

# Package ‘umap’

November 4, 2020

**Title** Uniform Manifold Approximation and Projection

**Version** 0.2.7.0

**Author** Tomasz Konopka [aut, cre]

**Maintainer** Tomasz Konopka <tokonopka@gmail.com>

**Description** Uniform manifold approximation and projection is a technique for dimension reduction. The algorithm was described by McInnes and Healy (2018) in <arXiv:1802.03426>. This package provides an interface for two implementations. One is written from scratch, including components for nearest-neighbor search and for embedding. The second implementation is a wrapper for 'python' package 'umap-learn' (requires separate installation, see vignette for more details).

**Depends** R (>= 3.1.2)

**Imports** methods, openssl, reticulate, Rcpp (>= 0.12.6), RSpectra, stats

**License** MIT + file LICENSE

**URL** <https://github.com/tkonopka/umap>

**BugReports** <https://github.com/tkonopka/umap/issues>

**LinkingTo** Rcpp

**LazyData** true

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-11-04 06:50:02 UTC

## R topics documented:

|               |   |
|---------------|---|
| predict.umap  | 2 |
| umap          | 3 |
| umap.defaults | 4 |
| umap.knn      | 5 |

|              |          |
|--------------|----------|
| <b>Index</b> | <b>7</b> |
|--------------|----------|

---

|              |  |
|--------------|--|
| predict.umap | <i>project data points onto an existing umap embedding</i> |
|--------------|--|

---

### Description

project data points onto an existing umap embedding

### Usage

```
## S3 method for class 'umap'
predict(object, data, ...)
```

### Arguments

|        |                                 |
|--------|---------------------------------|
| object | trained object of class umap    |
| data   | matrix with data                |
| ...    | additional arguments (not used) |

### Value

new matrix

### Examples

```
# embedd iris dataset using default settings
iris.umap = umap(iris[,1:4])

# create a dataset with structure like iris, but with perturbation
iris.perturbed = iris[,1:4] + matrix(rnorm(nrow(iris)*4, 0, 0.1), ncol=4)

# project perturbed dataset
perturbed.embedding = predict(iris.umap, iris.perturbed)

# output is a matrix with embedding coordinates
head(perturbed.embedding)
```

---

`umap`*Computes a manifold approximation and projection*

---

**Description**

Computes a manifold approximation and projection

**Usage**

```
umap(  
  d,  
  config = umap.defaults,  
  method = c("naive", "umap-learn"),  
  preserve.seed = TRUE,  
  ...  
)
```

**Arguments**

|                            |   |
|----------------------------|---|
| <code>d</code>             | matrix, input data  |
| <code>config</code>        | object of class <code>umap.config</code>  |
| <code>method</code>        | character, implementation. Available methods are 'naive' (an implementation written in pure R) and 'umap-learn' (requires python package 'umap-learn')  |
| <code>preserve.seed</code> | logical, leave TRUE to insulate external code from randomness within the umap algorithms; set FALSE to allow randomness used in umap algorithms to alter the external random-number generator |
| <code>...</code>           | list of settings; values overwrite defaults from config; see documentation of <code>umap.default</code> for details about available settings  |

**Value**

object of class `umap`, containing at least a component with an embedding and a component with configuration settings

**Examples**

```
# embedd iris dataset using default settings  
iris.umap = umap(iris[,1:4])  
  
# display object summary  
iris.umap  
  
# display embedding coordinates  
head(iris.umap$layout)
```

---

`umap.defaults`*Default configuration for umap*

---

## Description

A list with parameters customizing a UMAP embedding. Each component of the list is an effective argument for `umap()`.

## Usage

```
umap.defaults
```

## Format

An object of class `umap.config` of length 22.

## Details

`n_neighbors`: integer; number of nearest neighbors

`n_components`: integer; dimension of target (output) space

`metric`: character or function; determines how distances between data points are computed. When using a string, available metrics are: euclidean, manhattan. Other available generalized metrics are: cosine, pearson, pearson2. Note the triangle inequality may not be satisfied by some generalized metrics, hence knn search may not be optimal. When using `metric.function` as a function, the signature must be `function(matrix, origin, target)` and should compute a distance between the origin column and the target columns

`n_epochs`: integer; number of iterations performed during layout optimization

`input`: character, use either "data" or "dist"; determines whether the primary input argument to `umap()` is treated as a data matrix or as a distance matrix

`init`: character or matrix. The default string "spectral" computes an initial embedding using eigenvectors of the connectivity graph matrix. An alternative is the string "random", which creates an initial layout based on random coordinates. This setting can also be set to a matrix, in which case layout optimization begins from the provided coordinates.

`min_dist`: numeric; determines how close points appear in the final layout

`set_op_ratio_mix_ratio`: numeric in range [0,1]; determines who the knn-graph is used to create a fuzzy simplicial graph

`local_connectivity`: numeric; used during construction of fuzzy simplicial set

`bandwidth`: numeric; used during construction of fuzzy simplicial set

`alpha`: numeric; initial value of "learning rate" of layout optimization

`gamma`: numeric; determines, together with alpha, the learning rate of layout optimization

`negative_sample_rate`: integer; determines how many non-neighbor points are used per point and per iteration during layout optimization

a: numeric; contributes to gradient calculations during layout optimization. When left at NA, a suitable value will be estimated automatically.

b: numeric; contributes to gradient calculations during layout optimization.

spread: numeric; used during automatic estimation of a/b parameters.

random\_state: integer; seed for random number generation used during umap()

transform\_state: integer; seed for random number generation used during predict()

knn: object of class umap.knn; precomputed nearest neighbors

knn.repeat: number of times to restart knn search

verbose: logical or integer; determines whether to show progress messages

umap\_learn\_args: vector of arguments to python package umap-learn

## Examples

```
# display all default settings
umap.defaults

# create a new settings object with n_neighbors set to 5
custom.settings = umap.defaults
custom.settings$n_neighbors = 5
custom.settings
```

---

umap.knn

*construct a umap.knn object describing nearest neighbors*

---

## Description

construct a umap.knn object describing nearest neighbors

## Usage

```
umap.knn(indexes, distances)
```

## Arguments

indexes            matrix, integers linking data points to nearest neighbors

distances         matrix, distance values between pairs of points specified in the matrix of indexes

## Value

object of class umap.knn, which is a list with matrices with indexes of nearest neighbors and distances to those neighbors

**Examples**

```
# this example describes a set of three data points (indexes 1,2,3)
# which are equidistant from each other. Hence the distance between
# pairs (i, j) is 0 for i=j and 1 otherwise.
three.indexes = matrix(c(1,2,3,
                        2,1,3,
                        3,1,2), nrow=3, ncol=3)
three.distances = matrix(c(0, 1, 1,
                          0, 1, 1,
                          0, 1, 1), nrow=3, ncol=3)

umap.knn(three.indexes, three.distances)
```

# Index

\* **datasets**

umap.defaults, 4

predict.umap, 2

umap, 3

umap.defaults, 4

umap.knn, 5