

# Package ‘SOMMD’

July 21, 2025

**Title** Self Organising Maps for the Analysis of Molecular Dynamics Data

**Version** 0.1.2

**Description** Processes data from Molecular Dynamics simulations using Self Organising Maps. Features include the ability to read different input formats. Trajectories can be analysed to identify groups of important frames. Output visualisation can be generated for maps and pathways. Methodological details can be found in Motta S et al (2022)

[doi:10.1021/acs.jctc.1c01163](https://doi.org/10.1021/acs.jctc.1c01163).

I/O functions for xtc format files were implemented using the 'xdrfile' library available under open source license. The relevant information can be found in inst/COPYRIGHT.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Depends** R (>= 3.5.0)

**Imports** bio3d, kohonen, abind, cluster, methods, igraph

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Alessandro Pandini [aut, cph] (ORCID:  
<https://orcid.org/0000-0002-4158-233X>),  
Stefano Motta [aut, cre, cph] (ORCID:  
<https://orcid.org/0000-0002-0812-6834>),  
Erik Lindahl [ctb] (Author of xdrfile C library),  
David van der Spoel [ctb] (Author of xdrfile C library)

**Maintainer** Stefano Motta <stefano.motta@unimib.it>

**Repository** CRAN

**Date/Publication** 2024-10-02 13:50:06 UTC

## Contents

average.neur.property . . . . .	3
calc.distances . . . . .	3
calc.dists . . . . .	4
cat.trj . . . . .	5
cluster.pathways . . . . .	6
cluster.representatives . . . . .	7
comp.trans.mat . . . . .	8
fit.trj . . . . .	9
map.color . . . . .	10
matrix2graph . . . . .	10
native.cont . . . . .	11
neur.population . . . . .	12
neur.representatives . . . . .	13
print.struct . . . . .	14
print.trj . . . . .	14
read.gro . . . . .	15
read.struct . . . . .	16
read.trj . . . . .	17
remap.data . . . . .	18
rio_read_xtc . . . . .	19
rio_read_xtc2xyz . . . . .	19
rio_read_xtc_natoms . . . . .	20
rio_read_xtc_nframes . . . . .	20
rio_write_xtc . . . . .	21
silhouette.profile . . . . .	21
silhouette.score . . . . .	22
som.add.circles . . . . .	23
som.add.clusters.legend . . . . .	24
som.add.numbers . . . . .	25
stride.trj . . . . .	25
struct2pdb . . . . .	26
summary.struct . . . . .	27
summary.trj . . . . .	27
trace.path . . . . .	28
trj2pdb . . . . .	29
trj2xyz . . . . .	30

## Index

**31**

---

average.neur.property *Compute average property*

---

**Description**

Function to compute the average value of a property for each neuron.

**Usage**

```
average.neur.property(SOM, P)
```

**Arguments**

SOM	the SOM object to cluster
P	the property for each frame of the simulation

**Value**

The a vector with the per-neuron average of the property.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Compute distance between two atoms in every frame of the simulation
Distance <- apply(trj$coord[c(162,1794),,], 3, dist)
#Compute average property value for each neuron
avg.p <- average.neur.property(som_model, Distance)
```

---

calc.distances *calc.distances*

---

**Description**

Function to compute distances to be used to train the SOM

**Usage**

```
calc.distances(trj, mol.2 = FALSE, sele = FALSE, atoms = NULL, cap = NULL)
```

**Arguments**

trj	contains the trajectory coordinates (array with three dimensions obtained by ri-oxdr)
mol.2	contains the atom indexes of the second molecule in case only intermolecular distances should be computed
sele	contains the selection of distances coming from the native_contacts function
atoms	contains a list of atoms indexes on which the distances will be computed
cap	If a number is given, distances greater than this value are set at the cap value

**Value**

A matrix containing the set of distances computed for all the frames.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
# Read reference structure file with native conformation
struct <- read.struct(system.file("extdata", "HIF2a.gro", package = "SOMMD"))
# Read the trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
# Select only Cbeta atoms to perform the analysis
sele_atoms <- which(trj$top$seley=="CB")
# Choose only native contacts
sele_dists <- native.cont(struct=struct, distance=1.0, atoms=sele_atoms)
# Compute distances for SOM training.
DIST <- calc.distances(trj, mol.2=FALSE, sele=sele_dists, atoms=sele_atoms)
```

---

calc.dists

*Calculation of Distances*

---

**Description**

Compute the pairwise distance matrix of a given set of coordinates, and only retain to some selected distances

**Usage**

```
calc.dists(coord, mol.1_id = FALSE, mol.2_id = FALSE, sele = FALSE)
```

**Arguments**

coord	matrix of N atomic coordinates (N rows, 3 columns)
mol.1_id	vector containing the index of the first molecule for intermolecular distances only
mol.2_id	vector containing the index of the second molecule for intermolecular distances only
sele	contains the selection of distances coming from the native_contacts function

**Value**

A matrix containing the selected distances for a frame

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

---

cat.trj

*Concatenate simulations*

---

**Description**

Function to concatenate two simulations.

**Usage**

```
cat.trj(trj1, ...)
```

**Arguments**

trj1	the first trj file
...	additional trj files

**Value**

A trj object with the simulations concatenated

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

## Examples

```
# Read the simulations
trj1 <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
trj2 <- read.trj(trjfile = system.file("extdata", "HIF2a-MD-2.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
# Concatenate the simulations
trj <- cat.trj(trj1, trj2)
```

---

cluster.pathways      *Clustering of Pathways*

---

## Description

Cluster pathways according to a time dependent or independent scheme

## Usage

```
cluster.pathways(
  SOM,
  start,
  end,
  time.dep = "independent",
  method = "complete"
)
```

## Arguments

SOM	a kohonen SOM object.
start	the vector specifying the starting frame of each replicas
end	the vector specifying the ending frame of each replicas
time.dep	choose whether to use time "dependent" or "independent" clustering of pathways
method	the method to be passed to hclust for the clustering

## Value

representatives a vector of frames representatives of each neuron

## Author(s)

Stefano Motta<stefano.motta@unimib.it>

## Examples

```
#Read trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
#Assign length of the replicas
trj$start <- seq(1, 25, by=5)
trj$end <- seq(5, 25, by=5)
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Cluster Pathways using the time dependent algorithm
clus.paths.tdep <- cluster.pathways(som_model, start=trj$start, end=trj$end,
  time.dep="dependent")
#Cluster Pathways using the time independent algorithm
clus.paths.tindep <- cluster.pathways(som_model,
  start=trj$start, end=trj$end, time.dep="independent")
```

---

cluster.representatives

*Cluster Representatives*

---

## Description

Compute the cluster representatives

## Usage

```
cluster.representatives(SOM, clusters)
```

## Arguments

SOM	a kohonen SOM object.
clusters	a vector of clusters assignment for each neuron, as returned for example by hclust.

## Value

A vector of frames representatives of each neuron

## Author(s)

Stefano Motta <stefano.motta@unimib.it>

## Examples

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
# Divide the SOM in the selected number of clusters
som_cl <- cutree(hclust(dist(som_model$codes[[1]]), method="euclidean"), method="complete", 4)
#Get representative frames for each cluster
cl_repres <- cluster.representatives(som_model, som_cl)
```

---

comp.trans.mat

*Compute transition matrix*

---

## Description

Compute the transition matrix starting from a vector of subsequent classifications

## Usage

```
comp.trans.mat(SOM, start = 1)
```

## Arguments

SOM	a kohonen object on which transitions between neurons will be computed
start	a vector containing the start frames of each replica (usually contained in trj\$start if replicas were merged with cat_trj)

## Value

A matrix of pairwise transitions between neurons

## Author(s)

Stefano Motta <stefano.motta@unimib.it>

## Examples

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Compute transition Matrix
tr_mat <- comp.trans.mat(som_model, start = 1)
```



---

`fit.trj`*Coordinate superposition*

---

### Description

Coordinate superposition with the Kabsch algorithm. This function make use of the bio3d `fit.xyz` function to align a SOMMD trj object. If `ref` is not specified, the trj object is aligned to the first frame of the simulation, otherwise it is aligned to the reference input object.

### Usage

```
fit.trj(trj, ref = NULL, trj.inds = NULL, ref.inds = NULL)
```

### Arguments

<code>trj</code>	an object with class trj
<code>ref</code>	a struct object read with <code>read.struct()</code> to be used as reference
<code>trj.inds</code>	a vector of indices that selects the trj atoms upon which fitting should be based. If not specified all atoms will be used.
<code>ref.inds</code>	a vector of indices that selects the ref atoms upon which fitting should be based. If not specified all atoms will be used.

### Value

A trj object aligned

### Author(s)

Stefano Motta <stefano.motta@unimib.it>

### Examples

```
#Read trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
# Fit a trajectory to the first frame based on alpha carbons:
ca.inds <- which(trj$top$seley=="CA")
trj.fit <- fit.trj(trj, trj.inds=ca.inds)
```

---

`map.color`*Map the property vector to colours*

---

**Description**

Function map a numeric vector of a property to a vector of colors for that property according to that property value.

**Usage**

```
map.color(x, pal, limits = NULL, na.col = "grey")
```

**Arguments**

<code>x</code>	a numeric vector
<code>pal</code>	a color palette
<code>limits</code>	the values of the extremes for the colorscale
<code>na.col</code>	the color that will be assigned to the na.values of the vector

**Value**

A vector with colors proportional to the values of `x`

---

`matrix2graph`*Convert transition matrix to an igraph object*

---

**Description**

Function to convert a transition matrix to an igraph object

**Usage**

```
matrix2graph(trans, SOM, SOM.hc, col.set, diag = FALSE)
```

**Arguments**

<code>trans</code>	a transition matrix (usually obtained from <code>comp.trans.mat</code> )
<code>SOM</code>	a kohonen object that form the network
<code>SOM.hc</code>	a vector of cluster assignment for SOM neurons
<code>col.set</code>	a vector of colors used for the SOM clusters
<code>diag</code>	boolean condition to include diagonal elements

**Value**

An igraph object, with SOM properties annotated

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Divide the SOM in the selected number of clusters
som_cl <- cutree(hclust(dist(som_model$codes[[1]]), method="euclidean"), method="complete"), 4)
#Compute transition matrix
tr_mat <- comp.trans.mat(som_model, start = 1)
#Define a set of colors
colors <- c("#1f78b4", "#33a02c", "#e31a1c", "#ffff88", "#6a3d9a")
#Create graph object
net <- matrix2graph(tr_mat, som_model, som_cl, colors, diag=FALSE)
```

---

native.cont

*Select native contact distances*

---

**Description**

Function to select only distances between residues making contacts in reference file or a frame of the simulation

**Usage**

```
native.cont(
  struct = NULL,
  trj = NULL,
  trj.frame = 1,
  distance,
  mol.2 = FALSE,
  atoms = NULL
)
```

**Arguments**

struct	a struct object read with read.struct() to compute the native.cont
trj	a trj object to compute the native.cont
trj.frame	The frame of the trj on which the native.cont are computed
distance	the distance cut-off

mol.2	can be FALSE (default), use the whole distance matrix, or a vector containing the atomic number of the second molecule (and compute only intermolecular distances)
atoms	can be NULL (default), consider all the atoms present in coords, or a vector containing a set of atomic numbers to consider in the calculation (e.g. only CB). atoms can be obtained with the bio3d atom.select function

**Value**

A vector containing the index of a subset of selected distances

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
# Read reference structure file with native conformation
struct <- read.struct(system.file("extdata", "HIF2a.gro", package = "SOMMD"))
#Select only Cbeta atoms to perform the analysis
sele_atoms <- which(struct$atom$seley=="CB")
#Choose only native contacts
sele_dists <- native.cont(struct=struct, distance=1.0, atoms=sele_atoms)
```

---

neur.population	<i>Get Neuron Population</i>
-----------------	------------------------------

---

**Description**

Function to compute the per-neuron population

**Usage**

```
neur.population(SOM, start = 1, end = length(SOM$unit.classif), N = 1)
```

**Arguments**

SOM	the SOM object
start	a vector containing the start frames of each replica (usually contained in trj\$start if replicas were merged with cat_trj)
end	a vector containing the end frames of each replica (usually contained in trj\$end if replicas were merged with cat_trj)
N	An integer for the portion (replica) of the simulations to be plotted

**Value**

A vector containing the per-neuron population

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
pop <- neur.population(som_model)
```

---

neur.representatives    *Neuron representative*

---

**Description**

Compute the representative frame of each neuron (the closest to the neuron codebook)

**Usage**

```
neur.representatives(SOM)
```

**Arguments**

SOM                    a kohonen SOM object.

**Value**

A vector containing the index of the representative frames for each neuron

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Compute representative frame for each neuron
neuron_representatives <- neur.representatives(som_model)
```

---

print.struct	<i>print.struct</i>
--------------	---------------------

---

**Description**

A short description...

**Usage**

```
## S3 method for class 'struct'  
print(x, ...)
```

**Arguments**

x	trj object
...	additional arguments to be passed to further methods

**Value**

Called for its effect.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
# Read structure file  
struct <- read.struct(system.file("extdata", "HIF2a.gro", package = "SOMMD"))  
#Print basic information  
print(struct)
```

---

print.trj	<i>Print Trajectory</i>
-----------	-------------------------

---

**Description**

A short description...

**Usage**

```
## S3 method for class 'trj'  
print(x, ...)
```

**Arguments**

x                   trj object  
...                 additional arguments to be passed to further methods

**Value**

Called for its effect.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
#Print basic informations
print(trj)
```

---

read.gro                    *Read gro file*

---

**Description**

Function to read gro files

**Usage**

```
read.gro(file)
```

**Arguments**

file                   contains the name and the path to the gro file to be read

**Value**

Returns a list of class "gro" with the following components:

atom                   a data frame containing all atomic coordinate with a row per atom and a column per record type.  
xyz                    a numeric matrix of class "xyz" containing the atomic coordinate data.  
box                    a vector of box size.  
call                   the matched call.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

---

read.struct	<i>Read structure files</i>
-------------	-----------------------------

---

### Description

Function to read pdb and gro files

### Usage

```
read.struct(file)
```

### Arguments

file                    contains the name and the path to the pdb or gro file to be read

### Value

Returns a list of class "struct" with the following components:

atom	a data frame containing all atomic coordinate with a row per atom and a column per record type.
xyz	a numeric matrix of class "xyz" containing the atomic coordinate data.
box	a vector of box size.
format	The format of the original file
call	the matched call.

### Author(s)

Stefano Motta <stefano.motta@unimib.it>

### Examples

```
# Read structure file
struct <- read.struct(system.file("extdata", "HIF2a.gro", package = "SOMMD"))
```



---

read.trj	<i>Read trj file</i>
----------	----------------------

---

**Description**

Function to read a trajectory file

**Usage**

```
read.trj(trjfile, topfile)
```

**Arguments**

trjfile	contains the name and the path to the reference file (pdb or gro files are accepted)
topfile	contains the name and the path to the trajectory file (xtc or dcd files are accepted)

**Value**

Returns a list of class "trj" with the following components:

topfile	the input topology file.
topformat	the format of the input topology.
trjfile	the input trajectory file.
trjformat	the format of the input trajectory.
coord	a three dimensional array containing atomic coordinates for all the frames. Dimensions are: Natoms:3:Nframes.
top	a data.frame containing topological informations with a row per atom and a column per record type (resno, resid, eley, eleno, chain).
start	a vector with the first frame of the simulation. When multiple simulations are concatenated with <code>cat.trj</code> the vector indicates the first frame of each simulation.
end	a vector with the last frame of the simulation. When multiple simulations are concatenated with <code>cat.trj</code> the vector indicates the last frame of each simulation.
call	the matched call.

**Author(s)**

Alessandro Pandini

**Examples**

```
#Read trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
```

---

remap.data	<i>map data to existing SOM</i>
------------	---------------------------------

---

## Description

Assign new data to a pre-trained SOM

## Usage

```
remap.data(SOM, X, add = FALSE)
```

## Arguments

SOM	a trained SOM
X	a data set with the same number of features of the dataset used to train the SOM
add	whether to append the new data to the ones used to train the SOM

## Value

An object of class "kohonen" with the new data mapped

## Author(s)

Stefano Motta <stefano.motta@unimib.it>

## Examples

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Read a trajectory that was not used to train the som
trj_2 <- read.trj(trjfile = system.file("extdata", "HIF2a-MD-2.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
#Read reference structure file
gro <- read.struct(system.file("extdata", "HIF2a.gro", package = "SOMMD"))
#Selection of the same intermolecular distances used to train the SOM
protein.sele <- which(gro$atom$resid!="020")
ligand.sele <- which(gro$atom$resid=="020")
heavy.atoms <- which(startsWith(gro$atom$eley, "H")==FALSE)
sele.dists <- native.cont(struct=gro, distance=0.6, mol.2=ligand.sele, atoms=heavy.atoms)
# Compute distances on new simulations (the same used for SOM training)
dist_2 <- calc.distances(trj_2, mol.2=ligand.sele, sele=sele.dists, atoms=heavy.atoms)
# Map new data on the existing SOM
som_model_2 <- remap.data(SOM=som_model, X=dist_2)
```

---

rio_read_xtc	<i>Read xtc trajectory file</i>
--------------	---------------------------------

---

**Description**

Function to read an xtc trajectory file

**Usage**

```
rio_read_xtc(xtc_filename)
```

**Arguments**

xtc\_filename contains the name and the path to the xtc file

**Value**

Returns 3D array of cartesian coordinates

**Author(s)**

Alessandro Pandini

---

rio_read_xtc2xyz	<i>Read xtc trajectory file</i>
------------------	---------------------------------

---

**Description**

Function to read an xtc trajectory file

**Usage**

```
rio_read_xtc2xyz(xtc_filename)
```

**Arguments**

xtc\_filename contains the name and the path to the xtc file

**Value**

Returns bio3d xyz array of Cartesian coordinates

**Author(s)**

Alessandro Pandini

---

rio\_read\_xtc\_natoms    *Read xtc trajectory file*

---

**Description**

Function to read a xtc trajectory file

**Usage**

```
rio_read_xtc_natoms(xtc_filename)
```

**Arguments**

xtc\_filename    contains the name and the path to the xtc file

**Value**

Returns number of atoms in the structure

**Author(s)**

Alessandro Pandini

---

rio\_read\_xtc\_nframes    *Read xtc trajectory file*

---

**Description**

Function to read an xtc trajectory file

**Usage**

```
rio_read_xtc_nframes(xtc_filename)
```

**Arguments**

xtc\_filename    contains the name and the path to the xtc file

**Value**

Returns number of frames in the trajectory

**Author(s)**

Alessandro Pandini

---

rio_write_xtc	<i>Write xtc trajectory file</i>
---------------	----------------------------------

---

**Description**

Function to write an xtc trajectory file

**Usage**

```
rio_write_xtc(xtc_filename, trj)
```

**Arguments**

xtc_filename	contains the name and the path to the xtc file to write
trj	trajectory object to save

**Value**

Returns status of write execution

**Author(s)**

Alessandro Pandini

---

silhouette.profile	<i>Silhouette profile</i>
--------------------	---------------------------

---

**Description**

Function to compute the silhouette profile for the Nclus cluster of the SOM neurons

**Usage**

```
silhouette.profile(  
  SOM,  
  Nclus,  
  dist_clust = "euclidean",  
  clust_method = "complete"  
)
```

**Arguments**

SOM	the SOM object to cluster
Nclus	the cluster number on which the silhouette profile will be computed
dist_clust	the metric for the distance calculation
clust_method	the method for the clustering (passed to the hclust function)

**Value**

A vector of silhouette profile computed with the cluster package

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Compute the silhouette profile
sil_pro <- silhouette.profile(som_model, Nclus=5, clust_method="complete")
```

---

silhouette.score	<i>Silhouette score</i>
------------------	-------------------------

---

**Description**

Function to compute the silhouette score for the clustering of SOM neurons

**Usage**

```
silhouette.score(  
  SOM,  
  dist_clust = "euclidean",  
  clust_method = "complete",  
  interval = seq(2, 30)  
)
```

**Arguments**

SOM	the SOM object to cluster
dist_clust	the metric for the distance calculation
clust_method	the method for the clustering (passed to the hclust function)
interval	the cluster number on which the silhouette score will be computed

**Value**

A vector with the silhouette scores for all the frames

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

## Examples

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Compute the silhouette profile
sil_score <- silhouette.score(som_model, clust_method="complete", interval=seq(2,8))
```

---

som.add.circles	<i>Add circles to SOM</i>
-----------------	---------------------------

---

## Description

Function to add circles to a SOM plot, with dimension proportional to a selected property

## Usage

```
som.add.circles(SOM, P, scale = 1, col.circles = "white")
```

## Arguments

SOM	the SOM object
P	a vector containing the per-neuron property to plot
scale	a number to scale up or down the size of the drawn circles
col.circles	the background color of the drawn circles

## Value

Called for its effect.

## Author(s)

Stefano Motta <stefano.motta@unimib.it>

## Examples

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
# Compute per neuron population
pop <- neur.population(som_model)
#Plot the som
plot(som_model, type = "count", bgcol=c("red", "blue", "yellow", "green"), shape='straight')
# Add circles to the SOM plot
som.add.circles(som_model, pop, scale=0.9)
```

---

`som.add.clusters.legend`*Add legend clusters*

---

**Description**

Function to apply a legend of clusters to a SOM map image

**Usage**

```
som.add.clusters.legend(Nclus, color.scale)
```

**Arguments**

<code>Nclus</code>	the number of clusters to which put the legend
<code>color.scale</code>	the color scale used for the image

**Value**

Called for its effect.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Divide the SOM in the selected number of clusters
som_cl <- cutree(hclust(dist(som_model$codes[[1]]), method="euclidean"), method="complete", 4)
#Define a set of colors
colors <- c("#1f78b4", "#33a02c", "#e31a1c", "#ffff88", "#6a3d9a")
#Plot the som with neurons colored according to clusters
plot(som_model, type = "mapping", bgcol=colors[som_cl], col=rgb(0,0,0,0), shape='straight', main="")
kohonen::add.cluster.boundaries(som_model, som_cl, lwd=5)
#Add legend to the plot
som.add.clusters.legend(Nclus=4, color.scale=colors)
```



---

som.add.numbers	<i>Add Neuron Numbering</i>
-----------------	-----------------------------

---

**Description**

Add the neuron numbering scheme to the SOM plot

**Usage**

```
som.add.numbers(SOM, scale = 1, col = "black")
```

**Arguments**

SOM	the SOM object
scale	a number to scale up or down the size of the text
col	the color of the text

**Value**

Called for its effect.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#Plot the som
plot(som_model, type = "count", bgcol=c("red", "blue", "yellow", "green"), shape='straight')
#Add neuron numbers on the som
som.add.numbers(som_model, scale=0.5, col="black")
```

---

stride.trj	<i>Stride a trj</i>
------------	---------------------

---

**Description**

Apply a stride to the frame of a trj object to reduce the number of frames

**Usage**

```
stride.trj(trj, stride)
```

**Arguments**

trj                a trj object.  
stride            the stride to apply to the trajectory

**Value**

An object of class trj with a frame every stride

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
# Read the simulation
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
# keep a frame every 2 frame
trj_strd <- stride.trj(trj, 2)
```

---

struct2pdb

*Convert structure to pdb object*

---

**Description**

Convert a struct object into a pdb object

**Usage**

```
struct2pdb(struct)
```

**Arguments**

struct            contains the struct object to convert

**Value**

Returns an object with class "pdb"

An object of class "pdb"

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
# Read structure file
struct <- read.struct(system.file("extdata", "HIF2a.gro", package = "SOMMD"))
#Convert structure to pdb object
pdb <- struct2pdb(struct)
```

---

summary.struct	<i>Summarizing a structure object</i>
----------------	---------------------------------------

---

**Description**

summary method for class struct

**Usage**

```
## S3 method for class 'struct'
summary(object, ...)
```

**Arguments**

object	struct object
...	additional arguments to be passed to further methods

**Value**

Called for its effect.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

---

summary.trj	<i>Summarizing a trajectory object</i>
-------------	--

---

**Description**

summary method for class trj

**Usage**

```
## S3 method for class 'trj'
summary(object, ...)
```

**Arguments**

object            trajectory object  
 ...                additional arguments to be passed to further methods

**Value**

Called for its effect.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

---

trace.path	<i>Trace pathway</i>
------------	----------------------

---

**Description**

Function trace pathway sampled on the SOM

**Usage**

```
trace.path(
  SOM,
  start = 1,
  end = length(SOM$unit.classif),
  N = 1,
  draw.stride = 1,
  pts.scale = 1,
  lwd.scale = 1
)
```

**Arguments**

SOM                the SOM object  
 start              a vector containing the start frames of each replica (usually contained in trj\$start if replicas were merged with cat\_trj)  
 end                a vector containing the end frames of each replica (usually contained in trj\$end if replicas were merged with cat\_trj)  
 N                  The portion of simulation that one want to plot  
 draw.stride        used to plot the pathways with a stride (useful for very complex pathways)  
 pts.scale          a number to scale up or down the size of the circles  
 lwd.scale          a number to scale up or down the size of the lines

**Value**

Called for its effect.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
# Read the trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
#Read example SOM data
som_model <- readRDS(system.file("extdata", "SOM_HIFa.rds", package = "SOMMD"))
#trace pathway sampled on the SOM
trace.path(som_model, start=trj$start, end=trj$end, N=1, pts.scale=0.5)
```

---

trj2pdb

*Extract frame to pdb*

---

**Description**

Extract a trj frame to a pdb object

**Usage**

```
trj2pdb(trj, frame, filename)
```

**Arguments**

trj	a trj object.
frame	the frame to extract.
filename	for the output pdb file

**Value**

a pdb object of the selected frame  
Called for its effect.

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
# Read the trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
# Write the pdb file for a specific frame
trj2pdb(trj = trj, frame=5, filename = tempfile(fileext = '.pdb' ))
```

---

`trj2xyz`*Convert Trajectory to xyz*

---

**Description**

Convert the trj coordinates 3D-array in a 2D matrix.

**Usage**

```
trj2xyz(trj, inds = NULL)
```

**Arguments**

<code>trj</code>	an object with class trj
<code>inds</code>	indices for the output coordinates

**Value**

a xyz matrix with frames on rows and coordinates as columns

**Author(s)**

Stefano Motta <stefano.motta@unimib.it>

**Examples**

```
#Read trajectory
trj <- read.trj(trjfile = system.file("extdata", "HIF2a-MD.xtc", package = "SOMMD"),
  topfile = system.file("extdata", "HIF2a.gro", package = "SOMMD"))
trj2xyz(trj)
```

# Index

average.neur.property, 3

calc.distances, 3  
calc.dists, 4  
cat.trj, 5  
cluster.pathways, 6  
cluster.representatives, 7  
comp.trans.mat, 8

fit.trj, 9

map.color, 10  
matrix2graph, 10

native.cont, 11  
neur.population, 12  
neur.representatives, 13

print.struct, 14  
print.trj, 14

read.gro, 15  
read.struct, 16  
read.trj, 17  
remap.data, 18  
rio\_read\_xtc, 19  
rio\_read\_xtc2xyz, 19  
rio\_read\_xtc\_natoms, 20  
rio\_read\_xtc\_nframes, 20  
rio\_write\_xtc, 21

silhouette.profile, 21  
silhouette.score, 22  
som.add.circles, 23  
som.add.clusters.legend, 24  
som.add.numbers, 25  
stride.trj, 25  
struct2pdb, 26  
summary.struct, 27  
summary.trj, 27

trace.path, 28  
trj2pdb, 29  
trj2xyz, 30