.

`approx_type` Character. Approximation method for computing repulsive forces. One of `"bh"` for Barnes-Hut, `"bh_qd"` for the quick-and-dirty Barnes-Hut of qdtsne (tree depth capped at `max_depth` in `params_tsne()`, repulsion computed once per leaf), `"fft"` for FFT-accelerated interpolation or `"fft_3k"` for the three-kernel variant of the latter (one forward and three inverse FFTs per epoch instead of four each). Defaults to `"bh"`.

`knn_method` Character. (Approximate) Nearest neighbour method to use. One of `"kmknn"`, `"hnsw"`, `"annoy"`, `"nndescent"`, `"balltree"`, `"ivf"` or `"exhaustive"`. Defaults to `"kmknn"`.

`nn_params` Named list. Nearest neighbour search parameters, see `params_nn()`.

`tsne_params` Named list. t-SNE algorithm parameters, see `params_tsne()`.

`dreams_params` Named list. DREAMS parameters, see `params_dreams()`.

`seed` Integer. Random seed for reproducibility.

`use_high_precision` Optional boolean. Gives fine-grained control over `fp32` vs `fp64` usage.

`.verbose` Logical. Controls verbosity. Defaults to `TRUE`.

Details

The number of neighbours will be `3 * perplexity`, as this is a usual default in tSNE. Setting `lambda` to `0` in `params_dreams()` gives t-SNE started from the reference, `1` returns the (scaled) reference. The `init` element of `tsne_params` is ignored: DREAMS always initialises from its reference.

Value

A numerical matrix with dimensions samples x n_dim containing the DREAMS embedding.

References

Kury, Kobak & Damrich, Transactions on Machine Learning Research, 2026

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- dreams(d$data, .verbose = FALSE)
dim(emb)
```

evoc *Rust-based EVoC clustering*

Description

Performs EVoC (Embedding Vector Oriented Clustering) on the input data. Combines a UMAP-like node embedding with HDBSCAN-style density-based clustering and multi-layer persistence analysis.

Usage

```
evoc(
  data,
  knn = NULL,
  n_neighbours = 15L,
 knn_method = c("kmknn", "hnsw", "annoy", "nndescent", "balltree", "ivf", "exhaustive"),
  nn_params = params_nn(),
  evoc_params = params_evoc(),
  return_knn = FALSE,
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

`data`	Numerical matrix or data frame. The data to cluster, of shape samples x features. Will be coerced to a matrix.
`knn`	Optional `NearestNeighbours` class. If provided, EVoC will skip the kNN graph generation and use this one. Defaults to `NULL`.
`n_neighbours`	Integer. Number of nearest neighbours for graph construction. Defaults to `15L`.
`knn_method`	Character. (Approximate) Nearest neighbour method to use. One of `"kmknn"`, `"hnsw"`, `"annoy"`, `"nndescent"`, `"balltree"`, `"ivf"` or `"exhaustive"`. Defaults to `"kmknn"`.
`nn_params`	Named list. Nearest neighbour search parameters, see `params_nn()`.
`evoc_params`	Named list. EVoC algorithm parameters, see `params_evoc()`.
`return_knn`	Logical. Whether to return the kNN graph. If `knn` is provided, it is returned as-is. Defaults to `FALSE`.
`seed`	Integer. Random seed for reproducibility. Defaults to `42L`.
`use_high_precision`	Optional boolean. Gives fine-grained control over `fp32` vs `fp64` usage.
`.verbose`	Logical. Controls verbosity. Defaults to `TRUE`.

Value

An Evoc S3 object (see print.Evoc, best_membership, get_layer).

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
res <- evoc(d$data, .verbose = FALSE)
res
```

`forceatlas2` *Rust-based ForceAtlas2*

Description

Lays out the kNN graph of the data with ForceAtlas2, the way scanpy's `draw_graph` does: the kNN graph is turned into the UMAP fuzzy union graph and every edge pulls, every node pair pushes (`1/d`, with masses `1 + degree`), plus gravity towards the origin. Repulsion runs on a Barnes-Hut tree. Already have a graph? Use `forceatlas2_from_graph()`.

Usage

```
forceatlas2(
  data,
  knn = NULL,
  k = 15L,
 knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  fa2_params = params_fa2(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

`data`	Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.
`knn`	Optional `NearestNeighbours` class. If provided, the k-nearest neighbour search is skipped and this one is used. Defaults to `NULL`.
`k`	Integer. Number of nearest neighbours. Defaults to `15L`.
`knn_method`	Character. (Approximate) Nearest neighbour method to use. One of `"kmknn"`, `"hnsw"`, `"annoy"`, `"nndescent"`, `"balltree"`, `"ivf"` or `"exhaustive"`. Defaults to `"kmknn"`.
`nn_params`	Named list. Nearest neighbour search parameters, see `params_nn()`.
`fa2_params`	Named list. ForceAtlas2 parameters, see `params_fa2()`.

seed	Integer. Random seed for reproducibility. Defaults to 42L.
use_high_precision	Optional boolean. Gives fine-grained control over fp32 vs fp64 usage.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Value

A numerical matrix with dimensions samples x 2 containing the ForceAtlas2 layout.

References

Jacomy, et al., PLoS ONE, 2014

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- forceatlas2(d$data, .verbose = FALSE)
dim(emb)
```

---

forceatlas2_from_graph

Rust-based ForceAtlas2 on a pre-computed graph

Description

Lays out any undirected graph with ForceAtlas2, for example an SNN graph or a graph from another package. Edge weights come from the weight edge attribute; without one, every edge weighs 1. Self-loops are dropped. Only the optimisation parameters in fa2_params are used; the graph and initialisation ones do not apply here.

Usage

```
forceatlas2_from_graph(
  graph,
  init = NULL,
  fa2_params = params_fa2(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

graph	An undirected igraph object without multi-edges. Use igraph::as_undirected(mode = "collapse") or igraph::simplify() to get there. If present, the weight edge attribute needs to be finite and non-negative.
init	Optional numerical matrix of dimensions vcount(graph) x 2. The initial layout, e.g. a UMAP of the same samples. If NULL, a random layout is used. Defaults to NULL.
fa2_params	Named list. ForceAtlas2 parameters, see params_fa2().
seed	Integer. Random seed for reproducibility. Defaults to 42L.
use_high_precision	Optional boolean. Gives fine-grained control over fp32 vs fp64 usage.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Value

A numerical matrix with dimensions vcount(graph) x 2 containing the ForceAtlas2 layout, rows in vertex order.

References

Jacomy, et al., PLoS ONE, 2014

Examples

```
if (requireNamespace("igraph", quietly = TRUE)) {
  g <- igraph::sample_gnp(200L, 0.05)
  emb <- forceatlas2_from_graph(g, .verbose = FALSE)
  dim(emb)
}
```

---

generate_knn_graph *Generate a k-nearest neighbour graph.*

Description

This function generates a kNN graph based on a given numeric matrix. Seven different algorithms are implemented with different speed and accuracy trade-offs:

- "hnsw" - Hierarchical Navigable Small Worlds vector search with slower index generation (which ammortises on larger data sets) and high precision.
- "annoy" - Approximate Nearest Neighbours Oh Yeah algorithm, faster to index but querying on large data sets can be slow.
- "nndescent" - Rust-based implementation of the PyNNDescent algorithm, a good all-rounder that performs well on very large data sets.

- "balltree" - Ball tree structure for exact search with a configurable budget.
- "ivf" - Inverted file index that leverages k-means clustering and probing a few of the clusters.
- "exhaustive" - Exact nearest neighbour search.
- "kmknn" - Fast exact nearest neighbour search using triangle inequality

Usage

```
generate_knn_graph(
  data,
  k,
 knn_method = c("kmknn", "hnsw", "annoy", "nndescent", "balltree", "ivf", "exhaustive"),
  nn_params = params_nn(),
  seed = 42L,
  .verbose = TRUE
)
```

Arguments

data	Numeric matrix. The embedding or feature matrix to compute neighbours on. Rows are observations, columns are features.
k	Integer. The number of nearest neighbours to compute.
knn_method	Character. The algorithm to use for nearest neighbour search. One of c("kmknn", "hnsw", "annoy", "nndescent", "balltree", "ivf", "exhaustive"). Defaults to "kmknn".
nn_params	List. Output of params_nn(). Controls algorithm-specific parameters.
seed	Integer. For reproducibility. Defaults to 42L.
.verbose	Boolean. Controls verbosity.

Value

A nearest neighbours class object with 1-indexed neighbour indices and distances. Distances are on the scale of the metric they are named after, so dist_metric = "euclidean" returns Euclidean distance, not the squared Euclidean the backends sort on internally.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
knn <- generate_knn_graph(d$data, k = 15L, .verbose = FALSE)
knn
```

get_centroids *Get cluster centroids*

Description

Get cluster centroids

Usage

```
get_centroids(x)

## S3 method for class 'KMeansCluster'
get_centroids(x)
```

Arguments

x	A KMeansCluster object.

Value

Numeric matrix of shape k x features.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
km <- kmeans_cluster(d$data, k = 4L, .verbose = FALSE)
dim(get_centroids(km))
```

get_dist_flat *Get the distances as a flat vector*

Description

Get the distances as a flat vector

Usage

```
get_dist_flat(x)

## S3 method for class 'NearestNeighbours'
get_dist_flat(x)
```

Arguments

x	NearestNeighbours class

Value

The distances in a flat format like `c(dist1.1, dist1.2, ... , dist2.1, dist2.2, ... dist3.1, dist3.2)`

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
knn <- generate_knn_graph(d$data, k = 15L, .verbose = FALSE)
head(get_dist_flat(knn))
```

---

get_dist_mat *Get the indices as a matrix*

Description

Get the indices as a matrix

Usage

```
get_dist_mat(x)

## S3 method for class 'NearestNeighbours'
get_dist_mat(x)
```

Arguments

`x`	`NearestNeighbours` class

Value

The indices in samples x neighbours format

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
knn <- generate_knn_graph(d$data, k = 15L, .verbose = FALSE)
dim(get_dist_mat(knn))
```

get_idx_flat *Get the indices as a flat vector*

Description

Get the indices as a flat vector

Usage

```
get_idx_flat(x)

## S3 method for class 'NearestNeighbours'
get_idx_flat(x)
```

Arguments

x	NearestNeighbours class

Value

The indices in a flat format like c(idx1.1, idx1.2, ... , idx.2.1, idx2.2, ... idx3.1, idx3.2)

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
knn <- generate_knn_graph(d$data, k = 15L, .verbose = FALSE)
head(get_idx_flat(knn))
```

get_idx_mat *Get the indices as a matrix*

Description

Get the indices as a matrix

Usage

```
get_idx_mat(x)

## S3 method for class 'NearestNeighbours'
get_idx_mat(x)
```

Arguments

x	NearestNeighbours class

Value

The indices in samples x neighbours format

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
knn <- generate_knn_graph(d$data, k = 15L, .verbose = FALSE)
dim(get_idx_mat(knn))
```

---

get_layer *Get a specific EVoC layer*

Description

Get a specific EVoC layer

Usage

```
get_layer(x, i)

## S3 method for class 'Evoc'
get_layer(x, i)
```

Arguments

x	An Evoc object.
i	Integer. Layer index to retrieve.

Value

A named list with `labels`, `membership`, and `persistence` for the requested layer.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
res <- evoc(d$data, .verbose = FALSE)
str(get_layer(res, 1L))
```

get_nearest_neighbours *Get the kNN graph from an Evoc object*

Description

Get the kNN graph from an Evoc object

Usage

```
get_nearest_neighbours(x)

## S3 method for class 'Evoc'
get_nearest_neighbours(x)
```

Arguments

x	An Evoc object.

Value

A NearestNeighbours object, or NULL if the kNN was not stored.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
res <- evoc(d$data, return_knn = TRUE, .verbose = FALSE)
get_nearest_neighbours(res)
```

kmeans_cluster *K-means clustering*

Description

Performs k-means clustering on the input data. Supports both full Lloyd's iterations (with SIMD/GEMM acceleration) and mini-batch k-means for large data sets.

Usage

```
kmeans_cluster(
  data,
  k,
  method = c("full", "minibatch"),
  kmeans_params = params_kmeans(),
  seed = 42L,
  .verbose = TRUE
)
```

Arguments

`data`	Numerical matrix or data frame. The data to cluster, of shape samples x features. Will be coerced to a matrix.
`k`	Integer. Number of clusters to create. Must be >= 2.
`method`	Character. Clustering method. One of `"full"` (Lloyd's algorithm) or `"minibatch"` (mini-batch k-means). Defaults to `"full"`.
`kmeans_params`	Named list. K-means parameters, see `params_kmeans()`.
`seed`	Integer. Random seed for reproducibility. Defaults to `42L`.
`.verbose`	Logical. Controls verbosity. Defaults to `TRUE`.

Value

A named list with:

centroids Numeric matrix of shape k x features containing the final cluster centroids.

assignments Integer vector of length samples with cluster assignments (1-indexed).

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
km <- kmeans_cluster(d$data, k = 4L, .verbose = FALSE)
km
```

`manifold_synthetic_data` *Generate synthetic data for manifold learning*

Description

A unified wrapper function for generating various types of synthetic data to test manifold learning techniques and demonstrate differences.

Usage

```
manifold_synthetic_data(
  type = c("swiss_roll", "biased_swiss_roll", "clusters", "trajectory", "hierarchical"),
  n_samples,
  dim = 32L,
  seed = 42L,
  parameters = NULL
)
```

Arguments

type	Character. Type of synthetic data to generate. One of: • "swiss_roll" - Swiss roll manifold • "biased_swiss_roll" - Biased Swiss roll manifold • "clusters" - Clustered data • "trajectory" - Trajectory-like data with branching • "hierarchical" - Two-level hierarchical cluster structure
n_samples	Integer. Number of data points to generate.
dim	Integer. Dimensionality of the ambient space. Used for all types except "swiss_roll". Defaults to 32L.
seed	Integer. Seed for reproducibility. Defaults to 42L.
parameters	A named list of type-specific parameters, constructed via params_swiss_roll(), params_swiss_roll_biased() params_clusters(), params_trajectory(), or params_hierarchical(). If NULL, defaults for the chosen type are used. A plain list is accepted but must contain all required fields for the given type.

Value

A list with the following elements:

- data - Numeric matrix of shape n_samples x dim
- membership - Integer vector of cluster/branch assignments (NULL for "swiss_roll")

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
dim(d$data)
table(d$membership)

roll <- manifold_synthetic_data(type = "swiss_roll", n_samples = 500L)
dim(roll$data)
```

---

membership *Get cluster assignments*

Description

Get cluster assignments

Usage

```
membership(x)

## S3 method for class 'KMeansCluster'
membership(x)
```

Arguments

x	A KMeansCluster object.

Value

Integer vector of length samples with cluster assignments (1-indexed).

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
km <- kmeans_cluster(d$data, k = 4L, .verbose = FALSE)
table(membership(km))
```

---

new_nearest_neighbour *Generate a new NearestNeighbours*

Description

Generate a new NearestNeighbours

Usage

```
new_nearest_neighbour(indices, dist, k, n)

generate_nearest_neigbours_class(indices, dist, k, n)
```

Arguments

indices	Integer. Nearest neighbours in flat storage format. Need to be sorted!
dist	Numeric. Nearest neighbour distances in flat storarge format. Need to be sorted! On the scale of the metric they were generated with, i.e. Euclidean rather than squared Euclidean.
k	Integer. Number of k-neighbours per sample.
n	Integer. Number of samples.

Value

Initialised NearestNeighbours class.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
knn <- generate_knn_graph(d$data, k = 15L, .verbose = FALSE)
knn2 <- new_nearest_neighbour(
  indices = get_idx_flat(knn),
  dist = get_dist_flat(knn),
  k = 15L,
  n = 500L
)
knn2
```

---

pacmap *Rust-based PaCMAP*

Description

Performs PaCMAP dimensionality reduction on the input data. This function provides a user-friendly interface with input validation before calling the Rust implementation.

Usage

```
pacmap(
  data,
  knn = NULL,
  n_dim = 2L,
  knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  pacmap_params = params_pacmap(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

data	Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.
knn	Optional NearestNeighbours class. If provided, PaCMAP will skip the k-nearest neighbour graph generation and use this one. Defaults to NULL.
n_dim	Integer. Number of dimensions in the embedding space. Defaults to 2L.
knn_method	Character. (Approximate) Nearest neighbour method to use. One of "kmknn", "hnsw", "annoy", "nndescent", "balltree", "ivf" or "exhaustive". Defaults to "kmknn".
nn_params	Named list. Nearest neighbour search parameters, see params_nn().

pacmap_params	Named list. PaCMAP algorithm parameters, see params_pacmap(). Controls near/mid-near/further pair counts and the kNN search size (via mn_candidate_end).
seed	Integer. Random seed for reproducibility. Defaults to 42L.
use_high_precision	Optional boolean. Gives fine-grained control over fp32 vs fp64 usage.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Value

A numerical matrix with dimensions samples x n_dim containing the PaCMAP embedding.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- pacmap(d$data, .verbose = FALSE)
dim(emb)
```

---

params_clusters *Parameters for clustered data generation*

Description

For use with manifold_synthetic_data().

Usage

```
params_clusters(n_clusters = 15L)
```

Arguments

n_clusters	Integer. Number of clusters to generate. Defaults to 15L.

Value

A named list with the following elements:

- n_clusters - Integer. Number of clusters to generate. Defaults to 15L.

Examples

```
params_clusters()
params_clusters(n_clusters = 5L)
```

`params_densmap` *Wrapper function to generate densMAP parameters*

Description

The density-preservation knobs on top of the usual UMAP parameters. densMAP adds `-lambda * Corr(log Ro, log Re)` to the UMAP loss, where `Ro` is the local radius in the input space and `Re` the matching radius in the embedding. Setting `lambda` to `0` recovers plain UMAP.

Usage

```
params_densmap(lambda = 2, frac = 0.3, var_shift = 0.1)
```

Arguments

`lambda`	Numeric. Weight of the density term. `0` disables it. The default is the densMAP reference value. Defaults to `2.0`.
`frac`	Numeric. Fraction of the total epochs, at the end of the run, over which the density term is active. Defaults to `0.3`.
`var_shift`	Numeric. Additive shift on the variance of the embedding log-radii. Defaults to `0.1`.

Value

A named list with the following elements:

- lambda - Numeric. Weight of the density term. `0` disables it. The default is the densMAP reference value. Defaults to `2.0`.
- frac - Numeric. Fraction of the total epochs, at the end of the run, over which the density term is active. Defaults to `0.3`.
- var_shift - Numeric. Additive shift on the variance of the embedding log-radii. Defaults to `0.1`.

References

Narayan, Berger & Cho, Nat. Biotechnol., 2021

Examples

```
params_densmap()
params_densmap(lambda = 1)
```

params_densne *Wrapper function to generate den-SNE parameters*

Description

The density-preservation knobs on top of the usual t-SNE parameters. den-SNE adds `-lambda * Corr(log Ro, log Re)` to the t-SNE loss, where `Ro` is the local radius in the input space and `Re` the matching radius in the embedding. Setting `lambda` to `0` recovers plain t-SNE.

Usage

```
params_densne(lambda = 0.5, frac = 0.3, var_shift = 0.1)
```

Arguments

`lambda`	Numeric. Weight of the density term. `0` disables it. The default is higher than the den-SNE reference value (original: `0.1`). Defaults to `0.5`.
`frac`	Numeric. Fraction of the total epochs, at the end of the run, over which the density term is active. Defaults to `0.3`.
`var_shift`	Numeric. Additive shift on the variance of the embedding log-radii. Defaults to `0.1`.

Value

A named list with the following elements:

- lambda - Numeric. Weight of the density term. `0` disables it. The default is higher than the den-SNE reference value (original: `0.1`). Defaults to `0.5`.
- frac - Numeric. Fraction of the total epochs, at the end of the run, over which the density term is active. Defaults to `0.3`.
- var_shift - Numeric. Additive shift on the variance of the embedding log-radii. Defaults to `0.1`.

References

Narayan, Berger & Cho, Nat. Biotechnol., 2021

Examples

```
params_densne()
params_densne(lambda = 1)
```

`params_diffusion_maps` *Wrapper function to generate diffusion maps parameters*

Description

Wrapper function to generate diffusion maps parameters

Usage

```
params_diffusion_maps(
  bandwidth_scale = 1,
  thresh = 1e-04,
  graph_symmetry = c("additive", "multiplicative", "mnn", "none"),
  alpha_norm = 1,
  t_max = 100L,
  t_custom = NULL,
  n_landmarks = 2048L,
  landmark_method = c("spectral", "random", "density"),
  n_svd = NULL
)
```

Arguments

`bandwidth_scale`	Numeric. Multiplicative factor applied to the adaptive kernel bandwidth. Defaults to `1.0`.
`thresh`	Numeric. Sparsity threshold applied to kernel entries. Defaults to `1e-04`.
`graph_symmetry`	String. Method used to symmetrise the affinity graph. One of `c("additive", "multiplicative", "mnn", "none")`. Defaults to `"additive"`.
`alpha_norm`	Numeric. Anisotropic density-correction exponent. `0` gives the normalised graph Laplacian, `0.5` the Fokker-Planck operator, `1` the Laplace-Beltrami operator. Defaults to `1.0`.
`t_max`	Integer. Maximum diffusion time considered during automatic selection via Von Neumann entropy knee point detection. Defaults to `100L`.
`t_custom`	Integer or `NULL`. Fixed diffusion time. If set, overrides automatic time selection. Defaults to `NULL`.
`n_landmarks`	Integer or `NULL`. Number of landmarks for compressed diffusion. If `NULL`, the full N x N diffusion operator is used. Defaults to `2048L`.
`landmark_method`	String. Method used to select landmarks. One of `c("spectral", "random", "density")`. Defaults to `"spectral"`.
`n_svd`	Integer or `NULL`. Number of SVD components used in landmark construction. If `NULL`, the library selects a sensible default. Defaults to `NULL`.

Value

A named list with the following elements:

- bandwidth_scale - Numeric. Multiplicative factor applied to the adaptive kernel bandwidth. Defaults to `1.0`.
- thresh - Numeric. Sparsity threshold applied to kernel entries. Defaults to `1e-04`.
- graph_symmetry - String. Method used to symmetrise the affinity graph. One of `c("additive", "multiplicative", "mnn", "none")`. Defaults to `"additive"`.
- alpha_norm - Numeric. Anisotropic density-correction exponent. `0` gives the normalised graph Laplacian, `0.5` the Fokker-Planck operator, `1` the Laplace-Beltrami operator. Defaults to `1.0`.
- t_max - Integer. Maximum diffusion time considered during automatic selection via Von Neumann entropy knee point detection. Defaults to `100L`.
- t_custom - Integer or `NULL`. Fixed diffusion time. If set, overrides automatic time selection. Defaults to `NULL`.
- n_landmarks - Integer or `NULL`. Number of landmarks for compressed diffusion. If `NULL`, the full N x N diffusion operator is used. Defaults to `2048L`.
- landmark_method - String. Method used to select landmarks. One of `c("spectral", "random", "density")`. Defaults to `"spectral"`.
- n_svd - Integer or `NULL`. Number of SVD components used in landmark construction. If `NULL`, the library selects a sensible default. Defaults to `NULL`.

Examples

```
params_diffusion_maps()
params_diffusion_maps(t_max = 50L)
```

---

params_dreams *Wrapper function to generate DREAMS parameters*

Description

The regularisation knob on top of the usual t-SNE parameters. DREAMS pulls every t-SNE epoch towards a scaled global reference embedding (PCA unless one is supplied). Setting `lambda` to `0` recovers plain t-SNE, `1` the reference.

Usage

```
params_dreams(lambda = 0.15)
```

Arguments

`lambda`	Numeric. Regularisation strength. Larger values pull harder towards the reference layout. The default is the DREAMS reference value. Defaults to `0.15`.

Value

A named list with the following elements:

- lambda - Numeric. Regularisation strength. Larger values pull harder towards the reference layout. The default is the DREAMS reference value. Defaults to `0.15`.

References

Kury, Kobak & Damrich, Transactions on Machine Learning Research, 2026

Examples

```
params_dreams()
params_dreams(lambda = 0.3)
```

---

params_evoc *Wrapper function to generate EVoC parameters*

Description

Wrapper function to generate EVoC parameters

Usage

```
params_evoc(
  noise_level = 0.5,
  n_epochs = 50L,
  embedding_dim = NULL,
  neighbour_scale = 1,
  symmetrise = TRUE,
  min_samples = 5L,
  base_min_cluster_size = 5L,
  approx_n_clusters = NULL,
  min_similarity_threshold = 0.2,
  max_layers = 10L
)
```

Arguments

`noise_level`	Numeric. Noise level for the embedding gradient. `0.0` = aggressive, `1.0` = conservative. Defaults to `0.5`.
`n_epochs`	Integer. Number of embedding optimisation epochs. Defaults to `50L`.
`embedding_dim`	Integer or `NULL`. Embedding dimensionality. If `NULL`, uses `min(max(n_neighbours / 4, 4), 16)`. Defaults to `NULL`.
`neighbour_scale`	Numeric. Multiplier on effective neighbours for fuzzy graph construction. Defaults to `1.0`.

`symmetrise` Boolean. Whether to symmetrise the fuzzy graph. Defaults to `TRUE`.

`min_samples` Integer. Minimum samples for core distance in MST density estimation. Defaults to `5L`.

`base_min_cluster_size` Integer. Base minimum cluster size for the finest layer. Defaults to `5L`.

`approx_n_clusters` Integer or `NULL`. If set, binary-searches for approximately this many clusters (single layer output). Defaults to `NULL`.

`min_similarity_threshold` Numeric. Jaccard similarity threshold for filtering redundant layers. Defaults to `0.2`.

`max_layers` Integer. Maximum number of cluster layers to return. Defaults to `10L`.

Value

A named list with the following elements:

- noise_level - Numeric. Noise level for the embedding gradient. `0.0` = aggressive, `1.0` = conservative. Defaults to `0.5`.
- n_epochs - Integer. Number of embedding optimisation epochs. Defaults to `50L`.
- embedding_dim - Integer or `NULL`. Embedding dimensionality. If `NULL`, uses `min(max(n_neighbours / 4, 4), 16)`. Defaults to `NULL`.
- neighbour_scale - Numeric. Multiplier on effective neighbours for fuzzy graph construction. Defaults to `1.0`.
- symmetrise - Boolean. Whether to symmetrise the fuzzy graph. Defaults to `TRUE`.
- min_samples - Integer. Minimum samples for core distance in MST density estimation. Defaults to `5L`.
- base_min_cluster_size - Integer. Base minimum cluster size for the finest layer. Defaults to `5L`.
- approx_n_clusters - Integer or `NULL`. If set, binary-searches for approximately this many clusters (single layer output). Defaults to `NULL`.
- min_similarity_threshold - Numeric. Jaccard similarity threshold for filtering redundant layers. Defaults to `0.2`.
- max_layers - Integer. Maximum number of cluster layers to return. Defaults to `10L`.

Examples

```
params_evoc()
params_evoc(min_samples = 10L, base_min_cluster_size = 10L)
```

params_fa2 *Wrapper function to generate ForceAtlas2 parameters*

Description

The graph, initialisation and optimisation knobs for ForceAtlas2. `local_connectivity`, `bandwidth`, `init` and `randomised` only apply when the graph is built from the data via `forceatlas2()`; `forceatlas2_from_graph()` ignores them.

Usage

```
params_fa2(
  local_connectivity = 1,
  bandwidth = 1e-05,
  n_epochs = 500L,
  scaling_ratio = 2,
  gravity = 1,
  strong_gravity = FALSE,
  lin_log = FALSE,
  dissuade_hubs = FALSE,
  edge_weight_influence = 1,
  jitter_tolerance = 1,
  theta = 1.2,
  init = c("spectral", "pca", "random"),
  randomised = FALSE
)
```

Arguments

`local_connectivity`	Numeric. Number of nearest neighbours assumed to be at distance zero. Defaults to `1.0`.
`bandwidth`	Numeric. Convergence tolerance for smooth kNN distance binary search. Defaults to `1e-05`.
`n_epochs`	Integer. Number of optimisation epochs. Defaults to `500L`.
`scaling_ratio`	Numeric. Repulsion strength. Larger values spread the layout. Defaults to `2.0`.
`gravity`	Numeric. Pull towards the origin. Keeps disconnected components from drifting off. Defaults to `1.0`.
`strong_gravity`	Boolean. Distance-independent gravity, scaled by `scaling_ratio`. Defaults to `FALSE`.
`lin_log`	Boolean. Logarithmic attraction (LinLog). Gives tighter communities. Defaults to `FALSE`.
`dissuade_hubs`	Boolean. Divide the attraction by the node mass. Pushes hubs to the periphery. Defaults to `FALSE`.

edge_weight_influence
: Numeric. Exponent applied to the edge weights. `0` ignores the weights. Defaults to `1.0`.

jitter_tolerance
: Numeric. Tolerated swinging. Larger is faster but less precise. Defaults to `1.0`.

theta
: Numeric. Barnes-Hut opening parameter on Gephi's scale. `0` gives the exact repulsion. Defaults to `1.2`.

init
: String. Embedding initialisation method. One of `c("spectral", "pca", "random")`. Defaults to `"spectral"`.

randomised
: Boolean. Use randomised SVD for PCA initialisation. Defaults to `FALSE`.

Value

A named list with the following elements:

- local_connectivity - Numeric. Number of nearest neighbours assumed to be at distance zero. Defaults to `1.0`.
- bandwidth - Numeric. Convergence tolerance for smooth kNN distance binary search. Defaults to `1e-05`.
- n_epochs - Integer. Number of optimisation epochs. Defaults to `500L`.
- scaling_ratio - Numeric. Repulsion strength. Larger values spread the layout. Defaults to `2.0`.
- gravity - Numeric. Pull towards the origin. Keeps disconnected components from drifting off. Defaults to `1.0`.
- strong_gravity - Boolean. Distance-independent gravity, scaled by `scaling_ratio`. Defaults to `FALSE`.
- lin_log - Boolean. Logarithmic attraction (LinLog). Gives tighter communities. Defaults to `FALSE`.
- dissuade_hubs - Boolean. Divide the attraction by the node mass. Pushes hubs to the periphery. Defaults to `FALSE`.
- edge_weight_influence - Numeric. Exponent applied to the edge weights. `0` ignores the weights. Defaults to `1.0`.
- jitter_tolerance - Numeric. Tolerated swinging. Larger is faster but less precise. Defaults to `1.0`.
- theta - Numeric. Barnes-Hut opening parameter on Gephi's scale. `0` gives the exact repulsion. Defaults to `1.2`.
- init - String. Embedding initialisation method. One of `c("spectral", "pca", "random")`. Defaults to `"spectral"`.
- randomised - Boolean. Use randomised SVD for PCA initialisation. Defaults to `FALSE`.

References

Jacomy, et al., PLoS ONE, 2014

Examples

```
params_fa2()
params_fa2(n_epochs = 200L, lin_log = TRUE)
```

`params_hierarchical` *Parameters for hierarchical cluster data generation*

Description

For use with `manifold_synthetic_data()`.

Usage

```
params_hierarchical(
  n_supergroups = 3L,
  n_subclusts = 3L,
  supergroup_spread = 15,
  subcluster_spread = 2,
  point_std = 0.4
)
```

Arguments

`n_supergroups`	Integer. Number of top-level groups. Defaults to `3L`.
`n_subclusts`	Integer. Number of subclusters per supergroup. Defaults to `3L`.
`supergroup_spread`	Numeric. Spread of supergroup centres in the ambient space. Defaults to `15.0`.
`subcluster_spread`	Numeric. Spread of subcluster centres around their supergroup centre. Defaults to `2.0`.
`point_std`	Numeric. Within-subcluster Gaussian noise. Defaults to `0.4`.

Value

A named list with the following elements:

- n_supergroups - Integer. Number of top-level groups. Defaults to `3L`.
- n_subclusts - Integer. Number of subclusters per supergroup. Defaults to `3L`.
- supergroup_spread - Numeric. Spread of supergroup centres in the ambient space. Defaults to `15.0`.
- subcluster_spread - Numeric. Spread of subcluster centres around their supergroup centre. Defaults to `2.0`.
- point_std - Numeric. Within-subcluster Gaussian noise. Defaults to `0.4`.

Examples

```
params_hierarchical()
params_hierarchical(n_supergroups = 4L, n_subclusts = 2L)
```

params_kmeans *Wrapper function to generate k-means parameters*

Description

Wrapper function to generate k-means parameters

Usage

```
params_kmeans(
  metric = c("euclidean", "cosine"),
  max_iters = 1000L,
  batch_size = 4096L,
  drift_threshold = 1e-04,
  lr_alpha = 1,
  init = c("parallel", "random"),
  use_hamerly = NULL,
  use_gemm = NULL
)
```

Arguments

metric	String. Distance metric to use. One of `c("euclidean", "cosine")`. Defaults to `"euclidean"`.
max_iters	Integer. Maximum number of iterations. Defaults to `1000L`.
batch_size	Integer. Mini-batch size. Only used when `method = "minibatch"`. Defaults to `4096L`.
drift_threshold	Numeric. Below which centroid drift the mini-batch k-means is considered converged. Only used when `method = "minibatch"`. Defaults to `1e-04`.
lr_alpha	Numeric. Learning rate decay for the mini-batch k-means. Original paper uses `1.0`. Defaults to `1.0`.
init	String. The initialisation of the centroids. One of `c("parallel", "random")`. Defaults to `"parallel"`.
use_hamerly	Boolean or `NULL`. Shall Hamerly's method be used (only if `metric == "euclidean"`). Defaults to `NULL`.
use_gemm	Boolean or `NULL`. Shall the GEMM path be used. Useful on high dimensional data. If `NULL`, choice will be based on heuristics. Defaults to `NULL`.

Value

A named list with the following elements:

- metric - String. Distance metric to use. One of `c("euclidean", "cosine")`. Defaults to `"euclidean"`.

- max_iters - Integer. Maximum number of iterations. Defaults to `1000L`.
- batch_size - Integer. Mini-batch size. Only used when `method = "minibatch"`. Defaults to `4096L`.
- drift_threshold - Numeric. Below which centroid drift the mini-batch k-means is considered converged. Only used when `method = "minibatch"`. Defaults to `1e-04`.
- lr_alpha - Numeric. Learning rate decay for the mini-batch k-means. Original paper uses `1.0`. Defaults to `1.0`.
- init - String. The initialisation of the centroids. One of `c("parallel", "random")`. Defaults to `"parallel"`.
- use_hamerly - Boolean or `NULL`. Shall Hamerly's method be used (only if `metric == "euclidean"`). Defaults to `NULL`.
- use_gemm - Boolean or `NULL`. Shall the GEMM path be used. Useful on high dimensional data. If `NULL`, choice will be based on heuristics. Defaults to `NULL`.

Examples

```
params_kmeans()
params_kmeans(metric = "cosine", max_iters = 100L)
```

`params_nn` *Wrapper function to generate nearest neighbour parameters*

Description

Wrapper function to generate nearest neighbour parameters

Usage

```
params_nn(
  dist_metric = c("euclidean", "cosine"),
  n_tree = 50L,
  search_budget = NULL,
  m = 16L,
  ef_construction = 100L,
  ef_search = 100L,
  diversify_prob = 0,
  delta = 0.001,
  ef_budget = NULL,
  extract_knn = TRUE,
  bt_budget = 0.1,
  n_list = NULL,
  n_probes = NULL
)
```

Arguments

dist_metric	String. The distance metric to use. One of c("euclidean", "cosine"). Defaults to "euclidean".
n_tree	Integer. Number of trees for Annoy. Defaults to 50L.
search_budget	Integer or NULL. Search budget for Annoy. Defaults to NULL.
m	Integer. Number of bidirectional links for HNSW. Defaults to 16L.
ef_construction	Integer. Size of the dynamic candidate list during HNSW construction. Defaults to 100L.
ef_search	Integer. Size of the dynamic candidate list during HNSW search. Defaults to 100L.
diversify_prob	Numeric. Diversification probability for NN descent. Defaults to 0.0.
delta	Numeric. Precision parameter for NN descent. Defaults to 0.001.
ef_budget	Integer or NULL. Effort budget for NN descent. Ignored when extract_knn is TRUE, as no search runs. Defaults to NULL.
extract_knn	Boolean. Only affects the "nndescent" backend. If TRUE, the descent hands back the graph it just built instead of running a beam search over it. Faster, at a recall of roughly 0.98 rather than 0.99-1.00. Defaults to TRUE.
bt_budget	Numeric. Budget for ball tree search. Defaults to 0.1.
n_list	Integer or NULL. Number of clusters to use for IVF. If NULL, uses sqrt(n). Defaults to NULL.
n_probes	Integer or NULL. Number of clusters to probe for IVF. If NULL, uses sqrt(n_list). Defaults to NULL.

Value

A named list with the following elements:

- dist_metric - String. The distance metric to use. One of c("euclidean", "cosine"). Defaults to "euclidean".
- n_tree - Integer. Number of trees for Annoy. Defaults to 50L.
- search_budget - Integer or NULL. Search budget for Annoy. Defaults to NULL.
- m - Integer. Number of bidirectional links for HNSW. Defaults to 16L.
- ef_construction - Integer. Size of the dynamic candidate list during HNSW construction. Defaults to 100L.
- ef_search - Integer. Size of the dynamic candidate list during HNSW search. Defaults to 100L.
- diversify_prob - Numeric. Diversification probability for NN descent. Defaults to 0.0.
- delta - Numeric. Precision parameter for NN descent. Defaults to 0.001.
- ef_budget - Integer or NULL. Effort budget for NN descent. Ignored when extract_knn is TRUE, as no search runs. Defaults to NULL.
- extract_knn - Boolean. Only affects the "nndescent" backend. If TRUE, the descent hands back the graph it just built instead of running a beam search over it. Faster, at a recall of roughly 0.98 rather than 0.99-1.00. Defaults to TRUE.

- bt_budget - Numeric. Budget for ball tree search. Defaults to `0.1`.
- n_list - Integer or `NULL`. Number of clusters to use for IVF. If `NULL`, uses `sqrt(n)`. Defaults to `NULL`.
- n_probes - Integer or `NULL`. Number of clusters to probe for IVF. If `NULL`, uses `sqrt(n_list)`. Defaults to `NULL`.

Examples

```
params_nn()
params_nn(dist_metric = "cosine", ef_search = 200L)
```

`params_pacmap` *Wrapper function to generate PaCMAP parameters*

Description

Wrapper function to generate PaCMAP parameters

Usage

```
params_pacmap(
  n_near = 10L,
  n_mid_near = 5L,
  n_further = 20L,
  mn_candidate_start = 4L,
  mn_candidate_end = 50L,
  init = c("pca", "random"),
  optimiser = c("adam_parallel", "adam"),
  lr = 1,
  n_epochs = NULL,
  beta1 = 0.9,
  beta2 = 0.999,
  eps = 1e-07,
  phase1_end = NULL,
  phase2_end = NULL
)
```

Arguments

`n_near`	Integer. Near pairs per point (attractive). Defaults to `10L`.
`n_mid_near`	Integer. Mid-near pairs per point. Defaults to `5L`.
`n_further`	Integer. Further (random) pairs per point. Defaults to `20L`.
`mn_candidate_start`	Integer. Start index into kNN list for mid-near candidate window. Defaults to `4L`.

mn_candidate_end
: Integer. End index into kNN list for mid-near candidate window. Also determines the kNN search size. Defaults to `50L`.

init
: String. Embedding initialisation. One of `c("pca", "random")`. Defaults to `"pca"`.

optimiser
: String. The optimiser. One of `c("adam_parallel", "adam")`. Defaults to `"adam_parallel"`.

lr
: Numeric. Adam learning rate. Defaults to `1.0`.

n_epochs
: Integer or `NULL`. Total optimisation epochs. If `NULL`, resolved downstream to `450`. Defaults to `NULL`.

beta1
: Numeric. Adam first moment decay. Defaults to `0.9`.

beta2
: Numeric. Adam second moment decay. Defaults to `0.999`.

eps
: Numeric. Adam numerical stability constant. Defaults to `1e-07`.

phase1_end
: Integer or `NULL`. Epoch at which phase 1 ends. If `NULL`, resolved downstream to `100`. Defaults to `NULL`.

phase2_end
: Integer or `NULL`. Epoch at which phase 2 ends. If `NULL`, resolved downstream to `200`. Defaults to `NULL`.

Value

A named list with the following elements:

- n_near - Integer. Near pairs per point (attractive). Defaults to `10L`.
- n_mid_near - Integer. Mid-near pairs per point. Defaults to `5L`.
- n_further - Integer. Further (random) pairs per point. Defaults to `20L`.
- mn_candidate_start - Integer. Start index into kNN list for mid-near candidate window. Defaults to `4L`.
- mn_candidate_end - Integer. End index into kNN list for mid-near candidate window. Also determines the kNN search size. Defaults to `50L`.
- init - String. Embedding initialisation. One of `c("pca", "random")`. Defaults to `"pca"`.
- optimiser - String. The optimiser. One of `c("adam_parallel", "adam")`. Defaults to `"adam_parallel"`.
- lr - Numeric. Adam learning rate. Defaults to `1.0`.
- n_epochs - Integer or `NULL`. Total optimisation epochs. If `NULL`, resolved downstream to `450`. Defaults to `NULL`.
- beta1 - Numeric. Adam first moment decay. Defaults to `0.9`.
- beta2 - Numeric. Adam second moment decay. Defaults to `0.999`.
- eps - Numeric. Adam numerical stability constant. Defaults to `1e-07`.
- phase1_end - Integer or `NULL`. Epoch at which phase 1 ends. If `NULL`, resolved downstream to `100`. Defaults to `NULL`.
- phase2_end - Integer or `NULL`. Epoch at which phase 2 ends. If `NULL`, resolved downstream to `200`. Defaults to `NULL`.

Examples

```
params_pacmap()
params_pacmap(n_near = 15L, n_further = 30L)
```

params_phate *Wrapper function to generate PHATE parameters*

Description

Wrapper function to generate PHATE parameters

Usage

```
params_phate(
  decay = 40,
  bandwidth_scale = NULL,
  t_max = 100L,
  t_custom = NULL,
  gamma = 1,
  n_landmarks = 2048L,
  landmark_method = c("spectral", "random", "density"),
  n_svd = NULL,
  mds_method = c("sgd_dense", "classic"),
  graph_symmetry = c("additive", "multiplicative", "mnn", "none")
)
```

Arguments

decay	Numeric. Alpha decay parameter controlling the kernel bandwidth. Higher values create sharper transitions. Defaults to 40.0.
bandwidth_scale	Numeric or NULL. Scaling factor applied to the kernel bandwidth. If NULL, the library selects a sensible default. Defaults to NULL.
t_max	Integer. Maximum diffusion time considered during automatic selection via Von Neumann entropy knee point detection. Defaults to 100L.
t_custom	Integer or NULL. Fixed diffusion time. If set, overrides automatic time selection. Defaults to NULL.
gamma	Numeric. Informational distance parameter for the diffusion potential. Defaults to 1.0.
n_landmarks	Integer or NULL. Number of landmarks for compressed diffusion. If NULL, the full N x N diffusion operator is used. Defaults to 2048L.
landmark_method	String. Method used to select landmarks. One of c("spectral", "random", "density"). Defaults to "spectral".
n_svd	Integer or NULL. Number of SVD components used in landmark construction. If NULL, the library selects a sensible default. Defaults to NULL.
mds_method	String. MDS algorithm used for the final embedding. One of c("sgd_dense", "classic"). Defaults to "sgd_dense".
graph_symmetry	String. Method used to symmetrise the affinity graph. One of c("additive", "multiplicative", "mnn", "none"). Defaults to "additive".

Value

A named list with the following elements:

- decay - Numeric. Alpha decay parameter controlling the kernel bandwidth. Higher values create sharper transitions. Defaults to `40.0`.
- bandwidth_scale - Numeric or `NULL`. Scaling factor applied to the kernel bandwidth. If `NULL`, the library selects a sensible default. Defaults to `NULL`.
- t_max - Integer. Maximum diffusion time considered during automatic selection via Von Neumann entropy knee point detection. Defaults to `100L`.
- t_custom - Integer or `NULL`. Fixed diffusion time. If set, overrides automatic time selection. Defaults to `NULL`.
- gamma - Numeric. Informational distance parameter for the diffusion potential. Defaults to `1.0`.
- n_landmarks - Integer or `NULL`. Number of landmarks for compressed diffusion. If `NULL`, the full N x N diffusion operator is used. Defaults to `2048L`.
- landmark_method - String. Method used to select landmarks. One of `c("spectral", "random", "density")`. Defaults to `"spectral"`.
- n_svd - Integer or `NULL`. Number of SVD components used in landmark construction. If `NULL`, the library selects a sensible default. Defaults to `NULL`.
- mds_method - String. MDS algorithm used for the final embedding. One of `c("sgd_dense", "classic")`. Defaults to `"sgd_dense"`.
- graph_symmetry - String. Method used to symmetrise the affinity graph. One of `c("additive", "multiplicative", "mnn", "none")`. Defaults to `"additive"`.

Examples

```
params_phate()
params_phate(decay = 20, n_landmarks = 500L)
```

---

params_swiss_roll *Parameters for swiss roll data generation*

Description

For use with `manifold_synthetic_data()`.

Usage

```
params_swiss_roll(noise = 0.1)
```

Arguments

`noise`	Numeric. Amount of noise to add. Defaults to `0.1`.

Value

A named list with the following elements:

- noise - Numeric. Amount of noise to add. Defaults to `0.1`.

Examples

```
params_swiss_roll()
params_swiss_roll(noise = 0.2)
```

`params_swiss_roll_biased` *Parameters for biased swiss roll data generation*

Description

For use with `manifold_synthetic_data()`.

Usage

```
params_swiss_roll_biased(noise = 0.1, bias = 2.5)
```

Arguments

`noise`	Numeric. Amount of noise to add. Defaults to `0.1`.
`bias`	Numeric. The sampling bias across the manifold. Defaults to `2.5`.

Value

A named list with the following elements:

- noise - Numeric. Amount of noise to add. Defaults to `0.1`.
- bias - Numeric. The sampling bias across the manifold. Defaults to `2.5`.

Examples

```
params_swiss_roll_biased()
params_swiss_roll_biased(noise = 0.2, bias = 1)
```

`params_trajectory` *Parameters for trajectory data generation*

Description

For use with `manifold_synthetic_data()`.

Usage

```
params_trajectory(
  topology = c("bifurcation", "linear", "combination"),
  cell_trajectories = NULL,
  noise = 0.1
)
```

Arguments

`topology`	String. Ignored if `cell_trajectories` is not `NULL`. One of `c("bifurcation", "linear", "combination")`. Defaults to `"bifurcation"`.
`cell_trajectories`	Any. Optional named list with three equal-length vectors: `parent` (integer, `NA` for root, zero-indexed), `split_at` (numeric, fraction along parent where branch starts), and `length` (numeric, length of the branch). If `NULL`, `topology` is used instead. Defaults to `NULL`.
`noise`	Numeric. Amount of noise to add. Defaults to `0.1`.

Value

A named list with the following elements:

- topology - String. Ignored if `cell_trajectories` is not `NULL`. One of `c("bifurcation", "linear", "combination")`. Defaults to `"bifurcation"`.
- cell_trajectories - Any. Optional named list with three equal-length vectors: `parent` (integer, `NA` for root, zero-indexed), `split_at` (numeric, fraction along parent where branch starts), and `length` (numeric, length of the branch). If `NULL`, `topology` is used instead. Defaults to `NULL`.
- noise - Numeric. Amount of noise to add. Defaults to `0.1`.

Examples

```
params_trajectory()
```

`params_tsne` *Wrapper function to generate t-SNE parameters*

Description

Wrapper function to generate t-SNE parameters

Usage

```
params_tsne(
  lr = NULL,
  n_epochs = 1000L,
  early_exag_iter = 250L,
  early_exag_factor = 12,
  late_exag_factor = NULL,
  theta = 0.5,
  n_interp_points = 3L,
  max_depth = 7L,
  init = c("pca", "spectral", "random"),
  randomised = TRUE
)
```

Arguments

`lr`	Numeric or `NULL`. Learning rate. If `NULL`, the Rust backend sets it to `max((n_samples / 12), 200)`, following the N-dependent heuristic of Belkina et al. (2019). Defaults to `NULL`.
`n_epochs`	Integer. Number of optimisation epochs. Defaults to `1000L`.
`early_exag_iter`	Integer. Number of early exaggeration iterations. Defaults to `250L`.
`early_exag_factor`	Numeric. Early exaggeration factor. Defaults to `12.0`.
`late_exag_factor`	Numeric or `NULL`. Late exaggeration factor, if you wish to use one. Can be useful on large data sets (set it to `2.0` to `4.0`). Defaults to `NULL`.
`theta`	Numeric. Barnes-Hut approximation angle. Lower values increase accuracy at the cost of speed. Defaults to `0.5`.
`n_interp_points`	Integer. Number of interpolation points per grid cell for FFT acceleration (`"fft"` and `"fft_3k"`). Defaults to `3L`.
`max_depth`	Integer. Maximum Barnes-Hut tree depth for the quick-and-dirty optimiser (`"bh_qd"`). Smaller is faster and coarser; qdtsne recommends `7` to `10`. No effect on the other optimisers. Defaults to `7L`.
`init`	String. Embedding initialisation method. One of `c("pca", "spectral", "random")`. Defaults to `"pca"`.
`randomised`	Boolean. Use randomised SVD for PCA initialisation. Defaults to `TRUE`.

Value

A named list with the following elements:

- lr - Numeric or NULL. Learning rate. If NULL, the Rust backend sets it to max((n_samples / 12), 200), following the N-dependent heuristic of Belkina et al. (2019). Defaults to NULL.
- n_epochs - Integer. Number of optimisation epochs. Defaults to 1000L.
- early_exag_iter - Integer. Number of early exaggeration iterations. Defaults to 250L.
- early_exag_factor - Numeric. Early exaggeration factor. Defaults to 12.0.
- late_exag_factor - Numeric or NULL. Late exaggeration factor, if you wish to use one. Can be useful on large data sets (set it to 2.0 to 4.0). Defaults to NULL.
- theta - Numeric. Barnes-Hut approximation angle. Lower values increase accuracy at the cost of speed. Defaults to 0.5.
- n_interp_points - Integer. Number of interpolation points per grid cell for FFT acceleration ("fft" and "fft_3k"). Defaults to 3L.
- max_depth - Integer. Maximum Barnes-Hut tree depth for the quick-and-dirty optimiser ("bh_qd"). Smaller is faster and coarser; qdtsne recommends 7 to 10. No effect on the other optimisers. Defaults to 7L.
- init - String. Embedding initialisation method. One of c("pca", "spectral", "random"). Defaults to "pca".
- randomised - Boolean. Use randomised SVD for PCA initialisation. Defaults to TRUE.

References

Belkina, et al., Nat. Commun., 2019

Examples

```
params_tsne()
params_tsne(n_epochs = 500L, theta = 0.8)
```

---

params_umap *Wrapper function to generate UMAP parameters*

Description

Wrapper function to generate UMAP parameters

Usage

```
params_umap(
  local_connectivity = 1,
  bandwidth = 1e-05,
  mix_weight = 1,
  lr = 1,
  n_epochs = NULL,
```

```
  neg_sample_rate = 5L,
  gamma = 1,
  optimiser = c("adam_parallel", "sgd", "adam"),
  init = c("spectral", "pca", "random"),
  randomised = TRUE
)
```

Arguments

local_connectivity	Numeric. Number of nearest neighbours assumed to be at distance zero. Defaults to 1.0.
bandwidth	Numeric. Convergence tolerance for smooth kNN distance binary search. Defaults to 1e-05.
mix_weight	Numeric. Balance between fuzzy union and directed graph during symmetrisation. Defaults to 1.0.
lr	Numeric. Learning rate. Defaults to 1.0.
n_epochs	Integer or NULL. Number of optimisation epochs. If NULL, resolved downstream based on data size. Defaults to NULL.
neg_sample_rate	Integer. Number of negative samples per positive sample. Defaults to 5L.
gamma	Numeric. Repulsion strength. Defaults to 1.0.
optimiser	String. The optimiser. One of c("adam_parallel", "sgd", "adam"). Defaults to "adam_parallel".
init	String. Embedding initialisation method. One of c("spectral", "pca", "random"). Defaults to "spectral".
randomised	Boolean. Use randomised SVD for PCA initialisation. Defaults to TRUE.

Value

A named list with the following elements:

- local_connectivity - Numeric. Number of nearest neighbours assumed to be at distance zero. Defaults to 1.0.
- bandwidth - Numeric. Convergence tolerance for smooth kNN distance binary search. Defaults to 1e-05.
- mix_weight - Numeric. Balance between fuzzy union and directed graph during symmetrisation. Defaults to 1.0.
- lr - Numeric. Learning rate. Defaults to 1.0.
- n_epochs - Integer or NULL. Number of optimisation epochs. If NULL, resolved downstream based on data size. Defaults to NULL.
- neg_sample_rate - Integer. Number of negative samples per positive sample. Defaults to 5L.
- gamma - Numeric. Repulsion strength. Defaults to 1.0.
- optimiser - String. The optimiser. One of c("adam_parallel", "sgd", "adam"). Defaults to "adam_parallel".

- init - String. Embedding initialisation method. One of `c("spectral", "pca", "random")`. Defaults to `"spectral"`.
- randomised - Boolean. Use randomised SVD for PCA initialisation. Defaults to `TRUE`.

Examples

```
params_umap()
params_umap(n_epochs = 200L, neg_sample_rate = 10L)
```

`phate` *Rust-based PHATE*

Description

Performs PHATE dimensionality reduction on the input data. This function provides a user-friendly interface with input validation before calling the Rust implementation.

Usage

```
phate(
  data,
  knn = NULL,
  n_dim = 2L,
  k = 5L,
 knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  phate_params = params_phate(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

`data`	Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.
`knn`	Optional `NearestNeighbours` class. If provided, PHATE will skip the k-nearest neighbour graph generation and use this one instead. The `k` argument must match the k used to generate this graph. Defaults to `NULL`.
`n_dim`	Integer. Number of dimensions in the embedding space. Currently only `2L` is supported. Defaults to `2L`.
`k`	Integer. Number of nearest neighbours for graph construction. Defaults to `5L`.
`knn_method`	Character. (Approximate) Nearest neighbour method to use. One of `"kmknn"`, `"hnsw"`, `"annoy"`, `"nndescent"`, `"balltree"`, `"ivf"` or `"exhaustive"`. Defaults to `"kmknn"`.
`nn_params`	Named list. Nearest neighbour search parameters, see `params_nn()`.

phate_params	Named list. PHATE algorithm parameters, see params_phate().
seed	Integer. Random seed for reproducibility. Defaults to 42L.
use_high_precision	Optional boolean. Gives fine-grained control over fp32 vs fp64 usage.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Value

A numerical matrix with dimensions samples x n_dim containing the PHATE embedding.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- phate(d$data, .verbose = FALSE)
dim(emb)
```

---

rs_approx_nearest_neighbours

Wrapper around some nearest neighbour searches integrated into manifold-rs

---

Description

[Experimental] This is an interface into various (approximate) nearest neighbour searches implemented in the Rust crate ann-search-rs. They designed for incredible fast in-memory searches for kNN generation.

Usage

```
rs_approx_nearest_neighbours(data, k, ann_method, ann_params, seed, verbose)
```

Arguments

data	Numeric matrix. Shape of samples x n_dim for which to get the (approximate) nearest neighbours
k	Integer. Number of neighbours to return
ann_method	String. Which of the methods to use. One of c("hsnw", "balltree", "annoy", "nndescent")
ann_params	Named list. Contains the nearest neighbour parameters.
seed	Integer. Seed for reproducibility
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

A list with the following elements

- indices - flat representation of the indices.
- dist - flat representaitons of the distances.
- k - number of neighbours.
- n - number of samples.

`rs_ari` *Adjusted Rand index*

Description

[Experimental] Calculates the adjusted Rand index in Rust between two membership vectors.

Usage

```
rs_ari(cluster_membership_a, cluster_membership_b)
```

Arguments

`cluster_membership_a`
: Integers. Cluster memberships in group a.

`cluster_membership_b`
: Integers. Cluster memberships in group b.

Value

Returns the adjusted Rand index between the two groups.

`rs_data_biased_swiss_roll`
Generates biased Swiss roll data

Description

[Experimental] Generates synthetic data, i.e., the Swiss roll to test different manifold learning techniques

Usage

```
rs_data_biased_swiss_roll(n_samples, noise, bias, seed)
```

Arguments

n_samples	Integer. Number of data points to generate.
noise	Numeric. How much noise to add.
bias	Numeric. Sampling bias along t. 0.0 recovers uniform sampling; higher values concentrate samples at the inner end of the roll. A value of 2.5 mirrors the trajectory accumulation behaviour.
seed	Integer. For reproducibility purposes

Value

The biased Swiss roll synthetic data with the desired parameters.

---

rs_data_clusters *Generates clustered data*

Description

[Experimental] Generates synthetic data with clear cluster structure.

Usage

```
rs_data_clusters(n_samples, dim, n_clusters, seed)
```

Arguments

n_samples	Integer. Number of data points to generate.
dim	Integer. Dimensionality of the data
n_clusters	Integer. Number of clusters to produce in the data.
seed	Integer. For reproducibility purposes

Value

A list with the following elements:

- data - Numerical matrix with the data.
- clusters - Cluster assignments

`rs_data_hierarchical` *Generate hierarchical cluster data*

Description

[Experimental] Generates synthetic data with a two-level cluster hierarchy: `n_supergroups` top-level groups each containing `n_subclusts` tight subclusters. Supergroup centres are spread far apart; subcluster centres sit tightly around their supergroup centre.

Note that the actual number of samples returned may be slightly less than `n_samples` if it is not evenly divisible by `n_supergroups * n_subclusts`.

Usage

```
rs_data_hierarchical(
  n_samples,
  dim,
  n_supergroups,
  n_subclusts,
  supergroup_spread,
  subcluster_spread,
  point_std,
  seed
)
```

Arguments

`n_samples`	Integer. Total number of points, distributed evenly across all subclusters.
`dim`	Integer. Dimensionality of the ambient space.
`n_supergroups`	Integer. Number of top-level groups. Defaults to `3`.
`n_subclusts`	Integer. Number of subclusters per supergroup. Defaults to `3`.
`supergroup_spread`	Numeric. Spread of supergroup centres. Defaults to `15.0`.
`subcluster_spread`	Numeric. Spread of subcluster centres around their supergroup centre. Defaults to `2.0`.
`point_std`	Numeric. Within-subcluster Gaussian noise. Defaults to `0.4`.
`seed`	Integer. Seed for reproducibility.

Value

A named list with three elements: `data`, a numeric matrix of shape samples x `dim`; `supergroup`, an integer vector of supergroup labels (`0..n_supergroups`) one per sample; and `subgroup`, an integer vector of subcluster labels (`0..n_supergroups * n_subclusts`) one per sample.

rs_data_swiss_roll *Generates Swiss roll data*

Description

[Experimental] Generates synthetic data, i.e., the Swiss roll to test different manifold learning techniques

Usage

```
rs_data_swiss_roll(n_samples, noise, seed)
```

Arguments

n_samples	Integer. Number of data points to generate.
noise	Numeric. How much noise to add.
seed	Integer. For reproducibility purposes

Value

The Swiss roll synthetic data with the desired parameters.

rs_data_trajectory *Generates tree-like data with branches*

Description

[Experimental] Generates synthetic data that has a tree-like structure to simulate evolution/trajectory of data.

Usage

```
rs_data_trajectory(n_samples, dim, topology, cell_trajectories, noise, seed)
```

Arguments

n_samples	Integer. Number of data points to generate.
dim	Integer. Dimensionality of the data.
topology	String. One of c("bifurcation", "linear", "combination").
cell_trajectories	List or NULL. Named list with three equal-length vectors: parent (integer, NA for root, zero-indexed), split_at (numeric, fraction along parent where branch starts), and length (numeric, length of the branch). If NULL, will use the topology specified in topology.
noise	Numeric. How much noise to add.
seed	Integer. For reproducibility purposes.

Value

A list with the following elements:

- data - Numerical matrix with the generated data.
- branches - Branch assignments for each sample.

`rs_densmap` *densMAP implementation*

Description

[Experimental] Leverages the densMAP implementation in manifolds-rs - a very fast Rust-based implementation. densMAP is UMAP with an added density-preserving term, so tight clusters stay tight and diffuse ones stay diffuse. Setting `lambda` to `0` in the parameters recovers plain UMAP.

Usage

```
rs_densmap(
  embd,
  n_dim,
  min_dist,
  spread,
  k,
  densmap_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

`embd`	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
`n_dim`	Integer. Number of densMAP dimensions to return.
`min_dist`	Numeric. Minimum distance to use.
`spread`	Numeric. Spread parameter to use.
`k`	Integer. Number of nearest neighbours to consider
`densmap_params`	Named list. List that contains all of the key parameters for the densMAP generation, i.e. the UMAP ones plus `lambda`, `frac` and `var_shift`.
`seed`	Integer. Seed for reproducibility.
`use_high_precision`	Optional logical. Controls `fp32` vs `fp64` for. If `NULL` will use sensible default thresholding.
`verbose`	Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

The densMAP embeddings.

References

Narayan, Berger & Cho, Nature Biotechnology, 2021

`rs_densmap_from_knn` *densMAP implementation*

Description

[Experimental] Leverages the densMAP implementation in manifolds-rs - a very fast Rust-based implementation. densMAP is UMAP with an added density-preserving term, so tight clusters stay tight and diffuse ones stay diffuse. This version uses a pre-computed kNN graph, please see `new_nearest_neighbour()`.

Usage

```
rs_densmap_from_knn(
  embd,
  knn_data,
  n_dim,
  min_dist,
  spread,
  k,
  densmap_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

`embd`	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
`knn_data`	`NearestNeighbours` class from R.
`n_dim`	Integer. Number of densMAP dimensions to return.
`min_dist`	Numeric. Minimum distance to use.
`spread`	Numeric. Spread parameter to use.
`k`	Integer. Number of nearest neighbours to consider
`densmap_params`	Named list. List that contains all of the key parameters for the densMAP generation, i.e. the UMAP ones plus `lambda`, `frac` and `var_shift`.
`seed`	Integer. Seed for reproducibility.

use_high_precision
: Optional logical. Controls `fp32` vs `fp64` for. If `NULL` will use sensible default thresholding.

verbose
: Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

The densMAP embeddings.

References

Narayan, Berger & Cho, Nature Biotechnology, 2021

---

rs_densne *den-SNE implementation*

Description

[Experimental] Leverages the den-SNE implementation in manifolds-rs - a very fast Rust-based implementation. den-SNE is tSNE with an added density-preserving term, so tight clusters stay tight and diffuse ones stay diffuse. Setting `lambda` to `0` in the parameters recovers plain tSNE.

Usage

```
rs_densne(
  embd,
  n_dim,
  perplexity,
  approx_type,
  densne_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

embd
: Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.

n_dim
: Integer. Number of den-SNE dimensions to return. Needs to be two, others are not supported.

perplexity
: Numeric. The tSNE perplexity parameter.

approx_type
: String. One of `c("bh", "bh_qd", "fft", "fft_3k")`. Which of the approximations to use.

densne_params
: Named list. List that contains all of the key parameters for the den-SNE generation, i.e. the tSNE ones plus `lambda`, `frac` and `var_shift`.

seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The den-SNE embeddings.

References

Narayan, Berger & Cho, Nature Biotechnology, 2021

---

rs_densne_from_knn *den-SNE implementation*

Description

[Experimental] Leverages the den-SNE implementation in manifolds-rs - a very fast Rust-based implementation. den-SNE is tSNE with an added density-preserving term, so tight clusters stay tight and diffuse ones stay diffuse. This version uses a pre-computed kNN graph, please see new_nearest_neighbour().

Usage

```
rs_densne_from_knn(
  embd,
  knn_data,
  n_dim,
  perplexity,
  approx_type,
  densne_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

embd	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
knn_data	NearestNeighbours class from R.
n_dim	Integer. Number of den-SNE dimensions to return. Needs to be two, others are not supported.
perplexity	Numeric. The tSNE perplexity parameter.

approx_type	String. One of c("bh", "bh_qd", "fft", "fft_3k"). Which of the approximations to use.
densne_params	Named list. List that contains all of the key parameters for the den-SNE generation, i.e. the tSNE ones plus lambda, frac and var_shift.
seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The den-SNE embeddings.

References

Narayan, Berger & Cho, Nature Biotechnology, 2021

`rs_diffusion_maps` *Diffusion maps implementation*

Description

[Experimental] Leverages the Diffusion Maps implementation in manifolds-rs - a very fast Rust-based implementation.

Usage

```
rs_diffusion_maps(embd, n_dim, k, dm_params, seed, use_high_precision, verbose)
```

Arguments

embd	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
n_dim	Integer. Number of dimensions to return.
k	Integer. Number of nearest neighbours to consider.
dm_params	Named list. List that contains all of the key parameters for the diffusion maps generation.
seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The diffusion maps embeddings.

`rs_diffusion_maps_from_knn` *Diffusion maps implementation with pre-computed kNN*

Description

[Experimental] Leverages the Diffusion Maps implementation in manifolds-rs - a very fast Rust-based implementation. This version uses a pre-computed kNN graph, please see `new_nearest_neighbour()`.

Usage

```
rs_diffusion_maps_from_knn(
  embd,
  knn_data,
  n_dim,
  k,
  dm_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

`embd`	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
`knn_data`	`NearestNeighbours` class from R.
`n_dim`	Integer. Number of dimensions to return.
`k`	Integer. Number of nearest neighbours to consider.
`dm_params`	Named list. List that contains all of the key parameters for the diffusion maps generation.
`seed`	Integer. Seed for reproducibility.
`use_high_precision`	Optional logical. Controls `fp32` vs `fp64` for. If `NULL` will use sensible default thresholding.
`verbose`	Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

The diffusion maps embeddings.

`rs_dreams` *DREAMS implementation*

Description

[Experimental] Leverages the DREAMS implementation in manifolds-rs. DREAMS is tSNE with an added pull towards a global reference embedding (PCA unless one is supplied), so you keep the local structure of tSNE and most of the global layout of the reference. `lambda = 0` is plain tSNE, `lambda = 1` the reference.

Usage

```
rs_dreams(
  embd,
  reference,
  n_dim,
  perplexity,
  approx_type,
  dreams_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

`embd`	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
`reference`	Optional numerical matrix. Global reference embedding of dimensions samples x 2. If `NULL`, the first two principal components are used.
`n_dim`	Integer. Number of DREAMS dimensions to return. Needs to be two, others are not supported.
`perplexity`	Numeric. The tSNE perplexity parameter.
`approx_type`	String. One of `c("bh", "bh_qd", "fft", "fft_3k")`. Which of the approximations to use.
`dreams_params`	Named list. List that contains all of the key parameters for the DREAMS generation, i.e. the tSNE ones plus `lambda`.
`seed`	Integer. Seed for reproducibility.
`use_high_precision`	Optional logical. Controls `fp32` vs `fp64` for. If `NULL` will use sensible default thresholding.
`verbose`	Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

The DREAMS embeddings.

References

Kury, Kobak & Damrich, Transactions on Machine Learning Research, 2026

`rs_dreams_from_knn` *DREAMS implementation*

Description

[Experimental] Leverages the DREAMS implementation in manifolds-rs. DREAMS is tSNE with an added pull towards a global reference embedding (PCA unless one is supplied). This version uses a pre-computed kNN graph, please see `new_nearest_neighbour()`.

Usage

```
rs_dreams_from_knn(
  embd,
  knn_data,
  reference,
  n_dim,
  perplexity,
  approx_type,
  dreams_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

`embd`	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
`knn_data`	`NearestNeighbours` class from R.
`reference`	Optional numerical matrix. Global reference embedding of dimensions samples x 2. If `NULL`, the first two principal components are used.
`n_dim`	Integer. Number of DREAMS dimensions to return. Needs to be two, others are not supported.
`perplexity`	Numeric. The tSNE perplexity parameter.
`approx_type`	String. One of `c("bh", "bh_qd", "fft", "fft_3k")`. Which of the approximations to use.
`dreams_params`	Named list. List that contains all of the key parameters for the DREAMS generation, i.e. the tSNE ones plus `lambda`.
`seed`	Integer. Seed for reproducibility.
`use_high_precision`	Optional logical. Controls `fp32` vs `fp64` for. If `NULL` will use sensible default thresholding.
`verbose`	Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

The DREAMS embeddings.

References

Kury, Kobak & Damrich, Transactions on Machine Learning Research, 2026

`rs_evoc` *EVoC clustering*

Description

[Experimental] Wrapper function into the Rust interface for EVoC clustering.

Usage

```
rs_evoc(
  embd,
  n_neighbours,
  evoc_params,
  return_knn,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

`embd`	Numerical matrix. The data to cluster. Should be of dimensions samples x features.
`n_neighbours`	Integer. Number of nearest neighbours for graph construction.
`evoc_params`	Named list. List that contains all of the key parameters for EVoC clustering.
`return_knn`	Boolean. Shall the kNN graph be returned.
`seed`	Integer. Seed for reproducibility.
`use_high_precision`	Optional logical. Controls `fp32` vs `fp64` for. If `NULL` will use sensible default thresholding.
`verbose`	Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

A named list with:

- evoc_res - List with the EVoC results
- knn - Optional list (can be NULL) with the kNN graph

`rs_evoc_from_knn` *EVoC clustering from pre-computed kNN*

Description

[Experimental] Wrapper function into the Rust interface for EVoC clustering. This version uses a pre-computed kNN graph, please see `new_nearest_neighbour()`.

Usage

```
rs_evoc_from_knn(
  embd,
  knn_data,
  n_neighbours,
  evoc_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

`embd`	Numerical matrix. The data to cluster. Should be of dimensions samples x features.
`knn_data`	List. A NearestNeighbours object with `k`, `indices`, and `dist` elements.
`n_neighbours`	Integer. Number of nearest neighbours for graph construction.
`evoc_params`	Named list. List that contains all of the key parameters for EVoC clustering.
`seed`	Integer. Seed for reproducibility.
`use_high_precision`	Optional logical. Controls `fp32` vs `fp64` for. If `NULL` will use sensible default thresholding.
`verbose`	Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

A named list with cluster layers, membership strengths, persistence scores, and the kNN graph.

rs_forceatlas2 *ForceAtlas2 implementation*

Description

[Experimental] Leverages the ForceAtlas2 implementation in manifolds-rs. Builds the UMAP fuzzy union graph from the data and lays it out with ForceAtlas2.

Usage

```
rs_forceatlas2(embd, k, fa2_params, seed, use_high_precision, verbose)
```

Arguments

`embd`	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
`k`	Integer. Number of nearest neighbours to consider.
`fa2_params`	Named list. List that contains all of the key parameters for the kNN search, graph generation and ForceAtlas2 optimisation.
`seed`	Integer. Seed for reproducibility.
`use_high_precision`	Optional logical. Controls `fp32` vs `fp64`. If `NULL` will use sensible default thresholding.
`verbose`	Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

The ForceAtlas2 embedding, samples x 2.

rs_forceatlas2_from_graph *ForceAtlas2 on a pre-computed graph*

Description

[Experimental] Leverages the ForceAtlas2 implementation in manifolds-rs and lays out a caller-supplied undirected graph. Each edge is passed once; the Rust side adds the reverse direction.

Usage

```
rs_forceatlas2_from_graph(
  from,
  to,
  weight,
  n,
  init,
  fa2_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

from	Integer vector. 1-based source vertex of each edge.
to	Integer vector. 1-based target vertex of each edge.
weight	Numeric vector. Weight of each edge.
n	Integer. Number of vertices.
init	Optional numerical matrix of dimensions n x 2. The initial layout. If NULL, a random layout is used.
fa2_params	Named list. List that contains the ForceAtlas2 optimisation parameters.
seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The ForceAtlas2 embedding, n x 2.

---

rs_forceatlas2_from_knn

ForceAtlas2 implementation

Description

[Experimental] Leverages the ForceAtlas2 implementation in manifolds-rs. This version uses a pre-computed kNN graph, please see new_nearest_neighbour().

Usage

```
rs_forceatlas2_from_knn(
  embd,
  knn_data,
  k,
  fa2_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

embd	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
knn_data	NearestNeighbours class from R.
k	Integer. Number of nearest neighbours to consider.
fa2_params	Named list. List that contains all of the key parameters for the graph generation and ForceAtlas2 optimisation.
seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The ForceAtlas2 embedding, samples x 2.

---

rs_inertia *Calculates the inertia for k-means clustering*

Description

[Experimental] Calculates inertia for k-means clustering given the data, centroids and membership

Usage

```
rs_inertia(data, centroids, cluster_membership)
```

Arguments

data	Numeric matrix. The data in shape of sample x features.
centroids	Numeric matrix. The centroid data in shape k x features.
cluster_membership	Integers. Cluster memberships as integers.

Value

The inertia score

`rs_k_means` *K-means clustering (full)*

Description

[Experimental] Rust interface for k-means clustering using Lloyd's algorithm with SIMD or GEMM acceleration depending on dimensionality. It exposes the high performance k-means clustering code from the Rust crate `ann-search-rs` for rapid clustering.

Usage

```
rs_k_means(data, k, kmeans_params, seed, verbose)
```

Arguments

`data`	Numerical matrix. The data to cluster, of dimensions samples x features.
`k`	Integer. Number of clusters.
`kmeans_params`	Named list. Parameters produced by `params_kmeans()`.
`seed`	Integer. Seed for reproducibility.
`verbose`	Boolean. Controls verbosity.

Value

A named list with:

- centroids - Numeric matrix of shape k x features.
- assignments - Integer vector of length samples (1-indexed).

`rs_k_means_mini_batch` *Mini-batch k-means clustering*

Description

[Experimental] Rust interface for mini batch k-means clustering from `bixverse-rs`. This version can be very useful in cases where you overcluster the data.

Usage

```
rs_k_means_mini_batch(data, k, kmeans_params, seed, verbose)
```

Arguments

data	Numerical matrix. The data to cluster, of dimensions samples x features.
k	Integer. Number of clusters.
kmeans_params	Named list. Parameters produced by params_kmeans().
seed	Integer. Seed for reproducibility.
verbose	Boolean. Controls verbosity.

Value

A named list with:

- centroids - Numeric matrix of shape k x features.
- assignments - Integer vector of length samples (1-indexed).

---

rs_pacmap *PaCMAP implementation*

Description

[Experimental] Leverages the PacMAP implementation in manifolds-rs - a very fast Rust-based implementation.

Usage

```
rs_pacmap(embd, n_dim, pacmap_params, seed, use_high_precision, verbose)
```

Arguments

embd	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
n_dim	Integer. Number of dimensions to return.
pacmap_params	Named list. List that contains all of the key parameters for the PaCMAP generation.
seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The PaCMAP embeddings.

rs_pacmap_from_knn *PaCMAP implementation with pre-computed kNN*

Description

[Experimental] Leverages the PacMAP implementation in manifolds-rs - a very fast Rust-based implementation. This version uses a pre-computed kNN graph, please see `new_nearest_neighbour()`.

Usage

```
rs_pacmap_from_knn(
  embd,
  knn_data,
  n_dim,
  pacmap_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

`embd`	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
`knn_data`	`NearestNeighbours` class from R.
`n_dim`	Integer. Number of dimensions to return.
`pacmap_params`	Named list. List that contains all of the key parameters for the PaCMAP generation.
`seed`	Integer. Seed for reproducibility.
`use_high_precision`	Optional logical. Controls `fp32` vs `fp64` for. If `NULL` will use sensible default thresholding.
`verbose`	Integer. If `0L` -> silent or `1L` for normal verbosity; `2L` for detailed verbosity.

Value

The PaCMAP embeddings.

rs_phate *Run PHATE dimensionality reduction*

Description

[Experimental] Leverages the PHATE implementation in manifolds-rs - a very fast Rust-based implementation.

Usage

```
rs_phate(embd, n_dim, k, phate_params, seed, use_high_precision, verbose)
```

Arguments

embd	Numerical matrix. The data to embed of shape samples x features.
n_dim	Integer. Number of PHATE dimensions to return. Currently only 2L is supported.
k	Integer. Number of nearest neighbours for graph construction.
phate_params	Named list. Contains all key parameters for PHATE, see params_phate() and params_nn().
seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The PHATE embedding as a matrix of shape samples x n_dim.

rs_phate_from_knn *Run PHATE dimensionality reduction from a precomputed kNN graph*

Description

[Experimental] Leverages the PHATE implementation in manifolds-rs - a very fast Rust-based implementation. This version uses a pre-computed kNN graph, please see new_nearest_neighbour().

Usage

```
rs_phate_from_knn(
  embd,
  knn_data,
  n_dim,
  k,
  phate_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

embd	Numerical matrix. The data to embed of shape samples x features.
knn_data	NearestNeighbours class from R.
n_dim	Integer. Number of PHATE dimensions to return. Currently only 2L is supported.
k	Integer. Number of nearest neighbours used during graph construction. Must match the k used to generate knn_data.
phate_params	Named list. Contains all key parameters for PHATE, see params_phate() and params_nn().
seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The PHATE embedding as a matrix of shape samples x n_dim.

---

rs_silhouette_score *Calculates the cluster silhouette scores*

Description

[Experimental] Calculates the Silhouette in Rust given the original data and a membership vector. a(i) and b(i) are mean **squared** Euclidean distances, which keeps the whole thing closed-form and avoids materialising any pairwise distance. The trade-off is that the scores are not the numbers a plain-Euclidean silhouette gives: squaring inflates b more than a, so the values run higher. The sign is unaffected, so which points sit in the wrong cluster is the same call, but do not compare the magnitudes against implementations that use plain Euclidean.

Usage

```
rs_silhouette_score(data, cluster_membership)
```

Arguments

data	Numeric matrix. The data in shape of sample x features.
cluster_membership	Integers. Cluster memberships as integers.

Value

A list with the following items

- mean_silhouette - Mean silhouette scores per cluster.
- silhouette_scores - Silhouette scores per given data point.

---

rs_tsne *tSNE implementation*

Description

[Experimental] Leverages the tSNE implementation in manifolds-rs - a very fast Rust-based implementation. You have three optimiser options: `"bh"` which tends to be faster on smaller datasets, and `"fft"` or its three-kernel variant `"fft_3k"` for large data sets.

Usage

```
rs_tsne(
  embd,
  n_dim,
  perplexity,
  approx_type,
  tsne_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

embd	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
n_dim	Integer. Number of tSNE dimensions to return. Needs to be two, others are not supported.
perplexity	Numeric. The tSNE perplexity parameter.

approx_type — String. One of c("bh", "bh_qd", "fft", "fft_3k"). Which of the approximations to use.

tsne_params — Named list. List that contains all of the key parameters for the tSNE generation.

seed — Integer. Seed for reproducibility.

use_high_precision — Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.

verbose — Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The tSNE embeddings.

---

rs_tsne_from_knn *tSNE implementation*

Description

[Experimental] Leverages the tSNE implementation in manifolds-rs - a very fast Rust-based implementation. You have three optimiser options: "bh" which tends to be faster on smaller datasets, and "fft" or its three-kernel variant "fft_3k" for large data sets. This version uses a pre-computed kNN graph, please see new_nearest_neighbour().

Usage

```
rs_tsne_from_knn(
  embd,
  knn_data,
  n_dim,
  perplexity,
  approx_type,
  tsne_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

embd — Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.

knn_data — NearestNeighbours class from R.

n_dim — Integer. Number of tSNE dimensions to return. Needs to be two, others are not supported.

perplexity — Numeric. The tSNE perplexity parameter.

approx_type	String. One of c("bh", "bh_qd", "fft", "fft_3k"). Which of the approximations to use.
tsne_params	Named list. List that contains all of the key parameters for the tSNE generation.
seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The tSNE embeddings.

`rs_umap` *UMAP implementation*

Description

[Experimental] Leverages the UMAP implementation in manifolds-rs - a very fast Rust-based implementation.

Usage

```
rs_umap(
  embd,
  n_dim,
  min_dist,
  spread,
  k,
  umap_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

embd	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
n_dim	Integer. Number of UMAP dimensions to return.
min_dist	Numeric. Minimum distance to use.
spread	Numeric. Spread parameter to use.
k	Integer. Number of nearest neighbours to consider
umap_params	Named list. List that contains all of the key parameters for the UMAP generation.

seed	Integer. Seed for reproducibility.
use_high_precision	Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.
verbose	Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The UMAP embeddings.

`rs_umap_from_knn` *UMAP implementation*

Description

[Experimental] Leverages the UMAP implementation in manifolds-rs - a very fast Rust-based implementation. This version uses a pre-computed kNN graph, please see `new_nearest_neighbour()`.

Usage

```
rs_umap_from_knn(
  embd,
  knn_data,
  n_dim,
  min_dist,
  spread,
  k,
  umap_params,
  seed,
  use_high_precision,
  verbose
)
```

Arguments

embd	Numerical matrix. The data to use to generate the embeddings. Should be of dimensions samples x features.
knn_data	NearestNeighbours class from R.
n_dim	Integer. Number of UMAP dimensions to return.
min_dist	Numeric. Minimum distance to use.
spread	Numeric. Spread parameter to use.
k	Integer. Number of nearest neighbours to consider
umap_params	Named list. List that contains all of the key parameters for the UMAP generation.
seed	Integer. Seed for reproducibility.

use_high_precision
: Optional logical. Controls fp32 vs fp64 for. If NULL will use sensible default thresholding.

verbose
: Integer. If 0L -> silent or 1L for normal verbosity; 2L for detailed verbosity.

Value

The UMAP embeddings.

---

tsne *Rust-based t-SNE*

Description

Performs t-SNE dimensionality reduction on the input data. This function provides a user-friendly interface with input validation before calling the Rust implementation.

Usage

```
tsne(
  data,
  knn = NULL,
  n_dim = 2L,
  perplexity = 20,
  approx_type = c("bh", "bh_qd", "fft", "fft_3k"),
 knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  tsne_params = params_tsne(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

data
: Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.

knn
: Optional NearestNeighbours class. If provided, t-SNE will skip the k-nearest neighbour graph generation and use this one. Defaults to NULL.

n_dim
: Integer. Number of dimensions in the embedding space. Currently only 2L is supported. Defaults to 2L.

perplexity
: Numeric. Perplexity parameter, related to the number of nearest neighbours used in manifold learning. Typical values are between 5 and 50. Defaults to 20.0.

approx_type	Character. Approximation method for computing repulsive forces. One of "bh" for Barnes-Hut, "bh_qd" for the quick-and-dirty Barnes-Hut of qdtsne (tree depth capped at max_depth in params_tsne(), repulsion computed once per leaf), "fft" for FFT-accelerated interpolation or "fft_3k" for the three-kernel variant of the latter (one forward and three inverse FFTs per epoch instead of four each). Defaults to "bh".
knn_method	Character. (Approximate) Nearest neighbour method to use. One of "kmknn", "hnsw", "annoy", "nndescent", "balltree", "ivf" or "exhaustive". Defaults to "kmknn".
nn_params	Named list. Nearest neighbour search parameters, see params_nn().
tsne_params	Named list. t-SNE algorithm parameters, see params_tsne().
seed	Integer. Random seed for reproducibility.
use_high_precision	Optional boolean. Gives fine-grained control over fp32 vs fp64 usage.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Details

The number of neighbours will be '3 * perplexity", as this is a usual default in tSNE.

Value

A numerical matrix with dimensions samples x n_dim containing the t-SNE embedding.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- tsne(d$data, .verbose = FALSE)
dim(emb)
```

---

umap *Rust-based UMAP*

Description

Performs UMAP dimensionality reduction on the input data. This function provides a user-friendly interface with input validation before calling the Rust implementation.

Usage

```
umap(
  data,
  knn = NULL,
  n_dim = 2L,
  k = 15L,
```

```
  min_dist = 0.5,
  spread = 1,
 knn_method = c("kmknn", "balltree", "hnsw", "annoy", "nndescent", "exhaustive", "ivf"),
  nn_params = params_nn(),
  umap_params = params_umap(),
  seed = 42L,
  use_high_precision = NULL,
  .verbose = TRUE
)
```

Arguments

data	Numerical matrix or data frame. The data to embed of shape samples x features. Will be coerced to a matrix.
knn	Optional NearestNeighbours class. If provided, UMAP will skip the k-nearest neighbour graph generation and use this one. Defaults to NULL.
n_dim	Integer. Number of dimensions in the embedding space. Defaults to 2L.
k	Integer. Number of nearest neighbours to consider for manifold approximation. Larger values result in more global structure being preserved. Defaults to 15L.
min_dist	Numeric. Minimum distance between points in the embedding. Controls how tightly points are packed. Smaller values result in more clustered embeddings. Must be >= 0. Defaults to 0.5. If you use SGD, consider reducing this!
spread	Numeric. Effective scale of embedded points. Determines the scale at which embedded points will be spread out. Defaults to 1.0.
knn_method	Character. (Approximate) Nearest neighbour method to use. One of "kmknn", "hnsw", "annoy", "nndescent", "balltree", "ivf" or "exhaustive". Defaults to "kmknn".
nn_params	Named list. Nearest neighbour search parameters, see params_nn().
umap_params	Named list. UMAP algorithm parameters, see params_umap().
seed	Integer. Random seed for reproducibility. Defaults to 42L.
use_high_precision	Optional boolean. Gives fine-grained control over fp32 vs fp64 usage.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Value

A numerical matrix with dimensions samples x n_dim containing the UMAP embedding.

Examples

```
d <- manifold_synthetic_data(type = "clusters", n_samples = 500L)
emb <- umap(d$data, .verbose = FALSE)
dim(emb)

# reuse one kNN graph across methods
knn <- generate_knn_graph(d$data, k = 15L, .verbose = FALSE)
emb <- umap(d$data, knn = knn, .verbose = FALSE)
```

Index

best_membership, 4, *14*

calc_ari, 4
calc_inertia, 5
calc_silhouette_score, 6

densmap, 6
densne, 8
diffusion_maps, 10
dreams, 11

evoc, 13

forceatlas2, 14
forceatlas2(), *34*
forceatlas2_from_graph, 15
forceatlas2_from_graph(), *14*, *34*

generate_knn_graph, 16
generate_nearest_neigbours_class
(new_nearest_neighbour), 25
get_centroids, 18
get_dist_flat, 18
get_dist_mat, 19
get_idx_flat, 20
get_idx_mat, 20
get_layer, *14*, 21
get_nearest_neighbours, 22

kmeans_cluster, 22

manifold_synthetic_data, 23
manifold_synthetic_data(), *27*, *36*, *43–45*
membership, 24

new_nearest_neighbour, 25
new_nearest_neighbour(), *56*, *58*, *60*, *62*, *64*, *66*, *70, 71*, *74*, *76*

pacmap, 26
params_clusters, 27
params_clusters(), *24*
params_densmap, 28
params_densmap(), *7*
params_densne, 29
params_densne(), *9*
params_diffusion_maps, 30
params_diffusion_maps(), *10*
params_dreams, 31
params_dreams(), *12*
params_evoc, 32
params_evoc(), *13*
params_fa2, 34
params_fa2(), *14*, *16*
params_hierarchical, 36
params_hierarchical(), *24*
params_kmeans, 37
params_kmeans(), *23*
params_nn, 38
params_nn(), *7*, *9, 10*, *12–14*, *17*, *26*, *49*, *71, 72*, *78, 79*
params_pacmap, 40
params_pacmap(), *27*
params_phate, 42
params_phate(), *50*, *71, 72*
params_swiss_roll, 43
params_swiss_roll(), *24*
params_swiss_roll_biased, 44
params_swiss_roll_biased(), *24*
params_trajectory, 45
params_trajectory(), *24*
params_tsne, 46
params_tsne(), *9*, *12*, *78*
params_umap, 47
params_umap(), *7*, *79*
phate, 49
print.Evoc, *14*

rs_approx_nearest_neighbours, 50
rs_ari, 51
rs_data_biased_swiss_roll, 51

rs_data_clusters, 52
rs_data_hierarchical, 53
rs_data_swiss_roll, 54
rs_data_trajectory, 54
rs_densmap, 55
rs_densmap_from_knn, 56
rs_densne, 57
rs_densne_from_knn, 58
rs_diffusion_maps, 59
rs_diffusion_maps_from_knn, 60
rs_dreams, 61
rs_dreams_from_knn, 62
rs_evoc, 63
rs_evoc_from_knn, 64
rs_forceatlas2, 65
rs_forceatlas2_from_graph, 65
rs_forceatlas2_from_knn, 66
rs_inertia, 67
rs_k_means, 68
rs_k_means_mini_batch, 68
rs_pacmap, 69
rs_pacmap_from_knn, 70
rs_phate, 71
rs_phate_from_knn, 71
rs_silhouette_score, 72
rs_tsne, 73
rs_tsne_from_knn, 74
rs_umap, 75
rs_umap_from_knn, 76

tsne, 77
tsne(), *9*

umap, 78
umap(), *7*