Package 'aweSOM'

May 7, 2026

Title Interactive Self-Organizing Maps

Version 1.3

Date 2022-08-30

Description
Self-organizing maps (also known as SOM, see Kohonen (2001) <doi:10.1007/978-3-642-56927-2>) are a method for dimensionality reduction and clustering of continuous data. This package introduces interactive (html) graphics for easier analysis of SOM results. It also features an interactive interface, for push-button training and visualization of SOM on numeric, categorical or mixed data, as well as tools to evaluate the quality of SOM.

License GPL (>= 2)

Depends R (>= 3.1.0)

Imports kohonen (>= 2.0), shiny (>= 1.6), htmlwidgets, bslib, rmarkdown, htmltools, rclipboard, ggplot2, RColorBrewer, viridis, data.table, DT, kernlab, fields, stats, cluster, e1071, haven, foreign, readxl, readODS

Suggests knitr

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Author Julien Boelaert [aut, cre], Etienne Ollion [aut], Jan Sodoge [aut], Mohamed Megdoud [ctb], Otmane Naji [ctb], Arnaud Lemba Kote [ctb], Theo Renoud [ctb], Samuel Hym [ctb]

Maintainer Julien Boelaert <julien.boelaert@univ-lille.fr>

Repository CRAN

Date/Publication 2022-08-30 15:40:02 UTC

Contents

`aweSOM-package` *Interactive Self-Organizing Maps*

Description

Interactive plots and interface for Kohonen self-organizing maps.

References

Kohonen T. (2001) *Self-Organizing Maps*, 3rd edition, Springer Press, Berlin. <doi:10.1007/978-3-642-56927-2>

See Also

`aweSOM`, `aweSOMplot`, `somInit`, `somQuality`.

`aweSOM` *aweSOM interface*

Description

Launches the (offline) web-based interface for training and visualizing self-organizing maps.

Usage

```
aweSOM()
```

Details

If the interface does not open automatically, open the printed link in a web browser.

To open large files within the interface, use `options(shiny.maxRequestSize=2^30)` (or a suitably large file size) before lauching the interface.

Value

No return value, used for side effects.

References

Kohonen T. (2001) *Self-Organizing Maps*, 3rd edition, Springer Press, Berlin. <doi:10.1007/978-3-642-56927-2>

---

aweSOMdendrogram *Dendogram of hierarchical clustering of SOM cells*

Description

Plots the dendogram of a hierarchical clustering of the SOM prototypes.

Usage

```
aweSOMdendrogram(clust, nclass)
```

Arguments

`clust`	an object of class `hclust`, the result of a hierarchical clustering performed by `stats::hclust`.
`nclass`	an integer, number of superclasses

Value

Returns `NULL` if `nclass` is 1, or else a `list` containing the indices of the SOM cells in each superclass.

Examples

```
## Build training data
dat <- iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]
### Scale training data
dat <- scale(dat)
## Train SOM
### Initialization (PCA grid)
init <- somInit(dat, 4, 4)
ok.som <- kohonen::som(dat, grid = kohonen::somgrid(4, 4, 'hexagonal'),
                       rlen = 100, alpha = c(0.05, 0.01),
                       radius = c(2.65,-2.65),
```

```
                      init = init, dist.fcts = 'sumofsquares')
## Group cells into superclasses (hierarchical clustering)
superclust <- hclust(dist(ok.som$codes[[1]]), 'complete')
## Plot superclasses dendrogram
aweSOMdendrogram(superclust, 2)
```

---

aweSOMplot *Interactive SOM plots*

Description

Plot interactive visualizations of self-organizing maps (SOM), as an html page. The plot can represent general map informations, or selected categorical or numeric variables (not necessarily the ones used during training). Hover over the map to focus on the selected cell or variable, and display further information.

Usage

```
aweSOMplot(
  som,
  type = c("Hitmap", "Cloud", "UMatrix", "Circular", "Barplot", "Boxplot", "Radar",
    "Line", "Color", "Pie", "CatBarplot"),
  data = NULL,
  variables = NULL,
  superclass = NULL,
  obsNames = NULL,
  scales = c("contrast", "range", "same"),
  values = c("mean", "median", "prototypes"),
  size = 400,
  palsc = c("Set3", "viridis", "grey", "rainbow", "heat", "terrain", "topo", "cm",
    rownames(RColorBrewer::brewer.pal.info)),
  palvar = c("viridis", "grey", "rainbow", "heat", "terrain", "topo", "cm",
    rownames(RColorBrewer::brewer.pal.info)),
  palrev = FALSE,
  showAxes = TRUE,
  transparency = TRUE,
  boxOutliers = TRUE,
  showSC = TRUE,
  pieEqualSize = FALSE,
  showNames = TRUE,
  legendPos = c("beside", "below", "none"),
  legendFontsize = 14,
  cloudType = c("cellPCA", "kPCA", "PCA", "proximity", "random"),
  cloudSeed = NA,
  elementId = NULL
)
```

Arguments

som	kohonen object, a SOM created by the kohonen::som function.
type	character, the plot type. The default "Hitmap" is a population map. "Cloud" plots the observations as a scatterplot within each cell (see Details). "UMatrix" plots the average distance of each cell to its neighbors, on a color scale. "Circular" (barplot), "Barplot", "Boxplot", "Radar" and "Line" are for numeric variables. "Color" (heat map) is for a single numeric variable. "Pie" (pie chart) and "CatBarplot" are for a single categorical (factor) variable.
data	data.frame containing the variables to plot. This is typically not the training data, but rather the unscaled original data, as it is easier to read the results in the original units, and this allows to plot extra variables not used in training. If not provided, the training data is used.
variables	character vector containing the names of the variable(s) to plot. See Details.
superclass	integer vector, the superclass of each cell of the SOM.
obsNames	character vector, names of the observations to be displayed when hovering over the cells of the SOM. Must have a length equal to the number of data rows. If not provided, the row names of data will be used.
scales	character, controls the scaling of the variables on the plot. See Details.
values	character, the type of value to be displayed. The default "mean" uses the observation means (from data) for each cell. Alternatively, "median" uses the observation medians for each cell, and "prototypes" uses the SOM's prototypes values.
size	numeric, plot size, in pixels. Default 400.
palsc	character, the color palette used to represent the superclasses as background of the cells. Default is "Set3". Can be "viridis", "grey", "rainbow", "heat", "terrain", "topo", "cm", or any palette name of the RColorBrewer package.
palvar	character, the color palette used to represent the variables. Default is "viridis", available choices are the same as for palsc.
palrev	logical, whether color palette for variables is reversed. Default is FALSE.
showAxes	logical, whether to display the axes (for "Circular", "Barplot", "Boxplot", "Star", "Line", "CatBarplot"), default TRUE.
transparency	logical, whether to use transparency when focusing on a variable, default TRUE.
boxOutliers	logical, whether outliers in "Boxplot" are displayed, default TRUE.
showSC	logical, whether to display superclasses as labels in the "Color" and "UMatrix" plots, default TRUE.
pieEqualSize	logical, whether "Pie" should display pies of equal size. The default FALSE displays pies with areas proportional to the number of observations in the cells.
showNames	logical, whether to display the observations names in a box below the plot.
legendPos	character, whether and where to display the legend (if applicable). Possible values are "beside", "below" or "none".
legendFontsize	numeric, font size to use for the legend, and for the tooltip information of the "Cloud" plot. Default is 14.

`cloudType`	character, for "Cloud" type, controls how the point coordinates are computed, see Details.
`cloudSeed`	numeric, for "random Cloud" type, seed for the pseudo-random placement of the points. If NA (the default), no seed will be set.
`elementId`	character, user-defined elementId of the widget. Can be useful for user extensions when embedding the result in an html page.

Details

The selected `variables` must be numeric for types "Circular", "Barplot", "Boxplot", "Radar", "Color" and "Line", or factor for types "Pie" and "CatBarplot". If not provided, all columns of data will be selected. If a numeric variable is provided to a "Cloud", "Pie" or "CatBarplot", it will be split into a maximum of 8 classes. For "Cloud" plots, the first element of `variables` is used to color the points (and can be "None" for no coloring), the following elements (if any) are used in the information box of each point.

Variables scales: All values that are used for the plots (means, medians, prototypes) are scaled to 0-1 for display (minimum height to maximum height). The `scales` parameter controls how this scaling is done.

- "contrast": for each variable, the minimum height is the minimum observed mean/median/prototype on the map, the maximum height is the maximum on the map. This ensures maximal contrast on the plot.
- "range": observation range; for each variable, the minimum height corresponds to the minimum of that variable over the whole dataset, the maximum height to the maximum of the variable on the whole dataset.
- "same": same scales; all heights are displayed on the same scale, using the global minimum and maximum of the dataset.

Cloud plot: three types of cloud plots are available, controlled by the `cloudType` argument:

- "cellPCA": (default) the point coordinates are computed cell by cell, by computing a PCA on the training data of that cell only. Points close to the center of the cell are close to the mean of its observations. Points far apart within a cell are likely to have different characteristics.
- "kPCA": the point coordinates are computed globally, by a kernel PCA performed on all the differences between the training data and their winning prototypes. Points close to the center of their cell are close to their prototype, and points with similar placements in the clouds thus have a similar difference to their prototype. Not recommended for large datasets (eg. > 1000 observations), as it tends to take too much memory.
- "PCA": the point coordinates are computed globally, by a PCA performed on all the differences between the training data and their winning prototypes. Points close to the center of their cell are close to their prototype, and points with similar placements in the clouds thus have a similar difference to their prototype.
- "proximity": the point coordinates are computed one by one, based on the distances of the observation's training data to its cell's prototype and to its second best matching prototypes among its cell's neighbors. Points close to their cell's center are close to their closest prototype, while points close to another cell are close to that cell's prototype.
- "random": the point coordinates are random samples from a uniform distribution.

Value

Returns an object of class `htmlwidget`.

Examples

```
## Build training data
dat <- iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]
### Scale training data
dat <- scale(dat)
## Train SOM
### Initialization (PCA grid)
init <- somInit(dat, 4, 4)
ok.som <- kohonen::som(dat, grid = kohonen::somgrid(4, 4, 'hexagonal'),
                       rlen = 100, alpha = c(0.05, 0.01),
                       radius = c(2.65,-2.65), init = init,
                       dist.fcts = 'sumofsquares')
## Group cells into superclasses (PAM clustering)
superclust <- cluster::pam(ok.som$codes[[1]], 2)
superclasses <- superclust$clustering

## Observations cloud ('Cloud')
variables <- c("Sepal.Length", "Sepal.Width",  "Petal.Length", "Petal.Width")
aweSOMplot(som = ok.som, type = 'Cloud', data = iris,
           variables = c("Species", variables), superclass = superclasses)

## Not run:
## Population map ('Hitmap')
aweSOMplot(som = ok.som, type = 'Hitmap', superclass = superclasses)

## Plots for numerical variables
## Circular barplot
aweSOMplot(som = ok.som, type = 'Circular', data = iris,
           variables= variables, superclass = superclasses)
## Barplot (numeric variables)
aweSOMplot(som = ok.som, type = 'Barplot', data = iris,
           variables= variables, superclass = superclasses)

## Plots for categorial variables (iris species, not used for training)
## Pie
aweSOMplot(som = ok.som, type = 'Pie', data = iris,
           variables= "Species", superclass = superclasses)
## Barplot (categorical variables)
aweSOMplot(som = ok.som, type = 'CatBarplot', data = iris,
           variables= "Species", superclass = superclasses)

## End(Not run)
```

`aweSOMreorder` *Reorder variables for SOM plot*

Description

Reorders a set of variables for prettier display on SOM plots. Variables that have similar variations along the cell plots while be ordered close together. Reordering is computed from the first component of a kernel PCA performed on the matrix of displayed values (with the variables as rows, and the cells as columns).

Usage

```
aweSOMreorder(
  som,
  data = NULL,
  variables = NULL,
  scales = c("contrast", "range", "same"),
  values = c("mean", "median", "prototypes")
)
```

Arguments

`som`	kohonen object, a SOM created by the `kohonen::som` function.
`data`	`data.frame` containing the variables to plot. This is typically not the training data, but rather the unscaled original data, as it is easier to read the results in the original units, and this allows to plot extra variables not used in training. If not provided, the training data is used.
`variables`	character vector containing the names of the variables to plot. If not provided, all columns of data will be selected. All variables must be numeric.
`scales`	character, controls the scaling of the variables on the plot. The default "constrast" maximizes the displayed contrast by scaling the displayed heights of each variable from minimum to maximum of the displayed value. Alternatively, "range" uses the minimum and maximum of the observations for each variable, and "same" displays all variables on the same scale, using the global minimum and maximum of the data.
`values`	character, the type of value to be displayed. The default "mean" uses the observation means (from data) for each cell. Alternatively, "median" uses the observation medians for each cell, and "prototypes" uses the SOM's prototypes values.

Value

Returns a character vector containing the reordered variables names.

Examples

```
## Build training data
dat <- iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]
### Scale training data
dat <- scale(dat)
## Train SOM
### Initialization (PCA grid)
```

```
init <- somInit(dat, 4, 4)
ok.som <- kohonen::som(dat, grid = kohonen::somgrid(4, 4, 'hexagonal'),
                       rlen = 100, alpha = c(0.05, 0.01),
                       radius = c(2.65,-2.65), init = init,
                       dist.fcts = 'sumofsquares')
## Reorder variables
ordered.vars <- aweSOMreorder(ok.som)
## Not run:
## Plot with reordered variables
aweSOMplot(som = ok.som, type = 'Circular', data = iris,
           variables= ordered.vars)

## End(Not run)
```

`aweSOMscreeplot` *Screeplot of SOM superclasses*

Description

The screeplot, helps deciding the optimal number of superclasses. Available for both PAM and hierarchical clustering.

Usage

```
aweSOMscreeplot(
  som,
  nclass = 2,
  method = c("hierarchical", "pam"),
 hmethod = c("complete", "ward.D2", "ward.D", "single", "average", "mcquitty", "median",
    "centroid")
)
```

Arguments

`som`	kohonen object, a SOM created by the `kohonen::som` function.
`nclass`	number of superclasses to be visualized in the screeplot. Default is 2.
`method`	Method used for clustering. Hierarchical clustering ("hierarchical") and Partitioning around medoids ("pam") can be used. Default is hierarchical clustering.
`hmethod`	For hierarchicical clustering, the clustering method, by default "complete". See the `stats::hclust` documentation for more details.

Value

No return value, called for side effects.

Examples

```
## Build training data
dat <- iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]
### Scale training data
dat <- scale(dat)
## Train SOM
### Initialization (PCA grid)
init <- somInit(dat, 4, 4)
ok.som <- kohonen::som(dat, grid = kohonen::somgrid(4, 4, 'hexagonal'),
                       rlen = 100, alpha = c(0.05, 0.01),
                       radius = c(2.65,-2.65),
                       init = init, dist.fcts = 'sumofsquares')
## Group cells into superclasses (PAM clustering)
superclust <- cluster::pam(ok.som$codes[[1]], 2)
superclasses <- superclust$clustering
aweSOMscreeplot(ok.som, method = 'hierarchical',
                hmethod = 'complete', nclass = 2)
```

`aweSOMsilhouette` *Silhouette plot of SOM superclasses*

Description

Plots a silhouette plot, used to assess the quality of the super-clustering of SOM prototypes into superclasses. Available for both PAM and hierarchical clustering.

Usage

```
aweSOMsilhouette(som, clust)
```

Arguments

`som`	kohonen object, a SOM created by the `kohonen::som` function.
`clust`	object containing the result of the super-clustering of the SOM prototypes (either a `hclust` or a `pam` object).

Value

No return value, called for side effects.

Examples

```
## Build training data
dat <- iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]
### Scale training data
dat <- scale(dat)
## Train SOM
### Initialization (PCA grid)
init <- somInit(dat, 4, 4)
```

```
ok.som <- kohonen::som(dat, grid = kohonen::somgrid(4, 4, 'hexagonal'),
                       rlen = 100, alpha = c(0.05, 0.01),
                       radius = c(2.65,-2.65), init = init,
                       dist.fcts = 'sumofsquares')
## Group cells into superclasses (PAM clustering)
superclust <- cluster::pam(ok.som$codes[[1]], 2)
superclasses <- superclust$clustering
aweSOMsilhouette(ok.som, superclasses)
```

---

aweSOMsmoothdist *Smooth Distance Plot for SOM*

Description

Plots a visualization of the distances between the SOM cells. Based on the U-Matrix, which is computed for each cell as the mean distance to its immediate neighbors.

Usage

```
aweSOMsmoothdist(
  som,
  pal = c("viridis", "grey", "rainbow", "heat", "terrain", "topo", "cm",
    rownames(RColorBrewer::brewer.pal.info)),
  reversePal = FALSE,
  legendFontsize = 14
)
```

Arguments

som	kohonen object, a SOM created by the kohonen::som function.
pal	character, the color palette. Default is "viridis". Can be "viridis", "grey", "rainbow", "heat", "terrain", "topo", "cm", or any palette name of the RColorBrewer package.
reversePal	logical, whether color palette should be reversed. Default is FALSE.
legendFontsize	numeric, the font size for the legend. Default 14.

Value

Returns an object of classes gg and ggplot.

Examples

```
## Build training data
dat <- iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]
### Scale training data
dat <- scale(dat)
## Train SOM
### Initialization (PCA grid)
```

```
init <- somInit(dat, 4, 4)
ok.som <- kohonen::som(dat, grid = kohonen::somgrid(4, 4, 'rectangular'),
                       init = init)
aweSOMsmoothdist(ok.som)
```

---

cdt *Complete disjunctive table*

Description

Computes the complete disjunctive table of a set of factors, where each factor (ie categorical variable) is encoded as a set of dummy variables, one for each level (category).

Usage

```
cdt(x)
```

Arguments

x	data.frame on which the table is computed. All columns will be treated as factors.

Value

A matrix of dummy variables, with nrow(x) rows and a number of columns equal to the sum of numbers of levels in all the variables of x.

---

somDist *Distance measures on a SOM*

Description

Several distance measures between cells or prototypes of a trained SOM (in grid space, in data space).

Usage

```
somDist(som)
```

Arguments

som	kohonen object, a SOM created by the som function.

Value

A list with distance measures: between cells on the grid, between prototypes in data space, and the neighborhood matrix on the grid.

somInit *Initialize SOM prototypes*

Description

Prototypes are the artificial points in data space that are used to cluster observations: each observation is assigned to the cluster of its closest prototype. In self-organizing maps, each cell of the map has its own prototype, and training is performed by iteratively adjusting the prototypes. This function creates an initial guess for the prototypes of a SOM grid, to be used as the `init` argument to the `kohonen::som` function (see example).

Usage

```
somInit(traindat, nrows, ncols, method = c("pca.sample", "pca", "random"))
```

Arguments

`traindat`	Matrix of training data, that will also be used to train the SOM.
`nrows`	Number of rows on the map.
`ncols`	Number of columns on the map.
`method`	Method used, see Details. "pca" or "random"

Details

The default method "pca.sample" takes as prototypes the observations that are closest to the nodes of a 2d grid placed along the first two components of a PCA. The "pca" method uses the nodes instead of the observations. The "random" method samples random observations.

Value

A matrix of prototype coordinates.

Examples

```
dat <- iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]
### Scale training data
dat <- scale(dat)
## Train SOM
### Initialization (PCA grid)
init <- somInit(dat, 4, 4)
the.som <- kohonen::som(dat, grid = kohonen::somgrid(4, 4, 'hexagonal'),
                        rlen = 100, alpha = c(0.05, 0.01),
                        radius = c(2.65,-2.65), init = init,
                        dist.fcts = 'sumofsquares')
```

somQuality *SOM quality measures*

Description

Computes several quality measures on a trained SOM (see Details).

Usage

```
somQuality(som, traindat)
```

Arguments

som	kohonen object, a SOM created by the kohonen::som function.
traindat	matrix containing the training data.

Details

Four measures of SOM quality are returned :

Quantization error: Average squared distance between the data points and the map's prototypes to which they are mapped. Lower is better.

Percentage of explained variance: Similar to other clustering methods, the share of total variance that is explained by the clustering (equal to 1 minus the ratio of quantization error to total variance). Higher is better.

Topographic error: Measures how well the topographic structure of the data is preserved on the map. It is computed as the share of observations for which the best-matching node is not a neighbor of the second-best matching node on the map. Lower is better: 0 indicates excellent topographic representation (all best and second-best matching nodes are neighbors), 1 is the maximum error (best and second-best nodes are never neighbors).

Kaski-Lagus error: Combines aspects of the quantization and topographic error. It is the sum of the mean distance between points and their best-matching prototypes, and of the mean geodesic distance (pairwise prototype distances following the SOM grid) between the points and their second-best matching prototype.

Value

A list containing quality measures : quantization error, share of explained variance, topographic error and Kaski-Lagus error (see Details).

References

Kohonen T. (2001) *Self-Organizing Maps*, 3rd edition, Springer Press, Berlin. <doi:10.1007/978-3-642-56927-2>

Kaski, S. and Lagus, K. (1996) Comparing Self-Organizing Maps. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendho (Eds.) *Proceedings of ICANN96, International Conference on Articial Neural Networks , Lecture Notes in Computer Science* vol. 1112, pp. 809-814. Springer, Berlin. <doi:10.1007/3-540-61510-5_136>

Index