

Package ‘SOMbrero’

November 24, 2013

Title SOM Bound to Realize Euclidean and Relational Outputs

Version 0.4-1

Date 2013-11-24

Author Laura Bendhaiba, Julien Boelaert, Madalina Olteanu, Nathalie Villa-Vialaneix

Maintainer Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

Description SOMbrero provides an implementation of the stochastic (also called on-line) Self-Organising Map (SOM) algorithm for numeric and relational data

Depends R (>= 3.0), wordcloud, igraph (>= 0.6), RColorBrewer, scatterplot3d, knitr, shiny

License GPL (>= 2)

URL <http://sombbrero.r-forge.r-project.org/>

Repository R-Forge

VignetteBuilder knitr

R topics documented:

initGrid	1
initSOM	3
lesmis	5
plot.myGrid	6
plot.somRes	7
predict.somRes	9
presidentielles2002	10
protoDist	11
quality	12
SOMbrero	13
sombbreroGUI	14
somRes.plotting	15
superClass	18
trainSOM	20

initGrid

Create an empty Self-Organizing Map structure

Description

Create an empty Self-Organizing Map structure which is a list of 2-dimensional points.

Usage

```
initGrid(dimension=c(5,5), topo=c("square"),
         dist.type=c("letremy", "maximum", "euclidean", "manhattan",
                    "canberra", "binary", "minkowski"))
## S3 method for class 'myGrid'
print(x, ...)
## S3 method for class 'myGrid'
summary(object, ...)
```

Arguments

dimension	Vector of two integer points corresponding to the x dimension and the y dimension. Default values are: (5,5).
topo	The topology to be used to build the grid. Default value is square.
dist.type	The distance type to be used. Default value is letremy, which corresponds to the original implementation by Patrick Letremy.
x, object	an object of class myGrid
...	not used

Value

initGrid function returns an object of class myGrid, including the following components:

coord	A two columns matrix which provides the coordinates of the points of the Self-Organizing Map structure,
topo	Same value as the arguments given to initGrid function,
dim	Same values as the arguments given to initGrid function.
dist.type	Same value as the arguments given to initGrid function.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

References

Letremy, P. (2005) Programmes bases sur l'algorithme de Kohonen et dedies a l'analyse des donnees. SAS/IML programs for 'korresp'. <http://samos.univ-paris1.fr/Programmes-bases-sur-l-algo>

See Also

See `plot.myGrid` for plotting a `myGrid` class object in a graphical window.

Examples

```
# creating a default grid
# default parameters are: 5x5 dimension, squared topology
# and letremy distance type
initGrid()

# creating a 5x7 squared grid
initGrid(dimension=c(5, 7), topo="square", dist.type="maximum")
```

initSOM

Initialize parameters for the SOM algorithm

Description

The `initSOM` function returns a `paramSOM` class object which contains the parameters needed to run the SOM algorithm.

Usage

```
initSOM(dimension=c(5,5), topo=c("square"),
        dist.type=c("letremy", "maximum", "euclidean", "manhattan",
                    "canberra", "binary", "minkowski"),
        type=c("numeric", "relational", "korresp"), mode=c("online"),
        maxit=500, nb.save=0, verbose=FALSE, proto0=NULL,
        init.proto=switch(type,
                          "numeric"="random",
                          "relational"="obs",
                          "korresp"="random"),
        scaling=switch(type,
                       "numeric"="unitvar",
                       "relational"="none",
                       "korresp"="chi2"),
        radius.type=c("letremy"), eps0=1)
## S3 method for class 'paramSOM'
print(x, ...)
## S3 method for class 'paramSOM'
summary(object, ...)
```

Arguments

<code>dimension</code>	Vector of two integer points corresponding to the x dimension and the y dimension of the <code>myGrid</code> class object. Default values are: (5, 5). Other data-driven defaults are set by function <code>trainSOM</code> .
------------------------	---

<code>topo</code>	The topology to be used to build the grid of the <code>myGrid</code> class object. Default value is <code>square</code> .
<code>dist.type</code>	The neighborhood relationship on the grid. Default value is <code>"letremy"</code> which is the original implementation by Patrick Letremy. The other possible values are the values passed to <code>method</code> in function <code>dist</code> .
<code>type</code>	The SOM algorithm type. Possible values are: <code>numeric</code> (default value), <code>korresp</code> and <code>relational</code> .
<code>mode</code>	The SOM algorithm mode. Default value is <code>online</code> .
<code>maxit</code>	The maximum number of iterations to be done during the SOM algorithm process. Default value is 500. Other data-driven defaults are set by function <code>trainSOM</code> .
<code>nb.save</code>	The number of intermediate back-ups to be done during the algorithm process. Default value is 0.
<code>verbose</code>	The boolean value which activates the verbose mode during the SOM algorithm process. Default value is <code>FALSE</code> .
<code>proto0</code>	The initial prototypes. Default value is <code>NULL</code> .
<code>init.proto</code>	The method to be used to initialize the prototypes, either randomization or an observation sample. This argument is only considered when <code>proto0=NULL</code> . Default value is <code>random</code> for the <code>numeric</code> and <code>korresp</code> types, and <code>obs</code> for the <code>relational</code> type.
<code>scaling</code>	The type of data pre-processing. For <code>numeric</code> SOM, possibilities are <code>unitvar</code> (data are centered and scaled; this is the default value for a <code>numeric</code> SOM) and <code>none</code> (no pre-processing and this is the only value allowed for a <code>relational</code> SOM), <code>center</code> (data are centered but not scaled). For <code>korresp</code> SOM, the only available value (and default) is <code>chi2</code> .
<code>radius.type</code>	The neighbourhood type. Default value is <code>"letremy"</code> , which corresponds to the original implementation by Patrick Letremy.
<code>eps0</code>	The scaling value for the stochastic gradient descent step in the prototypes' update. The scaling value for the stochastic gradient descent step is equal to $\frac{0.3\epsilon_0}{1+0.2t/\text{dim}}$ where t is the current step number and <code>dim</code> is the grid dimension (width multiplied by height).
<code>x, object</code>	an object of class <code>paramSOM</code> .
<code>...</code>	not used

Value

The `initSOM` function returns an object of class `paramSOM` which is a list of the parameters passed to the `initSOM` function, plus the default parameters for the ones not specified by the user.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

References

Letremy, P. (2005) Programmes bases sur l'algorithme de Kohonen et dedies a l'analyse des donnees. SAS/IML programs for 'korresp'. <http://samos.univ-paris1.fr/Programmes-bases-sur-l-algo>

See Also

See `initGrid` for creating a SOM prior structure (grid).

Examples

```
# create a default 'paramSOM' class object
default.paramSOM <- initSOM()
summary(default.paramSOM)
```

lesmis

Dataset "Les Miserables"

Description

This dataset contains the coappearance network (igraph object) of characters in the novel Les Miserables (written by the French writter Victor Hugo).

Usage

```
data(lesmis)
```

Format

`lesmis` is an `igraph` object. Its vertices are the characters of the novel and an edge indicates that the two characters appear together in the same chapter of the novel, at least once. `dissim.lesmis` is a dissimilarity matrix computed with the function `shortest.paths` and containing the length of the shortest paths between pairs of nodes.

Details

Les Miserables is a French historical novel, written by Victor Hugo and published in 1862. The co-appearance network has been extracted by D.E. Knuth (1993).

Source

<http://people.sc.fsu.edu/~jburkardt/datasets/sgb/jean.dat>

References

Hugo, H. (1862) *Les Miserables*.

Knuth, D.E. (1993) *The Stanford GraphBase: A Platform for Combinatorial Computing*. Reading (MA): Addison-Wesley.

Examples

```
data(lesmis)
## Not run: plot(lesmis,vertex.size=0)
```

```
plot.myGrid
```

Draw a myGrid class object

Description

Draw a grid corresponding to a myGrid class object in a graphical window.

Usage

```
## S3 method for class 'myGrid'
plot(x, neuron.col="white", ...)
```

Arguments

<code>x</code>	The myGrid class object to be drawn.
<code>neuron.col</code>	Color(s) used to depict the neurons. Default value is <code>white</code> . If the argument is composed of one single color, neurons will all be filled with the same color. If the argument is composed of many colors, the number of colors must match the total number of neurons.
<code>...</code>	Further arguments to the <code>plot</code> function.

Details

Color filling process uses the coordinates of the object `x`, having class `myGrid` and included in `x$coord`.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

See Also

`initGrid` to define a myGrid class object.

Examples

```
# creating grid
a.grid <- initGrid(dimension=c(5,5), topo="square", dist.type="maximum")

# plotting grid
# without any color specification
plot(a.grid)
# generating colors from rainbow() function
my.colors <- rainbow(5*5)
plot(a.grid, neuron.col=my.colors)
```

plot.somRes	<i>Draw a somRes class object</i>
-------------	-----------------------------------

Description

Produce graphics to help interpreting a somRes object.

Usage

```
## S3 method for class 'somRes'
plot(x, what=c("obs", "prototypes", "energy", "add"),
      type=switch(what,
                  "obs"="hitmap",
                  "prototypes"="color",
                  "add"="pie",
                  "energy"=NULL),
      variable = if (what=="add") NULL else
        if (type=="boxplot") 1:min(5,ncol(x$data)) else 1,
      my.palette=NULL,
      is.scaled = if (x$parameters$type=="numeric") TRUE else
        FALSE,
      proportional=TRUE, print.title=FALSE, s.radius=1,
      pie.graph=FALSE, pie.variable=NULL,
      the.titles=if (what!="energy")
        switch(type,
                "graph"=
                  1:prod(x$parameters$the.grid$dim),
                paste("Cluster",
                      1:prod(x$parameters$
                            the.grid$dim))),
      view = if (x$parameters$type=="korresp") "r" else NULL,
      ...)
```

Arguments

x A somRes class object.

<code>what</code>	What you want to plot. Either the observations (<code>obs</code> , default case), the evolution of energy (<code>energy</code>), the prototypes (<code>prototypes</code>) or an additional variable (<code>add</code>).
<code>type</code>	Further argument indicating which type of chart you want to have. Choices depend on the value of <code>what</code> (<code>what="energy"</code> has no <code>type</code> argument). Default values are <code>"hitmap"</code> for <code>obs</code> , <code>"color"</code> for <code>prototypes</code> and <code>"pie"</code> for <code>add</code> . See section “Details” below for further details.
<code>variable</code>	Either the variable to be used for <code>what="add"</code> or the index of the variable of the data set to consider. For <code>type="boxplot"</code> , the default value is the sequence from 1 to the minimum between 5 and the number of columns of the data set. In all other cases, default value is 1. See <code>somRes.plotting</code> for further details.
<code>my.palette</code>	A vector of colors. If omitted, predefined palettes are used, depending on the plot case. This argument is used for the following combinations: all <code>"color"</code> types and <code>"prototypes"/"poly.dist"</code> .
<code>is.scaled</code>	A boolean indicating whether values should be scaled prior to plotting or not. Default value is <code>TRUE</code> when <code>type="numeric"</code> and <code>FALSE</code> in the other cases.
<code>proportional</code>	Boolean used when <code>what="add"</code> and <code>type="pie"</code> . It indicates if the pies should be proportional to the number of observations in the class. Default value is <code>TRUE</code> .
<code>print.title</code>	Boolean used to indicate whether each neuron should have a title or not. Default value is <code>FALSE</code> . It is feasible on the following cases: all <code>"color"</code> types, all <code>"lines"</code> types, all <code>"barplot"</code> types, all <code>"radar"</code> types, all <code>"boxplot"</code> types, all <code>"names"</code> types, <code>"add"/"pie"</code> , <code>"prototypes"/"umatrix"</code> , <code>"prototypes"/"poly.dist"</code> and <code>"add"/"words"</code> .
<code>s.radius</code>	The size of the pies to be plotted (maximum size when <code>proportional=TRUE</code>). The default value is 0.9.
<code>pie.graph</code>	Boolean used when <code>what="add"</code> and <code>type="graph"</code> . It indicates if the vertices should be pies or not.
<code>pie.variable</code>	The variable needed to plot the pies when <code>what="add"</code> , <code>type="graph"</code> and argument <code>pie.graph</code> is set to <code>TRUE</code> .
<code>the.titles</code>	The titles to be printed for each neuron.
<code>view</code>	Used only when the algorithm’s type is <code>"korresp"</code> . It indicates whether rows (<code>"r"</code>) or columns (<code>"c"</code>) must be drawn.
<code>...</code>	Further arguments to be passed to the underlined plot function (which can be <code>plot</code> , <code>barplot</code> , <code>pie...</code> depending on <code>type</code> ; see <code>somRes.plotting</code> for further details).

Details

See `somRes.plotting` for further details.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>

Madalina Olteanu <madalina.olteanu@univ-paris1.fr>

Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

See Also

`trainSOM` to run the SOM algorithm, that returns a `somRes` class object.

<code>predict.somRes</code>	<i>Predict the classification of a new observation</i>
-----------------------------	--

Description

Predict the neuron where a new observation is classified

Usage

```
## S3 method for class 'somRes'
predict(object, x.new, ...)
```

Arguments

<code>object</code>	a <code>somRes</code> object
<code>x.new</code>	the new observation
<code>...</code>	not used

Details

The length of the new observation must match the number of columns of the data set given to the SOM algorithm (see `trainSOM`).

Value

`predict.somRes` returns the number of the neuron to which the new observation is assigned (i.e., neuron with the closest prototype).

When the algorithm's type is "korresp", `x.new` must be the original contingency table passed to the algorithm.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

See Also

`trainSOM`

Examples

```
set.seed(2343)
my.som <- trainSOM(x.data=iris[-100,1:4], dimension=c(5,5))
predict(my.som, iris[100,1:4])
```

`presidentielles2002`*2002 French presidential election data set*

Description

This data set provides the number of votes at the first round of the 2002 French presidential election for each of the 16 candidates for 106 administrative districts called "Departements".

Usage

```
data(presidentielles2002)
```

Format

`presidentielles2002` is a data frame of 106 rows (the French administrative districts called "Departements") and 16 columns (the candidates).

Source

The data are provided by the French minister "Ministere de l'Interieur". The original data can be downloaded at <http://www.interieur.gouv.fr/Elections/Les-resultats/Presidentielles> (2002 elections and "Resultats par departements")

References

The 2002 French presidential election consisted of two rounds. The second round attracted a greater than usual amount of international attention because of far-right candidate Le Pen's unexpected victory over Socialist candidate Lionel Jospin. The event is known because, on the one hand, the number of candidates was unusually high (16) and, on the other hand, because the polls had failed to predict that Jean-Marie would be on the second round.

Further comments at http://en.wikipedia.org/wiki/French_presidential_election,_2002 or at http://fr.wikipedia.org/wiki/%C3%89lection_pr%C3%A9sidentielle_fran%C3%A7aise_de_2002 (in French).

Examples

```
data(presidentielles2002)
apply(presidentielles2002, 2, sum)
```

protoDist	<i>Compute distances between prototypes</i>
-----------	---

Description

Compute distances, either between all prototypes (mode="complete") or only between prototypes' neighbours (mode="neighbors").

Usage

```
## S3 method for class 'somRes'
protoDist(object, mode=c("complete", "neighbors"), ...)
```

Arguments

object	a somRes object.
mode	Specifies which distances should be computed.
...	Not used.

Details

When mode="complete", distances between all prototypes are computed. When mode="neighbors", distances are computed only between the prototypes and their neighbors.

Value

When mode="complete", the function returns a square matrix which dimensions are equal to the product of the grid dimensions.

When mode="neighbors", the function returns a list which length is equal to the product of the grid dimensions; the length of each item is equal to the number of neighbors.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

See Also

trainSOM

Examples

```
set.seed(2343)
my.som <- trainSOM(x.data=iris[,1:4], dimension=c(5,5))
protoDist(my.som)
```

quality

Compute SOM algorithm quality criteria

Description

The `quality` function computes several quality criteria for the result of a SOM algorithm.

Usage

```
## S3 method for class 'somRes'
quality(sommap,
        quality.type=c("all", "quantization", "topographic"), ...)
```

Arguments

<code>sommap</code>	A <code>somRes</code> object (see <code>trainSOM</code> for details).
<code>quality.type</code>	The quality type to compute. Two types are implemented: <code>quantization</code> and <code>topographic</code> . The output of the function is one those or both of them using the option <code>"all"</code> . Default value is the latter.
<code>...</code>	Not used.

Value

The `quality` function returns either a numeric value (if only one type is computed) or a list a numeric values (if all types are computed)

The quantization error calculates the mean distance between the sample vectors and their respective cluster prototypes. It is a decreasing function of the size of the map.

The topographic error is the simplest of the topology preservation measure: it calculates the ratio of sample vectors for which the second best matching unit is in the direct neighborhood of the best matching unit.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

References

Polzlbauer, G. (2004) Survey and comparison of quality measures for self-organizing maps. In: *Proceedings of the Fifth Workshop on Data Analysis (WDA'04)*, Paralic, J., Polzlbauer, G., Rauber, A. (eds) Sliezsky dom, Vysoke Tatry, Slovakia: Elfa Academic Press, 67–82.

See Also

`trainSOM`, `plot.somRes`

Examples

```
my.som <- trainSOM(x.data=iris[,1:4])
quality(my.som, quality.type="all")
quality(my.som, quality.type="topographic")
```

SOMbrero

Self Organizing Maps Bound to Realize Euclidean and Relational Outputs

Description

This package implements the stochastic (also called on-line) Self-Organising Map (SOM) algorithm for numeric and relational data.

It is based on a grid (see `initGrid`) which is part of the parameters given to the algorithm (see `initSOM` and `trainSOM`). Many graphs can help you with the results (see `plot.somRes`).

Details

```
Package:  SOMbrero
Type:     Package
Version:  0.4-1
Date:     2013-11-24
License:  GPL (>= 2)
URL:      http://sombbrero.r-forge.r-project.org/
```

The version of the SOM algorithm implemented in this package is the stochastic version.

Several variants able to handle non-vectorial data are also implemented in their stochastic versions:

`type="korresp"` for contingency tables, as described in Cottrel et al., 1993 and `type="relational"` for dissimilarity data, as described in Olteanu and Villa-Vialaneix, 2013.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>

Julien Boelaert <julien.boelaert@gmail.com>

Madalina Olteanu <madalina.olteanu@univ-paris1.fr>

Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

Maintainer: Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

References

Kohonen T. (2001) *Self-Organizing Maps*. Berlin/Heidelberg: Springer-Verlag, 3rd edition.

Cottrell, M., Letremy, P., Roy, E. (1993) Analyzing a contingency table with Kohonen maps: a Factorial Correspondence Analysis. In: *Proceedings of IWANN'93*, J. Cabestany, J. Mary, A. Prieto (Eds.), *Lecture Notes in Computer Science*, Springer-Verlag, 305–311.

Letremy, P. (2005) Programmes bases sur l'algorithme de Kohonen et dedies a l'analyse des donnees. SAS/IML programs for 'korresp'. <http://samos.univ-paris1.fr/Programmes-bases-sur-l-algo>

Olteanu, M., Villa-Vialaneix, N. (2013) On-line relational and multiple relational SOM. *Neurocomputing*. Forthcoming.

See Also

`initGrid`, `trainSOM`, `plot.somRes` and `sombbreroGUI`.

sombbreroGUI

Graphical Web User Interface for SOMbrero

Description

Starts the SOMbrero GUI

Usage

```
sombbreroGUI()
```

Arguments

No arguments.

Value

This function starts the graphical user interface with the default system browser. This interface is more likely to work properly with Firefox <http://www.mozilla.org/fr/firefox/new/>. In case Firefox is not your default browser, copy/paste <http://localhost:8100/> into the address bar.

Author(s)

Julien Boelaert <julien.boelaert@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

References

RStudio and Inc. (2013). shiny: Web Application Framework for R. R package version 0.7.0. <http://cran.r-project.org/package=shiny>

Description

Useful details on how to produce graphics to help interpreting a somRes object. Important: all these graphics are available when the algorithm's type is "numeric" ; those which are available for a korresp SOM are marked by a * and those which are available for a relational SOM are marked with a #.

Graphics on the observations: what="obs"

The possible values for type are: "hitmap"(*, #), "color", "lines", "barplot", "names"(*, #), "boxplot" and "radar".

For the cases what="obs" and what="add", if a neuron is empty, nothing will be plotted at its location.

- "hitmap" (*, #) plots proportional areas according to the number of observations per neuron. It is the default plot when what="obs".
- "color" can have two more arguments, var, the index of the variable to be considered (default, 1), and my.palette for the colors to be used. Neurons are filled using the given colors according to the average value level of the observations for the chosen variable.
- "lines" plots, for each neuron, the average value level of the observations, with lines. One point represents a variable. All variables of the dataset used to train the algorithm are plotted.
- "barplot" is similar to "lines" but using barplots. Then, a bar represents a variable.
- "radar" is similar to "lines" but using radars. Then, a slice represents a variable. If needed, a legend can be added ; its location has to be passed by the key.loc argument (see stars).
- "names" (*, #) prints on the grid the element names (i.e., the names of the rows) in the neuron to which it belongs.
- "boxplot" plots boxplots for several observations in every neuron. This case can handle 5 variables at most. The default behavior is to plot the boxplots for the first 5 variables of the data set; if there is less than 5 variables in the data set, they will all be plotted.

When the algorithm's type is korresp or relational, only the types "hitmap" and "names" are available.

Graphic on the energy: what="energy" (*, #)

This graphic is only available if some intermediate backups have been registered (i.e., x\$parameters\$nb.save>1). Graphic plots the evolution of the level of the energy according to the registered steps.

Graphics on the prototypes: `what="prototypes"`

The possible values for `type` are: `"3d"(*, #)`, `"lines"(*, #)`, `"barplot"(*, #)`, `"radar"(*, #)`, `"color"(*, #)`, `"smooth.dist"(*, #)`, `"poly.dist"(*, #)`, `"umatrix"(*, #)`, `"mds"(*, #)` and `"grid.dist"(*, #)`.

- `"lines"(*, #)` has the same behavior as the `"lines"` case described in the observations section, but according to the prototypes level;
- `"barplot"(*, #)` has the same behavior as the `"barplot"` case described in the observations section, but according to the prototypes level;
- `"radar"(*, #)` has the same behavior as the `"radar"` case described in the observations section, but according to the prototypes level;
- `"color"(*, #)` has the same behavior as the `"color"` case described in the observations section, but according to the prototypes level;
- `"3d"` case is similar to the `"color"` case, but in 3 dimensions, with `x` and `y` the coordinates of the grid and `z` the value of the prototypes for the considered variable;
- `"smooth.dist"(*, #)` depicts the average distance between a prototypes and its neighbors on a map where `x` and `y` are the coordinates of the prototypes on the grid;
- `"poly.dist"(*, #)` also represents the distances between prototypes but with polygons plotted for each neuron. The closest from the border the polygon point is, the closest the pairs of prototypes are. The color used for filling the polygon shows the number of observations in each neuron. A white polygon means that there is no observation. With the default colors, a red polygon means a high number of observations;
- `"umatrix"(*, #)` is another way of plotting distances between prototypes. The grid is plotted and filled with `my.palette` colors according to the mean distance between the current neuron and the neighboring neurons. With the default colors, red indicates proximity.
- `"mds"(*, #)` plots the number of the neuron on a map according to a Multi Dimensional Scaling (MDS) projection on a two dimensional space.
- `"grid.dist"(*, #)` plots on a 2 dimension map all distances. The number of points on this picture is equal to: $\frac{\text{number of neurons} \times (\text{number of neurons} - 1)}{2}$. On the `x` axis corresponds to the prototype distances whereas the `y` axis depicts the grid distances.

Graphics on an additional variable: `what="add" (#)`

The case `what="add"` considers an additional variable, which has to be given to the argument `variable`. Its length must match the number of observations in the original data. Then the possible values for `type` are: `"pie"(#)`, `"color"(#)`, `"lines"(#)`, `"boxplot"(#)`, `"barplot"(#)`, `"radar"(#)`, `"names"(#)`, `"words"(#)` and `"graph"(#)`.

- `"color"(#)` has the same behavior as the `"color"` case described in the observations section. Then, the additional variable must be a numerical vector;
- `"lines"(#)` has the same behavior as the `"color"` case described in the observations section. Then, the additional variable must be a numerical matrix or a data frame;
- `"boxplot"(#)` has the same behavior as the `"color"` case described in the observations section. Then, the additional variable must be either a numeric vector or a numeric matrix/data frame;

- "barplot" (#) has the same behavior as the "color" case described in the observations section. Then, the additional variable must be either a numeric vector or a numeric matrix/data frame;
- "radar" (#) has the same behavior as the "color" case described in the observations section. Then, the additional variable must be a numerical matrix or data frame;
- "pie" requires the argument `variable` to be a factor vector and plots one pie for each neuron according to this factor;
- "names" (#) has the same behavior as the "names" case described in the observations section. Then, the names to be printed are the elements of the variable given to the `variable` argument;
- "words" (#) needs the argument `variable` be a contingency table: names of the columns will be used as words and the values express the frequency of a given word in the observation. Then, for each neuron of the grid, the words will be printed with sizes proportional to their frequency in the neuron;
- Last option is "graph" (#). The argument `variable` must be an `igraph` object (see `library(igraph)`). According to the existing edges in the graph and to the clustering obtained with the SOM algorithm, a clustered graph will be produced where a vertex between two vertices represents a neuron and the width of an edge is proportional to the number of edges in the given graph between the vertices affected to the corresponding neurons. The option can handle two more arguments: `pie.graph` and `pie.variable`. These are used to display the vertex as pie charts. For this case, `pie.graph` must be set to `TRUE` and a factor vector is supplied by `pie.variable`.

When the algorithm's type is `korresp`, no graphic is available for `what="add"`.

All these graphics are available for a `relational SOM`.

Further arguments via ...

Further arguments, their reference functions and the `plot.somRes` cases are summarized in the following list:

- `plot` is called by the cases:
 - `what="energy"`
 - `type="lines"`
 - `what="prototypes"/type="mds"`
- `plot.myGrid` is called by the cases:
 - `what="obs"/type="hitmap"`
 - `type="color"`
 - `what="prototypes"/type="poly.dist"`
 - `what="prototypes"/type="umatrix"`
- `plot.igraph` is called by the case `what="add"/type="graph"`
- `pie` is called by the case `what="add"/type="pie"`
- `barplot` is called by the cases `type="barplot"`
- `boxplot` is called by the cases `type="boxplot"`

- stars is called by the cases type="radar"
- persp is called by the case what="prototypes"/type="3d"
- wordcloud is called by the cases:
 - type="names"
 - what="add"/type="words"

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

Examples

```
# run the SOM algorithm on the numerical data of 'iris' data set
iris.som <- trainSOM(x.data=iris[,1:4], nb.save=2)

# plots
# on energy
plot(iris.som, what="energy") # energy
# on prototypes
plot(iris.som, what="prototypes", type="3d", variable="Sepal.Length")
# on an additional variable: the flower species
plot(iris.som, what="add", type="pie", variable=iris$Species)
```

superClass

Create super-clusters from SOM clustering

Description

Aggregate the resulting clustering of the SOM algorithm into super-clusters.

Usage

```
## S3 method for class 'somRes'
superClass(sommap, method="ward", members=NULL, k=NULL,
h=NULL, ...)
## S3 method for class 'somSC'
print(x, ...)
## S3 method for class 'somSC'
summary(object, ...)
## S3 method for class 'somSC'
plot(x, type=c("dendrogram", "grid", "hitmap", "lines",
               "barplot", "boxplot", "mds", "color",
               "poly.dist", "pie", "graph", "dendro3d",
               "radar"),
      plot.var=TRUE, plot.legend=FALSE, add.type=FALSE,
      ...)
```

Arguments

sommap	A somRes object
method, members	Arguments passed to the <code>hclust</code> function.
k, h	Arguments passed to the <code>cutree</code> function (respectively, the number of super-clusters or the height where to cut the dendrogram).
x, object	A somSC object
type	The type of plot to draw. Default value is "dendrogram", to plot the dendrogram of the clustering. Case "grid" plots the grid in color according to the super clustering. All other cases are those available in the function <code>plot.somRes</code> .
plot.var	A boolean indicating whether a graph showing the evolution of the explained variance should be plotted. This argument is only used when <code>type="dendrogram"</code> , its default value is <code>TRUE</code> .
plot.legend	A boolean indicating whether a legend should be added to the plot. This argument is only used when <code>type</code> is either "grid" or "hitmap" or "mds". Its default value is <code>FALSE</code> .
add.type	A boolean, which default value is <code>FALSE</code> , indicating whether you are giving an additional variable to the argument <code>variable</code> or not. If you do, the function <code>plot.somRes</code> will be called with the argument <code>what</code> set to "add".
...	Used for <code>plot.somSC</code> : further arguments passed either to the function <code>plot</code> (case <code>type="dendro"</code>) or to <code>plot.myGrid</code> (case <code>type="grid"</code>) or to <code>plot.somRes</code> (all other cases).

Details

The `superClass` function can be used in 2 ways:

- to choose the number of super clusters via an `hclust` object: then, both arguments `k` and `h` are not filled.
- to cut the clustering into super clusters: then, either argument `k` or argument `h` must be filled. See `cutree` for details on these arguments.

The squared distance between prototypes is passed to the algorithm.

The different super classes are identified in the following ways:

- either with different color, when `type` is set among: "grid" (*, #), "hitmap" (*, #), "lines" (*, #), "barplot" (*, #), "boxplot", "mds" (*, #), "dendro3d" (*, #), "graph" (*, #)
- or with title, when `type` is set among: "color" (*), "poly.dist" (*, #), "pie" (#), "radar" (#)

In the list above, graphics available for a `korresp` SOM are marked with a * whereas those available for a `relational` SOM are marked with a #.

Value

The `superClass` function returns an object of class `somSC` which is a list of the following elements:

<code>cluster</code>	The super clustering of the prototypes (only if either <code>k</code> or <code>h</code> are given by user).
<code>tree</code>	An <code>hclust</code> object.
<code>som</code>	The <code>somRes</code> object given as argument (see <code>trainSOM</code> for details).

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
 Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
 Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

See Also

`hclust`, `cutree`, `trainSOM`, `plot.somRes`

Examples

```
set.seed(11051729)
my.som <- trainSOM(x.data=iris[,1:4])
# choose the number of super-clusters
sc <- superClass(my.som)
plot(sc)
# cut the clustering
sc <- superClass(my.som, k=4)
summary(sc)
plot(sc)
plot(sc, type="hitmap", plot.legend=TRUE)
```

trainSOM

Run the SOM algorithm

Description

The `trainSOM` function returns a `somRes` class object which contains the outputs of the algorithm.

Usage

```
trainSOM(x.data, ...)
## S3 method for class 'somRes'
print(x, ...)
## S3 method for class 'somRes'
summary(object, ...)
```

Arguments

<code>x.data</code>	a data frame or matrix containing the observations to be mapped on the grid by the SOM algorithm.
<code>...</code>	Further arguments to be passed to the function <code>initSOM</code> for specifying the parameters of the algorithm. The default values of the arguments <code>maxit</code> and <code>dimension</code> are calculated according to the SOM type if the user does not set them: <ul style="list-style-type: none"> • <code>maxit</code> is equal to $(\text{number of rows} + \text{number of columns}) * 5$ if the SOM type is <code>korresp</code>. It is equal to $\text{number of rows} * 5$ in all other SOM types • <code>dimension</code>: for a <code>korresp</code> SOM, is approximately equal to the square root of the number of observations to be classified divided by 10 but it is never smaller than 5 or larger than 10.
<code>x, object</code>	an object of class <code>somRes</code>

Details

The version of the SOM algorithm implemented in this package is the stochastic version.

Several variants able to handle non-vectorial data are also implemented in their stochastic versions: `type="korresp"` for contingency tables, as described in Cottrel et al., 1993; `type="relational"` for dissimilarity matrices, as described in Olteanu et al., 2013.

Value

The `trainSOM` function returns an object of class `somRes` which contains the following components:

<code>clustering</code>	the final classification of the data.
<code>prototypes</code>	the final coordinates of the prototypes.
<code>energy</code>	the final energy of the map.
<code>backup</code>	a list containing some intermediate backups of the prototypes coordinates, clustering, energy and the indexes of the recorded backups, if <code>nb.save</code> is set to a value larger than 1.
<code>data</code>	the original dataset used to train the algorithm.
<code>parameters</code>	a list of the map's parameters, which is an object of class <code>paramSOM</code> as produced by the function <code>initSOM</code> .

The function `summary.somRes` also provides an ANOVA (ANalysis Of VAriance) of each input numeric variables in function of the map's clusters. This is helpful to see which variables participate to the clustering.

Note

Warning! Recording intermediate backups with the argument `nb.save` can strongly increase the computational time since calculating the entire clustering and the energy is time consuming. Use this option with care and only when it is strictly necessary.

Author(s)

Laura Bendhaiba <laurabendhaiba@gmail.com>
Madalina Olteanu <madalina.olteanu@univ-paris1.fr>
Nathalie Villa-Vialaneix <nathalie.villa@univ-paris1.fr>

References

Kohonen T. (2001) *Self-Organizing Maps*. Berlin/Heidelberg: Springer-Verlag, 3rd edition.
Cottrell, M., Letremy, P., Roy, E. (1993) Analyzing a contingency table with Kohonen maps: a Factorial Correspondence Analysis. In: *Proceedings of IWANN'93*, J. Cabestany, J. Mary, A. Prieto (Eds.), *Lecture Notes in Computer Science*, Springer-Verlag, 305–311.
Olteanu, M., Villa-Vialaneix, N. (2013) On-line relational and multiple relational SOM. *Neurocomputing*. Forthcoming.

See Also

See plot.somRes to plot the outputs of the algorithm.

Examples

```
# Run trainSOM algorithm on the iris data with 500 iterations
iris.som <- trainSOM(x.data=iris[,1:4])
iris.som
summary(iris.som)
```