Package 'uwotlite'

January 17, 2026

Title The Uniform Manifold Approximation and Projection (UMAP) Method for Dimensionality Reduction (MIT-Compatible Fork)

Version 0.2.4.9000

Description MIT-compatible fork of the 'uwot' package. An implementation of the Uniform Manifold Approximation and Projection dimensionality reduction by McInnes et al. (2018) <doi:10.48550/arXiv.1802.03426>. It also provides means to transform new data and to carry out supervised dimensionality reduction. An implementation of the related LargeVis method of Tang et al. (2016) <doi:10.48550/arXiv.1602.00370> is also provided. This fork replaces the AGPL-licensed dqrng with MIT-licensed sitmo for the PCG random number generator.

License MIT + file LICENSE

URL https://github.com/rmsharp/uwotlite

BugReports https://github.com/rmsharp/uwotlite/issues

Depends Matrix

Imports FNN, irlba, methods, Rcpp, RcppAnnoy (>= 0.0.17), RSpectra

Suggests bigstatsr, covr, dplyr, ggplot2, knitr, modeldata, RcppHNSW, rmarkdown, rnndescent, scales, testthat

LinkingTo sitmo, Rcpp, RcppAnnoy, RcppProgress

VignetteBuilder knitr

Config/Needs/website rmarkdown

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Author James Melville [aut, cph],
R. Mark Sharp [cre, ctb],
Aaron Lun [ctb],
Mohamed Nadhir Djekidel [ctb],
Yuhan Hao [ctb],
Dirk Eddelbuettel [ctb],
Wouter van der Bijl [ctb],
Hugo Gruson [ctb]

Maintainer R. Mark Sharp <rmsharp@me.com>

Contents

`load_uwot` *Save or Load a Model*

Description

Functions to write a UMAP model to a file, and to restore.

Usage

```
load_uwot(file, verbose = FALSE)
```

Arguments

`file`	name of the file where the model is to be saved or read from.
`verbose`	if `TRUE`, log information to the console.

Value

The model saved at `file`, for use with `umap_transform`. Additionally, it contains an extra item: `mod_dir`, which contains the path to the temporary working directory used during loading of the model. This directory cannot be removed until this model has been unloaded by using `unload_uwot`.

See Also

`save_uwot`, `unload_uwot`

Examples

```
library(RSpectra)

iris_train <- iris[c(1:10, 51:60), ]
iris_test <- iris[100:110, ]

# create model
model <- umap(iris_train, ret_model = TRUE, n_epochs = 20)
```

```
# save without unloading: this leaves behind a temporary working directory
model_file <- tempfile("iris_umap")
model <- save_uwot(model, file = model_file)

# The model can continue to be used
test_embedding <- umap_transform(iris_test, model)

# To manually unload the model from memory when finished and to clean up
# the working directory (this doesn't touch your model file)
unload_uwot(model)

# At this point, model cannot be used with umap_transform, this would fail:
# test_embedding2 <- umap_transform(iris_test, model)

# restore the model: this also creates a temporary working directory
model2 <- load_uwot(file = model_file)
test_embedding2 <- umap_transform(iris_test, model2)

# Unload and clean up the loaded model temp directory
unload_uwot(model2)

# clean up the model file
unlink(model_file)

# save with unloading: this deletes the temporary working directory but
# doesn't allow the model to be re-used
model3 <- umap(iris_train, ret_model = TRUE, n_epochs = 20)
model_file3 <- tempfile("iris_umap")
model3 <- save_uwot(model3, file = model_file3, unload = TRUE)
```

`lvish` *Dimensionality Reduction with a LargeVis-like method*

Description

Carry out dimensionality reduction of a dataset using a method similar to LargeVis (Tang et al., 2016).

Usage

```
lvish(
  X,
  perplexity = 50,
  n_neighbors = perplexity * 3,
  n_components = 2,
  metric = "euclidean",
  n_epochs = -1,
  learning_rate = 1,
  scale = "maxabs",
  init = "lvrandom",
  init_sdev = NULL,
  repulsion_strength = 7,
```

```
  negative_sample_rate = 5,
  nn_method = NULL,
  n_trees = 50,
  search_k = 2 * n_neighbors * n_trees,
  n_threads = NULL,
  n_build_threads = NULL,
  n_sgd_threads = 0,
  grain_size = 1,
  kernel = "gauss",
  pca = NULL,
  pca_center = TRUE,
  pcg_rand = TRUE,
  fast_sgd = FALSE,
  ret_nn = FALSE,
  ret_extra = c(),
  tmpdir = tempdir(),
  verbose = getOption("verbose", TRUE),
  batch = FALSE,
  opt_args = NULL,
  epoch_callback = NULL,
  pca_method = NULL,
  binary_edge_weights = FALSE,
  nn_args = list(),
  rng_type = NULL
)
```

Arguments

X	Input data. Can be a data.frame, matrix, dist object or sparseMatrix. Matrix and data frames should contain one observation per row. Data frames will have any non-numeric columns removed, although factor columns will be used if explicitly included via metric (see the help for metric for details). A sparse matrix is interpreted as a distance matrix, and is assumed to be symmetric, so you can also pass in an explicitly upper or lower triangular sparse matrix to save storage. There must be at least n_neighbors non-zero distances for each row. Both implicit and explicit zero entries are ignored. Set zero distances you want to keep to an arbitrarily small non-zero value (e.g. 1e-10). X can also be NULL if pre-computed nearest neighbor data is passed to nn_method, and init is not "spca" or "pca".
perplexity	Controls the size of the local neighborhood used for manifold approximation. This is the analogous to n_neighbors in umap. Change this, rather than n_neighbors.
n_neighbors	The number of neighbors to use when calculating the perplexity. Usually set to three times the value of the perplexity. Must be at least as large as perplexity.
n_components	The dimension of the space to embed into. This defaults to 2 to provide easy visualization, but can reasonably be set to any integer value in the range 2 to 100.
metric	Type of distance metric to use to find nearest neighbors. For nn_method = "annoy" this can be one of: • "euclidean" (the default) • "cosine"

- "manhattan"
- "hamming"
- "correlation" (a distance based on the Pearson correlation)
- "categorical" (see below)

For nn_method = "hnsw" this can be one of:

- "euclidean"
- "cosine"
- "correlation"

If rnndescent is installed and nn_method = "nndescent" is specified then many more metrics are avaiable, including:

- "braycurtis"
- "canberra"
- "chebyshev"
- "dice"
- "hamming"
- "hellinger"
- "jaccard"
- "jensenshannon"
- "kulsinski"
- "rogerstanimoto"
- "russellrao"
- "sokalmichener"
- "sokalsneath"
- "spearmanr"
- "symmetrickl"
- "tsss"
- "yule"

For more details see the package documentation of rnndescent. For nn_method = "fnn", the distance metric is always "euclidean".

If X is a data frame or matrix, then multiple metrics can be specified, by passing a list to this argument, where the name of each item in the list is one of the metric names above. The value of each list item should be a vector giving the names or integer ids of the columns to be included in a calculation, e.g. metric = list(euclidean = 1:4, manhattan = 5:10).

Each metric calculation results in a separate fuzzy simplicial set, which are intersected together to produce the final set. Metric names can be repeated. Because non-numeric columns are removed from the data frame, it is safer to use column names than integer ids.

Factor columns can also be used by specifying the metric name "categorical". Factor columns are treated different from numeric columns and although multiple factor columns can be specified in a vector, each factor column specified is processed individually. If you specify a non-factor column, it will be coerced to a factor.

For a given data block, you may override the pca and pca_center arguments for that block, by providing a list with one unnamed item containing the column names or ids, and then any of the pca or pca_center overrides as named items, e.g. metric = list(euclidean = 1:4, manhattan = list(5:10, pca_center

= FALSE)). This exists to allow mixed binary and real-valued data to be included and to have PCA applied to both, but with centering applied only to the real-valued data (it is typical not to apply centering to binary data before PCA is applied).

n_epochs Number of epochs to use during the optimization of the embedded coordinates. The default is calculate the number of epochs dynamically based on dataset size, to give the same number of edge samples as the LargeVis defaults. This is usually substantially larger than the UMAP defaults. If n_epochs = 0, then coordinates determined by "init" will be returned.

learning_rate Initial learning rate used in optimization of the coordinates.

scale Scaling to apply to X if it is a data frame or matrix:

- "none" or FALSE or NULL No scaling.
- "Z" or "scale" or TRUE Scale each column to zero mean and variance 1.
- "maxabs" Center each column to mean 0, then divide each element by the maximum absolute value over the entire matrix.
- "range" Range scale the entire matrix, so the smallest element is 0 and the largest is 1.
- "colrange" Scale each column in the range (0,1).

For lvish, the default is "maxabs", for consistency with LargeVis.

init Type of initialization for the coordinates. Options are:

- "spectral" Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, with Gaussian noise added.
- "normlaplacian". Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, without noise.
- "random". Coordinates assigned using a uniform random distribution between -10 and 10.
- "lvrandom". Coordinates assigned using a Gaussian distribution with standard deviation 1e-4, as used in LargeVis (Tang et al., 2016) and t-SNE.
- "laplacian". Spectral embedding using the Laplacian Eigenmap (Belkin and Niyogi, 2002).
- "pca". The first two principal components from PCA of X if X is a data frame, and from a 2-dimensional classical MDS if X is of class "dist".
- "spca". Like "pca", but each dimension is then scaled so the standard deviation is 1e-4, to give a distribution similar to that used in t-SNE and LargeVis. This is an alias for init = "pca", init_sdev = 1e-4.
- "agspectral" An "approximate global" modification of "spectral" which all edges in the graph to a value of 1, and then sets a random number of edges (negative_sample_rate edges per vertex) to 0.1, to approximate the effect of non-local affinities.
- A matrix of initial coordinates.

For spectral initializations, ("spectral", "normlaplacian", "laplacian", "agspectral"), if more than one connected component is identified, no spectral initialization is attempted. Instead a PCA-based initialization is attempted. If verbose = TRUE the number of connected components are logged to the console. The existence of multiple connected components implies that a global view of the data cannot be attained with this initialization. Increasing the value of n_neighbors may help.

`init_sdev` If non-`NULL`, scales each dimension of the initialized coordinates (including any user-supplied matrix) to this standard deviation. By default no scaling is carried out, except when `init = "spca"`, in which case the value is `0.0001`. Scaling the input may help if the unscaled versions result in initial coordinates with large inter-point distances or outliers. This usually results in small gradients during optimization and very little progress being made to the layout. Shrinking the initial embedding by rescaling can help under these circumstances. Scaling the result of `init = "pca"` is usually recommended and `init = "spca"` as an alias for `init = "pca", init_sdev = 1e-4` but for the spectral initializations the scaled versions usually aren't necessary unless you are using a large value of `n_neighbors` (e.g. `n_neighbors = 150` or higher). For compatibility with recent versions of the Python UMAP package, if you are using `init = "spectral"`, then you should also set `init_sdev = "range"`, which will range scale each of the columns containing the initial data between 0-10. This is not set by default to maintain backwards compatibility with previous versions of uwot.

`repulsion_strength` Weighting applied to negative samples in low dimensional embedding optimization. Values higher than one will result in greater weight being given to negative samples.

`negative_sample_rate` The number of negative edge/1-simplex samples to use per positive edge/1-simplex sample in optimizing the low dimensional embedding.

`nn_method` Method for finding nearest neighbors. Options are:

- `"fnn"`. Use exact nearest neighbors via the FNN package.
- `"annoy"` Use approximate nearest neighbors via the RcppAnnoy package.
- `"hnsw"` Use approximate nearest neighbors with the Hierarchical Navigable Small World (HNSW) method (Malkov and Yashunin, 2018) via the RcppHNSW package. `RcppHNSW` is not a dependency of this package: this option is only available if you have installed `RcppHNSW` yourself. Also, HNSW only supports the following arguments for `metric`: `"euclidean"`, `"cosine"` and `"correlation"`.
- `"nndescent"` Use approximate nearest neighbors with the Nearest Neighbor Descent method (Dong et al., 2011) via the rnndescent package. `rnndescent` is not a dependency of this package: this option is only available if you have installed `rnndescent` yourself.

By default, if `X` has less than 4,096 vertices, the exact nearest neighbors are found. Otherwise, approximate nearest neighbors are used. You may also pass precalculated nearest neighbor data to this argument. It must be a list consisting of two elements:

- `"idx"` (or `"index"`). A `n_vertices` x `n_neighbors` matrix containing the integer indexes of the nearest neighbors in `X`. *Each vertex is considered to be its own nearest neighbor, i.e.* `idx[, 1] == 1:n_vertices`.
- `"dist"` (or `"distance"`). A `n_vertices` x `n_neighbors` matrix containing the distances of the nearest neighbors.

Multiple nearest neighbor data (e.g. from two different precomputed metrics) can be passed by passing a list containing the nearest neighbor data lists as items. The `n_neighbors` parameter is ignored when using precomputed nearest neighbor data.

n_trees — Number of trees to build when constructing the nearest neighbor index. The more trees specified, the larger the index, but the better the results. With search_k, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy". Sensible values are between 10 to 100.

search_k — Number of nodes to search during the neighbor retrieval. The larger k, the more the accurate results, but the longer the search takes. With n_trees, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy".

n_threads — Number of threads to use (except during stochastic gradient descent). Default is half the number of concurrent threads supported by the system. For nearest neighbor search, this controls the search phase. For nn_method = "annoy", if n_threads > 1, then the Annoy index will be temporarily written to disk in the location determined by tempfile.

n_build_threads — Number of threads to use when building nearest neighbor indexes. Default is NULL, which uses n_threads. To improve determinism, use n_build_threads = 1. Only applies for nn_method = "hnsw" or nn_method = "nndescent". The Annoy-based index always use a single thread.

n_sgd_threads — Number of threads to use during stochastic gradient descent. If set to > 1, then be aware that if batch = FALSE, results will *not* be reproducible, even if set.seed is called with a fixed seed before running. Set to "auto" to use the same value as n_threads.

grain_size — The minimum amount of work to do on each thread. If this value is set high enough, then less than n_threads or n_sgd_threads will be used for processing, which might give a performance improvement if the overhead of thread management and context switching was outweighing the improvement due to concurrent processing. This should be left at default (1) and work will be spread evenly over all the threads specified.

kernel — Type of kernel function to create input probabilities. Can be one of "gauss" (the default) or "knn". "gauss" uses the usual Gaussian weighted similarities. "knn" assigns equal probabilities to every edge in the nearest neighbor graph, and zero otherwise, using perplexity nearest neighbors. The n_neighbors parameter is ignored in this case.

pca — If set to a positive integer value, reduce data to this number of columns using PCA. Doesn't applied if the distance metric is "hamming", or the dimensions of the data is larger than the number specified (i.e. number of rows and columns must be larger than the value of this parameter). If you have > 100 columns in a data frame or matrix, reducing the number of columns in this way may substantially increase the performance of the nearest neighbor search at the cost of a potential decrease in accuracy. In many t-SNE applications, a value of 50 is recommended, although there's no guarantee that this is appropriate for all settings.

pca_center — If TRUE, center the columns of X before carrying out PCA. For binary data, it's recommended to set this to FALSE.

pcg_rand — If TRUE, use the PCG random number generator (O'Neill, 2014) during optimization. Otherwise, use the faster (but probably less statistically good) Tausworthe "taus88" generator. The default is TRUE. This parameter has been superseded by rng_type – if both are set, rng_type takes precedence.

fast_sgd — If TRUE, then the following combination of parameters is set: pcg_rand = TRUE and n_sgd_threads = "auto". The default is FALSE. Setting this to TRUE will

speed up the stochastic optimization phase, but give a potentially less accurate embedding, and which will not be exactly reproducible even with a fixed seed. For visualization, `fast_sgd = TRUE` will give perfectly good results. For more generic dimensionality reduction, it's safer to leave `fast_sgd = FALSE`. If `fast_sgd = TRUE`, then user-supplied values of `pcg_rand` and `n_sgd_threads`, are ignored.

`ret_nn` If `TRUE`, then in addition to the embedding, also return nearest neighbor data that can be used as input to `nn_method` to avoid the overhead of repeatedly calculating the nearest neighbors when manipulating unrelated parameters (e.g. `min_dist`, `n_epochs`, `init`). See the "Value" section for the names of the list items. If `FALSE`, just return the coordinates. Note that the nearest neighbors could be sensitive to data scaling, so be wary of reusing nearest neighbor data if modifying the `scale` parameter.

`ret_extra` A vector indicating what extra data to return. May contain any combination of the following strings:

- `"nn"` same as setting `ret_nn = TRUE`.
- `"P"` the high dimensional probability matrix. The graph is returned as a sparse symmetric N x N matrix of class dgCMatrix-class, where a non-zero entry (i, j) gives the input probability (or similarity or affinity) of the edge connecting vertex i and vertex j. Note that the graph is further sparsified by removing edges with sufficiently low membership strength that they would not be sampled by the probabilistic edge sampling employed for optimization and therefore the number of non-zero elements in the matrix is dependent on `n_epochs`. If you are only interested in the fuzzy input graph (e.g. for clustering), setting `n_epochs = 0` will avoid any further sparsifying. Be aware that setting `binary_edge_weights = TRUE` will affect this graph (all non-zero edge weights will be 1).
- `sigma` a vector of the bandwidths used to calibrate the input Gaussians to reproduce the target `"perplexity"`.

`tmpdir` Temporary directory to store nearest neighbor indexes during nearest neighbor search. Default is `tempdir`. The index is only written to disk if `n_threads > 1` and `nn_method = "annoy"`; otherwise, this parameter is ignored.

`verbose` If `TRUE`, log details to the console.

`batch` If `TRUE`, then embedding coordinates are updated at the end of each epoch rather than during the epoch. In batch mode, results are reproducible with a fixed random seed even with `n_sgd_threads > 1`, at the cost of a slightly higher memory use. You may also have to modify `learning_rate` and increase `n_epochs`, so whether this provides a speed increase over the single-threaded optimization is likely to be dataset and hardware-dependent.

`opt_args` A list of optimizer parameters, used when `batch = TRUE`. The default optimization method used is Adam (Kingma and Ba, 2014).

- `method` The optimization method to use. Either `"adam"` or `"sgd"` (stochastic gradient descent). Default: `"adam"`.
- `beta1` (Adam only). The weighting parameter for the exponential moving average of the first moment estimator. Effectively the momentum parameter. Should be a floating point value between 0 and 1. Higher values can smooth oscillatory updates in poorly-conditioned situations and may allow for a larger `learning_rate` to be specified, but too high can cause divergence. Default: `0.5`.

- beta2 (Adam only). The weighting parameter for the exponential moving average of the uncentered second moment estimator. Should be a floating point value between 0 and 1. Controls the degree of adaptivity in the step-size. Higher values put more weight on previous time steps. Default: 0.9.
- eps (Adam only). Intended to be a small value to prevent division by zero, but in practice can also affect convergence due to its interaction with beta2. Higher values reduce the effect of the step-size adaptivity and bring the behavior closer to stochastic gradient descent with momentum. Typical values are between 1e-8 and 1e-3. Default: 1e-7.
- alpha The initial learning rate. Default: the value of the learning_rate parameter.

epoch_callback A function which will be invoked at the end of every epoch. Its signature should be: (epoch, n_epochs, coords), where:

- epoch The current epoch number (between 1 and n_epochs).
- n_epochs Number of epochs to use during the optimization of the embedded coordinates.
- coords The embedded coordinates as of the end of the current epoch, as a matrix with dimensions (N, n_components).

pca_method Method to carry out any PCA dimensionality reduction when the pca parameter is specified. Allowed values are:

- "irlba". Uses prcomp_irlba from the irlba package.
- "rsvd". Uses 5 iterations of svdr from the irlba package. This is likely to give much faster but potentially less accurate results than using "irlba". For the purposes of nearest neighbor calculation and coordinates initialization, any loss of accuracy doesn't seem to matter much.
- "bigstatsr". Uses big_randomSVD from the bigstatsr package. The SVD methods used in bigstatsr may be faster on systems without access to efficient linear algebra libraries (e.g. Windows). **Note**: bigstatsr is *not* a dependency of uwot: if you choose to use this package for PCA, you *must* install it yourself.
- "svd". Uses svd for the SVD. This is likely to be slow for all but the smallest datasets.
- "auto" (the default). Uses "irlba", unless more than 50 case "svd" is used.

binary_edge_weights

If TRUE then edge weights in the input graph are treated as binary (0/1) rather than real valued. This affects the sampling frequency of neighbors and is the strategy used by the PaCMAP method (Wang and co-workers, 2020). Practical (Böhm and co-workers, 2020) and theoretical (Damrich and Hamprecht, 2021) work suggests this has little effect on UMAP's performance.

nn_args A list containing additional arguments to pass to the nearest neighbor method. For nn_method = "annoy", you can specify "n_trees" and "search_k", and these will override the n_trees and search_k parameters. For nn_method = "hnsw", you may specify the following arguments:

- M The maximum number of neighbors to keep for each vertex. Reasonable values are 2 to 100. Higher values give better recall at the cost of more memory. Default value is 16.
- ef_construction A positive integer specifying the size of the dynamic list used during index construction. A higher value will provide better results at the cost of a longer time to build the index. Default is 200.

- `ef` A positive integer specifying the size of the dynamic list used during search. This cannot be smaller than `n_neighbors` and cannot be higher than the number of items in the index. Default is `10`.

For `nn_method = "nndescent"`, you may specify the following arguments:

- `n_trees` The number of trees to use in a random projection forest to initialize the search. A larger number will give more accurate results at the cost of a longer computation time. The default of `NULL` means that the number is chosen based on the number of observations in `X`.
- `max_candidates` The number of potential neighbors to explore per iteration. By default, this is set to `n_neighbors` or `60`, whichever is smaller. A larger number will give more accurate results at the cost of a longer computation time.
- `n_iters` The number of iterations to run the search. A larger number will give more accurate results at the cost of a longer computation time. By default, this will be chosen based on the number of observations in `X`. You may also need to modify the convergence criterion `delta`.
- `delta` The minimum relative change in the neighbor graph allowed before early stopping. Should be a value between 0 and 1. The smaller the value, the smaller the amount of progress between iterations is allowed. Default value of `0.001` means that at least 0.1 neighbor graph must be updated at each iteration.
- `init` How to initialize the nearest neighbor descent. By default this is set to `"tree"` and uses a random project forest. If you set this to `"rand"`, then a random selection is used. Usually this is less accurate than using RP trees, but for high-dimensional cases, there may be little difference in the quality of the initialization and random initialization will be a lot faster. If you set this to `"rand"`, then the `n_trees` parameter is ignored.

`rng_type` The type of random number generator to use during optimization. One of:

- `"pcg"`. Use the PCG random number generator (O'Neill, 2014).
- `"tausworthe"`. Use the Tausworthe "taus88" generator.
- `"deterministic"`. Use a deterministic number generator. This isn't actually random, but may provide enough variation in the negative sampling to give a good embedding and can provide a noticeable speed-up.

For backwards compatibility, by default this is unset and the choice of `pcg_rand` is used (making "pcg" the effective default).

Details

`lvish` differs from the official LargeVis implementation in the following:

- Only the nearest-neighbor index search phase is multi-threaded.
- Matrix input data is not normalized.
- The `n_trees` parameter cannot be dynamically chosen based on data set size.
- Nearest neighbor results are not refined via the neighbor-of-my-neighbor method. The `search_k` parameter is twice as large than default to compensate.
- Gradient values are clipped to `4.0` rather than `5.0`.
- Negative edges are generated by uniform sampling of vertexes rather than their degree ^ 0.75.
- The default number of samples is much reduced. The default number of epochs, `n_epochs`, is set to `5000`, much larger than for `umap`, but may need to be increased further depending on your dataset. Using `init = "spectral"` can help.

Value

A matrix of optimized coordinates, or:

- if `ret_nn = TRUE` (or `ret_extra` contains `"nn"`), returns the nearest neighbor data as a list called `nn`. This contains one list for each `metric` calculated, itself containing a matrix `idx` with the integer ids of the neighbors; and a matrix `dist` with the distances. The `nn` list (or a sub-list) can be used as input to the `nn_method` parameter.
- if `ret_extra` contains `"P"`, returns the high dimensional probability matrix as a sparse matrix called `P`, of type dgCMatrix-class.
- if `ret_extra` contains `"sigma"`, returns a vector of the high dimensional gaussian bandwidths for each point, and `"dint"` a vector of estimates of the intrinsic dimensionality at each point, based on the method given by Lee and co-workers (2015).

The returned list contains the combined data from any combination of specifying `ret_nn` and `ret_extra`.

References

Belkin, M., & Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems* (pp. 585-591). http://papers.nips.cc/paper/1961-laplacian-eigenmaps-and-spectral-techniques-for-embedding-and-clustering.pdf

Böhm, J. N., Berens, P., & Kobak, D. (2020). A unifying perspective on neighbor embeddings along the attraction-repulsion spectrum. *arXiv preprint arXiv:2007.08902*. https://arxiv.org/abs/2007.08902

Damrich, S., & Hamprecht, F. A. (2021). On UMAP's true loss function. *Advances in Neural Information Processing Systems*, *34*. https://proceedings.neurips.cc/paper/2021/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html

Dong, W., Moses, C., & Li, K. (2011, March). Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World Wide Web* (pp. 577-586). ACM. doi:10.1145/1963405.1963487.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv*:1412.6980. https://arxiv.org/abs/1412.6980

Lee, J. A., Peluffo-Ordóñez, D. H., & Verleysen, M. (2015). Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. *Neurocomputing*, *169*, 246-261.

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, *42*(4), 824-836.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction *arXiv preprint arXiv*:1802.03426. https://arxiv.org/abs/1802.03426

O'Neill, M. E. (2014). *PCG: A family of simple fast space-efficient statistically good algorithms for random number generation* (Report No. HMC-CS-2014-0905). Harvey Mudd College.

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016, April). Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 287-297). International World Wide Web Conferences Steering Committee. https://arxiv.org/abs/1602.00370

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9* (2579-2605). https://www.jmlr.org/papers/v9/vandermaaten08a.html

Wang, Y., Huang, H., Rudin, C., & Shaposhnik, Y. (2021). Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *Journal of Machine Learning Research*, *22*(201), 1-73. https://www.jmlr.org/papers/v22/20-1061.html

Examples

```
# Default number of epochs is much larger than for UMAP, assumes random
# initialization. Use perplexity rather than n_neighbors to control the size
# of the local neighborhood 20 epochs may be too small for a random
# initialization
iris_lvish <- lvish(iris,
  perplexity = 50, learning_rate = 0.5,
  init = "random", n_epochs = 20
)
```

---

optimize_graph_layout *Optimize Graph Layout*

Description

Carry out dimensionality reduction on an input graph, where the distances in the low dimensional space attempt to reproduce the neighbor relations in the input data. By default, the cost function used to optimize the output coordinates use the Uniform Manifold Approximation and Projection (UMAP) method (McInnes et al., 2018), but the approach from LargeVis (Tang et al., 2016) can also be used. This function can be used to produce a low dimensional representation of the graph produced by similarity_graph.

Usage

```
optimize_graph_layout(
  graph,
  X = NULL,
  n_components = 2,
  n_epochs = NULL,
  learning_rate = 1,
  init = "spectral",
  init_sdev = NULL,
  spread = 1,
  min_dist = 0.01,
  repulsion_strength = 1,
  negative_sample_rate = 5,
  a = NULL,
  b = NULL,
  method = "umap",
  approx_pow = FALSE,
  pcg_rand = TRUE,
  fast_sgd = FALSE,
  n_sgd_threads = 0,
```

```
  grain_size = 1,
  verbose = getOption("verbose", TRUE),
  batch = FALSE,
  opt_args = NULL,
  epoch_callback = NULL,
  pca_method = NULL,
  binary_edge_weights = FALSE,
  rng_type = NULL
)
```

Arguments

graph	A sparse, symmetric N x N weighted adjacency matrix representing a graph. Non-zero entries indicate an edge between two nodes with a given edge weight. There can be a varying number of non-zero entries in each row/column.
X	Optional input data. Used only for PCA-based initialization.
n_components	The dimension of the space to embed into. This defaults to 2 to provide easy visualization, but can reasonably be set to any integer value in the range 2 to 100.
n_epochs	Number of epochs to use during the optimization of the embedded coordinates. By default, this value is set to 500 for datasets containing 10,000 vertices or less, and 200 otherwise. If n_epochs = 0, then coordinates determined by "init" will be returned. For UMAP, the default is "none".
learning_rate	Initial learning rate used in optimization of the coordinates.
init	Type of initialization for the coordinates. Options are: • "spectral" Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, with Gaussian noise added. • "normlaplacian". Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, without noise. • "random". Coordinates assigned using a uniform random distribution between -10 and 10. • "lvrandom". Coordinates assigned using a Gaussian distribution with standard deviation 1e-4, as used in LargeVis (Tang et al., 2016) and t-SNE. • "laplacian". Spectral embedding using the Laplacian Eigenmap. • "pca". The first two principal components from PCA of X if X is a data frame, and from a 2-dimensional classical MDS if X is of class "dist". • "spca". Like "pca", but each dimension is then scaled so the standard deviation is 1e-4, to give a distribution similar to that used in t-SNE. This is an alias for init = "pca", init_sdev = 1e-4. • "agspectral" An "approximate global" modification of "spectral" which all edges in the graph to a value of 1, and then sets a random number of edges (negative_sample_rate edges per vertex) to 0.1, to approximate the effect of non-local affinities. • A matrix of initial coordinates. For spectral initializations, ("spectral", "normlaplacian", "laplacian", "agspectral"), if more than one connected component is identified, no spectral initialization is attempted. Instead a PCA-based initialization is attempted. If verbose = TRUE the number of connected components are logged to the console. The existence of multiple connected components implies that a global view of the data cannot

be attained with this initialization. Increasing the value of n_neighbors may help.

init_sdev
: If non-NULL, scales each dimension of the initialized coordinates (including any user-supplied matrix) to this standard deviation. By default no scaling is carried out, except when init = "spca", in which case the value is 0.0001. Scaling the input may help if the unscaled versions result in initial coordinates with large inter-point distances or outliers. This usually results in small gradients during optimization and very little progress being made to the layout. Shrinking the initial embedding by rescaling can help under these circumstances. Scaling the result of init = "pca" is usually recommended and init = "spca" as an alias for init = "pca", init_sdev = 1e-4 but for the spectral initializations the scaled versions usually aren't necessary unless you are using a large value of n_neighbors (e.g. n_neighbors = 150 or higher). For compatibility with recent versions of the Python UMAP package, if you are using init = "spectral", then you should also set init_sdev = "range", which will range scale each of the columns containing the initial data between 0-10. This is not set by default to maintain backwards compatibility with previous versions of uwot.

spread
: The effective scale of embedded points. In combination with min_dist, this determines how clustered/clumped the embedded points are.

min_dist
: The effective minimum distance between embedded points. Smaller values will result in a more clustered/clumped embedding where nearby points on the manifold are drawn closer together, while larger values will result on a more even dispersal of points. The value should be set relative to the spread value, which determines the scale at which embedded points will be spread out.

repulsion_strength
: Weighting applied to negative samples in low dimensional embedding optimization. Values higher than one will result in greater weight being given to negative samples.

negative_sample_rate
: The number of negative edge/1-simplex samples to use per positive edge/1-simplex sample in optimizing the low dimensional embedding.

a
: More specific parameters controlling the embedding. If NULL these values are set automatically as determined by min_dist and spread.

b
: More specific parameters controlling the embedding. If NULL these values are set automatically as determined by min_dist and spread.

method
: Cost function to optimize. One of:

- "umap". The UMAP method of McInnes and co-workers (2018).
- "tumap". UMAP with the a and b parameters fixed to 1.
- "largevis". The LargeVis method Tang and co-workers (2016).

approx_pow
: If TRUE, use an approximation to the power function in the UMAP gradient, from https://martin.ankerl.com/2012/01/25/optimized-approximative-pow-in-c-and-cpp/.

pcg_rand
: If TRUE, use the PCG random number generator (O'Neill, 2014) during optimization. Otherwise, use the faster (but probably less statistically good) Tausworthe "taus88" generator. The default is TRUE. This parameter has been superseded by rng_type – if both are set, rng_type takes precedence.

fast_sgd
: If TRUE, then the following combination of parameters is set: pcg_rand = TRUE, n_sgd_threads = "auto" and approx_pow = TRUE. The default is FALSE. Setting this to TRUE will speed up the stochastic optimization phase, but give a

potentially less accurate embedding, and which will not be exactly reproducible even with a fixed seed. For visualization, `fast_sgd = TRUE` will give perfectly good results. For more generic dimensionality reduction, it's safer to leave `fast_sgd = FALSE`. If `fast_sgd = TRUE`, then user-supplied values of `pcg_rand`, `n_sgd_threads`, and `approx_pow` are ignored.

`n_sgd_threads` Number of threads to use during stochastic gradient descent. If set to > 1, then be aware that if `batch = FALSE`, results will *not* be reproducible, even if `set.seed` is called with a fixed seed before running. If set to `"auto"` then half the number of concurrent threads supported by the system will be used.

`grain_size` The minimum amount of work to do on each thread. If this value is set high enough, then less than `n_threads` or `n_sgd_threads` will be used for processing, which might give a performance improvement if the overhead of thread management and context switching was outweighing the improvement due to concurrent processing. This should be left at default (1) and work will be spread evenly over all the threads specified.

`verbose` If `TRUE`, log details to the console.

`batch` If `TRUE`, then embedding coordinates are updated at the end of each epoch rather than during the epoch. In batch mode, results are reproducible with a fixed random seed even with `n_sgd_threads > 1`, at the cost of a slightly higher memory use. You may also have to modify `learning_rate` and increase `n_epochs`, so whether this provides a speed increase over the single-threaded optimization is likely to be dataset and hardware-dependent.

`opt_args` A list of optimizer parameters, used when `batch = TRUE`. The default optimization method used is Adam (Kingma and Ba, 2014).

- `method` The optimization method to use. Either `"adam"` or `"sgd"` (stochastic gradient descent). Default: `"adam"`.
- `beta1` (Adam only). The weighting parameter for the exponential moving average of the first moment estimator. Effectively the momentum parameter. Should be a floating point value between 0 and 1. Higher values can smooth oscillatory updates in poorly-conditioned situations and may allow for a larger `learning_rate` to be specified, but too high can cause divergence. Default: `0.5`.
- `beta2` (Adam only). The weighting parameter for the exponential moving average of the uncentered second moment estimator. Should be a floating point value between 0 and 1. Controls the degree of adaptivity in the step-size. Higher values put more weight on previous time steps. Default: `0.9`.
- `eps` (Adam only). Intended to be a small value to prevent division by zero, but in practice can also affect convergence due to its interaction with `beta2`. Higher values reduce the effect of the step-size adaptivity and bring the behavior closer to stochastic gradient descent with momentum. Typical values are between 1e-8 and 1e-3. Default: `1e-7`.
- `alpha` The initial learning rate. Default: the value of the `learning_rate` parameter.

`epoch_callback` A function which will be invoked at the end of every epoch. Its signature should be: `(epoch, n_epochs, coords)`, where:

- `epoch` The current epoch number (between 1 and `n_epochs`).
- `n_epochs` Number of epochs to use during the optimization of the embedded coordinates.
- `coords` The embedded coordinates as of the end of the current epoch, as a matrix with dimensions (N, `n_components`).

pca_method
: Method to carry out any PCA dimensionality reduction when the pca parameter is specified. Allowed values are:
  - "irlba". Uses prcomp_irlba from the irlba package.
  - "rsvd". Uses 5 iterations of svdr from the irlba package. This is likely to give much faster but potentially less accurate results than using "irlba". For the purposes of nearest neighbor calculation and coordinates initialization, any loss of accuracy doesn't seem to matter much.
  - "bigstatsr". Uses big_randomSVD from the bigstatsr package. The SVD methods used in bigstatsr may be faster on systems without access to efficient linear algebra libraries (e.g. Windows). **Note**: bigstatsr is *not* a dependency of uwot: if you choose to use this package for PCA, you *must* install it yourself.
  - "svd". Uses svd for the SVD. This is likely to be slow for all but the smallest datasets.
  - "auto" (the default). Uses "irlba", unless more than 50 case "svd" is used.

binary_edge_weights
: If TRUE then edge weights in the input graph are treated as binary (0/1) rather than real valued.

rng_type
: The type of random number generator to use during optimization. One of:
  - "pcg". Use the PCG random number generator (O'Neill, 2014).
  - "tausworthe". Use the Tausworthe "taus88" generator.
  - "deterministic". Use a deterministic number generator. This isn't actually random, but may provide enough variation in the negative sampling to give a good embedding and can provide a noticeable speed-up.

  For backwards compatibility, by default this is unset and the choice of pcg_rand is used (making "pcg" the effective default).

Value

A matrix of optimized coordinates.

References

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv*:1412.6980. https://arxiv.org/abs/1412.6980

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction *arXiv preprint arXiv*:1802.03426. https://arxiv.org/abs/1802.03426

O'Neill, M. E. (2014). *PCG: A family of simple fast space-efficient statistically good algorithms for random number generation* (Report No. HMC-CS-2014-0905). Harvey Mudd College.

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016, April). Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 287-297). International World Wide Web Conferences Steering Committee. https://arxiv.org/abs/1602.00370

Examples

```
iris30 <- iris[c(1:10, 51:60, 101:110), ]
```

```
# return a 30 x 30 sparse matrix with similarity data based on 10 nearest
# neighbors per item
iris30_sim_graph <- similarity_graph(iris30, n_neighbors = 10)
# produce 2D coordinates replicating the neighbor relations in the similarity
# graph
set.seed(42)
iris30_opt <- optimize_graph_layout(iris30_sim_graph, X = iris30)

# the above two steps are the same as:
# set.seed(42); iris_umap <- umap(iris30, n_neighbors = 10)
```

---

save_uwot *Save or Load a Model*

Description

Functions to write a UMAP model to a file, and to restore.

Usage

```
save_uwot(model, file, unload = FALSE, verbose = FALSE)
```

Arguments

model	a UMAP model create by umap.
file	name of the file where the model is to be saved or read from.
unload	if TRUE, unload all nearest neighbor indexes for the model. The model will no longer be valid for use in umap_transform and the temporary working directory used during model saving will be deleted. You will need to reload the model with load_uwot to use the model. If FALSE, then the model can be re-used without reloading, but you must manually unload the NN index when you are finished using it if you want to delete the temporary working directory. To unload manually, use unload_uwot. The absolute path of the working directory is found in the mod_dir item of the return value.
verbose	if TRUE, log information to the console.

Value

model with one extra item: mod_dir, which contains the path to the working directory. If unload = FALSE then this directory still exists after this function returns, and can be cleaned up with unload_uwot. If you don't care about cleaning up this directory, or unload = TRUE, then you can ignore the return value.

See Also

load_uwot, unload_uwot

Examples

```
iris_train <- iris[c(1:10, 51:60), ]
iris_test <- iris[100:110, ]

# create model
model <- umap(iris_train, ret_model = TRUE, n_epochs = 20)

# save without unloading: this leaves behind a temporary working directory
model_file <- tempfile("iris_umap")
model <- save_uwot(model, file = model_file)

# The model can continue to be used
test_embedding <- umap_transform(iris_test, model)

# To manually unload the model from memory when finished and to clean up
# the working directory (this doesn't touch your model file)
unload_uwot(model)

# At this point, model cannot be used with umap_transform, this would fail:
# test_embedding2 <- umap_transform(iris_test, model)

# restore the model: this also creates a temporary working directory
model2 <- load_uwot(file = model_file)
test_embedding2 <- umap_transform(iris_test, model2)

# Unload and clean up the loaded model temp directory
unload_uwot(model2)

# clean up the model file
unlink(model_file)

# save with unloading: this deletes the temporary working directory but
# doesn't allow the model to be re-used
model3 <- umap(iris_train, ret_model = TRUE, n_epochs = 20)
model_file3 <- tempfile("iris_umap")
model3 <- save_uwot(model3, file = model_file3, unload = TRUE)
```

---

similarity_graph *Similarity Graph*

Description

Create a graph (as a sparse symmetric weighted adjacency matrix) representing the similarities between items in a data set. No dimensionality reduction is carried out. By default, the similarities are calculated using the merged fuzzy simplicial set approach in the Uniform Manifold Approximation and Projection (UMAP) method (McInnes et al., 2018), but the approach from LargeVis (Tang et al., 2016) can also be used.

Usage

```
similarity_graph(
  X = NULL,
```

```
  n_neighbors = NULL,
  metric = "euclidean",
  scale = NULL,
  set_op_mix_ratio = 1,
  local_connectivity = 1,
  nn_method = NULL,
  n_trees = 50,
  search_k = 2 * n_neighbors * n_trees,
  perplexity = 50,
  method = "umap",
  y = NULL,
  target_n_neighbors = n_neighbors,
  target_metric = "euclidean",
  target_weight = 0.5,
  pca = NULL,
  pca_center = TRUE,
  ret_extra = c(),
  n_threads = NULL,
  n_build_threads = NULL,
  grain_size = 1,
  kernel = "gauss",
  tmpdir = tempdir(),
  verbose = getOption("verbose", TRUE),
  pca_method = NULL,
  binary_edge_weights = FALSE,
  nn_args = list()
)
```

Arguments

X	Input data. Can be a data.frame, matrix, dist object or sparseMatrix. Matrix and data frames should contain one observation per row. Data frames will have any non-numeric columns removed, although factor columns will be used if explicitly included via metric (see the help for metric for details). A sparse matrix is interpreted as a distance matrix, and is assumed to be symmetric, so you can also pass in an explicitly upper or lower triangular sparse matrix to save storage. There must be at least n_neighbors non-zero distances for each row. Both implicit and explicit zero entries are ignored. Set zero distances you want to keep to an arbitrarily small non-zero value (e.g. 1e-10). X can also be NULL if pre-computed nearest neighbor data is passed to nn_method.
n_neighbors	The size of local neighborhood (in terms of number of neighboring sample points) used for manifold approximation. Larger values result in more global views of the manifold, while smaller values result in more local data being preserved. In general values should be in the range 2 to 100.
metric	Type of distance metric to use to find nearest neighbors. For nn_method = "annoy" this can be one of: • "euclidean" (the default) • "cosine" • "manhattan" • "hamming" • "correlation" (a distance based on the Pearson correlation)

- "categorical" (see below)

For nn_method = "hnsw" this can be one of:

- "euclidean"
- "cosine"
- "correlation"

If rnndescent is installed and nn_method = "nndescent" is specified then many more metrics are avaiable, including:

- "braycurtis"
- "canberra"
- "chebyshev"
- "dice"
- "hamming"
- "hellinger"
- "jaccard"
- "jensenshannon"
- "kulsinski"
- "rogerstanimoto"
- "russellrao"
- "sokalmichener"
- "sokalsneath"
- "spearmanr"
- "symmetrickl"
- "tsss"
- "yule"

For more details see the package documentation of rnndescent. For nn_method = "fnn", the distance metric is always "euclidean".

If X is a data frame or matrix, then multiple metrics can be specified, by passing a list to this argument, where the name of each item in the list is one of the metric names above. The value of each list item should be a vector giving the names or integer ids of the columns to be included in a calculation, e.g. metric = list(euclidean = 1:4, manhattan = 5:10).

Each metric calculation results in a separate fuzzy simplicial set, which are intersected together to produce the final set. Metric names can be repeated. Because non-numeric columns are removed from the data frame, it is safer to use column names than integer ids.

Factor columns can also be used by specifying the metric name "categorical". Factor columns are treated different from numeric columns and although multiple factor columns can be specified in a vector, each factor column specified is processed individually. If you specify a non-factor column, it will be coerced to a factor.

For a given data block, you may override the pca and pca_center arguments for that block, by providing a list with one unnamed item containing the column names or ids, and then any of the pca or pca_center overrides as named items, e.g. metric = list(euclidean = 1:4, manhattan = list(5:10, pca_center = FALSE)). This exists to allow mixed binary and real-valued data to be included and to have PCA applied to both, but with centering applied only to the real-valued data (it is typical not to apply centering to binary data before PCA is applied).

scale Scaling to apply to X if it is a data frame or matrix:

- "none" or FALSE or NULL No scaling.
- "Z" or "scale" or TRUE Scale each column to zero mean and variance 1.
- "maxabs" Center each column to mean 0, then divide each element by the maximum absolute value over the entire matrix.
- "range" Range scale the entire matrix, so the smallest element is 0 and the largest is 1.
- "colrange" Scale each column in the range (0,1).

For method "umap", the default is "none". For "largevis", the default is "maxabs".

set_op_mix_ratio

Interpolate between (fuzzy) union and intersection as the set operation used to combine local fuzzy simplicial sets to obtain a global fuzzy simplicial sets. Both fuzzy set operations use the product t-norm. The value of this parameter should be between 0.0 and 1.0; a value of 1.0 will use a pure fuzzy union, while 0.0 will use a pure fuzzy intersection. Ignored if method = "largevis"

local_connectivity

The local connectivity required – i.e. the number of nearest neighbors that should be assumed to be connected at a local level. The higher this value the more connected the manifold becomes locally. In practice this should be not more than the local intrinsic dimension of the manifold. Ignored if method = "largevis".

nn_method Method for finding nearest neighbors. Options are:

- "fnn". Use exact nearest neighbors via the FNN package.
- "annoy" Use approximate nearest neighbors via the RcppAnnoy package.
- "hnsw" Use approximate nearest neighbors with the Hierarchical Navigable Small World (HNSW) method (Malkov and Yashunin, 2018) via the RcppHNSW package. RcppHNSW is not a dependency of this package: this option is only available if you have installed RcppHNSW yourself. Also, HNSW only supports the following arguments for metric and target_metric: "euclidean", "cosine" and "correlation".
- "nndescent" Use approximate nearest neighbors with the Nearest Neighbor Descent method (Dong et al., 2011) via the rnndescent package. rnndescent is not a dependency of this package: this option is only available if you have installed rnndescent yourself.

By default, if X has less than 4,096 vertices, the exact nearest neighbors are found. Otherwise, approximate nearest neighbors are used. You may also pass pre-calculated nearest neighbor data to this argument. It must be one of two formats, either a list consisting of two elements:

- "idx" (or "index"). A n_vertices x n_neighbors matrix containing the integer indexes of the nearest neighbors in X. *Each vertex is considered to be its own nearest neighbor, i.e.* idx[, 1] == 1:n_vertices.
- "dist" (or "distance"). A n_vertices x n_neighbors matrix containing the distances of the nearest neighbors.

or a sparse distance matrix of type dgCMatrix, with dimensions n_vertices x n_vertices. Distances should be arranged by column, i.e. a non-zero entry in row j of the ith column indicates that the jth observation in X is a nearest neighbor of the ith observation with the distance given by the value of that element.

The n_neighbors parameter is ignored when using precomputed nearest neighbor data. If using the sparse distance matrix input, each column can contain a different number of neighbors.

n_trees — Number of trees to build when constructing the nearest neighbor index. The more trees specified, the larger the index, but the better the results. With search_k, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy". Sensible values are between 10 to 100.

search_k — Number of nodes to search during the neighbor retrieval. The larger k, the more the accurate results, but the longer the search takes. With n_trees, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy".

perplexity — Used only if method = "largevis". Controls the size of the local neighborhood used for manifold approximation. Should be a value between 1 and one less than the number of items in X. If specified, you should *not* specify a value for n_neighbors unless you know what you are doing.

method — How to generate the similarities between items. One of:

- "umap" The UMAP method of McInnes et al. (2018).
- "largevis" The LargeVis method of Tang et al. (2016).

y — Optional target data to add supervised or semi-supervised weighting to the similarity graph . Can be a vector, matrix or data frame. Use the target_metric parameter to specify the metrics to use, using the same syntax as metric. Usually either a single numeric or factor column is used, but more complex formats are possible. The following types are allowed:

- Factor columns with the same length as X. NA is allowed for any observation with an unknown level, in which case UMAP operates as a form of semi-supervised learning. Each column is treated separately.
- Numeric data. NA is *not* allowed in this case. Use the parameter target_n_neighbors to set the number of neighbors used with y. If unset, n_neighbors is used. Unlike factors, numeric columns are grouped into one block unless target_metric specifies otherwise. For example, if you wish columns a and b to be treated separately, specify target_metric = list(euclidean = "a", euclidean = "b"). Otherwise, the data will be effectively treated as a matrix with two columns.
- Nearest neighbor data, consisting of a list of two matrices, idx and dist. These represent the precalculated nearest neighbor indices and distances, respectively. This is the same format as that expected for precalculated data in nn_method. This format assumes that the underlying data was a numeric vector. Any user-supplied value of the target_n_neighbors parameter is ignored in this case, because the the number of columns in the matrices is used for the value. Multiple nearest neighbor data using different metrics can be supplied by passing a list of these lists.

Unlike X, all factor columns included in y are automatically used. This parameter is ignored if method = "largevis".

target_n_neighbors — Number of nearest neighbors to use to construct the target simplicial set. Default value is n_neighbors. Applies only if y is non-NULL and numeric. This parameter is ignored if method = "largevis".

target_metric — The metric used to measure distance for y if using supervised dimension reduction. Used only if y is numeric. This parameter is ignored if method = "largevis".

target_weight
: Weighting factor between data topology and target topology. A value of 0.0 weights entirely on data, a value of 1.0 weights entirely on target. The default of 0.5 balances the weighting equally between data and target. Only applies if y is non-NULL. This parameter is ignored if method = "largevis".

pca
: If set to a positive integer value, reduce data to this number of columns using PCA. Doesn't applied if the distance metric is "hamming", or the dimensions of the data is larger than the number specified (i.e. number of rows and columns must be larger than the value of this parameter). If you have > 100 columns in a data frame or matrix, reducing the number of columns in this way may substantially increase the performance of the nearest neighbor search at the cost of a potential decrease in accuracy. In many t-SNE applications, a value of 50 is recommended, although there's no guarantee that this is appropriate for all settings.

pca_center
: If TRUE, center the columns of X before carrying out PCA. For binary data, it's recommended to set this to FALSE.

ret_extra
: A vector indicating what extra data to return. May contain any combination of the following strings:

  - "nn" nearest neighbor data that can be used as input to nn_method to avoid the overhead of repeatedly calculating the nearest neighbors when manipulating unrelated parameters. See the "Value" section for the names of the list items. Note that the nearest neighbors could be sensitive to data scaling, so be wary of reusing nearest neighbor data if modifying the scale parameter.
  - "sigma" the normalization value for each observation in the dataset when constructing the smoothed distances to each of its neighbors. This gives some sense of the local density of each observation in the high dimensional space: higher values of sigma indicate a higher dispersion or lower density.

n_threads
: Number of threads to use. Default is half the number of concurrent threads supported by the system. For nearest neighbor search, this controls the search phase. For nn_method = "annoy", if n_threads > 1, then the Annoy index will be temporarily written to disk in the location determined by tempfile.

n_build_threads
: Number of threads to use when building nearest neighbor indexes. Default is NULL, which uses n_threads. To improve determinism, use n_build_threads = 1. Only applies for nn_method = "hnsw" or nn_method = "nndescent". The Annoy-based index always use a single thread.

grain_size
: The minimum amount of work to do on each thread. If this value is set high enough, then less than n_threads will be used for processing, which might give a performance improvement if the overhead of thread management and context switching was outweighing the improvement due to concurrent processing. This should be left at default (1) and work will be spread evenly over all the threads specified.

kernel
: Used only if method = "largevis". Type of kernel function to create input similiarties. Can be one of "gauss" (the default) or "knn". "gauss" uses the usual Gaussian weighted similarities. "knn" assigns equal similiarties. to every edge in the nearest neighbor graph, and zero otherwise, using perplexity nearest neighbors. The n_neighbors parameter is ignored in this case.

tmpdir
: Temporary directory to store nearest neighbor indexes during nearest neighbor search. Default is tempdir. The index is only written to disk if n_threads > 1 and nn_method = "annoy"; otherwise, this parameter is ignored.

verbose If TRUE, log details to the console.

pca_method Method to carry out any PCA dimensionality reduction when the pca parameter is specified. Allowed values are:

- "irlba". Uses prcomp_irlba from the irlba package.
- "rsvd". Uses 5 iterations of svdr from the irlba package. This is likely to give much faster but potentially less accurate results than using "irlba". For the purposes of nearest neighbor calculation and coordinates initialization, any loss of accuracy doesn't seem to matter much.
- "bigstatsr". Uses big_randomSVD from the bigstatsr package. The SVD methods used in bigstatsr may be faster on systems without access to efficient linear algebra libraries (e.g. Windows). **Note**: bigstatsr is *not* a dependency of uwot: if you choose to use this package for PCA, you *must* install it yourself.
- "svd". Uses svd for the SVD. This is likely to be slow for all but the smallest datasets.
- "auto" (the default). Uses "irlba", unless more than 50 case "svd" is used.

binary_edge_weights
If TRUE then edge weights of the returned graph are binary (0/1) rather than reflecting the degree of similarity.

nn_args A list containing additional arguments to pass to the nearest neighbor method. For nn_method = "annoy", you can specify "n_trees" and "search_k", and these will override the n_trees and search_k parameters. For nn_method = "hnsw", you may specify the following arguments:

- M The maximum number of neighbors to keep for each vertex. Reasonable values are 2 to 100. Higher values give better recall at the cost of more memory. Default value is 16.
- ef_construction A positive integer specifying the size of the dynamic list used during index construction. A higher value will provide better results at the cost of a longer time to build the index. Default is 200.
- ef A positive integer specifying the size of the dynamic list used during search. This cannot be smaller than n_neighbors and cannot be higher than the number of items in the index. Default is 10.

For nn_method = "nndescent", you may specify the following arguments:

- n_trees The number of trees to use in a random projection forest to initialize the search. A larger number will give more accurate results at the cost of a longer computation time. The default of NULL means that the number is chosen based on the number of observations in X.
- max_candidates The number of potential neighbors to explore per iteration. By default, this is set to n_neighbors or 60, whichever is smaller. A larger number will give more accurate results at the cost of a longer computation time.
- n_iters The number of iterations to run the search. A larger number will give more accurate results at the cost of a longer computation time. By default, this will be chosen based on the number of observations in X. You may also need to modify the convergence criterion delta.
- delta The minimum relative change in the neighbor graph allowed before early stopping. Should be a value between 0 and 1. The smaller the value, the smaller the amount of progress between iterations is allowed. Default

value of 0.001 means that at least 0.1 neighbor graph must be updated at each iteration.

- init How to initialize the nearest neighbor descent. By default this is set to "tree" and uses a random project forest. If you set this to "rand", then a random selection is used. Usually this is less accurate than using RP trees, but for high-dimensional cases, there may be little difference in the quality of the initialization and random initialization will be a lot faster. If you set this to "rand", then the n_trees parameter is ignored.
- pruning_degree_multiplier The maximum number of edges per node to retain in the search graph, relative to n_neighbors. A larger value will give more accurate results at the cost of a longer computation time. Default is 1.5. This parameter only affects neighbor search when transforming new data with umap_transform.
- epsilon Controls the degree of the back-tracking when traversing the search graph. Setting this to 0.0 will do a greedy search with no back-tracking. A larger value will give more accurate results at the cost of a longer computation time. Default is 0.1. This parameter only affects neighbor search when transforming new data with umap_transform.
- max_search_fraction Specifies the maximum fraction of the search graph to traverse. By default, this is set to 1.0, so the entire graph (i.e. all items in X) may be visited. You may want to set this to a smaller value if you have a very large dataset (in conjunction with epsilon) to avoid an inefficient exhaustive search of the data in X. This parameter only affects neighbor search when transforming new data with umap_transform.

Details

This is equivalent to running umap with the ret_extra = c("fgraph") parameter, but without the overhead of calculating (or returning) the optimized low-dimensional coordinates.

Value

A sparse symmetrized matrix of the similarities between the items in X or if nn_method contains pre-computed nearest neighbor data, the items in nn_method. Because of the symmetrization, there may be more non-zero items in each column than the specified value of n_neighbors (or pre-computed neighbors in nn_method). If ret_extra is specified then the return value will be a list containing:

- similarity_graph the similarity graph as a sparse matrix as described above.
- nn (if ret_extra contained "nn") the nearest neighbor data as a list called nn. This contains one list for each metric calculated, itself containing a matrix idx with the integer ids of the neighbors; and a matrix dist with the distances. The nn list (or a sub-list) can be used as input to the nn_method parameter.
- sigma (if ret_extra contains "sigma"), a vector of calibrated parameters, one for each item in the input data, reflecting the local data density for that item. The exact definition of the values depends on the choice of the method parameter.
- rho (if ret_extra contains "sigma"), a vector containing the largest distance to the locally connected neighbors of each item in the input data. This will exist only if method = "umap".
- localr (if ret_extra contains "localr") a vector of the estimated local radii, the sum of "sigma" and "rho". This will exist only if method = "umap".

References

Dong, W., Moses, C., & Li, K. (2011, March). Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World Wide Web* (pp. 577-586). ACM. doi:10.1145/1963405.1963487.

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, *42*(4), 824-836.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction *arXiv preprint arXiv*:1802.03426. https://arxiv.org/abs/1802.03426

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016, April). Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 287-297). International World Wide Web Conferences Steering Committee. https://arxiv.org/abs/1602.00370

Examples

```
iris30 <- iris[c(1:10, 51:60, 101:110), ]

# return a 30 x 30 sparse matrix with similarity data based on 10 nearest
# neighbors per item
iris30_sim_graph <- similarity_graph(iris30, n_neighbors = 10)

# Default is to use the UMAP method of calculating similarities, but LargeVis
# is also available: for that method, use perplexity instead of n_neighbors
# to control neighborhood size. Use ret_extra = "nn" to return nearest
# neighbor data as well as the similarity graph. Return value is a list
# containing similarity_graph' and 'nn' items.
iris30_lv_graph <- similarity_graph(iris30,
  perplexity = 10,
  method = "largevis", ret_extra = "nn"
)
# If you have the neighbor information you don't need the original data
iris30_lv_graph_nn <- similarity_graph(
  nn_method = iris30_lv_graph$nn,
  perplexity = 10, method = "largevis"
)
all(iris30_lv_graph_nn == iris30_lv_graph$similarity_graph)
```

---

simplicial_set_intersect

Merge Similarity Graph by Simplicial Set Intersection

Description

Combine two similarity graphs by treating them as fuzzy topological sets and forming the intersection.

Usage

```
simplicial_set_intersect(x, y, weight = 0.5, n_threads = NULL, verbose = FALSE)
```

Arguments

x	A sparse matrix representing the first similarity graph in the intersection operation.
y	A sparse matrix representing the second similarity graph in the intersection operation.
weight	A value between 0 - 1, controlling the relative influence of x and y in the intersection. Default (0.5) gives equal influence. Values smaller than 0.5 put more weight on x. Values greater than 0.5 put more weight on y.
n_threads	Number of threads to use when resetting the local metric. Default is half the number of concurrent threads supported by the system.
verbose	If TRUE, log progress to the console.

Value

A sparse matrix containing the intersection of x and y.

Examples

```
# Form two different "views" of the same data
iris30 <- iris[c(1:10, 51:60, 101:110), ]
iris_sg12 <- similarity_graph(iris30[, 1:2], n_neighbors = 5)
iris_sg34 <- similarity_graph(iris30[, 3:4], n_neighbors = 5)

# Combine the two representations into one
iris_combined <- simplicial_set_intersect(iris_sg12, iris_sg34)

# Optimize the layout based on the combined view
iris_combined_umap <- optimize_graph_layout(iris_combined, n_epochs = 100)
```

---

simplicial_set_union *Merge Similarity Graph by Simplicial Set Union*

Description

Combine two similarity graphs by treating them as fuzzy topological sets and forming the union.

Usage

```
simplicial_set_union(x, y, n_threads = NULL, verbose = FALSE)
```

Arguments

x	A sparse matrix representing the first similarity graph in the union operation.
y	A sparse matrix representing the second similarity graph in the union operation.
n_threads	Number of threads to use when resetting the local metric. Default is half the number of concurrent threads supported by the system.
verbose	If TRUE, log progress to the console.

Value

A sparse matrix containing the union of x and y.

Examples

```
# Form two different "views" of the same data
iris30 <- iris[c(1:10, 51:60, 101:110), ]
iris_sg12 <- similarity_graph(iris30[, 1:2], n_neighbors = 5)
iris_sg34 <- similarity_graph(iris30[, 3:4], n_neighbors = 5)

# Combine the two representations into one
iris_combined <- simplicial_set_union(iris_sg12, iris_sg34)

# Optimize the layout based on the combined view
iris_combined_umap <- optimize_graph_layout(iris_combined, n_epochs = 100)
```

---

tumap *Dimensionality Reduction Using t-Distributed UMAP (t-UMAP)*

Description

A faster (but less flexible) version of the UMAP (McInnes et al, 2018) gradient. For more detail on UMAP, see the umap function.

Usage

```
tumap(
  X,
  n_neighbors = 15,
  n_components = 2,
  metric = "euclidean",
  n_epochs = NULL,
  learning_rate = 1,
  scale = FALSE,
  init = "spectral",
  init_sdev = NULL,
  set_op_mix_ratio = 1,
  local_connectivity = 1,
  bandwidth = 1,
  repulsion_strength = 1,
  negative_sample_rate = 5,
  nn_method = NULL,
  n_trees = 50,
  search_k = 2 * n_neighbors * n_trees,
  n_threads = NULL,
  n_build_threads = NULL,
  n_sgd_threads = 0,
  grain_size = 1,
  y = NULL,
  target_n_neighbors = n_neighbors,
  target_metric = "euclidean",
  target_weight = 0.5,
  pca = NULL,
  pca_center = TRUE,
  pcg_rand = TRUE,
```

```
  fast_sgd = FALSE,
  ret_model = FALSE,
  ret_nn = FALSE,
  ret_extra = c(),
  tmpdir = tempdir(),
  verbose = getOption("verbose", TRUE),
  batch = FALSE,
  opt_args = NULL,
  epoch_callback = NULL,
  pca_method = NULL,
  binary_edge_weights = FALSE,
  seed = NULL,
  nn_args = list(),
  rng_type = NULL
)
```

Arguments

X	Input data. Can be a data.frame, matrix, dist object or sparseMatrix. Matrix and data frames should contain one observation per row. Data frames will have any non-numeric columns removed, although factor columns will be used if explicitly included via metric (see the help for metric for details). A sparse matrix is interpreted as a distance matrix, and is assumed to be symmetric, so you can also pass in an explicitly upper or lower triangular sparse matrix to save storage. There must be at least n_neighbors non-zero distances for each row. Both implicit and explicit zero entries are ignored. Set zero distances you want to keep to an arbitrarily small non-zero value (e.g. 1e-10). X can also be NULL if pre-computed nearest neighbor data is passed to nn_method, and init is not "spca" or "pca".
n_neighbors	The size of local neighborhood (in terms of number of neighboring sample points) used for manifold approximation. Larger values result in more global views of the manifold, while smaller values result in more local data being preserved. In general values should be in the range 2 to 100.
n_components	The dimension of the space to embed into. This defaults to 2 to provide easy visualization, but can reasonably be set to any integer value in the range 2 to 100.
metric	Type of distance metric to use to find nearest neighbors. For nn_method = "annoy" this can be one of: • "euclidean" (the default) • "cosine" • "manhattan" • "hamming" • "correlation" (a distance based on the Pearson correlation) • "categorical" (see below) For nn_method = "hnsw" this can be one of: • "euclidean" • "cosine" • "correlation" If rnndescent is installed and nn_method = "nndescent" is specified then many more metrics are avaiable, including:

- "braycurtis"
- "canberra"
- "chebyshev"
- "dice"
- "hamming"
- "hellinger"
- "jaccard"
- "jensenshannon"
- "kulsinski"
- "rogerstanimoto"
- "russellrao"
- "sokalmichener"
- "sokalsneath"
- "spearmanr"
- "symmetrickl"
- "tsss"
- "yule"

For more details see the package documentation of rnndescent. For nn_method = "fnn", the distance metric is always "euclidean".

If X is a data frame or matrix, then multiple metrics can be specified, by passing a list to this argument, where the name of each item in the list is one of the metric names above. The value of each list item should be a vector giving the names or integer ids of the columns to be included in a calculation, e.g. metric = list(euclidean = 1:4, manhattan = 5:10).

Each metric calculation results in a separate fuzzy simplicial set, which are intersected together to produce the final set. Metric names can be repeated. Because non-numeric columns are removed from the data frame, it is safer to use column names than integer ids.

Factor columns can also be used by specifying the metric name "categorical". Factor columns are treated different from numeric columns and although multiple factor columns can be specified in a vector, each factor column specified is processed individually. If you specify a non-factor column, it will be coerced to a factor.

For a given data block, you may override the pca and pca_center arguments for that block, by providing a list with one unnamed item containing the column names or ids, and then any of the pca or pca_center overrides as named items, e.g. metric = list(euclidean = 1:4, manhattan = list(5:10, pca_center = FALSE)). This exists to allow mixed binary and real-valued data to be included and to have PCA applied to both, but with centering applied only to the real-valued data (it is typical not to apply centering to binary data before PCA is applied).

n_epochs — Number of epochs to use during the optimization of the embedded coordinates. By default, this value is set to 500 for datasets containing 10,000 vertices or less, and 200 otherwise. If n_epochs = 0, then coordinates determined by "init" will be returned.

learning_rate — Initial learning rate used in optimization of the coordinates.

scale — Scaling to apply to X if it is a data frame or matrix:

- "none" or FALSE or NULL No scaling.

- "Z" or "scale" or TRUE Scale each column to zero mean and variance 1.
- "maxabs" Center each column to mean 0, then divide each element by the maximum absolute value over the entire matrix.
- "range" Range scale the entire matrix, so the smallest element is 0 and the largest is 1.
- "colrange" Scale each column in the range (0,1).

For t-UMAP, the default is "none".

init Type of initialization for the coordinates. Options are:

- "spectral" Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, with Gaussian noise added.
- "normlaplacian". Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, without noise.
- "random". Coordinates assigned using a uniform random distribution between -10 and 10.
- "lvrandom". Coordinates assigned using a Gaussian distribution with standard deviation 1e-4, as used in LargeVis (Tang et al., 2016) and t-SNE.
- "laplacian". Spectral embedding using the Laplacian Eigenmap (Belkin and Niyogi, 2002).
- "pca". The first two principal components from PCA of X if X is a data frame, and from a 2-dimensional classical MDS if X is of class "dist".
- "spca". Like "pca", but each dimension is then scaled so the standard deviation is 1e-4, to give a distribution similar to that used in t-SNE. This is an alias for init = "pca", init_sdev = 1e-4.
- "agspectral" An "approximate global" modification of "spectral" which all edges in the graph to a value of 1, and then sets a random number of edges (negative_sample_rate edges per vertex) to 0.1, to approximate the effect of non-local affinities.
- A matrix of initial coordinates.

For spectral initializations, ("spectral", "normlaplacian", "laplacian", "agspectral"), if more than one connected component is identified, no spectral initialization is attempted. Instead a PCA-based initialization is attempted. If verbose = TRUE the number of connected components are logged to the console. The existence of multiple connected components implies that a global view of the data cannot be attained with this initialization. Increasing the value of n_neighbors may help.

init_sdev If non-NULL, scales each dimension of the initialized coordinates (including any user-supplied matrix) to this standard deviation. By default no scaling is carried out, except when init = "spca", in which case the value is 0.0001. Scaling the input may help if the unscaled versions result in initial coordinates with large inter-point distances or outliers. This usually results in small gradients during optimization and very little progress being made to the layout. Shrinking the initial embedding by rescaling can help under these circumstances. Scaling the result of init = "pca" is usually recommended and init = "spca" as an alias for init = "pca", init_sdev = 1e-4 but for the spectral initializations the scaled versions usually aren't necessary unless you are using a large value of n_neighbors (e.g. n_neighbors = 150 or higher). For compatibility with recent versions of the Python UMAP package, if you are using init = "spectral", then you should also set init_sdev = "range", which will range scale each of the columns containing the initial data between 0-10. This

is not set by default to maintain backwards compatibility with previous versions of uwot.

`set_op_mix_ratio`

Interpolate between (fuzzy) union and intersection as the set operation used to combine local fuzzy simplicial sets to obtain a global fuzzy simplicial sets. Both fuzzy set operations use the product t-norm. The value of this parameter should be between `0.0` and `1.0`; a value of `1.0` will use a pure fuzzy union, while `0.0` will use a pure fuzzy intersection.

`local_connectivity`

The local connectivity required – i.e. the number of nearest neighbors that should be assumed to be connected at a local level. The higher this value the more connected the manifold becomes locally. In practice this should be not more than the local intrinsic dimension of the manifold.

`bandwidth` The effective bandwidth of the kernel if we view the algorithm as similar to Laplacian Eigenmaps. Larger values induce more connectivity and a more global view of the data, smaller values concentrate more locally.

`repulsion_strength`

Weighting applied to negative samples in low dimensional embedding optimization. Values higher than one will result in greater weight being given to negative samples.

`negative_sample_rate`

The number of negative edge/1-simplex samples to use per positive edge/1-simplex sample in optimizing the low dimensional embedding.

`nn_method` Method for finding nearest neighbors. Options are:

- `"fnn"`. Use exact nearest neighbors via the FNN package.
- `"annoy"` Use approximate nearest neighbors via the RcppAnnoy package.
- `"hnsw"` Use approximate nearest neighbors with the Hierarchical Navigable Small World (HNSW) method (Malkov and Yashunin, 2018) via the RcppHNSW package. `RcppHNSW` is not a dependency of this package: this option is only available if you have installed `RcppHNSW` yourself. Also, HNSW only supports the following arguments for `metric` and `target_metric`: `"euclidean"`, `"cosine"` and `"correlation"`.
- `"nndescent"` Use approximate nearest neighbors with the Nearest Neighbor Descent method (Dong et al., 2011) via the rnndescent package. `rnndescent` is not a dependency of this package: this option is only available if you have installed `rnndescent` yourself.

By default, if `X` has less than 4,096 vertices, the exact nearest neighbors are found. Otherwise, approximate nearest neighbors are used. You may also pass pre-calculated nearest neighbor data to this argument. It must be one of two formats, either a list consisting of two elements:

- `"idx"` (or `"index"`). A `n_vertices` x `n_neighbors` matrix containing the integer indexes of the nearest neighbors in `X`. *Each vertex is considered to be its own nearest neighbor, i.e.* `idx[, 1] == 1:n_vertices`.
- `"dist"` (or `"distance"`). A `n_vertices` x `n_neighbors` matrix containing the distances of the nearest neighbors.

or a sparse distance matrix of type `dgCMatrix`, with dimensions `n_vertices` x `n_vertices`. Distances should be arranged by column, i.e. a non-zero entry in row `j` of the `i`th column indicates that the `j`th observation in `X` is a nearest neighbor of the `i`th observation with the distance given by the value of that element.

The n_neighbors parameter is ignored when using precomputed nearest neighbor data. If using the sparse distance matrix input, each column can contain a different number of neighbors.

n_trees — Number of trees to build when constructing the nearest neighbor index. The more trees specified, the larger the index, but the better the results. With search_k, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy". Sensible values are between 10 to 100.

search_k — Number of nodes to search during the neighbor retrieval. The larger k, the more the accurate results, but the longer the search takes. With n_trees, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy".

n_threads — Number of threads to use (except during stochastic gradient descent). Default is half the number of concurrent threads supported by the system. For nearest neighbor search, this controls the search phase. For nn_method = "annoy", if n_threads > 1, then the Annoy index will be temporarily written to disk in the location determined by tempfile.

n_build_threads — Number of threads to use when building nearest neighbor indexes. Default is NULL, which uses n_threads. To improve determinism, use n_build_threads = 1. Only applies for nn_method = "hnsw" or nn_method = "nndescent". The Annoy-based index always use a single thread.

n_sgd_threads — Number of threads to use during stochastic gradient descent. If set to > 1, then be aware that if batch = FALSE, results will *not* be reproducible, even if set.seed is called with a fixed seed before running. Set to "auto" to use the same value as n_threads.

grain_size — The minimum amount of work to do on each thread. If this value is set high enough, then less than n_threads or n_sgd_threads will be used for processing, which might give a performance improvement if the overhead of thread management and context switching was outweighing the improvement due to concurrent processing. This should be left at default (1) and work will be spread evenly over all the threads specified.

y — Optional target data for supervised dimension reduction. Can be a vector, matrix or data frame. Use the target_metric parameter to specify the metrics to use, using the same syntax as metric. Usually either a single numeric or factor column is used, but more complex formats are possible. The following types are allowed:

- Factor columns with the same length as X. NA is allowed for any observation with an unknown level, in which case UMAP operates as a form of semi-supervised learning. Each column is treated separately.
- Numeric data. NA is *not* allowed in this case. Use the parameter target_n_neighbors to set the number of neighbors used with y. If unset, n_neighbors is used. Unlike factors, numeric columns are grouped into one block unless target_metric specifies otherwise. For example, if you wish columns a and b to be treated separately, specify target_metric = list(euclidean = "a", euclidean = "b"). Otherwise, the data will be effectively treated as a matrix with two columns.
- Nearest neighbor data, consisting of a list of two matrices, idx and dist. These represent the precalculated nearest neighbor indices and distances, respectively. This is the same format as that expected for precalculated data in nn_method. This format assumes that the underlying data was a numeric

vector. Any user-supplied value of the `target_n_neighbors` parameter is ignored in this case, because the the number of columns in the matrices is used for the value. Multiple nearest neighbor data using different metrics can be supplied by passing a list of these lists.

Unlike `X`, all factor columns included in `y` are automatically used.

`target_n_neighbors`
: Number of nearest neighbors to use to construct the target simplicial set. Default value is `n_neighbors`. Applies only if `y` is non-`NULL` and `numeric`.

`target_metric`
: The metric used to measure distance for `y` if using supervised dimension reduction. Used only if `y` is numeric.

`target_weight`
: Weighting factor between data topology and target topology. A value of 0.0 weights entirely on data, a value of 1.0 weights entirely on target. The default of 0.5 balances the weighting equally between data and target. Only applies if `y` is non-`NULL`.

`pca`
: If set to a positive integer value, reduce data to this number of columns using PCA. Doesn't applied if the distance `metric` is `"hamming"`, or the dimensions of the data is larger than the number specified (i.e. number of rows and columns must be larger than the value of this parameter). If you have > 100 columns in a data frame or matrix, reducing the number of columns in this way may substantially increase the performance of the nearest neighbor search at the cost of a potential decrease in accuracy. In many t-SNE applications, a value of 50 is recommended, although there's no guarantee that this is appropriate for all settings.

`pca_center`
: If `TRUE`, center the columns of `X` before carrying out PCA. For binary data, it's recommended to set this to `FALSE`.

`pcg_rand`
: If `TRUE`, use the PCG random number generator (O'Neill, 2014) during optimization. Otherwise, use the faster (but probably less statistically good) Tausworthe "taus88" generator. The default is `TRUE`. This parameter has been superseded by `rng_type` – if both are set, `rng_type` takes precedence.

`fast_sgd`
: If `TRUE`, then the following combination of parameters is set: `pcg_rand = TRUE` and `n_sgd_threads = "auto"`. The default is `FALSE`. Setting this to `TRUE` will speed up the stochastic optimization phase, but give a potentially less accurate embedding, and which will not be exactly reproducible even with a fixed seed. For visualization, `fast_sgd = TRUE` will give perfectly good results. For more generic dimensionality reduction, it's safer to leave `fast_sgd = FALSE`. If `fast_sgd = TRUE`, then user-supplied values of `pcg_rand` and `n_sgd_threads`, are ignored.

`ret_model`
: If `TRUE`, then return extra data that can be used to add new data to an existing embedding via `umap_transform`. The embedded coordinates are returned as the list item `embedding`. If `FALSE`, just return the coordinates. This parameter can be used in conjunction with `ret_nn` and `ret_extra`. Note that some settings are incompatible with the production of a UMAP model: external neighbor data (passed via a list to `nn_method`), and factor columns that were included via the `metric` parameter. In the latter case, the model produced is based only on the numeric data. A transformation using new data is possible, but the factor columns in the new data are ignored. Note that setting `ret_model = TRUE` forces the use of the approximate nearest neighbors method. Because small datasets would otherwise use exact nearest neighbor calculations, setting `ret_model = TRUE` means that different results may be returned for small datasets in terms of both the returned nearest neighbors (if requested) and the final embedded

coordinates, compared to ret_model = FALSE, even if the random number seed is fixed. To avoid this, explicitly set nn_method = "annoy" in the ret_model = FALSE case.

ret_nn — If TRUE, then in addition to the embedding, also return nearest neighbor data that can be used as input to nn_method to avoid the overhead of repeatedly calculating the nearest neighbors when manipulating unrelated parameters (e.g. min_dist, n_epochs, init). See the "Value" section for the names of the list items. If FALSE, just return the coordinates. Note that the nearest neighbors could be sensitive to data scaling, so be wary of reusing nearest neighbor data if modifying the scale parameter. This parameter can be used in conjunction with ret_model and ret_extra.

ret_extra — A vector indicating what extra data to return. May contain any combination of the following strings:

- "model" Same as setting ret_model = TRUE.
- "nn" Same as setting ret_nn = TRUE.
- "fgraph" the high dimensional fuzzy graph (i.e. the fuzzy simplicial set of the merged local views of the input data). The graph is returned as a sparse symmetric N x N matrix of class dgCMatrix-class, where a non-zero entry (i, j) gives the membership strength of the edge connecting vertex i and vertex j. This can be considered analogous to the input probability (or similarity or affinity) used in t-SNE and LargeVis. Note that the graph is further sparsified by removing edges with sufficiently low membership strength that they would not be sampled by the probabilistic edge sampling employed for optimization and therefore the number of non-zero elements in the matrix is dependent on n_epochs. If you are only interested in the fuzzy input graph (e.g. for clustering), setting n_epochs = 0 will avoid any further sparsifying. Be aware that setting binary_edge_weights = TRUE will affect this graph (all non-zero edge weights will be 1).
- "sigma" the normalization value for each observation in the dataset when constructing the smoothed distances to each of its neighbors. This gives some sense of the local density of each observation in the high dimensional space: higher values of sigma indicate a higher dispersion or lower density.

tmpdir — Temporary directory to store nearest neighbor indexes during nearest neighbor search. Default is tempdir. The index is only written to disk if n_threads > 1 and nn_method = "annoy"; otherwise, this parameter is ignored.

verbose — If TRUE, log details to the console.

batch — If TRUE, then embedding coordinates are updated at the end of each epoch rather than during the epoch. In batch mode, results are reproducible with a fixed random seed even with n_sgd_threads > 1, at the cost of a slightly higher memory use. You may also have to modify learning_rate and increase n_epochs, so whether this provides a speed increase over the single-threaded optimization is likely to be dataset and hardware-dependent.

opt_args — A list of optimizer parameters, used when batch = TRUE. The default optimization method used is Adam (Kingma and Ba, 2014).

- method The optimization method to use. Either "adam" or "sgd" (stochastic gradient descent). Default: "adam".
- beta1 (Adam only). The weighting parameter for the exponential moving average of the first moment estimator. Effectively the momentum parameter. Should be a floating point value between 0 and 1. Higher values can smooth oscillatory updates in poorly-conditioned situations and may allow

for a larger learning_rate to be specified, but too high can cause divergence. Default: 0.5.

- beta2 (Adam only). The weighting parameter for the exponential moving average of the uncentered second moment estimator. Should be a floating point value between 0 and 1. Controls the degree of adaptivity in the step-size. Higher values put more weight on previous time steps. Default: 0.9.
- eps (Adam only). Intended to be a small value to prevent division by zero, but in practice can also affect convergence due to its interaction with beta2. Higher values reduce the effect of the step-size adaptivity and bring the behavior closer to stochastic gradient descent with momentum. Typical values are between 1e-8 and 1e-3. Default: 1e-7.
- alpha The initial learning rate. Default: the value of the learning_rate parameter.

epoch_callback A function which will be invoked at the end of every epoch. Its signature should be: (epoch, n_epochs, coords), where:

- epoch The current epoch number (between 1 and n_epochs).
- n_epochs Number of epochs to use during the optimization of the embedded coordinates.
- coords The embedded coordinates as of the end of the current epoch, as a matrix with dimensions (N, n_components).

pca_method Method to carry out any PCA dimensionality reduction when the pca parameter is specified. Allowed values are:

- "irlba". Uses prcomp_irlba from the irlba package.
- "rsvd". Uses 5 iterations of svdr from the irlba package. This is likely to give much faster but potentially less accurate results than using "irlba". For the purposes of nearest neighbor calculation and coordinates initialization, any loss of accuracy doesn't seem to matter much.
- "bigstatsr". Uses big_randomSVD from the bigstatsr package. The SVD methods used in bigstatsr may be faster on systems without access to efficient linear algebra libraries (e.g. Windows). **Note**: bigstatsr is *not* a dependency of uwot: if you choose to use this package for PCA, you *must* install it yourself.
- "svd". Uses svd for the SVD. This is likely to be slow for all but the smallest datasets.
- "auto" (the default). Uses "irlba", unless more than 50 case "svd" is used.

binary_edge_weights

If TRUE then edge weights in the input graph are treated as binary (0/1) rather than real valued. This affects the sampling frequency of neighbors and is the strategy used by the PaCMAP method (Wang and co-workers, 2020). Practical (Böhm and co-workers, 2020) and theoretical (Damrich and Hamprecht, 2021) work suggests this has little effect on UMAP's performance.

seed Integer seed to use to initialize the random number generator state. Combined with n_sgd_threads = 1 or batch = TRUE, this should give consistent output across multiple runs on a given installation. Setting this value is equivalent to calling set.seed, but it may be more convenient in some situations than having to call a separate function. The default is to not set a seed. If ret_model = TRUE, the seed will be stored in the output model and then used to set the seed inside umap_transform.

nn_args A list containing additional arguments to pass to the nearest neighbor method. For nn_method = "annoy", you can specify "n_trees" and "search_k", and these will override the n_trees and search_k parameters. For nn_method = "hnsw", you may specify the following arguments:

- M The maximum number of neighbors to keep for each vertex. Reasonable values are 2 to 100. Higher values give better recall at the cost of more memory. Default value is 16.
- ef_construction A positive integer specifying the size of the dynamic list used during index construction. A higher value will provide better results at the cost of a longer time to build the index. Default is 200.
- ef A positive integer specifying the size of the dynamic list used during search. This cannot be smaller than n_neighbors and cannot be higher than the number of items in the index. Default is 10.

For nn_method = "nndescent", you may specify the following arguments:

- n_trees The number of trees to use in a random projection forest to initialize the search. A larger number will give more accurate results at the cost of a longer computation time. The default of NULL means that the number is chosen based on the number of observations in X.
- max_candidates The number of potential neighbors to explore per iteration. By default, this is set to n_neighbors or 60, whichever is smaller. A larger number will give more accurate results at the cost of a longer computation time.
- n_iters The number of iterations to run the search. A larger number will give more accurate results at the cost of a longer computation time. By default, this will be chosen based on the number of observations in X. You may also need to modify the convergence criterion delta.
- delta The minimum relative change in the neighbor graph allowed before early stopping. Should be a value between 0 and 1. The smaller the value, the smaller the amount of progress between iterations is allowed. Default value of 0.001 means that at least 0.1 neighbor graph must be updated at each iteration.
- init How to initialize the nearest neighbor descent. By default this is set to "tree" and uses a random project forest. If you set this to "rand", then a random selection is used. Usually this is less accurate than using RP trees, but for high-dimensional cases, there may be little difference in the quality of the initialization and random initialization will be a lot faster. If you set this to "rand", then the n_trees parameter is ignored.
- pruning_degree_multiplier The maximum number of edges per node to retain in the search graph, relative to n_neighbors. A larger value will give more accurate results at the cost of a longer computation time. Default is 1.5. This parameter only affects neighbor search when transforming new data with umap_transform.
- epsilon Controls the degree of the back-tracking when traversing the search graph. Setting this to 0.0 will do a greedy search with no back-tracking. A larger value will give more accurate results at the cost of a longer computation time. Default is 0.1. This parameter only affects neighbor search when transforming new data with umap_transform.
- max_search_fraction Specifies the maximum fraction of the search graph to traverse. By default, this is set to 1.0, so the entire graph (i.e. all items in X) may be visited. You may want to set this to a smaller value if you have

a very large dataset (in conjunction with epsilon) to avoid an inefficient exhaustive search of the data in X. This parameter only affects neighbor search when transforming new data with umap_transform.

For nn_method = "nndescent", you may specify the following arguments:

- n_trees The number of trees to use in a random projection forest to initialize the search. A larger number will give more accurate results at the cost of a longer computation time. The default of NULL means that the number is chosen based on the number of observations in X.
- max_candidates The number of potential neighbors to explore per iteration. By default, this is set to n_neighbors or 60, whichever is smaller. A larger number will give more accurate results at the cost of a longer computation time.
- n_iters The number of iterations to run the search. A larger number will give more accurate results at the cost of a longer computation time. By default, this will be chosen based on the number of observations in X. You may also need to modify the convergence criterion delta.
- delta The minimum relative change in the neighbor graph allowed before early stopping. Should be a value between 0 and 1. The smaller the value, the smaller the amount of progress between iterations is allowed. Default value of 0.001 means that at least 0.1 neighbor graph must be updated at each iteration.
- init How to initialize the nearest neighbor descent. By default this is set to "tree" and uses a random project forest. If you set this to "rand", then a random selection is used. Usually this is less accurate than using RP trees, but for high-dimensional cases, there may be little difference in the quality of the initialization and random initialization will be a lot faster. If you set this to "rand", then the n_trees parameter is ignored.
- pruning_degree_multiplier The maximum number of edges per node to retain in the search graph, relative to n_neighbors. A larger value will give more accurate results at the cost of a longer computation time. Default is 1.5. This parameter only affects neighbor search when transforming new data with umap_transform.
- epsilon Controls the degree of the back-tracking when traversing the search graph. Setting this to 0.0 will do a greedy search with no back-tracking. A larger value will give more accurate results at the cost of a longer computation time. Default is 0.1. This parameter only affects neighbor search when transforming new data with umap_transform.
- max_search_fraction Specifies the maximum fraction of the search graph to traverse. By default, this is set to 1.0, so the entire graph (i.e. all items in X) may be visited. You may want to set this to a smaller value if you have a very large dataset (in conjunction with epsilon) to avoid an inefficient exhaustive search of the data in X. This parameter only affects neighbor search when transforming new data with umap_transform.

rng_type — The type of random number generator to use during optimization. One of:

- "pcg". Use the PCG random number generator (O'Neill, 2014).
- "tausworthe". Use the Tausworthe "taus88" generator.
- "deterministic". Use a deterministic number generator. This isn't actually random, but may provide enough variation in the negative sampling to give a good embedding and can provide a noticeable speed-up.

For backwards compatibility, by default this is unset and the choice of pcg_rand is used (making "pcg" the effective default).

Details

By setting the UMAP curve parameters a and b to 1, you get back the Cauchy distribution as used in t-SNE (van der Maaten and Hinton, 2008) and LargeVis (Tang et al., 2016). It also results in a substantially simplified gradient expression. This can give a speed improvement of around 50%.

Value

A matrix of optimized coordinates, or:

- if `ret_model = TRUE` (or `ret_extra` contains `"model"`), returns a list containing extra information that can be used to add new data to an existing embedding via `umap_transform`. In this case, the coordinates are available in the list item `embedding`. **NOTE**: The contents of the `model` list should *not* be considered stable or part of the public API, and are purposely left undocumented.
- if `ret_nn = TRUE` (or `ret_extra` contains `"nn"`), returns the nearest neighbor data as a list called `nn`. This contains one list for each `metric` calculated, itself containing a matrix `idx` with the integer ids of the neighbors; and a matrix `dist` with the distances. The `nn` list (or a sub-list) can be used as input to the `nn_method` parameter.
- if `ret_extra` contains `"fgraph"` returns the high dimensional fuzzy graph as a sparse matrix called `fgraph`, of type dgCMatrix-class.
- if `ret_extra` contains `"sigma"`, returns a vector of the smooth knn distance normalization terms for each observation as `"sigma"` and a vector `"rho"` containing the largest distance to the locally connected neighbors of each observation.
- if `ret_extra` contains `"localr"`, returns a vector of the estimated local radii, the sum of `"sigma"` and `"rho"`.

The returned list contains the combined data from any combination of specifying `ret_model`, `ret_nn` and `ret_extra`.

References

Belkin, M., & Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems* (pp. 585-591). http://papers.nips.cc/paper/1961-laplacian-eigenmaps-and-spectral-techniques-for-embedding-and-clustering.pdf

Böhm, J. N., Berens, P., & Kobak, D. (2020). A unifying perspective on neighbor embeddings along the attraction-repulsion spectrum. *arXiv preprint arXiv:2007.08902*. https://arxiv.org/abs/2007.08902

Damrich, S., & Hamprecht, F. A. (2021). On UMAP's true loss function. *Advances in Neural Information Processing Systems*, *34*. https://proceedings.neurips.cc/paper/2021/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html

Dong, W., Moses, C., & Li, K. (2011, March). Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World Wide Web* (pp. 577-586). ACM. doi:10.1145/1963405.1963487.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv*:1412.6980. https://arxiv.org/abs/1412.6980

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, *42*(4), 824-836.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction *arXiv preprint arXiv*:1802.03426. https://arxiv.org/abs/1802.03426

O'Neill, M. E. (2014). *PCG: A family of simple fast space-efficient statistically good algorithms for random number generation* (Report No. HMC-CS-2014-0905). Harvey Mudd College.

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016, April). Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 287-297). International World Wide Web Conferences Steering Committee. https://arxiv.org/abs/1602.00370

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9* (2579-2605). https://www.jmlr.org/papers/v9/vandermaaten08a.html

Wang, Y., Huang, H., Rudin, C., & Shaposhnik, Y. (2021). Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *Journal of Machine Learning Research*, *22*(201), 1-73. https://www.jmlr.org/papers/v22/20-1061.html

Examples

```
iris_tumap <- tumap(iris, n_neighbors = 50, learning_rate = 0.5)
```

---

umap *Dimensionality Reduction with UMAP*

Description

Carry out dimensionality reduction of a dataset using the Uniform Manifold Approximation and Projection (UMAP) method (McInnes et al., 2018). Some of the following help text is lifted verbatim from the Python reference implementation at https://github.com/lmcinnes/umap.

Usage

```
umap(
  X,
  n_neighbors = 15,
  n_components = 2,
  metric = "euclidean",
  n_epochs = NULL,
  learning_rate = 1,
  scale = FALSE,
  init = "spectral",
  init_sdev = NULL,
  spread = 1,
  min_dist = 0.01,
  set_op_mix_ratio = 1,
  local_connectivity = 1,
  bandwidth = 1,
  repulsion_strength = 1,
  negative_sample_rate = 5,
  a = NULL,
```

```
  b = NULL,
  nn_method = NULL,
  n_trees = 50,
  search_k = 2 * n_neighbors * n_trees,
  approx_pow = FALSE,
  y = NULL,
  target_n_neighbors = n_neighbors,
  target_metric = "euclidean",
  target_weight = 0.5,
  pca = NULL,
  pca_center = TRUE,
  pcg_rand = TRUE,
  fast_sgd = FALSE,
  ret_model = FALSE,
  ret_nn = FALSE,
  ret_extra = c(),
  n_threads = NULL,
  n_build_threads = NULL,
  n_sgd_threads = 0,
  grain_size = 1,
  tmpdir = tempdir(),
  verbose = getOption("verbose", TRUE),
  batch = FALSE,
  opt_args = NULL,
  epoch_callback = NULL,
  pca_method = NULL,
  binary_edge_weights = FALSE,
  dens_scale = NULL,
  seed = NULL,
  nn_args = list(),
  rng_type = NULL
)
```

Arguments

X	Input data. Can be a data.frame, matrix, dist object or sparseMatrix. Matrix and data frames should contain one observation per row. Data frames will have any non-numeric columns removed, although factor columns will be used if explicitly included via metric (see the help for metric for details). A sparse matrix is interpreted as a distance matrix, and is assumed to be symmetric, so you can also pass in an explicitly upper or lower triangular sparse matrix to save storage. There must be at least n_neighbors non-zero distances for each row. Both implicit and explicit zero entries are ignored. Set zero distances you want to keep to an arbitrarily small non-zero value (e.g. 1e-10). X can also be NULL if pre-computed nearest neighbor data is passed to nn_method, and init is not "spca" or "pca".
n_neighbors	The size of local neighborhood (in terms of number of neighboring sample points) used for manifold approximation. Larger values result in more global views of the manifold, while smaller values result in more local data being preserved. In general values should be in the range 2 to 100.
n_components	The dimension of the space to embed into. This defaults to 2 to provide easy visualization, but can reasonably be set to any integer value in the range 2 to

100.

metric

Type of distance metric to use to find nearest neighbors. For nn_method = "annoy" this can be one of:

- "euclidean" (the default)
- "cosine"
- "manhattan"
- "hamming"
- "correlation" (a distance based on the Pearson correlation)
- "categorical" (see below)

For nn_method = "hnsw" this can be one of:

- "euclidean"
- "cosine"
- "correlation"

If rnndescent is installed and nn_method = "nndescent" is specified then many more metrics are avaiable, including:

- "braycurtis"
- "canberra"
- "chebyshev"
- "dice"
- "hamming"
- "hellinger"
- "jaccard"
- "jensenshannon"
- "kulsinski"
- "rogerstanimoto"
- "russellrao"
- "sokalmichener"
- "sokalsneath"
- "spearmanr"
- "symmetrickl"
- "tsss"
- "yule"

For more details see the package documentation of rnndescent. For nn_method = "fnn", the distance metric is always "euclidean".

If X is a data frame or matrix, then multiple metrics can be specified, by passing a list to this argument, where the name of each item in the list is one of the metric names above. The value of each list item should be a vector giving the names or integer ids of the columns to be included in a calculation, e.g. metric = list(euclidean = 1:4, manhattan = 5:10).

Each metric calculation results in a separate fuzzy simplicial set, which are intersected together to produce the final set. Metric names can be repeated. Because non-numeric columns are removed from the data frame, it is safer to use column names than integer ids.

Factor columns can also be used by specifying the metric name "categorical". Factor columns are treated different from numeric columns and although multiple factor columns can be specified in a vector, each factor column specified is

processed individually. If you specify a non-factor column, it will be coerced to a factor.

For a given data block, you may override the pca and pca_center arguments for that block, by providing a list with one unnamed item containing the column names or ids, and then any of the pca or pca_center overrides as named items, e.g. metric = list(euclidean = 1:4, manhattan = list(5:10, pca_center = FALSE)). This exists to allow mixed binary and real-valued data to be included and to have PCA applied to both, but with centering applied only to the real-valued data (it is typical not to apply centering to binary data before PCA is applied).

n_epochs — Number of epochs to use during the optimization of the embedded coordinates. By default, this value is set to 500 for datasets containing 10,000 vertices or less, and 200 otherwise. If n_epochs = 0, then coordinates determined by "init" will be returned.

learning_rate — Initial learning rate used in optimization of the coordinates.

scale — Scaling to apply to X if it is a data frame or matrix:

- "none" or FALSE or NULL No scaling.
- "Z" or "scale" or TRUE Scale each column to zero mean and variance 1.
- "maxabs" Center each column to mean 0, then divide each element by the maximum absolute value over the entire matrix.
- "range" Range scale the entire matrix, so the smallest element is 0 and the largest is 1.
- "colrange" Scale each column in the range (0,1).

For UMAP, the default is "none".

init — Type of initialization for the coordinates. Options are:

- "spectral" Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, with Gaussian noise added.
- "normlaplacian". Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, without noise.
- "random". Coordinates assigned using a uniform random distribution between -10 and 10.
- "lvrandom". Coordinates assigned using a Gaussian distribution with standard deviation 1e-4, as used in LargeVis (Tang et al., 2016) and t-SNE.
- "laplacian". Spectral embedding using the Laplacian Eigenmap (Belkin and Niyogi, 2002).
- "pca". The first two principal components from PCA of X if X is a data frame, and from a 2-dimensional classical MDS if X is of class "dist".
- "spca". Like "pca", but each dimension is then scaled so the standard deviation is 1e-4, to give a distribution similar to that used in t-SNE. This is an alias for init = "pca", init_sdev = 1e-4.
- "agspectral" An "approximate global" modification of "spectral" which all edges in the graph to a value of 1, and then sets a random number of edges (negative_sample_rate edges per vertex) to 0.1, to approximate the effect of non-local affinities.
- A matrix of initial coordinates.

For spectral initializations, ("spectral", "normlaplacian", "laplacian", "agspectral"), if more than one connected component is identified, no spectral initialization is attempted. Instead a PCA-based initialization is attempted. If verbose = TRUE

the number of connected components are logged to the console. The existence of multiple connected components implies that a global view of the data cannot be attained with this initialization. Increasing the value of `n_neighbors` may help.

`init_sdev` If non-`NULL`, scales each dimension of the initialized coordinates (including any user-supplied matrix) to this standard deviation. By default no scaling is carried out, except when `init = "spca"`, in which case the value is `0.0001`. Scaling the input may help if the unscaled versions result in initial coordinates with large inter-point distances or outliers. This usually results in small gradients during optimization and very little progress being made to the layout. Shrinking the initial embedding by rescaling can help under these circumstances. Scaling the result of `init = "pca"` is usually recommended and `init = "spca"` as an alias for `init = "pca", init_sdev = 1e-4` but for the spectral initializations the scaled versions usually aren't necessary unless you are using a large value of `n_neighbors` (e.g. `n_neighbors = 150` or higher). For compatibility with recent versions of the Python UMAP package, if you are using `init = "spectral"`, then you should also set `init_sdev = "range"`, which will range scale each of the columns containing the initial data between 0-10. This is not set by default to maintain backwards compatibility with previous versions of uwot.

`spread` The effective scale of embedded points. In combination with `min_dist`, this determines how clustered/clumped the embedded points are.

`min_dist` The effective minimum distance between embedded points. Smaller values will result in a more clustered/clumped embedding where nearby points on the manifold are drawn closer together, while larger values will result on a more even dispersal of points. The value should be set relative to the `spread` value, which determines the scale at which embedded points will be spread out.

`set_op_mix_ratio`
Interpolate between (fuzzy) union and intersection as the set operation used to combine local fuzzy simplicial sets to obtain a global fuzzy simplicial sets. Both fuzzy set operations use the product t-norm. The value of this parameter should be between `0.0` and `1.0`; a value of `1.0` will use a pure fuzzy union, while `0.0` will use a pure fuzzy intersection.

`local_connectivity`
The local connectivity required – i.e. the number of nearest neighbors that should be assumed to be connected at a local level. The higher this value the more connected the manifold becomes locally. In practice this should be not more than the local intrinsic dimension of the manifold.

`bandwidth` The effective bandwidth of the kernel if we view the algorithm as similar to Laplacian Eigenmaps. Larger values induce more connectivity and a more global view of the data, smaller values concentrate more locally.

`repulsion_strength`
Weighting applied to negative samples in low dimensional embedding optimization. Values higher than one will result in greater weight being given to negative samples.

`negative_sample_rate`
The number of negative edge/1-simplex samples to use per positive edge/1-simplex sample in optimizing the low dimensional embedding.

`a` More specific parameters controlling the embedding. If `NULL` these values are set automatically as determined by `min_dist` and `spread`.

b — More specific parameters controlling the embedding. If NULL these values are set automatically as determined by min_dist and spread.

nn_method — Method for finding nearest neighbors. Options are:

- "fnn". Use exact nearest neighbors via the FNN package.
- "annoy" Use approximate nearest neighbors via the RcppAnnoy package.
- "hnsw" Use approximate nearest neighbors with the Hierarchical Navigable Small World (HNSW) method (Malkov and Yashunin, 2018) via the RcppHNSW package. RcppHNSW is not a dependency of this package: this option is only available if you have installed RcppHNSW yourself. Also, HNSW only supports the following arguments for metric and target_metric: "euclidean", "cosine" and "correlation".
- "nndescent" Use approximate nearest neighbors with the Nearest Neighbor Descent method (Dong et al., 2011) via the rnndescent package. rnndescent is not a dependency of this package: this option is only available if you have installed rnndescent yourself.

By default, if X has less than 4,096 vertices, the exact nearest neighbors are found. Otherwise, approximate nearest neighbors are used. You may also pass pre-calculated nearest neighbor data to this argument. It must be one of two formats, either a list consisting of two elements:

- "idx" (or "index"). A n_vertices x n_neighbors matrix containing the integer indexes of the nearest neighbors in X. *Each vertex is considered to be its own nearest neighbor, i.e.* idx[, 1] == 1:n_vertices.
- "dist" (or "distance"). A n_vertices x n_neighbors matrix containing the distances of the nearest neighbors.

or a sparse distance matrix of type dgCMatrix, with dimensions n_vertices x n_vertices. Distances should be arranged by column, i.e. a non-zero entry in row j of the ith column indicates that the jth observation in X is a nearest neighbor of the ith observation with the distance given by the value of that element. The n_neighbors parameter is ignored when using precomputed nearest neighbor data. If using the sparse distance matrix input, each column can contain a different number of neighbors.

n_trees — Number of trees to build when constructing the nearest neighbor index. The more trees specified, the larger the index, but the better the results. With search_k, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy". Sensible values are between 10 to 100.

search_k — Number of nodes to search during the neighbor retrieval. The larger k, the more the accurate results, but the longer the search takes. With n_trees, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy".

approx_pow — If TRUE, use an approximation to the power function in the UMAP gradient, from https://martin.ankerl.com/2012/01/25/optimized-approximative-pow-in-c-and-cpp/. Ignored if dens_scale is non-NULL.

y — Optional target data for supervised dimension reduction. Can be a vector, matrix or data frame. Use the target_metric parameter to specify the metrics to use, using the same syntax as metric. Usually either a single numeric or factor column is used, but more complex formats are possible. The following types are allowed:

- Factor columns with the same length as X. NA is allowed for any observation with an unknown level, in which case UMAP operates as a form of semi-supervised learning. Each column is treated separately.

- Numeric data. NA is *not* allowed in this case. Use the parameter target_n_neighbors to set the number of neighbors used with y. If unset, n_neighbors is used. Unlike factors, numeric columns are grouped into one block unless target_metric specifies otherwise. For example, if you wish columns a and b to be treated separately, specify target_metric = list(euclidean = "a", euclidean = "b"). Otherwise, the data will be effectively treated as a matrix with two columns.
- Nearest neighbor data, consisting of a list of two matrices, idx and dist. These represent the precalculated nearest neighbor indices and distances, respectively. This is the same format as that expected for precalculated data in nn_method. This format assumes that the underlying data was a numeric vector. Any user-supplied value of the target_n_neighbors parameter is ignored in this case, because the the number of columns in the matrices is used for the value. Multiple nearest neighbor data using different metrics can be supplied by passing a list of these lists.

Unlike X, all factor columns included in y are automatically used.

target_n_neighbors
: Number of nearest neighbors to use to construct the target simplicial set. Default value is n_neighbors. Applies only if y is non-NULL and numeric.

target_metric
: The metric used to measure distance for y if using supervised dimension reduction. Used only if y is numeric.

target_weight
: Weighting factor between data topology and target topology. A value of 0.0 weights entirely on data, a value of 1.0 weights entirely on target. The default of 0.5 balances the weighting equally between data and target. Only applies if y is non-NULL.

pca
: If set to a positive integer value, reduce data to this number of columns using PCA. Doesn't applied if the distance metric is "hamming", or the dimensions of the data is larger than the number specified (i.e. number of rows and columns must be larger than the value of this parameter). If you have > 100 columns in a data frame or matrix, reducing the number of columns in this way may substantially increase the performance of the nearest neighbor search at the cost of a potential decrease in accuracy. In many t-SNE applications, a value of 50 is recommended, although there's no guarantee that this is appropriate for all settings.

pca_center
: If TRUE, center the columns of X before carrying out PCA. For binary data, it's recommended to set this to FALSE.

pcg_rand
: If TRUE, use the PCG random number generator (O'Neill, 2014) during optimization. Otherwise, use the faster (but probably less statistically good) Tausworthe "taus88" generator. The default is TRUE. This parameter has been superseded by rng_type – if both are set, rng_type takes precedence.

fast_sgd
: If TRUE, then the following combination of parameters is set: pcg_rand = TRUE, n_sgd_threads = "auto" and approx_pow = TRUE. The default is FALSE. Setting this to TRUE will speed up the stochastic optimization phase, but give a potentially less accurate embedding, and which will not be exactly reproducible even with a fixed seed. For visualization, fast_sgd = TRUE will give perfectly good results. For more generic dimensionality reduction, it's safer to leave fast_sgd = FALSE. If fast_sgd = TRUE, then user-supplied values of pcg_rand, n_sgd_threads, and approx_pow are ignored.

ret_model
: If TRUE, then return extra data that can be used to add new data to an existing embedding via umap_transform. The embedded coordinates are returned as the

list item embedding. If FALSE, just return the coordinates. This parameter can be used in conjunction with ret_nn and ret_extra. Note that some settings are incompatible with the production of a UMAP model: external neighbor data (passed via a list to nn_method), and factor columns that were included via the metric parameter. In the latter case, the model produced is based only on the numeric data. A transformation using new data is possible, but the factor columns in the new data are ignored. Note that setting ret_model = TRUE forces the use of the approximate nearest neighbors method. Because small datasets would otherwise use exact nearest neighbor calculations, setting ret_model = TRUE means that different results may be returned for small datasets in terms of both the returned nearest neighbors (if requested) and the final embedded coordinates, compared to ret_model = FALSE, even if the random number seed is fixed. To avoid this, explicitly set nn_method = "annoy" in the ret_model = FALSE case.

ret_nn

If TRUE, then in addition to the embedding, also return nearest neighbor data that can be used as input to nn_method to avoid the overhead of repeatedly calculating the nearest neighbors when manipulating unrelated parameters (e.g. min_dist, n_epochs, init). See the "Value" section for the names of the list items. If FALSE, just return the coordinates. Note that the nearest neighbors could be sensitive to data scaling, so be wary of reusing nearest neighbor data if modifying the scale parameter. This parameter can be used in conjunction with ret_model and ret_extra.

ret_extra

A vector indicating what extra data to return. May contain any combination of the following strings:

- "model" Same as setting ret_model = TRUE.
- "nn" Same as setting ret_nn = TRUE.
- "fgraph" the high dimensional fuzzy graph (i.e. the fuzzy simplicial set of the merged local views of the input data). The graph is returned as a sparse symmetric N x N matrix of class dgCMatrix-class, where a non-zero entry (i, j) gives the membership strength of the edge connecting vertex i and vertex j. This can be considered analogous to the input probability (or similarity or affinity) used in t-SNE and LargeVis. Note that the graph is further sparsified by removing edges with sufficiently low membership strength that they would not be sampled by the probabilistic edge sampling employed for optimization and therefore the number of non-zero elements in the matrix is dependent on n_epochs. If you are only interested in the fuzzy input graph (e.g. for clustering), setting n_epochs = 0 will avoid any further sparsifying. Be aware that setting 'binary_edge_weights = TRUE' will affect this graph (all non-zero edge weights will be 1).
- "sigma" the normalization value for each observation in the dataset when constructing the smoothed distances to each of its neighbors. This gives some sense of the local density of each observation in the high dimensional space: higher values of sigma indicate a higher dispersion or lower density.

n_threads

Number of threads to use (except during stochastic gradient descent). Default is half the number of concurrent threads supported by the system. For nearest neighbor search, this controls the search phase. For nn_method = "annoy", if n_threads > 1, then the Annoy index will be temporarily written to disk in the location determined by tempfile.

n_build_threads

Number of threads to use when building nearest neighbor indexes. Default is NULL, which uses n_threads. To improve determinism, use n_build_threads

= 1. Only applies for nn_method = "hnsw" or nn_method = "nndescent". The Annoy-based index always use a single thread.

n_sgd_threads Number of threads to use during stochastic gradient descent. If set to > 1, then be aware that if batch = FALSE, results will *not* be reproducible, even if set.seed is called with a fixed seed before running. Set to "auto" to use the same value as n_threads.

grain_size The minimum amount of work to do on each thread. If this value is set high enough, then less than n_threads or n_sgd_threads will be used for processing, which might give a performance improvement if the overhead of thread management and context switching was outweighing the improvement due to concurrent processing. This should be left at default (1) and work will be spread evenly over all the threads specified.

tmpdir Temporary directory to store nearest neighbor indexes during nearest neighbor search. Default is tempdir. The index is only written to disk if n_threads > 1 and nn_method = "annoy"; otherwise, this parameter is ignored.

verbose If TRUE, log details to the console.

batch If TRUE, then embedding coordinates are updated at the end of each epoch rather than during the epoch. In batch mode, results are reproducible with a fixed random seed even with n_sgd_threads > 1, at the cost of a slightly higher memory use. You may also have to modify learning_rate and increase n_epochs, so whether this provides a speed increase over the single-threaded optimization is likely to be dataset and hardware-dependent.

opt_args A list of optimizer parameters, used when batch = TRUE. The default optimization method used is Adam (Kingma and Ba, 2014).

- method The optimization method to use. Either "adam" or "sgd" (stochastic gradient descent). Default: "adam".
- beta1 (Adam only). The weighting parameter for the exponential moving average of the first moment estimator. Effectively the momentum parameter. Should be a floating point value between 0 and 1. Higher values can smooth oscillatory updates in poorly-conditioned situations and may allow for a larger learning_rate to be specified, but too high can cause divergence. Default: 0.5.
- beta2 (Adam only). The weighting parameter for the exponential moving average of the uncentered second moment estimator. Should be a floating point value between 0 and 1. Controls the degree of adaptivity in the step-size. Higher values put more weight on previous time steps. Default: 0.9.
- eps (Adam only). Intended to be a small value to prevent division by zero, but in practice can also affect convergence due to its interaction with beta2. Higher values reduce the effect of the step-size adaptivity and bring the behavior closer to stochastic gradient descent with momentum. Typical values are between 1e-8 and 1e-3. Default: 1e-7.
- alpha The initial learning rate. Default: the value of the learning_rate parameter.

epoch_callback A function which will be invoked at the end of every epoch. Its signature should be: (epoch, n_epochs, coords), where:

- epoch The current epoch number (between 1 and n_epochs).
- n_epochs Number of epochs to use during the optimization of the embedded coordinates.

- coords The embedded coordinates as of the end of the current epoch, as a matrix with dimensions (N, n_components).

pca_method
: Method to carry out any PCA dimensionality reduction when the pca parameter is specified. Allowed values are:
  - "irlba". Uses prcomp_irlba from the irlba package.
  - "rsvd". Uses 5 iterations of svdr from the irlba package. This is likely to give much faster but potentially less accurate results than using "irlba". For the purposes of nearest neighbor calculation and coordinates initialization, any loss of accuracy doesn't seem to matter much.
  - "bigstatsr". Uses big_randomSVD from the bigstatsr package. The SVD methods used in bigstatsr may be faster on systems without access to efficient linear algebra libraries (e.g. Windows). **Note**: bigstatsr is *not* a dependency of uwot: if you choose to use this package for PCA, you *must* install it yourself.
  - "svd". Uses svd for the SVD. This is likely to be slow for all but the smallest datasets.
  - "auto" (the default). Uses "irlba", unless more than 50 case "svd" is used.

binary_edge_weights
: If TRUE then edge weights in the input graph are treated as binary (0/1) rather than real valued. This affects the sampling frequency of neighbors and is the strategy used by the PaCMAP method (Wang and co-workers, 2020). Practical (Böhm and co-workers, 2020) and theoretical (Damrich and Hamprecht, 2021) work suggests this has little effect on UMAP's performance.

dens_scale
: A value between 0 and 1. If > 0 then the output attempts to preserve relative local density around each observation. This uses an approximation to the densMAP method (Narayan and co-workers, 2021). The larger the value of dens_scale, the greater the range of output densities that will be used to map the input densities. This option is ignored if using multiple metric blocks.

seed
: Integer seed to use to initialize the random number generator state. Combined with n_sgd_threads = 1 or batch = TRUE, this should give consistent output across multiple runs on a given installation. Setting this value is equivalent to calling set.seed, but it may be more convenient in some situations than having to call a separate function. The default is to not set a seed. If ret_model = TRUE, the seed will be stored in the output model and then used to set the seed inside umap_transform.

nn_args
: A list containing additional arguments to pass to the nearest neighbor method. For nn_method = "annoy", you can specify "n_trees" and "search_k", and these will override the n_trees and search_k parameters. For nn_method = "hnsw", you may specify the following arguments:
  - M The maximum number of neighbors to keep for each vertex. Reasonable values are 2 to 100. Higher values give better recall at the cost of more memory. Default value is 16.
  - ef_construction A positive integer specifying the size of the dynamic list used during index construction. A higher value will provide better results at the cost of a longer time to build the index. Default is 200.
  - ef A positive integer specifying the size of the dynamic list used during search. This cannot be smaller than n_neighbors and cannot be higher than the number of items in the index. Default is 10.

  For nn_method = "nndescent", you may specify the following arguments:

- `n_trees` The number of trees to use in a random projection forest to initialize the search. A larger number will give more accurate results at the cost of a longer computation time. The default of `NULL` means that the number is chosen based on the number of observations in `X`.
- `max_candidates` The number of potential neighbors to explore per iteration. By default, this is set to `n_neighbors` or `60`, whichever is smaller. A larger number will give more accurate results at the cost of a longer computation time.
- `n_iters` The number of iterations to run the search. A larger number will give more accurate results at the cost of a longer computation time. By default, this will be chosen based on the number of observations in `X`. You may also need to modify the convergence criterion `delta`.
- `delta` The minimum relative change in the neighbor graph allowed before early stopping. Should be a value between 0 and 1. The smaller the value, the smaller the amount of progress between iterations is allowed. Default value of `0.001` means that at least 0.1 neighbor graph must be updated at each iteration.
- `init` How to initialize the nearest neighbor descent. By default this is set to `"tree"` and uses a random project forest. If you set this to `"rand"`, then a random selection is used. Usually this is less accurate than using RP trees, but for high-dimensional cases, there may be little difference in the quality of the initialization and random initialization will be a lot faster. If you set this to `"rand"`, then the `n_trees` parameter is ignored.
- `pruning_degree_multiplier` The maximum number of edges per node to retain in the search graph, relative to `n_neighbors`. A larger value will give more accurate results at the cost of a longer computation time. Default is `1.5`. This parameter only affects neighbor search when transforming new data with `umap_transform`.
- `epsilon` Controls the degree of the back-tracking when traversing the search graph. Setting this to `0.0` will do a greedy search with no back-tracking. A larger value will give more accurate results at the cost of a longer computation time. Default is `0.1`. This parameter only affects neighbor search when transforming new data with `umap_transform`.
- `max_search_fraction` Specifies the maximum fraction of the search graph to traverse. By default, this is set to `1.0`, so the entire graph (i.e. all items in `X`) may be visited. You may want to set this to a smaller value if you have a very large dataset (in conjunction with `epsilon`) to avoid an inefficient exhaustive search of the data in `X`. This parameter only affects neighbor search when transforming new data with `umap_transform`.

`rng_type` The type of random number generator to use during optimization. One of:

- `"pcg"`. Use the PCG random number generator (O'Neill, 2014).
- `"tausworthe"`. Use the Tausworthe "taus88" generator.
- `"deterministic"`. Use a deterministic number generator. This isn't actually random, but may provide enough variation in the negative sampling to give a good embedding and can provide a noticeable speed-up.

For backwards compatibility, by default this is unset and the choice of `pcg_rand` is used (making "pcg" the effective default).

Value

A matrix of optimized coordinates, or:

- if `ret_model = TRUE` (or `ret_extra` contains `"model"`), returns a list containing extra information that can be used to add new data to an existing embedding via `umap_transform`. In this case, the coordinates are available in the list item `embedding`. **NOTE**: The contents of the `model` list should *not* be considered stable or part of the public API, and are purposely left undocumented.
- if `ret_nn = TRUE` (or `ret_extra` contains `"nn"`), returns the nearest neighbor data as a list called `nn`. This contains one list for each `metric` calculated, itself containing a matrix `idx` with the integer ids of the neighbors; and a matrix `dist` with the distances. The `nn` list (or a sub-list) can be used as input to the `nn_method` parameter.
- if `ret_extra` contains `"fgraph"`, returns the high dimensional fuzzy graph as a sparse matrix called `fgraph`, of type dgCMatrix-class.
- if `ret_extra` contains `"sigma"`, returns a vector of the smooth knn distance normalization terms for each observation as `"sigma"` and a vector `"rho"` containing the largest distance to the locally connected neighbors of each observation.
- if `ret_extra` contains `"localr"`, returns a vector of the estimated local radii, the sum of `"sigma"` and `"rho"`.

The returned list contains the combined data from any combination of specifying `ret_model`, `ret_nn` and `ret_extra`.

References

Belkin, M., & Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems* (pp. 585-591). http://papers.nips.cc/paper/1961-laplacian-eigenmaps-and-spectral-techniques-for-embedding-and-clustering.pdf

Böhm, J. N., Berens, P., & Kobak, D. (2020). A unifying perspective on neighbor embeddings along the attraction-repulsion spectrum. *arXiv preprint arXiv:2007.08902*. https://arxiv.org/abs/2007.08902

Damrich, S., & Hamprecht, F. A. (2021). On UMAP's true loss function. *Advances in Neural Information Processing Systems*, *34*. https://proceedings.neurips.cc/paper/2021/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html

Dong, W., Moses, C., & Li, K. (2011, March). Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World Wide Web* (pp. 577-586). ACM. doi:10.1145/1963405.1963487.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv*:1412.6980. https://arxiv.org/abs/1412.6980

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, *42*(4), 824-836.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction *arXiv preprint arXiv*:1802.03426. https://arxiv.org/abs/1802.03426

Narayan, A., Berger, B., & Cho, H. (2021). Assessing single-cell transcriptomic variability through density-preserving data visualization. *Nature biotechnology*, *39*(6), 765-774. doi:10.1038/s41587-020008017

O'Neill, M. E. (2014). *PCG: A family of simple fast space-efficient statistically good algorithms for random number generation* (Report No. HMC-CS-2014-0905). Harvey Mudd College.

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016, April). Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 287-297). International World Wide Web Conferences Steering Committee. https://arxiv.org/abs/1602.00370

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9* (2579-2605). https://www.jmlr.org/papers/v9/vandermaaten08a.html

Wang, Y., Huang, H., Rudin, C., & Shaposhnik, Y. (2021). Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *Journal of Machine Learning Research*, *22*(201), 1-73. https://www.jmlr.org/papers/v22/20-1061.html

Examples

```
iris30 <- iris[c(1:10, 51:60, 101:110), ]

# Non-numeric columns are automatically removed so you can pass data frames
# directly in a lot of cases without pre-processing
iris_umap <- umap(iris30, n_neighbors = 5, learning_rate = 0.5, init = "random", n_epochs = 20)

# Faster approximation to the gradient and return nearest neighbors
iris_umap <- umap(iris30, n_neighbors = 5, approx_pow = TRUE, ret_nn = TRUE, n_epochs = 20)

# Can specify min_dist and spread parameters to control separation and size
# of clusters and reuse nearest neighbors for efficiency
nn <- iris_umap$nn
iris_umap <- umap(iris30, n_neighbors = 5, min_dist = 1, spread = 5, nn_method = nn, n_epochs = 20)

# Supervised dimension reduction using the 'Species' factor column
iris_sumap <- umap(iris30,
  n_neighbors = 5, min_dist = 0.001, y = iris30$Species,
  target_weight = 0.5, n_epochs = 20
)

# Calculate Petal and Sepal neighbors separately (uses intersection of the resulting sets):
iris_umap <- umap(iris30, metric = list(
  "euclidean" = c("Sepal.Length", "Sepal.Width"),
  "euclidean" = c("Petal.Length", "Petal.Width")
))
```

---

umap2 *Dimensionality Reduction with UMAP*

Description

Carry out dimensionality reduction of a dataset using the Uniform Manifold Approximation and Projection (UMAP) method (McInnes et al., 2018).

Usage

```
umap2(
  X,
  n_neighbors = 15,
```

```
  n_components = 2,
  metric = "euclidean",
  n_epochs = NULL,
  learning_rate = 1,
  scale = FALSE,
  init = "spectral",
  init_sdev = "range",
  spread = 1,
  min_dist = 0.1,
  set_op_mix_ratio = 1,
  local_connectivity = 1,
  bandwidth = 1,
  repulsion_strength = 1,
  negative_sample_rate = 5,
  a = NULL,
  b = NULL,
  nn_method = NULL,
  n_trees = 50,
  search_k = 2 * n_neighbors * n_trees,
  approx_pow = FALSE,
  y = NULL,
  target_n_neighbors = n_neighbors,
  target_metric = "euclidean",
  target_weight = 0.5,
  pca = NULL,
  pca_center = TRUE,
  pcg_rand = TRUE,
  fast_sgd = FALSE,
  ret_model = FALSE,
  ret_nn = FALSE,
  ret_extra = c(),
  n_threads = NULL,
  n_build_threads = NULL,
  n_sgd_threads = 0,
  grain_size = 1,
  tmpdir = tempdir(),
  verbose = getOption("verbose", TRUE),
  batch = TRUE,
  opt_args = NULL,
  epoch_callback = NULL,
  pca_method = NULL,
  binary_edge_weights = FALSE,
  dens_scale = NULL,
  seed = NULL,
  nn_args = list(),
  rng_type = NULL
)
```

Arguments

X	Input data. Can be a data.frame, matrix, dist object or sparseMatrix. Matrix and data frames should contain one observation per row. Data frames will have any non-numeric columns removed, although factor columns will be used

if explicitly included via `metric` (see the help for `metric` for details). Sparse matrices must be in the `dgCMatrix` format, and you must also install rnndescent and set `nn_method = "nndescent"` `X` can also be `NULL` if pre-computed nearest neighbor data is passed to `nn_method`, and `init` is not `"spca"` or `"pca"`.

`n_neighbors` The size of local neighborhood (in terms of number of neighboring sample points) used for manifold approximation. Larger values result in more global views of the manifold, while smaller values result in more local data being preserved. In general values should be in the range `2` to `100`.

`n_components` The dimension of the space to embed into. This defaults to `2` to provide easy visualization, but can reasonably be set to any integer value in the range `2` to `100`.

`metric` Type of distance metric to use to find nearest neighbors. For `nn_method = "annoy"` this can be one of:

- `"euclidean"` (the default)
- `"cosine"`
- `"manhattan"`
- `"hamming"`
- `"correlation"` (a distance based on the Pearson correlation)
- `"categorical"` (see below)

For `nn_method = "hnsw"` this can be one of:

- `"euclidean"`
- `"cosine"`
- `"correlation"`

If rnndescent is installed and `nn_method = "nndescent"` is specified then many more metrics are avaiable, including:

- `"braycurtis"`
- `"canberra"`
- `"chebyshev"`
- `"dice"`
- `"hamming"`
- `"hellinger"`
- `"jaccard"`
- `"jensenshannon"`
- `"kulsinski"`
- `"rogerstanimoto"`
- `"russellrao"`
- `"sokalmichener"`
- `"sokalsneath"`
- `"spearmanr"`
- `"symmetrickl"`
- `"tsss"`
- `"yule"`

For more details see the package documentation of `rnndescent`. For `nn_method = "fnn"`, the distance metric is always "euclidean".

If `X` is a data frame or matrix, then multiple metrics can be specified, by passing a list to this argument, where the name of each item in the list is one of the

metric names above. The value of each list item should be a vector giving the names or integer ids of the columns to be included in a calculation, e.g. `metric = list(euclidean = 1:4, manhattan = 5:10)`.

Each metric calculation results in a separate fuzzy simplicial set, which are intersected together to produce the final set. Metric names can be repeated. Because non-numeric columns are removed from the data frame, it is safer to use column names than integer ids.

Factor columns can also be used by specifying the metric name `"categorical"`. Factor columns are treated different from numeric columns and although multiple factor columns can be specified in a vector, each factor column specified is processed individually. If you specify a non-factor column, it will be coerced to a factor.

For a given data block, you may override the `pca` and `pca_center` arguments for that block, by providing a list with one unnamed item containing the column names or ids, and then any of the `pca` or `pca_center` overrides as named items, e.g. `metric = list(euclidean = 1:4, manhattan = list(5:10, pca_center = FALSE))`. This exists to allow mixed binary and real-valued data to be included and to have PCA applied to both, but with centering applied only to the real-valued data (it is typical not to apply centering to binary data before PCA is applied).

`n_epochs` Number of epochs to use during the optimization of the embedded coordinates. By default, this value is set to `500` for datasets containing 10,000 vertices or less, and `200` otherwise. If `n_epochs = 0`, then coordinates determined by `"init"` will be returned.

`learning_rate` Initial learning rate used in optimization of the coordinates.

`scale` Scaling to apply to `X` if it is a data frame or matrix:

- `"none"` or `FALSE` or `NULL` No scaling.
- `"Z"` or `"scale"` or `TRUE` Scale each column to zero mean and variance 1.
- `"maxabs"` Center each column to mean 0, then divide each element by the maximum absolute value over the entire matrix.
- `"range"` Range scale the entire matrix, so the smallest element is 0 and the largest is 1.
- `"colrange"` Scale each column in the range (0,1).

For UMAP, the default is `"none"`.

`init` Type of initialization for the coordinates. Options are:

- `"spectral"` Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, with Gaussian noise added.
- `"normlaplacian"`. Spectral embedding using the normalized Laplacian of the fuzzy 1-skeleton, without noise.
- `"random"`. Coordinates assigned using a uniform random distribution between -10 and 10.
- `"lvrandom"`. Coordinates assigned using a Gaussian distribution with standard deviation 1e-4, as used in LargeVis (Tang et al., 2016) and t-SNE.
- `"laplacian"`. Spectral embedding using the Laplacian Eigenmap (Belkin and Niyogi, 2002).
- `"pca"`. The first two principal components from PCA of `X` if `X` is a data frame, and from a 2-dimensional classical MDS if `X` is of class `"dist"`.

- "spca". Like "pca", but each dimension is then scaled so the standard deviation is 1e-4, to give a distribution similar to that used in t-SNE. This is an alias for init = "pca", init_sdev = 1e-4.
- "agspectral" An "approximate global" modification of "spectral" which all edges in the graph to a value of 1, and then sets a random number of edges (negative_sample_rate edges per vertex) to 0.1, to approximate the effect of non-local affinities.
- A matrix of initial coordinates.

For spectral initializations, ("spectral", "normlaplacian", "laplacian", "agspectral"), if more than one connected component is identified, no spectral initialization is attempted. Instead a PCA-based initialization is attempted. If verbose = TRUE the number of connected components are logged to the console. The existence of multiple connected components implies that a global view of the data cannot be attained with this initialization. Increasing the value of n_neighbors may help.

init_sdev
: If non-NULL, scales each dimension of the initialized coordinates (including any user-supplied matrix) to this standard deviation. By default, (init_sdev = "range"), each column of the initial coordinates are range scaled between 0-10. Scaling the input may help if the unscaled versions result in initial coordinates with large inter-point distances or outliers. This usually results in small gradients during optimization and very little progress being made to the layout. Shrinking the initial embedding by rescaling can help under these circumstances. Scaling the result of init = "pca" is usually recommended and init = "spca" as an alias for init = "pca", init_sdev = 1e-4 but for the spectral initializations the scaled versions usually aren't necessary unless you are using a large value of n_neighbors (e.g. n_neighbors = 150 or higher).

spread
: The effective scale of embedded points. In combination with min_dist, this determines how clustered/clumped the embedded points are.

min_dist
: The effective minimum distance between embedded points. Smaller values will result in a more clustered/clumped embedding where nearby points on the manifold are drawn closer together, while larger values will result on a more even dispersal of points. The value should be set relative to the spread value, which determines the scale at which embedded points will be spread out.

set_op_mix_ratio
: Interpolate between (fuzzy) union and intersection as the set operation used to combine local fuzzy simplicial sets to obtain a global fuzzy simplicial sets. Both fuzzy set operations use the product t-norm. The value of this parameter should be between 0.0 and 1.0; a value of 1.0 will use a pure fuzzy union, while 0.0 will use a pure fuzzy intersection.

local_connectivity
: The local connectivity required – i.e. the number of nearest neighbors that should be assumed to be connected at a local level. The higher this value the more connected the manifold becomes locally. In practice this should be not more than the local intrinsic dimension of the manifold.

bandwidth
: The effective bandwidth of the kernel if we view the algorithm as similar to Laplacian Eigenmaps. Larger values induce more connectivity and a more global view of the data, smaller values concentrate more locally.

repulsion_strength
: Weighting applied to negative samples in low dimensional embedding optimization. Values higher than one will result in greater weight being given to negative samples.

negative_sample_rate
: The number of negative edge/1-simplex samples to use per positive edge/1-simplex sample in optimizing the low dimensional embedding.

a
: More specific parameters controlling the embedding. If NULL these values are set automatically as determined by min_dist and spread.

b
: More specific parameters controlling the embedding. If NULL these values are set automatically as determined by min_dist and spread.

nn_method
: Method for finding nearest neighbors. Options are:

  - "fnn". Use exact nearest neighbors via the FNN package.
  - "annoy" Use approximate nearest neighbors via the RcppAnnoy package.
  - "hnsw" Use approximate nearest neighbors with the Hierarchical Navigable Small World (HNSW) method (Malkov and Yashunin, 2018) via the RcppHNSW package. RcppHNSW is not a dependency of this package: this option is only available if you have installed RcppHNSW yourself. Also, HNSW only supports the following arguments for metric and target_metric: "euclidean", "cosine" and "correlation".
  - "nndescent" Use approximate nearest neighbors with the Nearest Neighbor Descent method (Dong et al., 2011) via the rnndescent package. rnndescent is not a dependency of this package: this option is only available if you have installed rnndescent yourself.

  By default, if X has less than 4,096 vertices, the exact nearest neighbors are found. Otherwise, approximate nearest neighbors are used. You may also pass pre-calculated nearest neighbor data to this argument. It must be one of two formats, either a list consisting of two elements:

  - "idx". A n_vertices x n_neighbors matrix containing the integer indexes of the nearest neighbors in X. Each vertex is considered to be its own nearest neighbor, i.e. idx[, 1] == 1:n_vertices.
  - "dist". A n_vertices x n_neighbors matrix containing the distances of the nearest neighbors.

  or a sparse distance matrix of type dgCMatrix, with dimensions n_vertices x n_vertices. Distances should be arranged by column, i.e. a non-zero entry in row j of the ith column indicates that the jth observation in X is a nearest neighbor of the ith observation with the distance given by the value of that element. The n_neighbors parameter is ignored when using precomputed nearest neighbor data. If using the sparse distance matrix input, each column can contain a different number of neighbors.

n_trees
: Number of trees to build when constructing the nearest neighbor index. The more trees specified, the larger the index, but the better the results. With search_k, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy". Sensible values are between 10 to 100.

search_k
: Number of nodes to search during the neighbor retrieval. The larger k, the more the accurate results, but the longer the search takes. With n_trees, determines the accuracy of the Annoy nearest neighbor search. Only used if the nn_method is "annoy".

approx_pow
: If TRUE, use an approximation to the power function in the UMAP gradient, from https://martin.ankerl.com/2012/01/25/optimized-approximative-pow-in-c-and-cpp/. Ignored if dens_scale is non-NULL.

y
: Optional target data for supervised dimension reduction. Can be a vector, matrix or data frame. Use the target_metric parameter to specify the metrics to use,

using the same syntax as metric. Usually either a single numeric or factor column is used, but more complex formats are possible. The following types are allowed:

- Factor columns with the same length as X. NA is allowed for any observation with an unknown level, in which case UMAP operates as a form of semi-supervised learning. Each column is treated separately.
- Numeric data. NA is *not* allowed in this case. Use the parameter target_n_neighbors to set the number of neighbors used with y. If unset, n_neighbors is used. Unlike factors, numeric columns are grouped into one block unless target_metric specifies otherwise. For example, if you wish columns a and b to be treated separately, specify target_metric = list(euclidean = "a", euclidean = "b"). Otherwise, the data will be effectively treated as a matrix with two columns.
- Nearest neighbor data, consisting of a list of two matrices, idx and dist. These represent the precalculated nearest neighbor indices and distances, respectively. This is the same format as that expected for precalculated data in nn_method. This format assumes that the underlying data was a numeric vector. Any user-supplied value of the target_n_neighbors parameter is ignored in this case, because the the number of columns in the matrices is used for the value. Multiple nearest neighbor data using different metrics can be supplied by passing a list of these lists.

Unlike X, all factor columns included in y are automatically used.

target_n_neighbors

Number of nearest neighbors to use to construct the target simplicial set. Default value is n_neighbors. Applies only if y is non-NULL and numeric.

target_metric The metric used to measure distance for y if using supervised dimension reduction. Used only if y is numeric.

target_weight Weighting factor between data topology and target topology. A value of 0.0 weights entirely on data, a value of 1.0 weights entirely on target. The default of 0.5 balances the weighting equally between data and target. Only applies if y is non-NULL.

pca If set to a positive integer value, reduce data to this number of columns using PCA. Doesn't applied if the distance metric is "hamming", or the dimensions of the data is larger than the number specified (i.e. number of rows and columns must be larger than the value of this parameter). If you have > 100 columns in a data frame or matrix, reducing the number of columns in this way may substantially increase the performance of the nearest neighbor search at the cost of a potential decrease in accuracy. In many t-SNE applications, a value of 50 is recommended, although there's no guarantee that this is appropriate for all settings.

pca_center If TRUE, center the columns of X before carrying out PCA. For binary data, it's recommended to set this to FALSE.

pcg_rand If TRUE, use the PCG random number generator (O'Neill, 2014) during optimization. Otherwise, use the faster (but probably less statistically good) Tausworthe "taus88" generator. The default is TRUE. This parameter has been superseded by rng_type – if both are set, rng_type takes precedence.

fast_sgd If TRUE, then the following combination of parameters is set: pcg_rand = TRUE, n_sgd_threads = "auto" and approx_pow = TRUE. The default is FALSE. Setting this to TRUE will speed up the stochastic optimization phase, but give a potentially less accurate embedding, and which will not be exactly reproducible

even with a fixed seed. For visualization, `fast_sgd = TRUE` will give perfectly good results. For more generic dimensionality reduction, it's safer to leave `fast_sgd = FALSE`. If `fast_sgd = TRUE`, then user-supplied values of `pcg_rand`, `n_sgd_threads`, and `approx_pow` are ignored.

`ret_model` If `TRUE`, then return extra data that can be used to add new data to an existing embedding via `umap_transform`. The embedded coordinates are returned as the list item `embedding`. If `FALSE`, just return the coordinates. This parameter can be used in conjunction with `ret_nn` and `ret_extra`. Note that some settings are incompatible with the production of a UMAP model: external neighbor data (passed via a list to `nn_method`), and factor columns that were included via the `metric` parameter. In the latter case, the model produced is based only on the numeric data. A transformation using new data is possible, but the factor columns in the new data are ignored. Note that setting `ret_model = TRUE` forces the use of the approximate nearest neighbors method. Because small datasets would otherwise use exact nearest neighbor calculations, setting `ret_model = TRUE` means that different results may be returned for small datasets in terms of both the returned nearest neighbors (if requested) and the final embedded coordinates, compared to `ret_model = FALSE`, even if the random number seed is fixed. To avoid this, explicitly set `nn_method = "annoy"` in the `ret_model = FALSE` case.

`ret_nn` If `TRUE`, then in addition to the embedding, also return nearest neighbor data that can be used as input to `nn_method` to avoid the overhead of repeatedly calculating the nearest neighbors when manipulating unrelated parameters (e.g. `min_dist`, `n_epochs`, `init`). See the "Value" section for the names of the list items. If `FALSE`, just return the coordinates. Note that the nearest neighbors could be sensitive to data scaling, so be wary of reusing nearest neighbor data if modifying the `scale` parameter. This parameter can be used in conjunction with `ret_model` and `ret_extra`.

`ret_extra` A vector indicating what extra data to return. May contain any combination of the following strings:

- `"model"` Same as setting `ret_model = TRUE`.
- `"nn"` Same as setting `ret_nn = TRUE`.
- `"fgraph"` the high dimensional fuzzy graph (i.e. the fuzzy simplicial set of the merged local views of the input data). The graph is returned as a sparse symmetric N x N matrix of class dgCMatrix-class, where a non-zero entry (i, j) gives the membership strength of the edge connecting vertex i and vertex j. This can be considered analogous to the input probability (or similarity or affinity) used in t-SNE and LargeVis. Note that the graph is further sparsified by removing edges with sufficiently low membership strength that they would not be sampled by the probabilistic edge sampling employed for optimization and therefore the number of non-zero elements in the matrix is dependent on `n_epochs`. If you are only interested in the fuzzy input graph (e.g. for clustering), setting `n_epochs = 0` will avoid any further sparsifying. Be aware that setting 'binary_edge_weights = TRUE' will affect this graph (all non-zero edge weights will be 1).
- `"sigma"` the normalization value for each observation in the dataset when constructing the smoothed distances to each of its neighbors. This gives some sense of the local density of each observation in the high dimensional space: higher values of `sigma` indicate a higher dispersion or lower density.

`n_threads` Number of threads to use (except during stochastic gradient descent). Default is half the number of concurrent threads supported by the system. For nearest

neighbor search, this controls the search phase. For nn_method = "annoy", if n_threads > 1, then the Annoy index will be temporarily written to disk in the location determined by tempfile.

n_build_threads
: Number of threads to use when building nearest neighbor indexes. Default is NULL, which uses n_threads. To improve determinism, use n_build_threads = 1. Only applies for nn_method = "hnsw" or nn_method = "nndescent". The Annoy-based index always use a single thread.

n_sgd_threads
: Number of threads to use during stochastic gradient descent. If set to > 1, then be aware that if batch = FALSE, results will *not* be reproducible, even if set.seed is called with a fixed seed before running. Set to "auto" to use the same value as n_threads. Default is to use only one thread, unless batch = TRUE in which case "auto" used.

grain_size
: The minimum amount of work to do on each thread. If this value is set high enough, then less than n_threads or n_sgd_threads will be used for processing, which might give a performance improvement if the overhead of thread management and context switching was outweighing the improvement due to concurrent processing. This should be left at default (1) and work will be spread evenly over all the threads specified.

tmpdir
: Temporary directory to store nearest neighbor indexes during nearest neighbor search. Default is tempdir. The index is only written to disk if n_threads > 1 and nn_method = "annoy"; otherwise, this parameter is ignored.

verbose
: If TRUE, log details to the console.

batch
: If TRUE, then embedding coordinates are updated at the end of each epoch rather than during the epoch. In batch mode, results are reproducible with a fixed random seed even with n_sgd_threads > 1, at the cost of a slightly higher memory use. You may also have to modify learning_rate and increase n_epochs, so whether this provides a speed increase over the single-threaded optimization is likely to be dataset and hardware-dependent.

opt_args
: A list of optimizer parameters, used when batch = TRUE. The default optimization method used is Adam (Kingma and Ba, 2014).

  - method The optimization method to use. Either "adam" or "sgd" (stochastic gradient descent). Default: "adam".
  - beta1 (Adam only). The weighting parameter for the exponential moving average of the first moment estimator. Effectively the momentum parameter. Should be a floating point value between 0 and 1. Higher values can smooth oscillatory updates in poorly-conditioned situations and may allow for a larger learning_rate to be specified, but too high can cause divergence. Default: 0.5.
  - beta2 (Adam only). The weighting parameter for the exponential moving average of the uncentered second moment estimator. Should be a floating point value between 0 and 1. Controls the degree of adaptivity in the step-size. Higher values put more weight on previous time steps. Default: 0.9.
  - eps (Adam only). Intended to be a small value to prevent division by zero, but in practice can also affect convergence due to its interaction with beta2. Higher values reduce the effect of the step-size adaptivity and bring the behavior closer to stochastic gradient descent with momentum. Typical values are between 1e-8 and 1e-3. Default: 1e-7.
  - alpha The initial learning rate. Default: the value of the learning_rate parameter.

epoch_callback A function which will be invoked at the end of every epoch. Its signature should be: `(epoch, n_epochs, coords)`, where:

- `epoch` The current epoch number (between 1 and `n_epochs`).
- `n_epochs` Number of epochs to use during the optimization of the embedded coordinates.
- `coords` The embedded coordinates as of the end of the current epoch, as a matrix with dimensions (N, `n_components`).

pca_method Method to carry out any PCA dimensionality reduction when the `pca` parameter is specified. Allowed values are:

- `"irlba"`. Uses `prcomp_irlba` from the irlba package.
- `"rsvd"`. Uses 5 iterations of `svdr` from the irlba package. This is likely to give much faster but potentially less accurate results than using `"irlba"`. For the purposes of nearest neighbor calculation and coordinates initialization, any loss of accuracy doesn't seem to matter much.
- `"bigstatsr"`. Uses `big_randomSVD` from the bigstatsr package. The SVD methods used in `bigstatsr` may be faster on systems without access to efficient linear algebra libraries (e.g. Windows). **Note**: `bigstatsr` is *not* a dependency of uwot: if you choose to use this package for PCA, you *must* install it yourself.
- `"svd"`. Uses `svd` for the SVD. This is likely to be slow for all but the smallest datasets.
- `"auto"` (the default). Uses `"irlba"`, unless more than 50 case `"svd"` is used.

binary_edge_weights If `TRUE` then edge weights in the input graph are treated as binary (0/1) rather than real valued. This affects the sampling frequency of neighbors and is the strategy used by the PaCMAP method (Wang and co-workers, 2020). Practical (Böhm and co-workers, 2020) and theoretical (Damrich and Hamprecht, 2021) work suggests this has little effect on UMAP's performance.

dens_scale A value between 0 and 1. If > 0 then the output attempts to preserve relative local density around each observation. This uses an approximation to the densMAP method (Narayan and co-workers, 2021). The larger the value of `dens_scale`, the greater the range of output densities that will be used to map the input densities. This option is ignored if using multiple `metric` blocks.

seed Integer seed to use to initialize the random number generator state. Combined with `n_sgd_threads = 1` or `batch = TRUE`, this should give consistent output across multiple runs on a given installation. Setting this value is equivalent to calling `set.seed`, but it may be more convenient in some situations than having to call a separate function. The default is to not set a seed. If `ret_model = TRUE`, the seed will be stored in the output model and then used to set the seed inside `umap_transform`.

nn_args A list containing additional arguments to pass to the nearest neighbor method. For `nn_method = "annoy"`, you can specify `"n_trees"` and `"search_k"`, and these will override the `n_trees` and `search_k` parameters. For `nn_method = "hnsw"`, you may specify the following arguments:

- `M` The maximum number of neighbors to keep for each vertex. Reasonable values are `2` to `100`. Higher values give better recall at the cost of more memory. Default value is `16`.

- `ef_construction` A positive integer specifying the size of the dynamic list used during index construction. A higher value will provide better results at the cost of a longer time to build the index. Default is `200`.
- `ef` A positive integer specifying the size of the dynamic list used during search. This cannot be smaller than `n_neighbors` and cannot be higher than the number of items in the index. Default is `10`.

For `nn_method = "nndescent"`, you may specify the following arguments:

- `n_trees` The number of trees to use in a random projection forest to initialize the search. A larger number will give more accurate results at the cost of a longer computation time. The default of `NULL` means that the number is chosen based on the number of observations in `X`.
- `max_candidates` The number of potential neighbors to explore per iteration. By default, this is set to `n_neighbors` or `60`, whichever is smaller. A larger number will give more accurate results at the cost of a longer computation time.
- `n_iters` The number of iterations to run the search. A larger number will give more accurate results at the cost of a longer computation time. By default, this will be chosen based on the number of observations in `X`. You may also need to modify the convergence criterion `delta`.
- `delta` The minimum relative change in the neighbor graph allowed before early stopping. Should be a value between 0 and 1. The smaller the value, the smaller the amount of progress between iterations is allowed. Default value of `0.001` means that at least 0.1 neighbor graph must be updated at each iteration.
- `init` How to initialize the nearest neighbor descent. By default this is set to `"tree"` and uses a random project forest. If you set this to `"rand"`, then a random selection is used. Usually this is less accurate than using RP trees, but for high-dimensional cases, there may be little difference in the quality of the initialization and random initialization will be a lot faster. If you set this to `"rand"`, then the `n_trees` parameter is ignored.
- `pruning_degree_multiplier` The maximum number of edges per node to retain in the search graph, relative to `n_neighbors`. A larger value will give more accurate results at the cost of a longer computation time. Default is `1.5`. This parameter only affects neighbor search when transforming new data with `umap_transform`.
- `epsilon` Controls the degree of the back-tracking when traversing the search graph. Setting this to `0.0` will do a greedy search with no back-tracking. A larger value will give more accurate results at the cost of a longer computation time. Default is `0.1`. This parameter only affects neighbor search when transforming new data with `umap_transform`.
- `max_search_fraction` Specifies the maximum fraction of the search graph to traverse. By default, this is set to `1.0`, so the entire graph (i.e. all items in `X`) may be visited. You may want to set this to a smaller value if you have a very large dataset (in conjunction with `epsilon`) to avoid an inefficient exhaustive search of the data in `X`. This parameter only affects neighbor search when transforming new data with `umap_transform`.

`rng_type` The type of random number generator to use during optimization. One of:

- `"pcg"`. Use the PCG random number generator (O'Neill, 2014).
- `"tausworthe"`. Use the Tausworthe "taus88" generator.

- "deterministic". Use a deterministic number generator. This isn't actually random, but may provide enough variation in the negative sampling to give a good embedding and can provide a noticeable speed-up.

For backwards compatibility, by default this is unset and the choice of pcg_rand is used (making "pcg" the effective default).

Details

This function behaves like umap except with some updated defaults that make it behave more like the Python implementation and which cannot be added to umap without breaking backwards compatibility. In addition:

- if RcppHNSW is installed, it will be used in preference to Annoy if a compatible metric is requested.
- if RcppHNSW is not present, but rnndescent is installed, it will be used in preference to Annoy if a compatible metric is requested.
- if batch = TRUE then the default n_sgd_threads is set to the same value as n_threads.
- if the input data X is a sparse matrix, it is interpreted similarly to a dense matrix or dataframe, and not as a distance matrix. This requires rnndescent package to be installed.

Value

A matrix of optimized coordinates, or:

- if ret_model = TRUE (or ret_extra contains "model"), returns a list containing extra information that can be used to add new data to an existing embedding via umap_transform. In this case, the coordinates are available in the list item embedding. **NOTE**: The contents of the model list should *not* be considered stable or part of the public API, and are purposely left undocumented.
- if ret_nn = TRUE (or ret_extra contains "nn"), returns the nearest neighbor data as a list called nn. This contains one list for each metric calculated, itself containing a matrix idx with the integer ids of the neighbors; and a matrix dist with the distances. The nn list (or a sub-list) can be used as input to the nn_method parameter.
- if ret_extra contains "fgraph", returns the high dimensional fuzzy graph as a sparse matrix called fgraph, of type dgCMatrix-class.
- if ret_extra contains "sigma", returns a vector of the smooth knn distance normalization terms for each observation as "sigma" and a vector "rho" containing the largest distance to the locally connected neighbors of each observation.
- if ret_extra contains "localr", returns a vector of the estimated local radii, the sum of "sigma" and "rho".

The returned list contains the combined data from any combination of specifying ret_model, ret_nn and ret_extra.

References

Belkin, M., & Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems* (pp. 585-591). http://papers.nips.cc/paper/1961-laplacian-eigenmaps-and-spectral-techniques-for-embedding-and-clustering.pdf

Böhm, J. N., Berens, P., & Kobak, D. (2020). A unifying perspective on neighbor embeddings along the attraction-repulsion spectrum. *arXiv preprint arXiv:2007.08902*. https://arxiv.org/abs/2007.08902

Damrich, S., & Hamprecht, F. A. (2021). On UMAP's true loss function. *Advances in Neural Information Processing Systems*, *34*. https://proceedings.neurips.cc/paper/2021/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html

Dong, W., Moses, C., & Li, K. (2011, March). Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World Wide Web* (pp. 577-586). ACM. doi:10.1145/1963405.1963487.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv*:1412.6980. https://arxiv.org/abs/1412.6980

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, *42*(4), 824-836.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction *arXiv preprint arXiv*:1802.03426. https://arxiv.org/abs/1802.03426

Narayan, A., Berger, B., & Cho, H. (2021). Assessing single-cell transcriptomic variability through density-preserving data visualization. *Nature biotechnology*, *39*(6), 765-774. doi:10.1038/s41587-020008017

O'Neill, M. E. (2014). *PCG: A family of simple fast space-efficient statistically good algorithms for random number generation* (Report No. HMC-CS-2014-0905). Harvey Mudd College.

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016, April). Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 287-297). International World Wide Web Conferences Steering Committee. https://arxiv.org/abs/1602.00370

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9* (2579-2605). https://www.jmlr.org/papers/v9/vandermaaten08a.html

Wang, Y., Huang, H., Rudin, C., & Shaposhnik, Y. (2021). Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *Journal of Machine Learning Research*, *22*(201), 1-73. https://www.jmlr.org/papers/v22/20-1061.html

Examples

```
iris30 <- iris[c(1:10, 51:60, 101:110), ]
iris_umap <- umap2(iris30, n_neighbors = 5)
```

`umap_transform` *Add New Points to an Existing Embedding*

Description

Carry out an embedding of new data using an existing embedding. Requires using the result of calling `umap` or `tumap` with `ret_model = TRUE`.

Usage

```
umap_transform(
  X = NULL,
  model = NULL,
  nn_method = NULL,
  init_weighted = TRUE,
  search_k = NULL,
  tmpdir = tempdir(),
  n_epochs = NULL,
  n_threads = NULL,
  n_sgd_threads = 0,
  grain_size = 1,
  verbose = FALSE,
  init = "weighted",
  batch = NULL,
  learning_rate = NULL,
  opt_args = NULL,
  epoch_callback = NULL,
  ret_extra = NULL,
  seed = NULL
)
```

Arguments

X	The new data to be transformed, either a matrix of data frame. Must have the same columns in the same order as the input data used to generate the model.
model	Data associated with an existing embedding.
nn_method	Optional pre-calculated nearest neighbor data. There are two supported formats. The first is a list consisting of two elements: • "idx". A n_vertices x n_neighbors matrix where n_vertices is the number of observations in X. The contents of the matrix should be the integer indexes of the data used to generate the model, which are the n_neighbors-nearest neighbors of the data to be transformed. • "dist". A n_vertices x n_neighbors matrix containing the distances of the nearest neighbors. The second supported format is a sparse distance matrix of type dgCMatrix, with dimensions n_model_vertices x n_vertices. where n_model_vertices is the number of observations in the original data that generated the model. Distances should be arranged by column, i.e. a non-zero entry in row j of the ith column indicates that the jth observation in the original data used to generate the model is a nearest neighbor of the ith observation in the new data, with the distance given by the value of that element. In this format, a different number of neighbors is allowed for each observation, i.e. each column can contain a different number of non-zero values. Multiple nearest neighbor data (e.g. from two different pre-calculated metrics) can be passed by passing a list containing the nearest neighbor data lists as items.
init_weighted	If TRUE, then initialize the embedded coordinates of X using a weighted average of the coordinates of the nearest neighbors from the original embedding in model, where the weights used are the edge weights from the UMAP smoothed knn distances. Otherwise, use an un-weighted average. This parameter will be deprecated and removed at version 1.0 of this package. Use the init parameter

as a replacement, replacing `init_weighted = TRUE` with `init = "weighted"` and `init_weighted = FALSE` with `init = "average"`.

`search_k` Number of nodes to search during the neighbor retrieval. The larger k, the more the accurate results, but the longer the search takes. Default is the value used in building the `model` is used.

`tmpdir` Temporary directory to store nearest neighbor indexes during nearest neighbor search. Default is `tempdir`. The index is only written to disk if `n_threads > 1`; otherwise, this parameter is ignored.

`n_epochs` Number of epochs to use during the optimization of the embedded coordinates. A value between `30 - 100` is a reasonable trade off between speed and thoroughness. By default, this value is set to one third the number of epochs used to build the `model`.

`n_threads` Number of threads to use, (except during stochastic gradient descent). Default is half the number of concurrent threads supported by the system.

`n_sgd_threads` Number of threads to use during stochastic gradient descent. If set to > 1, then be aware that if `batch = FALSE`, results will *not* be reproducible, even if `set.seed` is called with a fixed seed before running. Set to `"auto"` to use the same value as `n_threads`.

`grain_size` Minimum batch size for multithreading. If the number of items to process in a thread falls below this number, then no threads will be used. Used in conjunction with `n_threads` and `n_sgd_threads`.

`verbose` If `TRUE`, log details to the console.

`init` how to initialize the transformed coordinates. One of:

- `"weighted"` (The default). Use a weighted average of the coordinates of the nearest neighbors from the original embedding in `model`, where the weights used are the edge weights from the UMAP smoothed knn distances. Equivalent to `init_weighted = TRUE`.
- `"average"`. Use the mean average of the coordinates of the nearest neighbors from the original embedding in `model`. Equivalent to `init_weighted = FALSE`.
- A matrix of user-specified input coordinates, which must have dimensions the same as `(nrow(X), ncol(model$embedding))`.

This parameter should be used in preference to `init_weighted`.

`batch` If `TRUE`, then embedding coordinates are updated at the end of each epoch rather than during the epoch. In batch mode, results are reproducible with a fixed random seed even with `n_sgd_threads > 1`, at the cost of a slightly higher memory use. You may also have to modify `learning_rate` and increase `n_epochs`, so whether this provides a speed increase over the single-threaded optimization is likely to be dataset and hardware-dependent. If `NULL`, the transform will use the value provided in the `model`, if available. Default: `FALSE`.

`learning_rate` Initial learning rate used in optimization of the coordinates. This overrides the value associated with the `model`. This should be left unspecified under most circumstances.

`opt_args` A list of optimizer parameters, used when `batch = TRUE`. The default optimization method used is Adam (Kingma and Ba, 2014).

- `method` The optimization method to use. Either `"adam"` or `"sgd"` (stochastic gradient descent). Default: `"adam"`.

- beta1 (Adam only). The weighting parameter for the exponential moving average of the first moment estimator. Effectively the momentum parameter. Should be a floating point value between 0 and 1. Higher values can smooth oscillatory updates in poorly-conditioned situations and may allow for a larger learning_rate to be specified, but too high can cause divergence. Default: 0.5.
- beta2 (Adam only). The weighting parameter for the exponential moving average of the uncentered second moment estimator. Should be a floating point value between 0 and 1. Controls the degree of adaptivity in the step-size. Higher values put more weight on previous time steps. Default: 0.9.
- eps (Adam only). Intended to be a small value to prevent division by zero, but in practice can also affect convergence due to its interaction with beta2. Higher values reduce the effect of the step-size adaptivity and bring the behavior closer to stochastic gradient descent with momentum. Typical values are between 1e-8 and 1e-3. Default: 1e-7.
- alpha The initial learning rate. Default: the value of the learning_rate parameter.

If NULL, the transform will use the value provided in the model, if available.

epoch_callback A function which will be invoked at the end of every epoch. Its signature should be: (epoch, n_epochs, coords, fixed_coords), where:

- epoch The current epoch number (between 1 and n_epochs).
- n_epochs Number of epochs to use during the optimization of the embedded coordinates.
- coords The embedded coordinates as of the end of the current epoch, as a matrix with dimensions (N, n_components).
- fixed_coords The originally embedded coordinates from the model. These are fixed and do not change. A matrix with dimensions (Nmodel, n_components) where Nmodel is the number of observations in the original data.

ret_extra A vector indicating what extra data to return. May contain any combination of the following strings:

- "fgraph" the high dimensional fuzzy graph (i.e. the fuzzy simplicial set of the merged local views of the input data). The graph is returned as a sparse matrix of class dgCMatrix-class with dimensions NX x Nmodel, where NX is the number of items in the data to transform in X, and NModel is the number of items in the data used to build the UMAP model. A non-zero entry (i, j) gives the membership strength of the edge connecting the vertex representing the ith item in X to the jth item in the data used to build the model. Note that the graph is further sparsified by removing edges with sufficiently low membership strength that they would not be sampled by the probabilistic edge sampling employed for optimization and therefore the number of non-zero elements in the matrix is dependent on n_epochs. If you are only interested in the fuzzy input graph (e.g. for clustering), setting n_epochs = 0 will avoid any further sparsifying.
- "nn" the nearest neighbor graph for X with respect to the observations in the model. The graph will be returned as a list of two items: idx a matrix of indices, with as many rows as there are items in X and as many columns as there are nearest neighbors to be computed (this value is determined by the model). The indices are those of the rows of the data used to build the model, so they're not necessarily of much use unless you have access to that data. The second item, dist is a matrix of the equivalent distances, with the same dimensions as idx.

seed — Integer seed to use to initialize the random number generator state. Combined with n_sgd_threads = 1 or batch = TRUE, this should give consistent output across multiple runs on a given installation. Setting this value is equivalent to calling set.seed, but it may be more convenient in some situations than having to call a separate function. The default is to not set a seed, in which case this function uses the behavior specified by the supplied model: If the model specifies a seed, then the model seed will be used to seed then random number generator, and results will still be consistent (if n_sgd_threads = 1). If you want to force the seed to not be set, even if it is set in model, set seed = FALSE.

Details

Note that some settings are incompatible with the production of a UMAP model via umap: external neighbor data (passed via a list to the argument of the nn_method parameter), and factor columns that were included in the UMAP calculation via the metric parameter. In the latter case, the model produced is based only on the numeric data. A transformation is possible, but factor columns in the new data are ignored.

Value

A matrix of coordinates for X transformed into the space of the model, or if ret_extra is specified, a list containing:

- embedding the matrix of optimized coordinates.
- if ret_extra contains "fgraph", an item of the same name containing the high-dimensional fuzzy graph as a sparse matrix, of type dgCMatrix-class.
- if ret_extra contains "sigma", returns a vector of the smooth knn distance normalization terms for each observation as "sigma" and a vector "rho" containing the largest distance to the locally connected neighbors of each observation.
- if ret_extra contains "localr", an item of the same name containing a vector of the estimated local radii, the sum of "sigma" and "rho".
- if ret_extra contains "nn", an item of the same name containing the nearest neighbors of each item in X (with respect to the items that created the model).

Examples

```
iris_train <- iris[1:100, ]
iris_test <- iris[101:150, ]

# You must set ret_model = TRUE to return extra data needed
iris_train_umap <- umap(iris_train, ret_model = TRUE)
iris_test_umap <- umap_transform(iris_test, iris_train_umap)
```

---

unload_uwot *Unload a Model*

Description

Unloads the UMAP model. This prevents the model being used with umap_transform, but allows the temporary working directory associated with saving or loading the model to be removed.

Usage

```
unload_uwot(model, cleanup = TRUE, verbose = FALSE)
```

Arguments

model	a UMAP model create by umap.
cleanup	if TRUE, attempt to delete the temporary working directory that was used in either the save or load of the model.
verbose	if TRUE, log information to the console.

See Also

save_uwot, load_uwot

Examples

```
iris_train <- iris[c(1:10, 51:60), ]
iris_test <- iris[100:110, ]

# create model
model <- umap(iris_train, ret_model = TRUE, n_epochs = 20)

# save without unloading: this leaves behind a temporary working directory
model_file <- tempfile("iris_umap")
model <- save_uwot(model, file = model_file)

# The model can continue to be used
test_embedding <- umap_transform(iris_test, model)

# To manually unload the model from memory when finished and to clean up
# the working directory (this doesn't touch your model file)
unload_uwot(model)

# At this point, model cannot be used with umap_transform, this would fail:
# test_embedding2 <- umap_transform(iris_test, model)

# restore the model: this also creates a temporary working directory
model2 <- load_uwot(file = model_file)
test_embedding2 <- umap_transform(iris_test, model2)

# Unload and clean up the loaded model temp directory
unload_uwot(model2)

# clean up the model file
unlink(model_file)

# save with unloading: this deletes the temporary working directory but
# doesn't allow the model to be re-used
model3 <- umap(iris_train, ret_model = TRUE, n_epochs = 20)
model_file3 <- tempfile("iris_umap")
model3 <- save_uwot(model3, file = model_file3, unload = TRUE)
```

Index