Package 'uwot'

December 2, 2021

Title The Uniform Manifold Approximation and Projection (UMAP) Method
for Dimensionality Reduction

Version 0.1.11

Description An implementation of the Uniform Manifold Approximation and
Projection dimensionality reduction by McInnes et al. (2018)
<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) <arXiv:1602.00370> is also
provided. This is a complete re-implementation in R (and C++, via the 'Rcpp'
package): no Python installation is required. See the uwot website
(<https://github.com/jlmelville/uwot>) for more documentation and examples.

License GPL (>= 3)

URL https://github.com/jlmelville/uwot

BugReports https://github.com/jlmelville/uwot/issues

Encoding UTF-8

Suggests testthat, covr, bigstatsr

RoxygenNote 7.1.2

Depends Matrix

LinkingTo Rcpp, RcppProgress, RcppAnnoy, dqrng

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

NeedsCompilation yes

Author James Melville [aut, cre],
Aaron Lun [ctb],
Mohamed Nadhir Djekidel [ctb],
Yuhan Hao [ctb]

Maintainer James Melville <jlmelville@gmail.com>

Repository CRAN

Date/Publication 2021-12-02 08:00:02 UTC

R topics documented:

---

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

```
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_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
)
```

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. One of: • "euclidean" (the default) • "cosine"

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

Only applies if nn_method = "annoy" (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"), if more than one connected component is identified, each connected component is initialized separately and the results are merged. 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. Either a PCA-based initialization or increasing the value of n_neighbors may be more appropriate.

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).

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.

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". 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.

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, only applies if nn_method = "annoy". If n_threads > 1, then the Annoy index will be temporarily written to disk in the location determined by tempfile.

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.

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.

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.

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]().

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

References

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

```
# 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
)
```

---

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 man-
                ually, 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)
```

---

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

Description

A faster (but less flexible) version of the UMAP 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_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
)
```

Arguments

X   Input data. Can be a [data.frame](), [matrix](), [dist]() object or [sparseMatrix](). Ma-
                trix 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 pre-
                served. 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. One of:

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

Only applies if `nn_method = "annoy"` (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"), if more than one connected component is identified, each connected component is initialized separately and the results are merged. 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. Either a PCA-based initialization or increasing the value of n_neighbors may be more appropriate.

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).

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.

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 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.

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. The n_neighbors parameter is ignored when using pre-calculated 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, only applies if nn_method = `"annoy"`. If n_threads > 1, then the Annoy index will be temporarily written to disk in the location determined by [tempfile](#).

| | |
|---|---|
| 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.

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.

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.

tmpdir        Temporary directory to store nearest neighbor indexes during nearest neighbor search. Default is [tempdir](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.

### Details

By setting the UMAP curve parameters a and b to 1, you get back the Cauchy distribution as used in t-SNE and LargeVis. 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.

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

### 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 & Healy, 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_sgd_threads = 0,
```

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

Arguments

X   Input data. Can be a [data.frame](), [matrix](), [dist]() object or [sparseMatrix](). Ma-
                     trix 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 pre-
                     served. 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. One of:

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

                     Only applies if nn_method = "annoy" (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 inter-
                     sected 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"`), if more than one connected component is identified, each connected component is initialized separately and the results are merged. 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. Either a PCA-based initialization or increasing the value of `n_neighbors` may be more appropriate.

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).

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.

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 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.

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".

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/>.

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.

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.

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.

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, only applies if `nn_method = "annoy"`. If `n_threads > 1`, then the Annoy index will be temporarily written to disk in the location determined by [tempfile](#).

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.

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](#).

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

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. (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

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")
))
```

---

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
)
```

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.
	The format is a list consisting of two elements:

- "idx". A n_vertices x n_neighbors matrix where n_vertices is the number of items to be transformed. 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.

	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. The X parameter is ignored when using pre-calculated nearest neighbor data.
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 stepsize. 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.

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.

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