Package ‘coppeCosenzaR’

October 12, 2022

Title COPPE-Cosenza Fuzzy Hierarchy Model

Date 2017-10-28

Version 0.1.3

Description The program implements the COPPE-Cosenza Fuzzy Hierarchy Model. The model was based on the evaluation of local alternatives, representing regional potentialities, so as to fulfill demands of economic projects. After defining demand profiles in terms of their technological coefficients, the degree of importance of factors is defined so as to represent the productive activity. The method can detect a surplus of supply without the restriction of the distance of classical algebra, defining a hierarchy of location alternatives. In COPPE-Cosenza Model, the distance between factors is measured in terms of the difference between grades of memberships of the same factors belonging to two or more sets under comparison. The required factors are classified under the following linguistic variables: Critical (CR); Conditioning (C); Little Conditioning (LC); and Irrelevant (I). And the alternatives can assume the following linguistic variables: Excellent (Ex), Good (G), Regular (R), Weak (W), Empty (Em), Zero (Z) and Inexistent (In). The model also provides flexibility, allowing different aggregation rules to be performed and defined by the Decision Maker. Such feature is considered in this package, allowing the user to define other aggregation matrices, since it considers the same linguistic variables mentioned.

Depends R (>= 3.2.2)

Imports methods

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests testthat

License GPL-2 | file LICENSE

URL https://github.com/ptaranti/coppeCosenzaR

BugReports https://github.com/ptaranti/coppeCosenzaR/issues

Collate 'aggregation-matrix.R' 'aggregation-matrix-default.R' 'aggregation-matrix-membership-difference.R' 'factor.R' 'factors-of-interest.R' 'option-factor-availability.R' 'option-resources.R' 'option.R' 'option-portfolio.R' 'project-criterion.R' 'project-criteria.R' 'project.R' 'project-portfolio.R' 'coppe-cosenza.R' 'coppe-cosenzaR.R'

NeedsCompilation no

Author Pier Taranti [aut, cre],
Leonardo Pessoa [aut],
Carlos Cosenza [aut]

Maintainer Pier Taranti <ptaranti@gmail.com>

Repository CRAN

Date/Publication 2017-10-28 13:29:46 UTC

R topics documented:

`Aggregate` *Aggregate*

Description

S4 method do not validate entries, since it is not exported and the data is validated by the constructors. The validation here would be resource consuming.

Usage

```
Aggregate(aggregation.matrix, factor.evaluation, resource.evaluation,
  factor.is.specific, nrfactors)


  ## S4 method for signature
## 'Aggregation.matrix.default,character,character,logical,numeric'
Aggregate(aggregation.matrix,
  factor.evaluation, resource.evaluation, factor.is.specific, nrfactors)


  ## S4 method for signature
## 'Aggregation.matrix.membership.difference,
##   character,
##   character,
##   logical,
##   numeric'
Aggregate(aggregation.matrix,
  factor.evaluation, resource.evaluation, factor.is.specific, nrfactors)
```

Arguments

`aggregation.matrix`	aggregation.matrix
`factor.evaluation`	character factor evaluation from project

`resource.evaluation`	character factor evaluation from option
`factor.is.specific`	logic indicates that this factor is specific for the project
`nrfactors`	numeric number of factors evaluated for each project/option

Value

numeric indicate the result factor per option. If a specific factor is not achieved it returns -1

`AggregateMatrix` *AggregateMatrix*

Description

S4 method to perform Aggregation.Matrix inheirited objects. If a implementation is not provided to a specific aggregation matrix, this implementation will be used. So it allows using different aggregation matrices. Such feature provides flexibility regarding compensatory effects among the criteria. Therefore it can be adjusted to different multicriteria problems. It is important to highlight that the entries must be compliant to the original described categories, using the same linguistic variables present in the default aggregation matrix.

Usage

```
AggregateMatrix(aggregation.matrix, project.portfolio.as.data.frame,
  project.portfolio.specifics.as.data.frame, option.portfolio.as.data.frame)


  ## S4 method for signature
## 'Aggregation.matrix,data.frame,data.frame,data.frame'
AggregateMatrix(aggregation.matrix,
  project.portfolio.as.data.frame, project.portfolio.specifics.as.data.frame,
  option.portfolio.as.data.frame)
```

Arguments

`aggregation.matrix`	aggregation.matrix
`project.portfolio.as.data.frame`	project.portfolio.as.data.frame
`project.portfolio.specifics.as.data.frame`	project.portfolio.specifics.as.data.frame
`option.portfolio.as.data.frame`	option.portfolio.as.data.frame

Value

data.frame

data.frame

`Aggregation.matrix-class`

Aggregation.matrix S4 Class

Description

This class was included to act as an abstract class to be inherited by concrete classes that implement their matrix in constructors.

Slots

`name` character

`Aggregation.matrix.default-class`

Aggregation.matrix.default

Description

This class represents extends the Aggregation.matrix S4 and is the default aggregation matrix, that presents a zero value, when the option does not provide an adequate level of the required factor. In other words, if the option level is below the required one, the evaluation of the criteria for the studied option will be zero. Such matrix provides a low compensatory effect. Nevertheless for problems which allows greater compensatory effects, the package allows using different aggregation matrices.

See Also

Aggregation.matrix

`Aggregation.matrix.membership.difference-class`

Aggregation.matrix.membership.difference

Description

This class represents extends the Aggregation.matrix S4 and implements the Membership Difference aggregation matrix.

See Also

Aggregation.matrix

`as.data.frame` *as.data.frame*

Description

This S4 method masks the `base::as.data.frame()` S3 function. If a call uses parameters other then the expected by this package, then it will be forward to the S3 function.

Usage

```
as.data.frame(x, row.names, optional, ...)

## S4 method for signature 'Option.portfolio'
as.data.frame(x)

## S4 method for signature 'Project.portfolio'
as.data.frame(x, row.names = NA,
  optional = FALSE)
```

Arguments

`x`	Option.portfolio or Project.portfolio
`row.names`	not used. It is inherited from `base::as.data.frame()`
`optional`	logical. To be used with Project.portfolio. Indicates if the return is a data.frame with factor evaluations or with the information about which factors are specific to a project. The default is `optional` = `FALSE`
`...`	not used.

Value

data.frame

Examples

```
## Not run: as.data.frame(option.portfolio)

## Not run: as.data.frame(project.portfolio, option = TRUE)
## Not run: as.data.frame(project.portfolio,  , TRUE)
## Not run: as.data.frame(project.portfolio, ANY, FALSE)
## Not run: as.data.frame(project.portfolio, option = FALSE)
## Not run: as.data.frame(project.portfolio) This infer option is FALSE, too.
```

Coppe.cosenza *Coppe.cosenza*

Description

S4 method to construct Coppe.cosenza objects. The package also provides a way to verify the consistency of the entry data. There are 3 different matrices which are considered for the evaluation purposes: The project' s required factors; The project's description of specific factors; and the options' available level of factors. All the factors must be evaluated by each project and by each option. The program deconstruct each evaluation so as to verify: if all the factors are evaluated for each project; if all the factors are evaluated for each option, and besides, if all the linguistic variables are the prescribed ones. Such verification avoids incomplete or incorrect evaluations presenting the correspondent error messages.

Usage

```
Coppe.cosenza(x, y, factors.of.interest, aggregation.matrix.name = "default",
  normalize = FALSE)

## S4 method for signature 'ANY,ANY,ANY,ANY,ANY'
Coppe.cosenza(x)


  ## S4 method for signature
## 'Project.portfolio,
##   Option.portfolio,
##   Factors.of.interest,
##   missing,
##   missing'
Coppe.cosenza(x,
  y, factors.of.interest, aggregation.matrix.name = "default",
  normalize = FALSE)


  ## S4 method for signature
## 'Project.portfolio,
##   Option.portfolio,
##   Factors.of.interest,
##   character,
##   missing'
Coppe.cosenza(x,
  y, factors.of.interest, aggregation.matrix.name = "default",
  normalize = FALSE)


  ## S4 method for signature
## 'Project.portfolio,
##   Option.portfolio,
##   Factors.of.interest,
##   missing,
##   logical'
Coppe.cosenza(x,
  y, factors.of.interest, aggregation.matrix.name = "default",
  normalize = FALSE)


  ## S4 method for signature
## 'Project.portfolio,
##   Option.portfolio,
##   Factors.of.interest,
##   character,
##   logical'
Coppe.cosenza(x,
  y, factors.of.interest, aggregation.matrix.name = "default",
  normalize = FALSE)

## S4 method for signature 'Project,ANY,ANY,ANY,ANY'
Coppe.cosenza(x, y, factors.of.interest,
  aggregation.matrix.name = "default", normalize = FALSE)

## S4 method for signature 'Project.portfolio,Option,ANY,ANY,ANY'
Coppe.cosenza(x, y,
  factors.of.interest, aggregation.matrix.name = "default",
  normalize = FALSE)
```

Arguments

`x`	Project.portfolio or Project S4 object
`y`	Option.portfolio or Option S4 object
`factors.of.interest`	Factors.of.interest S4 object
`aggregation.matrix.name`	character - the name of Aggregation.matrix to be used. If not provided the "default" implementation will be used
`normalize`	logical - if TRUE, the values will be normalized, dividing results by the number of factors.

Value

Coppe.cosenza S4 object

`Coppe.cosenza-class` *Coppe.cosenza S4 Class*

Description

Coppe.cosenza S4 class represents the solution of the COPPE-Cosenza method. In order to do so, this S4 class contains the final evaluation of the options regarding the studied projects. It presents a data frame presenting the final evaluation of the options regarding each project. If an option does not satisfies project´s specific factors, the option is discarded (a veto operation), with the value NA. The result also presents relevant messages list, describing if some evaluation could not be performed due to entry failures or missing evaluations.

Slots

`result` data.frame

`projects.names` character

`options.names` character

`factors.of.interest` Factors.of.interest

`aggregation.matrix` Aggregation.matrix

`messages` character

`coppeCosenzaR` *coppeCosenzaR*

Description

COPPE-Cosenza Fuzzy Hierarchy Model (coppeCosenzaR).

The program implements the COPPE-Cosenza Fuzzy Hierarchy Model .

The model was based on the evaluation of local alternatives, representing regional potentialities, so as to fulfill demands of economic projects. After defining demand profiles in terms of their technological coefficients, the degree of importance of factors is defined so as to represent the productive activity.

The method can detect a surplus of supply without the restriction of the distance of classical algebra, defining an hierarchy of location alternatives. In Coppe-Cosenza Model, the distance between factors is measured in terms of the difference between grades of memberships of the same factors belonging to two or more sets under comparison.

The required factors are classified under the following linguistic variables:

- Critical (CR),
- Contitioning (C),
- Little Conditioning (LC), and

- Irrelevant (I).

And the alternatives can assume the following linguistic variables:

- Excellent (Ex),
- Good (G),
- Regular (R),
- Weak (W),
- Empty (Em),
- Zero (Z), and
- Inexistent (In).

The model also provides flexibility, allowing different aggregation rules to be performed and defined by the Decision Maker. Such feature is considered in this package, allowing the user to define other aggregation matrices, since it considers the same linguistic variables mentioned.

The following matrices are avaiable in the package:

- Default Matrix (see Aggregation.matrix.default)
- Membership Difference Matrix (see Aggregation.matrix.membership.difference)

#' New matrices can be added when requested.

Author(s)

Pier-Giovanni Taranti <`ptaranti@gmail.com`>

Leonardo Antonio Monteiro Pessoa

Carlos Alberto Nunes Cosenza

References

Cosenza, Carlos Alberto Nunes, Francisco Antonio Doria, and Leonardo Antonio Monteiro Pessôa. Hierarchy Models for the Organization of Economic Spaces. Procedia Computer Science 55 (2015): 82-91. `https://doi.org/10.1016/j.procs.2015.07.010`

See Also

Useful links:

- `https://github.com/ptaranti/coppeCosenzaR`
- Report bugs at `https://github.com/ptaranti/coppeCosenzaR/issues`

`Factor` *Factor Constructor*

Description

Factor(name) is a constructor to Factor S4 objects. Factor S4 class contains a single slot with the factor name.

Usage

```
Factor(name)
```

Arguments

`name`	character the factor namecharacter (any other argument will be cast to character)

Value

a `Factor` S4 object

Examples

```
factor <- Factor("name")
Factor("name")
```

`Factor-class` *Factor S4 Class*

Description

Factor S4 class contains a single slot with the Factor name. A factor in the COPPE-Cosenza model represents an item to be considered both in the options and in projects.

Slots

`name` character

`Factors.of.interest` *Factors.of.interest Constructor*

Description

Factors.of.interest is a constructor. Factor elements inserted in list.of.factors are type-checked as S4 coppeCosenza::Factor objects. They must have distinct names.

Usage

```
Factors.of.interest(list.of.factors)
```

Arguments

`list.of.factors`	list of Factor S4 objects

Value

a `Factors.of.interest` S4 object

Examples

```
Factors.of.interest(list(Factor("factor1"), Factor("factor2"),
Factor("factor3")))
```

`Factors.of.interest-class` *Factors.of.interest S4 Class*

Description

Factors.of.interest S4 class contains a list of S4 Factor objects. This list is used as parameter when construction the output from Coppe-Cosenza method.

Slots

`list.of.factors` list of Factor. Has one or more distinct S4 Factor objects.

`getFactorsOfInterestNames` *getFactorsOfInterestNames*

Description

It provides a sorted vector with the names of factors.

Usage

```
getFactorsOfInterestNames(factors.of.interest)
```

Arguments

`factors.of.interest`	S4 Factors.of.interest object

Value

vector of character

Examples

```
## Not run: getFactorsOfInterestNames(factors.of.interest)
```

`getOptionFactorsNames` *getOptionFactorsNames*

Description

This function returns a sorted vector with all the factors names in a Option S4 object

Usage

```
getOptionFactorsNames(option)
```

Arguments

`option`	an Option S4 object

Value

It provides a sorted vector with the names of factors in an option.

Examples

```
## Not run: getOptionFactorsNames(option)
```

getOptionPortfolioFactors

getOptionPortfolioFactors

Description

function that provides a list of Factor S4 objects presents in a Option.portfolio S4 object

Usage

```
getOptionPortfolioFactors(option.portfolio)
```

Arguments

`option.portfolio`	S4 Option.portfolio object

Value

list of Factor S4 objects

Examples

```
## Not run: getOptionPortfolioFactors(option.portfolio)
```

getOptionPortfolioNames

getOptionPortfolioNames

Description

function that provides a sorted vector with option names.

Usage

```
getOptionPortfolioNames(option.portfolio)
```

Arguments

`option.portfolio`	S4 Option.portfolio object

Value

vector of character

Examples

```
## Not run: getOptionPortfolioNames(option.portfolio)
```

`getProjectFactorsNames` *getProjectFactorsNames*

Description

This function returns a sorted vector with all the factors names in a Project S4 object

Usage

```
getProjectFactorsNames(project)
```

Arguments

`project`	an Project S4 object

Value

It provides a sorted vector with the names of factors in an project

Examples

```
## Not run: getProjectFactorsNames(project)
```

`getProjectFactorsSpecific` *getProjectFactorsSpecific*

Description

This function returns a sorted vector with all the factors names in a Project S4 object which were classified as specific to the project under discussion.

Usage

```
getProjectFactorsSpecific(project)
```

Arguments

`project`	an Project S4 object

Value

It provides a sorted vector with the names of factors in an project which were classified as specific to the project under discussion.

Examples

```
## Not run: getProjectFactorsSpecific(project)
```

---

getProjectPortfolioFactors

getProjectPortfolioFactors

Description

function that provides a sorted vector with factors from the project list.

Usage

```
getProjectPortfolioFactors(project.portfolio)
```

Arguments

`project.portfolio`	S4 Project.portfolio object

Value

vector of character

Examples

```
## Not run: getProjectPortfolioFactors(project.portfolio)
```

getProjectPortfolioNames *getProjectPortfolioNames*

Description

function that provides a sorted vector with project names.

Usage

```
getProjectPortfolioNames(project.portfolio)
```

Arguments

`project.portfolio`	S4 Project.portfolio object

Value

vector of character

Examples

```
## Not run: getProjectPortfolioNames(project.portfolio)
```

Option *Option Constructor function*

Description

Constructs a Option S4 object, which represents a possible solution to projects. The object includes a list of Option.resource, which is type checked.

Usage

```
Option(name, option.resources)
```

Arguments

`name`	character character (any other argument will be cast to character)
`option.resources`	Option.resources S4 object. Cannot be empty.

Value

a `Option` S4 object

Examples

```
## Not run: Option <- Option(name, option.resources)
```

`Option-class` *Option S4 Class*

Description

Option S4 class represents a possible solution to projects. The object includes a list of Option.resource, which is type checked.

Slots

`name` character (any other argument will be cast to character)

`option.resources` Option.resources

`Option.factor.availability` *Option.factor.availability Constructor*

Description

Constructs a Option.factor.availability S4 class. This defines the criterion in association to a factor when evaluating projects .

Usage

```
Option.factor.availability(factor, availability)
```

Arguments

`factor`	Factor S4 class
`availability`	character, must mach the scale of degrees as provided in `Option.factor.availability` class documentation

Value

a `Option.factor.availability` S4 object

Examples

```
## Not run: Option.factor.availability <- Option.factor.availability(factor, availability)
Option.factor.availability(Factor("fator1"), "Ex")
```

`Option.factor.availability-class` *Option.factor.availability S4 Class*

Description

Option.factor.availability S4 class. It defines the availability to be used in association to a factor when evaluating projects .

Details

The accepted degrees are: Excellent (Ex), Good (G), Regular (R), Weak (W), Empty (Em), Zero (Z), and Inexistent (In).

Slots

`factor` Factor S4 class

`availability` character, must mach the scale of degrees to be used

`Option.portfolio` *Option.portfolio*

Description

S4 method to construct Option.portfolio S4 objects. It accepts different sets for parameters types.

Usage

```
Option.portfolio(x)

## S4 method for signature 'ANY'
Option.portfolio(x)

## S4 method for signature 'list'
Option.portfolio(x)

## S4 method for signature 'data.frame'
Option.portfolio(x)
```

Arguments

`x`	list of Option S4 object or a data.frame

Value

a Option.portfolio S4 object

Note

Arguments (ANY)
A call to `Project.portfolio( )` with no parameters will return an error message for mismatch argument.

Arguments list(). A non-empty list with Option S4 objects.

Arguments data.frame. A data.frame where columns represent factors and rows are the options. The data frame is checked for no columns and no rows. The constructors called subsequently will verify if acceptable values where used to factor evaluation and for distinct names of factors and options.

It is possible to obtain a dummy table to serve as example by construction a potrfolio using `Option.portfolio(list.of.opt` and after converting it in a data.frame using the function `as.data.frame(option.portfolio)`.

Examples

```
## Not run: option.portfolio <- Option.portfolio(list.of.options)

## Not run: option.portfolio <- Option.portfolio(my.option.portfolio.data.frame)
```

`Option.portfolio-class`
Option.portfolio S4 Class

Description

Option.portfolio S4 class contains a type-checked list of S4 Option objects. This object is an argument to construct the CoppeCosenza S4 objects, which, in turn, represents the method solution.

Details

Any S4 Option object can be included in the @list.of.options. This means we can have options with different set of factors. It is possible to export and import Option.portfolio to/from data.frame, allowing to store and edit information externally.

Slots

`list.of.option` list of Option S4 objects. The option names are checked and must be distinct.

`Option.resources` *Option.resources Constructor*

Description

A constructor to Option.resources S4 objects.

Usage

```
Option.resources(list.of.factor.availability)
```

Arguments

`list.of.factor.availability`	list of Option.factor.availability S4 objects

Value

a `Option.resources` S4 object

Examples

```
## Not run: Option.resources(list.of.factor.availability)
```

`Option.resources-class` *Option.resources S4 Class*

Description

Option.resources S4 class contains a list of one or more S4 Option.factor.availability objects. This list is type-checked and used to construct Option objects.

Slots

`list.of.factor.availability` list of Option.factor.availability

`Project` *Project Constructor function*

Description

Constructs a Project S4 object. ... TODO(Pessoa) VRF e Ampliar

Usage

```
Project(name, project.criteria)
```

Arguments

`name`	character
`project.criteria`	Project.criteria S4 object

Value

a `Project` S4 object

Examples

```
## Not run: Project <- Project(name, project.criteria)
```

`Project-class` *Project S4 Class*

Description

Project S4 class represents a potential project and its slots include a Project.criteria object, with the list of needed factors to the project and their degree of importance. The project has a non-empty name.

Slots

`name` character (any other argument will be cast to character)

`project.criteria` Project.criteria

`Project.criteria` *Project.criteria Constructor*

Description

Project.criteria is a constructor to Factor S4 objects.

Usage

```
Project.criteria(list.of.project.criterion)
```

Arguments

`list.of.project.criterion`	list of Project.criterion S4 objects. The list is type checked and cannot be empty. The factors of the used project.criterion must be distinct

Value

a `Project.criteria` S4 object

Examples

```
## Not run: Project.criteria(list(project.criterion1,project.criterion2))
```

`Project.criteria-class` *Project.criteria S4 Class*

Description

Project.criteria S4 class contains a list of S4 Project.criterion objects. This list is used to construct Projec objects, and is type checked.

Slots

`list.of.project.criterion` list of Project.criterion

`Project.criterion` *Project.criterion*

Description

This function is a constructor to Project.criterion S4 class. It defines the criterion to be used in association to a factor when evaluating projects.

Usage

```
Project.criterion(factor, importance.degree, specific)
```

Arguments

`factor`	Factor S4 class
`importance.degree`	character, must mach one item of the scale of degrees to be used ("Cr", "C", "LC","I")
`specific`	logical indicates the considered factors is specific for the project under consideration#'

Value

a `Project.criterion` S4 object

Examples

```
## Not run: Project.criterion <- Project.criterion(factor, importance.degree, specific)
Project.criterion(Factor("fator1"), "LC", FALSE)
```

`Project.criterion-class` *Project.criterion S4 Class*

Description

Project.criterion S4 class. It defines the criterion to be used in association to a factor when evaluating projects.

Details

The accepted degrees are: "Cr", "C", "LC","I"

Slots

`factor` Factor S4 class

`importance.degree` character, must mach the scale of degrees to be used

`specific` logical indicates the considered factors is specific for the project under consideration

---

`Project.portfolio` *Project.portfolio*

Description

S4 method to construct Project.portfolio S4 objects. It accepts different sets for parameters types.

Usage

```
Project.portfolio(x, y)

## S4 method for signature 'ANY,ANY'
Project.portfolio(x)

## S4 method for signature 'list,ANY'
Project.portfolio(x)

## S4 method for signature 'data.frame,data.frame'
Project.portfolio(x, y)
```

Arguments

x	list A non-empty list with Project S4 objects, or a data frame with factors evauation
y	data.frame with specfic factors, if x is also a data.frame

Value

a Project.portfolio S4 object

Note

Arguments (ANY)
A call to `Project.portfolio( )` with no parameters will return an error message for missing argument.

Arguments (data.frame, data.frame). Data.frame where columns represent factors and rows are the projects. The data.frame is checked for no-columns and no-rows. The firs data.frame contain the factors evaluation and the second, with same rows and columns, contain boolean information about the factor being specific or not to the project. The constructors called subsequently will verify if acceptable values where used to factor evaluation and for distinct names of factors and projects

It is possible to obtain a dummy table to serve as example by construction a portfolio using `Project.portfolio(list.of.pr`
and, after, converting it in a data.frame using the function `as.data.frame(project.portfolio)`.

Examples

```
## Not run: option.portfolio <- Project.portfolio(list.of.project)

## Not run: project.portfolio <-
(project.portfolio.as.data.frame, project.portfolio.specifics.as.data.frame)
## End(Not run)
```

Project.portfolio-class *Project.portfolio*

Description

Project.portfolio S4 class contains a type-checked list of S4 Project objects. This project.portfolio is an argument to construct the CoppeCosenza S4 objects, which, in turn, represents the method solution.

Slots

`list.of.project` list of Project S4 objects

Note

Any S4 Project object can be included in the @list.of.project. This means we can have projects with different set of factors. It is possible to export and import Project.portfolio to/from data.frame, allowing to store and edit information externally.

---

show,Aggregation.matrix-method *show*

Description

show

Usage

```
## S4 method for signature 'Aggregation.matrix'
show(object)

show(object)

## S4 method for signature 'Aggregation.matrix.default'
show(object)
```

```
## S4 method for signature 'Aggregation.matrix.membership.difference'
show(object)

## S4 method for signature 'Factor'
show(object)

## S4 method for signature 'Factors.of.interest'
show(object)

## S4 method for signature 'Option.factor.availability'
show(object)

## S4 method for signature 'Option.resources'
show(object)

## S4 method for signature 'Option'
show(object)

## S4 method for signature 'Option.portfolio'
show(object)

## S4 method for signature 'Project.criterion'
show(object)

## S4 method for signature 'Project.criteria'
show(object)

## S4 method for signature 'Project'
show(object)

## S4 method for signature 'Project.portfolio'
show(object)

## S4 method for signature 'Coppe.cosenza'
show(object)
```

Arguments

`object`	Aggregation.matrix
`Aggregation.matrix.default`	Aggregation.matrix.default
`Aggregation.matrix.membership.difference`	Aggregation.matrix.membership.difference
`Factor`	Factor
`Factors.of.interest`	Factors.of.interest

`Option.factor.availability`	Option.factor.availability
`Option.resources`	Option.resources
`Option`	Option
`Option.portfolio`	Option.portfolio
`Project.criterion`	Project.criterion
`Project.criteria`	Project.criteria
`Project`	Project
`Project.portfolio`	Project.portfolio
`Coppe.cosenza`	Coppe.cosenza

`summary` *summary*

Description

Generic S4 method to `summary`.

Usage

```
summary(object, ...)

## S4 method for signature 'Coppe.cosenza'
summary(object)
```

Arguments

`object`	Coppe.cosenza
`...`	not used.

Value

summary

Index