

# Package ‘FreeSortR’

December 17, 2017

**Type** Package

**Title** Free Sorting Data Analysis

**Version** 1.3

**Date** 2017-12-15

**Author** Philippe Courcoux

**Maintainer** Philippe Courcoux <philippe.courcoux@oniris-nantes.fr>

**Description** Provides tools for describing and analysing free sorting data. Main methods are computation of consensus partition and factorial analysis of the dissimilarity matrix between stimuli (using multidimensional scaling approach).

**License** GPL-2

**Depends** R (>= 3.1), methods, smacof, vegan, ellipse

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-12-17 20:26:27 UTC

## R topics documented:

FreeSortR-package . . . . .	2
AromaSort . . . . .	3
AromaTerms . . . . .	4
ConsensusPartition . . . . .	4
Cooccurrences . . . . .	6
DescriptionPartition . . . . .	6
Dissimil . . . . .	7
DissTot . . . . .	8
getConfig . . . . .	8
getPartition . . . . .	9
getPercent . . . . .	9
getStress . . . . .	10
MdsDimChoice . . . . .	11
MdsDiss . . . . .	11
MdsSort . . . . .	12

nGroups . . . . .	13
plotMds . . . . .	14
plotTerms . . . . .	14
RandIndex . . . . .	15
ReadSortFile . . . . .	16
SortingMds-class . . . . .	17
SortingPartition . . . . .	18
SortingPartition-class . . . . .	19

<b>Index</b>	<b>20</b>
--------------	-----------

---

FreeSortR-package	<i>Analysis of free sorting data.</i>
-------------------	---------------------------------------

---

## Description

This package gives several tools for analysing free sorting data.

## Details

Package:	FreeSortR
Type:	Package
Version:	1.0
Date:	2014-04-29
License:	GPL
Depends:	methods, smacof

The function for managing sorting data is `SortingPartition()`. Function for computing consensus partition is `ConsensusPartition()`. Multidimensional scaling of sorting data may be performed with the function `MdsSort()`.

## Author(s)

Philippe Courcoux

Maintainer: <philippe.courcoux@oniris-nantes.fr>

## References

Ph. Courcoux, P. Faye, E.M. Qannari (2014) Determination of the consensus partition and cluster analysis of subjects in a free sorting task experiment. *Food Quality and Preference*, 32, 107-112.

## See Also

[SortingPartition](#), [ConsensusPartition](#), [MdsSort](#)

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
res<-ConsensusPartition(Aroma)
DescriptionPartition(res$Consensus)
resMds<-MdsSort(Aroma, ndim=3)
plotMds(resMds)
```

---

AromaSort

*Aroma sorting data*

---

**Description**

Partitions of 16 aromas by 31 subjects (free sorting task)

**Usage**

```
data(AromaSort)
```

**Format**

A data frame with 16 observations (aromas) and 31 variables (subjects).

**Details**

List of stimuli : Lemon, Grapefruit, Pineapple, Pear, Honey, Butter, Grilledbread, Grilledhazelnut, Strawberry, Raspberry, Cherry, Blackcurrant, Greenpepper, Smoked, Pepper, Licorice.

**References**

Ph. Courcoux, P. Faye, E.M. Qannari (2014) Determination of the consensus partition and cluster analysis of subjects in a free sorting task experiment. Food Quality and Preference 32, 107-112

**Examples**

```
data(AromaSort)
```

AromaTerms

*Aroma data (verbalisation by 31 subjects)*

---

**Description**

Free sorting of 16 aromas described by 31 subjects. Data are occurrences of terms for describing stimuli. Partitions given by the subjects are described in the AromaSort data.

**Usage**

```
data(AromaTerms)
```

**Format**

A data frame with 16 observations (aromas) and 36 variables (terms). Rownames and colnames refer to stimuli and terms labels.

**Details**

List of terms : Acid, Smoked ,Heady, Citrus, Lemon, Cake, Milk, Woody, Grain, Low, Redfruit, Grilled, Strong, Fat, Vegetal, Medicine, Chemical, Licorice, Bread, Alcohol, Almond, Caramel, Coal, Unpleasant, Soft, Pepper, Flower, Fresh, Red, Fruit, Natural, Spicy, Sugar, Hot, Pleasant, Candy.

**Examples**

```
data(AromaTerms)
```

---

ConsensusPartition

*Consensus of Partitions*

---

**Description**

Returns the consensus partition among a set of partitions

**Usage**

```
ConsensusPartition(Part, ngroups = 0, type = "cutree", optim = FALSE,  
maxiter = 100, plotDendrogram = FALSE, verbose = FALSE)
```

**Arguments**

Part	Object of class <code>SortingPartition</code>
ngroups	Number of groups of the consensus (or <code>ngroups=0</code> for optimal choice)
type	Method ( <code>type="cutree"</code> or <code>type="fusion"</code> or <code>type="medoid"</code> )
optim	Optimisation of the consensus (default is <code>optim=FALSE</code> )
maxiter	Maximum number of iterations for fusion algorithm
plotDendrogram	Plot of the dendrogram (if <code>type="cutree"</code> initialisation)
verbose	Print the initialisation results

**Details**

The criterion for optimal consensus is the mean adjusted Rand Index between the consensus and the partitions given by the subjects.

If `ngroups=0`, consensus is computed between 2 and `nstimuli-1` and the best consensus is returned.

For `type="cutree"`, the initialisation step is based on cutting the tree generated by clustering the stimuli. For `type="fusion"`, the initialisation step is based on the fusion algorithm. In this case, results are more accurate but the algorithm might be time consuming. For `type="medoid"`, the consensus is the closest partition to all the partitions given by subjects.

For `optim=TRUE`, a transfer step is performed after the initialisation step.

**Value**

List of following components:

Consensus	Consensus
Crit	Criterion for consensus

**References**

Krieger & Green (1999) *J. of Classification*, 16:63-89

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
res<-ConsensusPartition(Aroma,ngroups=0,type="cutree")
res
##res<-ConsensusPartition(Aroma,ngroups=0,type="fusion",optim=TRUE)
##res
##res<-ConsensusPartition(Aroma,type="medoid")
##res
```

---

Cooccurrences	<i>Cooccurrences</i>
---------------	----------------------

---

**Description**

Returns the matrix of cooccurrences between stimuli.

**Usage**

```
Cooccurrences(Part)
```

**Arguments**

Part	Object of class <code>SortingPartition</code>
------	---

**Details**

Returns the matrix of cooccurrences between stimuli (number of times two stimuli have been sorted in the same group).

**Value**

A matrix of cooccurrences (stimuli x stimuli).

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
MatCooc<-Cooccurrences(Aroma)
```

---

DescriptionPartition	<i>Description of a partition</i>
----------------------	-----------------------------------

---

**Description**

DescriptionPartition() shows a partition given by a subject.

**Usage**

```
DescriptionPartition(Part, subject = 1, replicate = 1, Labels=NULL)
```

**Arguments**

Part	Object of class <code>SortingPartition</code> or vector giving a partition
subject	Subject identifier (number or label of a subject)
replicate	Number of the replicate to show (in the case of multiple partitions)
Labels	Labels of the stimuli

**Value**

Returns the partition with labels of stimuli bracketted in groups.

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
DescriptionPartition(Aroma,subject=1)
```

---

Dissimil

*Dissimilarities between stimuli*

---

**Description**

Creates a list of dissimilarity matrices from partitions given by the subjects.

**Usage**

```
Dissimil(Part)
```

**Arguments**

Part                    Object of class `SortingPartition`

**Details**

In the case of free sorting data, a list of dissimilarity matrices (the length of the list is equal to the number of subjects).

In the case of multiple sorting, dissimilarity matrix for a subject is the sum of the dissimilarity matrices computed from each of the different partitions given by this subject.

**Value**

A list of dissimilarity matrices (one matrix for each subject).

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
ListDiss<-Dissimil(Aroma)
```

---

DissTot	<i>Overall Dissimilarities between stimuli.</i>
---------	---

---

**Description**

Creates the matrix of dissimilarities between stimuli.

**Usage**

```
DissTot(Part)
```

**Arguments**

Part                    Object of class SortingPartition

**Value**

The matrix of dissimilarities between stimuli (number of times that two stimuli have not been grouped)

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
DisTot<-DissTot(Aroma)
```

---

getConfig	<i>Gets the Mds configuration.</i>
-----------	------------------------------------

---

**Description**

Gets the Mds config resulting from the function MdsSort().

**Usage**

```
getConfig(object)
```

**Arguments**

object                    An object of class SortingMds

**Value**

An array of Mds configuration



**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
resMds<-MdsSort(Aroma,ndim=2)
config<-getConfig(resMds)
```

---

getPartition                    *Gets the partitions.*

---

**Description**

Returns an array of the partitions given by the subjects.

**Usage**

```
getPartition(object)
```

**Arguments**

object                    An object of class SortingPartition

**Value**

An array of the stimuli as rows and the partitions as columns.

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
part<-getPartition(Aroma)
```

---

getPercent                    *Gets the percentages of variance*

---

**Description**

Returns the percentage of variance explained by the dimensions of a Mds solution returned by the function MdsSort().

**Usage**

```
getPercent(object)
```

**Arguments**

object                    An object of class SortingMds

**Value**

A vector of percentage of variance of the Mds configuration.

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
resMds<-MdsSort(Aroma, ndim=2)
perc<-getPercent(resMds)
```

---

getStress

*Gets the stress value*

---

**Description**

Get the Kruskal stress value of the Mds solution returned by the function `MdsSort()`.

**Usage**

```
getStress(object)
```

**Arguments**

object            An object of class `SortingMds`

**Value**

A numeric value of stress.

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
resMds<-MdsSort(Aroma, ndim=2)
stress<-getStress(resMds)
```

---

MdsDimChoice	<i>Computation of the stress of Mds solution</i>
--------------	--

---

**Description**

MdsDimChoice() returns a table of stress values of Multidimensionnal scaling for different dimensions. The different dimensions to test are given as an argument of the function.

The Mds is based on smacof algorithm and may be metric or not metric.

**Usage**

```
MdsDimChoice(Part, dimen = c(2, 4), metric = FALSE,
              ties = "primary", itmax = 5000, eps = 1e-06)
```

**Arguments**

Part	Part is an object of class SortingPartition
dimen	Vector of (minimum and maximum of) dimensions for Mds (default is dimen=c(2, 4))
metric	Metric or non metric Mds (default is metric=FALSE for non metric Mds)
ties	Treatment of ties in case of non metric Mds
itmax	Maximum number of iterations
eps	Epsilon for Mds computation

**Value**

Table of Kruskal stress for the chosen dimensions

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
tabres<-MdsDimChoice(Aroma)
```

---

MdsDiss	<i>Mds of a dissimilarity matrix</i>
---------	--------------------------------------

---

**Description**

Computes the multidimensional scaling of a matrix of dissimilarities between stimuli. Mds is based on smacof algorithm. The Mds configuration is rotated in order to get orthogonal dimensions sorted by decreasing variance.

**Usage**

```
MdsDiss(MatDissimil, ndim = 2, metric = TRUE, ties = "primary",
        itmax = 5000, eps = 1e-06)
```

**Arguments**

MatDissimil	A matrix of dissimilarities
ndim	Dimension of the Mds
metric	Metric or not metric Mds
ties	Treatment of ties in case of non metric Mds
itmax	Maximum number of iterations
eps	Epsilon for Mds computation

**Value**

List of the following components :

Config	Mds configuration of the stimuli
Percent	Percentage of inertia of the dimensions of Mds
Stress	Stress of the Mds solution

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
ListDissimil<-Dissimil(Aroma)
MatDissim<-apply(simplify2array(ListDissimil),c(1,2),'sum')
Mdsres<-MdsDiss(MatDissim)
```

---

MdsSort

*Mds of sorting data*


---

**Description**

MdsSort returns the results of the multidimensional scaling of a list of dissimilarities. The Mds is based on smacof algorithm and may be metric or not metric.

Botstrap on subjects allows to draw confidence regions for the stimuli.

**Usage**

```
MdsSort(Part,ndim=2,nboot=0,metric=FALSE,ties="primary",itmax=5000,eps=1e-06)
```

**Arguments**

Part	Part is an object of class <code>SortingPartition</code>
ndim	Dimension of the Mds (default is <code>ndim=2</code> )
nboot	Number of bootstrap samples (default is <code>nboot=0</code> for no bootstrap analysis)
metric	Metric or non metric Mds (default is <code>metric=FALSE</code> for non metric)
ties	Treatment of ties in case of non metric Mds
itmax	Number maximum of iterations
eps	Epsilon for Mds computation

**Value**

An object of class `SortingMds`

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
resMds<-MdsSort(Aroma,ndim=2)
```

---

nGroups	<i>Number of groups given by the subjects</i>
---------	---

---

**Description**

Returns the number of groups given by the subjects of a free sorting experiment.

**Usage**

```
nGroups(object)
```

**Arguments**

object	An object of class <code>SortingPartition</code>
--------	--

**Value**

A vector giving the number of groups made by the subjects

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
nGroups(Aroma)
```

---

plotMds *Plot of the configuration of Mds*

---

### Description

plotMds returns a plot of the configuration resulting from a Multidimensional scaling. Confidence ellipsoids are plotted if a bootstrap approach has been used in the MdsSort() step.

### Usage

```
plotMds(ResMds, dim=c(1,2), ellipse=FALSE, proba=0.90, col=NULL)
```

### Arguments

ResMds	ResMds is an object of class SortingMds
dim	Vector of dimensions to be plotted (default is dim=c(1,2))
ellipse	Indicates if ellipsoids have to be plotted (default if ellipse=FALSE)
proba	Probability for plotting ellipses (default is proba=.90)
col	The color to be used for the text, possibly vectors

### Value

plot of Mds configuration

### Examples

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
resMds<-MdsSort(Aroma,ndim=2)
plotMds(resMds)
```

---

plotTerms *Plot of the terms used during verbalisation step*

---

### Description

plotTerms() produces a plot of the terms. The rows of array MatTerms are the stimuli and the columns are the terms.

### Usage

```
plotTerms(MatTerms,ResMds,dim=c(1,2),type="correl",add=TRUE)
```

**Arguments**

MatTerms	Array of occurrences of terms
ResMds	Object of class <code>SortingMds</code>
dim	Vector of dimensions to be plotted (default is <code>dim=c(1,2)</code> )
type	Indicates the type of plotting (default is <code>type="correl"</code> for correlations)
add	Indicates if the stimuli are added to the plot (if <code>type="baryc"</code> )

**Details**

If `type="correl"`, the correlations between occurrences of terms and dimensions of a Mds configuration are plotted. `plotTerms()` returns the correlation matrix.

If `type="baryc"`, a barycentric representation of terms is used. If `add=TRUE`, the stimuli are added to this plot. `plotTerms()` returns the coordinates of terms.

**Value**

returns a matrix of correlation or a configuration of terms (depending on `type`).

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
resMds<-MdsSort(Aroma,ndim=2)
data(AromaTerms)
plotTerms(AromaTerms,resMds)
```

---

 RandIndex

*Rand Index between partitions*


---

**Description**

Computes the Rand Index and the Adjusted Rand Index between two partitions

**Usage**

```
RandIndex(Partition1, Partition2)
```

**Arguments**

Partition1	Vector describing the first partition
Partition2	Vector describing the second partition

**Details**

Supports incomplete partitions (value 0 is coding for missing stimulus)

**Value**

List of following components:

Rand	Rand Index between the partitions
AdjustedRand	Adjusted rand Index between the partitions

**References**

Rand (1971) *Jasa*, 66, 846-850  
 Hubert & Arabie (1985) *J. of Classification*, 2, 193-218

**Examples**

```
Partition1<-c(1,1,1,2,2,2)
Partition2<-c(1,1,2,2,2,3)
r<-RandIndex(Partition1,Partition2)
r
# $Rand
# [1] 0.6
# $AdjustedRand
# [1] 0.1176471
```

---

ReadSortFile	<i>Read a file of free sorting data</i>
--------------	---

---

**Description**

The function (ReadSortFile()) reads a csv file of free sorting data.

The file contains stimuli as rows and subjects as columns. For a subject, two stimuli in the same group are coded by the same symbol. First row contains the labels of subjects and first column contains the labels of stimuli.

For each subject, the coding of a group may be a number or a list of terms describing the group (terms have to be separated by a symbol, as a comma).

Returns a matrix of sorting and, if this is adequate, matrices describing the use of terms by subjects.

**Usage**

```
ReadSortFile(filename, terms=FALSE, sefterms=",", sep=";", dec=".")
```

**Arguments**

filename	File name (and address if necessary)
terms	Boolean indicating if groups are coded by terms (terms=TRUE) or numbers (default terms=FALSE)
sefterms	Symbol for separating terms (default is sefterms=",") if terms=TRUE
sep	Symbol for separating data in the csv file (default is sep=";")
dec	Decimal separator (default is dec=".")



**Value**

List of the following components :

MatSort	Matrix of sorting groups (may be used by the function <code>SortingPartition</code> )
MatTerms	Matrix of occurrences of the terms used by the subjects
Stress	List of terms used by each subject

**Examples**

```
# dat<-ReadSortFile ("FSdata.csv")
# Sort<-SortingPartition(dat$MatSort)
```

---

SortingMds-class	<i>Class</i> SortingMds
------------------	-------------------------

---

**Description**

A class for Mds results

**Objects from the Class**

Objects are created by the function `MdsSort()`.

**Slots**

**nstimuli:** Number of stimuli  
**nsubjects:** Number of subjects  
**LabStim:** Labels of stimuli  
**LabSubj:** Labels of subjects  
**ndim:** Dimension of the Mds  
**Config:** Array of the configuration of stimuli  
**Percent:** Vector of inertia of the dimensions  
**Stress:** Kruskal stress of the configuration  
**ResBoot:** (optional) Results of bootstrap on the subjects

**Methods**

**getConfig** signature(object = "SortingMds")  
**getPercent** signature(object = "SortingMds")  
**getStress** signature(object = "SortingMds")  
**show** show(object = "SortingMds")  
**summary** summary(object = "SortingMds")

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
resMds<-MdsSort(Aroma,ndim=2,metric=FALSE)
summary(resMds)
```

---

SortingPartition	<i>Creates an object of class SortingPartition</i>
------------------	--

---

**Description**

Returns an object of class `SortingPartition` from an array containing the partitions.

The array has stimuli as rows and subjects as columns. For a subject, two stimuli in the same group are coded by the same number.

**Usage**

```
SortingPartition(DataSort)
```

**Arguments**

`DataSort`      A dataframe containing the partitions of the subjects

**Details**

The first row contains the labels of subjects and the first column contains the labels of stimuli.

In the case of multiple sorting task, the different partitions given by the same subject are in columns sharing the same name (but with different subnames: A, A.1, A.2...) in the array.

A value of 0 indicates that the subject did not sort the given stimulus (in case of incomplete design).

**Value**

An object of class `SortingPartition`.

**Examples**

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
show(Aroma)
```

---

SortingPartition-class

*Class* SortingPartition

---

### Description

A class for free sorting data

### Objects from the Class

Objects can be created from an array by calls of the form `SortingPartition()`.

### Slots

**type:** Type of sorting procedure : type="Free" or type="Multiple"

**nstimuli:** Number of stimuli

**nsubjects:** Number of subjects

**LabStim:** Labels of stimuli

**LabSubj:** Labels of subjects

**Partition:** List of partitions of the stimuli given by subjects

### Methods

**show** `show(object = "SortingPartition")`

**summary** `summary(object = "SortingPartition")`

**getPartition** `getPartition(object = "SortingPartition")`

**nGroups** `nGroups(object = "SortingPartition")`

### Examples

```
data(AromaSort)
Aroma<-SortingPartition(AromaSort)
summary(Aroma)
show(Aroma)
getPartition(Aroma)
```

# Index

## \*Topic **package**

FreeSortR-package, 2

AromaSort, 3

AromaTerms, 4

ConsensusPartition, 2, 4

Cooccurrences, 6

DescriptionPartition, 6

Dissimil, 7

DissTot, 8

FreeSortR (FreeSortR-package), 2

FreeSortR-package, 2

getConfig, 8

getConfig, SortingMds-method  
(SortingMds-class), 17

getPartition, 9

getPartition, SortingPartition-method  
(SortingPartition-class), 19

getPercent, 9

getPercent, SortingMds-method  
(SortingMds-class), 17

getStress, 10

getStress, SortingMds-method  
(SortingMds-class), 17

MdsDimChoice, 11

MdsDiss, 11

MdsSort, 2, 12

nGroups, 13

nGroups, SortingPartition-method  
(SortingPartition-class), 19

plotMds, 14

plotTerms, 14

RandIndex, 15

ReadSortFile, 16

show, SortingMds-method  
(SortingMds-class), 17

show, SortingPartition-method  
(SortingPartition-class), 19

SortingMds-class, 17

SortingPartition, 2, 18

SortingPartition-class, 19

summary, SortingMds-method  
(SortingMds-class), 17

summary, SortingPartition-method  
(SortingPartition-class), 19