Package ‘EmbedSOM’

October 12, 2022

Version 2.1.2
Title Fast Embedding Guided by Self-Organizing Map
Depends R (>= 3.2)
Suggests knitr, rmarkdown
Imports FNN, ggplot2, igraph, Matrix, Rtsne, umap, uwot
Description Provides a smooth mapping of multidimensional points into low-dimensional space defined by a self-organizing map. Designed to work with 'FlowSOM' and flow-cytometry use-cases. See Kratochvil et al. (2019) <doi:10.12688/f1000research.21642.1>.
License GPL (>= 3)
URL https://github.com/exaexa/EmbedSOM
Encoding UTF-8
RoxygenNote 7.1.1
VignetteBuilder knitr
NeedsCompilation yes
Author Mirek Kratochvil [aut, cre],
Sofie Van Gassen [cph],
Britt Callebaut [cph],
Yvan Saeys [cph],
Ron Wehrens [cph]
Maintainer Mirek Kratochvil <exa.exa@gmail.com>
Repository CRAN
Date/Publication 2022-07-05 10:20:02 UTC

R topics documented:

ClusterPalette ................................................................. 2
EmbedSOM ................................................................. 3
ExprColors ................................................................. 4
ExpressionGradient .......................................................... 5
ClusterPalette

An acceptable cluster color palette

Description

An acceptable cluster color palette

Usage

ClusterPalette(n, vcycle = c(1, 0.7), scycle = c(0.7, 1), alpha = 1)

Arguments

n
How many colors to generate

vcycle, scycle
Small vectors with cycles of saturation/value for hsv

alpha
Opacity of the colors

Examples

EmbedSOM::ClusterPalette(10)
EmbedSOM

Process the cells with SOM into a nice embedding

Description

Process the cells with SOM into a nice embedding

Usage

```r
EmbedSOM(
  data = NULL,
  map = NULL,
  fsom = NULL,
  smooth = NULL,
  k = NULL,
  adjust = NULL,
  importance = NULL,
  coordsFn = NULL,
  coords = NULL,
  emcoords = NULL,
  emcoords.pow = 1,
  parallel = F,
  threads = if (parallel) 0 else 1
)
```

Arguments

data: Data matrix with points that optionally overrides the one from `fsom$data`
map: Map object in FlowSOM format, to optionally override `fsom$map`
fsom: FlowSOM object with a built SOM (used if data or map are missing)
smooth: Produce smoother (positive values) or more rough approximation (negative values).
k: How many neighboring landmarks (e.g. SOM nodes) to take into the whole computation
adjust: How much non-local information to remove from the approximation
importance: Scaling of the landmarks, will be used to scale the incoming data (should be same as used for training the SOM or to select the landmarks)
coordsFn: A coordinates-generating function (e.g. `tSNECoords()`) that overrides the existing `map$grid`.
coords: A matrix of embedding-space coordinates that correspond to `map$codes` (i.e. the "embedded landmarks"). Overrides `map$grid` if not NULL.
emcoords: Provided for backwards compatibility, will be removed. Use `coords` and `coordsFn` instead.
emcoords.pow

Provided for backwards compatibility, will be removed. Use a parametrized coordsFn instead.

parallel

Boolean flag whether the computation should be parallelized (this flag is just a nice name for threads and does not do anything directly – default FALSE sets threads=1, TRUE sets threads=0)

threads

Number of threads used for computation, 0 chooses hardware concurrency, 1 (default) turns off parallelization.

Value

matrix with 2D or 3D coordinates of the embedded data, depending on the map

Examples

d <- cbind(rnorm(10000), 3*runif(10000), rexp(10000))
colnames(d) <- paste0("col",1:3)
map <- EmbedSOM::SOM(d, xdim=10, ydim=10)
e <- EmbedSOM::EmbedSOM(data=d, map=map)
EmbedSOM::PlotEmbed(e, data=d, 'col1', pch=16)

ExprColors

Generate colors for multi-color marker expression labeling in a single plot

Arguments

exprs Matrix-like object with marker expressions (extract it manually from your data)
base, scale Base(s) and scale(s) for softmax (convertible to numeric vectors of size 1+ncol(exprs))
cutoff Gray level (expressed in sigmas of the sample distribution)
ExpressionGradient

Description

The ggplot2 scale gradient from ExpressionPalette.

Usage

ExpressionGradient(...)

Arguments

... Arguments passed to `ggplot2::scale_color_gradientn()`

Examples

```r
library(EmbedSOM)
library(ggplot2)

# simulate a simple dataset
e <- cbind(rnorm(10000), rnorm(10000))
data <- data.frame(Val=log(1+e[,1]^2+e[,2]^2))
PlotGG(e, data=data) +
  geom_point(aes_string(color="Val"), alpha=.5) +
  ExpressionGradient(guide=FALSE)
```

pow
Obsolette, now renamed to scale.

col
Colors to use, defaults to colors taken from ‘ClusterPalette’

nocolor
The color to use for sub-gray-level expression, default gray.

alpha
Default alpha value.
ExpressionPalette  

Marker expression palette generator based off ColorBrewer's RdYlBu, only better for plotting of half-transparent cells

Description

Marker expression palette generator based off ColorBrewer’s RdYlBu, only better for plotting of half-transparent cells

Usage

ExpressionPalette(n, alpha = 1)

Arguments

n  
How many colors to generate
alpha  
Opacity of the colors

Examples

EmbedSOM::ExpressionPalette(10)

GQTSOM  

Train a Growing Quadtree Self-Organizing Map

Description

Train a Growing Quadtree Self-Organizing Map

Usage

GQTSOM(
  data,
  init.dim = c(3, 3),
  target_codes = 100,
  rlen = 10,
  radius = c(sqrt(sum(init.dim^2)), 0.5),
  epochRadii = seq(radius[1], radius[2], length.out = rlen),
  coords = NULL,
  codes = NULL,
  coordsFn = NULL,
  importance = NULL,
  distf = 2,
  nhbr.distf = 2,
  noMapping = F,
  parallel = F,
  threads = if (parallel) 0 else 1
)
GraphCoords

**Arguments**

- **data**: Input data matrix
- **init.dim**: Initial size of the SOM, default c(3,3)
- **target_codes**: Make the SOM grow linearly to at most this amount of nodes (default 100)
- **rlen**: Number of training iterations
- **radius**: Start and end training radius, as in `SOM()`
- **epochRadii**: Precise radii for each epoch (must be of length `rlen`)
- **coords**: Quadtree coordinates of the initial SOM nodes.
- **codes**: Initial codebook
- **coordsFn**: Function to generate/transform grid coordinates (e.g. `tSNECoords()`). If `NULL` (default), the grid is the grid is the 2D coordinates of GQTSOM map.
- **importance**: Weights of input data dimensions
- **distf**: Distance measure to use in input data space (1=manhattan, 2=euclidean, 3=chebyshev, 4=cosine)
- **nhbr.distf**: Distance measure to use in output space (as in `distf`)
- **noMapping**: If `TRUE`, do not compute the assignment of input data to SOM nodes
- **parallel**: Paralllelize the training by setting appropriate `threads`. Defaults to `FALSE`.
- **threads**: Number of threads to use for training. Defaults to 0 (chooses maximum available hardware threads) if `parallel=TRUE` or 1 (single thread) if `parallel=FALSE`.

**Description**

This uses a complete graph on the map codebook, which brings overcrowding problems. It is therefore useful to transform the distances for avoiding that (e.g. by exponentiating them slightly).

**Usage**

```r
GraphCoords(
  dim = NULL,
  dist.method = NULL,
  distFn = function(x) x,
  layoutFn = igraph::layout_with_kk
)
```

**Arguments**

- **dim**: Dimension of the result (passed to `layoutFn`)
- **dist.method**: The method to compute distances, passed to `stats::dist()` as parameter `method`
- **distFn**: Custom transformation function of the distance matrix
- **layoutFn**: iGraph-compatible graph layouting function (default `igraph::layout_with_kk`)

Add Kamada-Kawai-generated embedding coordinates to the map
kMeansMap

Value
a function that transforms the map, usable as coordsFn parameter

---

Initialize_PCA Create a grid from first 2 PCA components

Description
Create a grid from first 2 PCA components

Usage
Initialize_PCA(data, xdim, ydim, zdim = NULL)

Arguments
data matrix in which each row represents a point
xdim, ydim, zdim Dimensions of the SOM grid

Value
array containing the selected selected rows

---

kMeansMap Create a map from k-Means clusters

Description
May give better results than 'RandomMap' on data where random sampling is complicated. This
does not use actual kMeans clustering, but re-uses the batch version of SOM() with tiny radius
(which makes it work the same as kMeans). In consequence, the speedup of SOM function is
applied here as well. Additionally, because we don’t need that amount of clustering precision,
patterns ‘batch=F, rlen=1’ may give a satisfactory result very quickly.

Usage
kMeansMap(data, k, coordsFn, batch = T, ...)

Arguments
data Input data matrix, with individual data points in rows
k How many points to sample
coordsFn a function to generate embedding coordinates (default none)
batch Use batch-SOM training (effectively kMeans, default TRUE)
... Passed to SOM(), useful e.g. for ‘parallel=T’ or ‘rlen=5’
kNNCoords

Value

map object (without the grid, if coordsFn was not specified)

Examples

d <- iris[,1:4]
EmbedSOM::PlotEmbed(
  EmbedSOM::EmbedSOM(
    data = d,
    map = EmbedSOM::kMeansMap(d, 10, EmbedSOM::GraphCoords()),
    pch=19, clust=iris[,5]
  )
)

kNNCoords

Add KNN-topology-based embedding coordinates to the map

Description

Internally, this uses FNN::get.knn() to compute the k-neighborhoods. That function only supports Euclidean metric, therefore kNNCoords throws a warning whenever a different metric is used.

Usage

kNNCoords(
  k = 4,
  dim = NULL,
  distFn = function(x) x,
  layoutFn = igraph::layout_with_kk
)

Arguments

k Size of the neighborhoods (default 4)
dim Dimension of the result (passed to layoutFn)
distFn Custom transformation function of the distance matrix
layoutFn iGraph-compatible graph layouting function (default igraph::layout_with_kk)

Value

a function that transforms the map, usable as coordsFn parameter
MapDataToCodes

Assign nearest node to each datapoint

Usage

MapDataToCodes(
codes,
data,
distf = 2,
parallel = F,
threads = if (parallel) 0 else 1
)

Arguments

codes matrix with nodes of the SOM
data datapoints to assign
distf Distance function (1=manhattan, 2=euclidean, 3=chebyshev, 4=cosine)
threads, parallel Use parallel computation (see SOM())

Value

array with nearest node id for each datapoint

MSTCoords

Add MST-style embedding coordinates to the map

Description

Add MST-style embedding coordinates to the map

Usage

MSTCoords(
dim = NULL,
dist.method = NULL,
distFn = function(x) x,
layoutFn = igraph::layout_with_kk
)
NormalizeColor

Arguments

dim          Dimension of the result (passed to layoutFn)
dist.method  The method to compute distances, passed to \texttt{stats::dist()} as parameter \texttt{method}
distFn       Custom transformation function of the distance matrix
layoutFn     iGraph-compatible graph layouting function (default \texttt{igraph::layout_with_kk()})

Value

a function that transforms the map, usable as \texttt{coordsFn} parameter

Description

Helper for computing colors for embedding plots

Usage

\begin{verbatim}
NormalizeColor(data, low = NULL, high = NULL, pow = 0, sds = 1)
\end{verbatim}

Arguments

data          Vector of scalar values to normalize between 0 and 1
low, high     Originally quantiles for clamping the color. Only kept for backwards compatibility, now ignored.
pow           The scaled data are transformed to data^{2^{pow}}. If set to 0, nothing happens. Positive values highlight differences in the data closer to 1, negative values highlight differences closer to 0.
sds           Inverse scale factor for measured standard deviation (greater value makes data look more extreme)

Examples

\begin{verbatim}
EmbedSOM::NormalizeColor(c(1,100,500))
\end{verbatim}
PlotData

Export a data frame for plotting with marker intensities and density.

Description

Export a data frame for plotting with marker intensities and density.

Usage

PlotData(
  embed, 
  fsom, 
  data = fsom$data, 
  cols, 
  names, 
  normalize = cols, 
  pow = 0, 
  sds = 1, 
  vf = PlotId, 
  density = "Density", 
  densBins = 256, 
  densLimit = NULL, 
  fdens = sqrt
)

Arguments

  embed, fsom, data, cols
    The embedding data, columns to select
  names
    Column names for output
  normalize
    List of columns to normalize using NormalizeColor(), default all
  pow, sds
    Parameters for the normalization
  vf
    Custom value-transforming function
  density
    Name of the density column
  densBins
    Number of bins for density calculation
  densLimit
    Upper limit of density (prevents outliers)
  fdens
    Density-transforming function; default sqrt
PlotDefault

**Description**
Default plot

**Usage**

```r
PlotDefault(pch = ".", cex = 1, ...)
```

**Arguments**

- `pch`, `cex`, `...` correctly defaulted and passed to `plot`

PlotEmbed

**Description**
Helper function for plotting the embedding

**Usage**

```r
PlotEmbed(
  embed,
  value = 0,
  red = 0,
  green = 0,
  blue = 0,
  fr = PlotId,
  fg = PlotId,
  fb = PlotId,
  fv = PlotId,
  powr = 0,
  powg = 0,
  powb = 0,
  powv = 0,
  sdsr = 1,
  sdsg = 1,
  sdsb = 1,
  sdsv = 1,
  clust = NULL,
  nbin = 256,
)```
maxDens = NULL,
fdens = sqrt,
limit = NULL,
alpha = NULL,
fsom,
data,
col,
cluster.colors = ClusterPalette,
expression.colors = ExpressionPalette,
na.color = grDevices::rgb(0.75, 0.75, 0.75, if (is.null(alpha)) 0.5 else alpha/2),
plotf = PlotDefault,
...
)

Arguments

embed The embedding from EmbedSOM(), or generally any 2-column matrix of coordinates
value The column of data to use for coloring the plotted points
red, green, blue The same, for individual RGB components
fv, fr, fg, fb Functions to transform the values before they are normalized
powv, powr, powg, powb Passed to corresponding NormalizeColor() calls as pow
sdsv, sdsr, sdsg, sdsb Passed to NormalizeColor() as sds
clust Cluster labels (used as a factor)
nbin, maxDens, fdens Parameters of density calculation, see PlotData()
limit Low/high offset for NormalizeColor() (obsolete & ignored, will be removed)
alpha Default alpha value of points
fsom FlowSOM object
data Data matrix, taken from fsom parameter by default
col Overrides the computed point colors with exact supplied colors.
cluster.colors Function to generate cluster colors, default ClusterPalette()
expression.colors Function to generate expression color scale, default ExpressionPalette()
na.color Color to assign to NA values
plotf Plot function, defaults to graphics::plot() slightly decorated with pch='.', cex=1
...
Extra params passed to the plot function

Examples

EmbedSOM::PlotEmbed(cbind(rnorm(1e5),rnorm(1e5)))
PlotGG

Wrap PlotData result in ggplot object.

Description

This creates a ggplot2 object for plotting.

Usage

PlotGG(embed, ...)

Arguments

embed Embedding data
...
Extra arguments passed to PlotData()

Examples

library(EmbedSOM)
library(ggplot2)

# simulate a simple dataset
e <- cbind(rnorm(10000), rnorm(10000))

PlotGG(e, data=data.frame(Expr=runif(10000))) + geom_point(aes_string(color="Expr"))

PlotId

Identity on whatever

Description

Identity on whatever

Usage

PlotId(x)

Arguments

x Just the x.

Value

The x.
RandomMap

Create a map by randomly selecting points

Description

Create a map by randomly selecting points

Usage

RandomMap(data, k, coordsFn)

Arguments

data Input data matrix, with individual data points in rows

k How many points to sample

coordsFn a function to generate embedding coordinates (default none)

Value

map object (without the grid, if coordsFn was not specified)

Examples

d <- iris[,1:4]
EmbedSOM::PlotEmbed(
  EmbedSOM::EmbedSOM(
    data = d,
    map = EmbedSOM::RandomMap(d, 30, EmbedSOM::GraphCoords()),
    pch=19, clust=iris[,5]
  )
)

SOM

Build a self-organizing map

Description

Build a self-organizing map
SOM

Usage

SOM(
  data,
  xdim = 10,
  ydim = 10,
  zdim = NULL,
  batch = F,
  rlen = 10,
  alphaA = c(0.05, 0.01),
  radiusA = stats::quantile(nhbrdist, 0.67) * c(1, 0),
  alphaB = alphaA * c(-negAlpha, -0.1 * negAlpha),
  radiusB = negRadius * radiusA,
  negRadius = 1.33,
  negAlpha = 0.1,
  epochRadii = seq(radiusA[1], radiusA[2], length.out = rlen),
  init = FALSE,
  initf = Initialize_PCA,
  distf = 2,
  codes = NULL,
  importance = NULL,
  coordsFn = NULL,
  nhbr.method = "maximum",
  noMapping = F,
  parallel = F,
  threads = if (parallel) 0 else 1
)

Arguments

data Matrix containing the training data
xdim Width of the grid
ydim Height of the grid
zdim Depth of the grid, causes the grid to be 3D if set
batch Use batch training (default FALSE chooses online training, which is more like FlowSOM)
rlen Number of training epochs; or number of times to loop over the training data in online training
alphaA Start and end learning rate for online learning (only for online training)
radiusA Start and end radius
alphaB Start and end learning rate for the second radius (only for online training)
radiusB Start and end radius (only for online training; make sure it is larger than radiusA)
negRadius easy way to set radiusB as a multiple of default radius (use lower value for higher dimensions)
negAlpha the same for alphaB
epochRadii Vector of length rlen with precise epoch radii (only for batch training)
tSNECoords

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>Initialize cluster centers in a non-random way</td>
</tr>
<tr>
<td>initf</td>
<td>Use the given initialization function if init==T (default: Initialize_PCA)</td>
</tr>
<tr>
<td>distf</td>
<td>Distance function (1=manhattan, 2=euclidean, 3=chebyshev, 4=cosine)</td>
</tr>
<tr>
<td>codes</td>
<td>Cluster centers to start with</td>
</tr>
<tr>
<td>importance</td>
<td>array with numeric values. Columns of data will be scaled according to importance.</td>
</tr>
<tr>
<td>coordsFn</td>
<td>Function to generate/transform grid coordinates (e.g. tSNECoords()). If NULL (default), the grid is the canonical SOM grid.</td>
</tr>
<tr>
<td>nhbr.method</td>
<td>Way of computing grid distances, passed as method= to stats::dist() function. Defaults to maximum (square neighborhoods); use euclidean for round neighborhoods.</td>
</tr>
<tr>
<td>noMapping</td>
<td>If TRUE, do not compute the mapping (default FALSE). Makes the process quicker by 1 rlen.</td>
</tr>
<tr>
<td>parallel</td>
<td>Parallelize the batch training by setting appropriate threads. Defaults to FALSE. Always use batch=TRUE for fully parallelized version, online training is not parallelizable. Passed to MapDataToCodes().</td>
</tr>
<tr>
<td>threads</td>
<td>Number of threads of the batch training (has no effect on online training). Defaults to 0 (chooses maximum available hardware threads) if parallel==TRUE or 1 (single thread) if parallel==FALSE. Passed to MapDataToCodes().</td>
</tr>
</tbody>
</table>

Value

A map useful for embedding (EmbedSOM() function) or further analysis, e.g. clustering.

See Also

FlowSOM::SOM

tSNECoords

Add tSNE-based coordinates to a map

Description

Add tSNE-based coordinates to a map

Usage

tSNECoords(dim = NULL, tSNEFn = Rtsne::Rtsne, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim</td>
<td>Dimension of the result (passed to tSNEFn as dims)</td>
</tr>
<tr>
<td>tSNEFn</td>
<td>tSNE function to run (default Rtsne::Rtsne)</td>
</tr>
<tr>
<td>...</td>
<td>passed to tSNEFn</td>
</tr>
</tbody>
</table>
**UMAPCoords**

Add UMAP-based coordinates to a map

**Usage**

```r
UMAPCoords(dim = NULL, UMAPFn = NULL)
```

**Arguments**

- `dim` : Dimension of the result (passed to `UMAPFn` as `n_components`)
- `UMAPFn` : UMAP function to run (default `umap::umap` configured by `umap::umap.defaults`)

**Value**

a function that transforms the map, usable as `coordsFn` parameter

---

**UMatrixCoords**

Add U-Matrix-optimized embedding coordinates to the map

**Description**

The map must already contain a SOM grid with corresponding `xdim`, `ydim` (possibly `zdim`)

**Usage**

```r
UMatrixCoords(
  dim = NULL, 
  dist.method = NULL, 
  distFn = function(x) x, 
  layoutFn = igraph::layout_with_kk 
)
```

**Arguments**

- `dim` : Dimension of the result (passed to `layoutFn`)
- `dist.method` : The method to compute distances, passed to `stats::dist()` as parameter `method`
- `distFn` : Custom transformation function of the distance matrix
- `layoutFn` : iGraph-compatible graph layouting function (default `igraph::layout_with_kk`)

**Value**

a function that transforms the map, usable as `coordsFn` parameter
uwotCoords

Description
Add UMAP-based coordinates to a map, using the 'uwot' package

Usage
uwotCoords(dim = NULL, uwotFn = uwot::umap, ...)

Arguments
- `dim`: Dimension of the result (passed to `uwotFn` as `dims`)
- `uwotFn`: UMAP function to run (default `uwot::umap`)
- `...`: passed to `uwotFn`

Value
a function that transforms the map, usable as `coordsFn` parameter
Index

ClusterPalette, 2
ClusterPalette(). 14

EmbedSOM, 3
EmbedSOM(). 13, 14, 18
ExprColors, 4
ExpressionGradient, 5
ExpressionPalette, 6
ExpressionPalette(). 14

FNN::get.knn(), 9
ggplot2::scale_color_gradientn(), 5
GQTSOM, 6
GraphCoords, 7
graphics::plot(). 14

igraph::layout_with_kk, 7, 9, 19
igraph::layout_with_kk(). 11
Initialize_PCA, 8

kMeansMap, 8
kNNCoords, 9

MapDataToCodes, 10
MapDataToCodes(). 18
MSTCoords, 10

NormalizeColor, 11
NormalizeColor(). 12, 14

PlotData, 12
PlotData(). 14, 15
PlotDefault, 13
PlotEmbed, 13
PlotGG, 15
PlotId, 15

RandomMap, 16
Rtsne::Rtsne, 18

SOM, 16
SOM(). 7, 8, 10
stats::dist(), 7, 11, 18, 19
tSNECoords, 18
tSNECoords(). 3, 7, 18
umap::umap, 19
umap::umap.defaults, 19
UMAPCoords, 19
UMatrixCoords, 19
uwot::umap, 20
uwotCoords, 20