

# Package ‘som’

July 23, 2025

**Version** 0.3-5.2

**Date** 2010-04-08

**Title** Self-Organizing Map

**Depends** R (>= 2.10)

**Description** Self-Organizing Map (with application in gene clustering).

**License** GPL (>= 3)

**Repository** CRAN

**Date/Publication** 2024-09-18 10:16:45 UTC

**NeedsCompilation** yes

**Author** Jun Yan [aut, cre]

**Maintainer** Jun Yan <jyan@stat.uconn.edu>

## Contents

filtering . . . . .	1
normalize . . . . .	2
plot.som . . . . .	3
qerror . . . . .	4
som . . . . .	4
summary.som . . . . .	7
yeast . . . . .	7

<b>Index</b>	<b>9</b>
--------------	----------

---

filtering	<i>Filter data before feeding som algorithm for gene expression data</i>
-----------	--

---

## Description

Filtering data by certain floor, ceiling, max/min ratio, and max - min difference.

**Usage**

```
filtering(x, lt=20, ut=16000, mmr=3, mmd=200)
```

**Arguments**

x	a data frame or matrix of input data.
lt	floor value replaces those less than it with the value
ut	ceiling value replaced those greater than it with the value
mmr	the max/min ratio, rows with max/min < mmr will be removed
mmd	the max - min difference, rows with (max - min) < mmd will be removed

**Value**

An dataframe or matrix after the filtering

**Author(s)**

Jun Yan <jyan@stat.uiowa.edu>

**See Also**

[normalize.](#)

---

normalize	<i>normalize data before feeding som algorithm</i>
-----------	--

---

**Description**

Normalize the data so that each row has mean 0 and variance 1.

**Usage**

```
normalize(x, byrow=TRUE)
```

**Arguments**

x	a data frame or matrix of input data.
byrow	whether normalizing by row or by column, default is byrow.

**Value**

An dataframe or matrix after the normalizing.

**Author(s)**

Jun Yan <jyan@stat.uiowa.edu>

**See Also**

[filtering](#).

---

plot.som

*Visualizing a SOM*

---

**Description**

Plot the SOM in a 2-dim map with means and sd bars.

**Usage**

```
## S3 method for class 'som'  
plot(x, sdbar=1, ylim=c(-3, 3), color=TRUE,  
      ntik=3, yadj=0.1, xlab="", ylab="", ...)
```

**Arguments**

x	a som object
sdbar	the length of sdbar in sd, no sdbar if sdbar=0
ylim	the range of y axes in each cell of the map
color	whether or not use color plotting
ntik	the number of tiks of the vertical axis
yadj	the proportion used to put the number of obs
xlab	x label
ylab	y label
...	other options to plot

**Note**

This function is not cleanly written. The original purpose was to mimic what GENECLUSTER does. The ylim is hardcoded so that only standardized data could be properly plotted.

There are visualization methods like umat and sammon in SOM\_PAK3.1, but not implemented here.

**Author(s)**

Jun Yan <jyan@stat.uiowa.edu>

**Examples**

```
foo <- som(matrix(rnorm(1000), 250), 3, 5)  
plot(foo, ylim=c(-1, 1))
```

qerror *quantization accuracy*

---

**Description**

get the average distortion measure

**Usage**

```
qerror(obj, err.radius=1)
```

**Arguments**

obj            a 'som' object  
err.radius    radius used calculating qerror

**Value**

An average of the following quantity (weighted distance measure) over all x in the sample,

$$\sum ||x - m_i|| h_{ci}$$

where  $h_{ci}$  is the neighbourhood kernel for the ith code.

**Author(s)**

Jun Yan <jyan@stat.uiowa.edu>

**Examples**

```
foo <- som(matrix(rnorm(1000), 100), 2, 4)  
qerror(foo, 3)
```

---

som *Function to train a Self-Organizing Map*

---

**Description**

Produces an object of class "som" which is a Self-Organizing Map fit of the data.

**Usage**

```

som.init(data, xdim, ydim, init="linear")
som(data, xdim, ydim, init="linear", alpha=NULL, alphaType="inverse",
neigh="gaussian", topol="rect", radius=NULL, rlen=NULL, err.radius=1,
inv.alp.c=NULL)
som.train(data, code, xdim, ydim, alpha=NULL, alphaType="inverse",
neigh="gaussian", topol="rect", radius=NULL, rlen=NULL, err.radius=1, inv.alp.c=NULL)
som.update(obj, alpha = NULL, radius = NULL, rlen = NULL, err.radius =
1, inv.alp.c = NULL)
som.project(obj, newdat)

```

**Arguments**

obj	a 'som' object.
newdat	a new dataset needs to be projected onto the map.
code	a matrix of initial code vector in the map.
data	a data frame or matrix of input data.
xdim	an integer specifying the x-dimension of the map.
ydim	an integer specifying the y-dimension of the map.
init	a character string specifying the initializing method. The following are permitted: "sample" uses a random sample from the data; "random" uses random draws from N(0,1); "linear" uses the linear grids upon the first two principle components directin.
alpha	a vector of initial learning rate parameter for the two training phases. Decreases linearly to zero during training.
alphaType	a character string specifying learning rate function type. Possible choices are linear function ("linear") and inverse-time type function ("inverse").
neigh	a character string specifying the neighborhood function type. The following are permitted: "bubble" "gaussian"
topol	a character string specifying the topology type when measuring distance in the map. The following are permitted: "hexa" "rect"
radius	a vector of initial radius of the training area in som-algorithm for the two training phases. Decreases linearly to one during training.
rlen	a vector of running length (number of steps) in the two training phases.
err.radius	a numeric value specifying the radius when calculating average distortion measure.
inv.alp.c	the constant C in the inverse learning rate function: $\alpha_0 * C / (C + t)$ ;

**Value**

'som.init' initializes a map and returns the code matrix. 'som' does the two-step som training in a batch fashion and return a 'som' object. 'som.train' takes data, code, and traing parameters and perform the requested som training. 'som.update' takes a 'som' object and further train it with updated paramters. 'som.project' projects new data onto the map.

An object of class "som" representing the fit, which is a list containing the following components:

data	the dataset on which som was applied.
init	a character string indicating the initializing method.
xdim	an integer specifying the x-dimension of the map.
ydim	an integer specifying the y-dimension of the map.
code	a matrix with nrow = xdim*ydim, each row corresponding to a code vector of a cell in the map. The mapping from cell coordinate (x, y) to the row index in the code matrix is: rownumber = x + y * xdim
visual	a data frame of three columns, with the same number of rows as in data: x and y are the coordinate of the corresponding observation in the map, and qerror is the quantization error computed as the squared distance (depends topol) between the observation vector and its coding vector.
alpha0	a vector of initial learning rate parameter for the two training phases.
alpha	a character string specifying learning rate funciton type.
neigh	a character string specifying the neighborhood function type.
topol	a character string specifying the topology type when measuring distance in the map.
radius0	a vector of initial radius of the training area in som-algorithm for the two training phases.
rln	a vector of running length in the two training phases.
qerror	a numeric value of average distortion measure.
code.sum	a dataframe summaries the number of observations in each map cell.

**Author(s)**

Jun Yan <jyan@stat.uiowa.edu>

**References**

Kohonen, Hynninen, Kangas, and Laaksonen (1995), SOM-PAK, the Self-Organizing Map Program Package (version 3.1). [http://www.cis.hut.fi/research/papers/som\\_tr96.ps.Z](http://www.cis.hut.fi/research/papers/som_tr96.ps.Z)

**Examples**

```
data(yeast)
yeast <- yeast[, -c(1, 11)]
yeast.f <- filtering(yeast)
yeast.f.n <- normalize(yeast.f)
foo <- som(yeast.f.n, xdim=5, ydim=6)
foo <- som(yeast.f.n, xdim=5, ydim=6, topol="hexa", neigh="gaussian")
plot(foo)
```

---

summary.som	<i>summarize a som object</i>
-------------	-------------------------------

---

**Description**

print out the configuration parameters of a som object

**Usage**

```
## S3 method for class 'som'  
summary(object, ...)  
## S3 method for class 'som'  
print(x, ...)
```

**Arguments**

object, x	a 'som' object
...	nothing yet

**Author(s)**

Jun Yan <jyan@stat.uiowa.edu>

---

yeast	<i>yeast cell cycle</i>
-------	-------------------------

---

**Description**

The yeast data frame has 6601 rows and 18 columns, i.e., 6601 genes, measured at 18 time points.

**Usage**

```
data(yeast)
```

**Format**

This data frame contains the following columns:

**Gene** a character vector of gene names

**zero** a numeric vector

**ten** a numeric vector

**twenty** a numeric vector

**thirty** a numeric vector

**fourty** a numeric vector

**fifty** a numeric vector

**sixty** a numeric vector

**seventy** a numeric vector

**eighty** a numeric vector

**ninety** a numeric vector

**hundred** a numeric vector

**one.ten** a numeric vector

**one.twenty** a numeric vector

**one.thirty** a numeric vector

**one.fourty** a numeric vector

**one.fifty** a numeric vector

**one.sixty** a numeric vector

### Source

<http://genomics.stanford.edu>

### References

Tamayo et. al. (1999), Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation, PNAS V96, pp2907-2912, March 1999.



# Index

- \* **arith**
  - qerror, 4
- \* **cluster**
  - som, 4
- \* **datasets**
  - yeast, 7
- \* **hplot**
  - plot.som, 3
- \* **manip**
  - filtering, 1
  - normalize, 2
- \* **print**
  - summary.som, 7

filtering, 1, 3

normalize, 2, 2

plot.som, 3

print.som (summary.som), 7

qerror, 4

som, 4

summary.som, 7

yeast, 7