Package 'POSetR'

March 20, 2021

Type Package

Title Partially Ordered Sets in R

Version 1.0.3

Date 2021-03-19

Author Alberto Arcagni [aut, cre],
Alessandro Avellone [aut],
Marco Fattore [aut]

Maintainer Alberto Arcagni <alberto.arcagni@uniroma1.it>

Description Provides a set of basic tools for generating, analyzing, summarizing and visualizing finite partially ordered sets. In particular, it implements flexible and very efficient algorithms for the extraction of linear extensions and for the computation of mutual ranking probabilities and other user-defined functionals, over them. The package is meant as a computationally efficient ``engine", for the implementation of data analysis procedures, on systems of multidimensional ordinal indicators and partially ordered data, in the spirit of Fattore, M. (2016) ``Partially ordered sets and the measurement of multidimensional ordinal deprivation", Social Indicators Research <DOI:10.1007/s11205-015-1059-6>, and Fattore M. and Arcagni, A. (2018) ``A reduced posetic approach to the measurement of multidimensional ordinal deprivation", Social Indicators Research <DOI:10.1007/s11205-016-1501-4>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.0), igraph, methods, Rdpack

RdMacros Rdpack

LinkingTo Rcpp

SystemRequirements C++17

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-03-20 17:00:02 UTC

R topics documented:

POSetR-package 2
antiChain 2
chain 3
intersection 4
LEapply 5
plot.Rcpp_POSet 7
poset 9
print.Rcpp_POSet 10
print.summary_poset 11
productOrder 12
summary.Rcpp_POSet 13

Index **14**

`POSetR-package` *Partially Ordered Sets in R*

Description

Provides a set of basic tools for generating, analyzing, summarizing and visualizing finite partially ordered sets. In particular, it implements flexible and very efficient algorithms for the extraction of linear extensions and for the computation of mutual ranking probabilities and other user-defined functionals, over them. The package is meant as a computationally efficient "engine", for the implementation of data analysis procedures, on systems of multidimensional ordinal indicators and partially ordered data, in the spirit of Fattore, M. (2016) "Partially ordered sets and the measurement of multidimensional ordinal deprivation", Social Indicators Research <DOI:10.1007/s11205-015-1059-6>, and Fattore M. and Arcagni, A. (2018) "A reduced posetic approach to the measurement of multidimensional ordinal deprivation", Social Indicators Research <DOI:10.1007/s11205-016-1501-4>.

Author(s)

Alberto Arcagni [aut, cre], Alessandro Avellone [aut], Marco Fattore [aut]

`antiChain` *Generate an anti-chain from a vector of elements' labels*

Description

The function produces an antichain from the vector of elements' labels `elements`. If `elements` is a numeric vector, it is first coerced to a character vector.

An antichain is a poset with no comparabilities, therefore the order of the labels in `elements` does not matter.

Usage

```
antiChain(elements)
```

Arguments

elements	a vector of characters with elements' labels.

Value

an S4 object of class Rcpp_POSet.

See Also

function poset for more details and to create a generic poset and function chain to create a complete order.

Examples

```
antiChain(LETTERS[1:5])
```

---

chain *Generate a complete order from a vector of elements' labels*

Description

The function produces a complete order from the vector of elements' labels elements. If elements is a numeric vector, it is first coerced to a character vector.

A complete order is a poset where all of the elements are comparable; the order of the labels in elements defines their position in the chain, from bottom to top.

Usage

```
chain(elements)
```

Arguments

elements	a vector of characters listing the elements' labels.

Value

an S4 object of class Rcpp_POSet.

See Also

function poset for more details and to create a generic poset and function antiChain to create an anti-chain.

Examples

```
chain(LETTERS[1:5])
```

`intersection` *Intersection of two posets*

Description

Given two posets `X` and `Y` on the same set, `intersection` returns the poset Z defined by `a < b` in `Z` if and only if `a < b` in `X` and `a < b` in `Y`.

Usage

```
intersection(x, y)

x %it% y
```

Arguments

`x`	an S4 object of class `Rcpp_POSet`, see `poset` for details.
`y`	an S4 object of class `Rcpp_POSet`, see `poset` for details.

Value

an S4 object of class `Rcpp_POSet`, see `poset` for details

References

Davey BA, Priestley HA (2002). *Introduction to lattices and order*. Cambridge university press.

See Also

`poset`

Examples

```
dom <- matrix(c(
  "a", "b",
  "c", "b",
  "b", "d"
), ncol = 2, byrow = TRUE)
p <- poset(x = dom)
plot(p)
dom <- matrix(c(
  "a", "b",
  "c", "b",
  "d", "b"
), ncol = 2, byrow = TRUE)
```

```
q <- poset(x = dom)
plot(q)
plot(p %it% q)
```

LEapply *Applies scalar functions over the set of linear extensions of a poset and returns the corresponding average values*

Description

"LEapply" is composed of three main elements: (i) the linear extensions generator, (ii) the application of the argument functions to the linear extensions and (iii) the computation of the averages of the results, for each function separately; see Fattore M (2016). "Partially ordered sets and the measurement of multidimensional ordinal deprivation." *Social Indicators Research*, **128**(2), 835–858..

Usage

```
LEapply(x, ...)

## S3 method for class 'Rcpp_POSet'
LEapply(
  x,
  FUN = "MutualRankingProbability",
  ...,
  generator = c("AllLE", "BubleyDyer"),
  bubleydyer.precision = 10,
  bubleydyer.nit = NULL,
  bubleydyer.progressBar = TRUE,
  degrees = NULL
)
```

Arguments

x	an S4 object of class Rcpp_POSet, see poset for details.
...	optional arguments to FUN.
FUN	the function, or a list of functions, to be applied to each linear extension: see 'Details'.
generator	a string specifying the method used to generate the linear extensions. The default value is "AllLE". See section 'Details' below.
bubleydyer.precision	considered only if "BubleyDyer" generator is selected. It corresponds to the number of digit precision of the frequencies in the sampling distributions of linear extensions.

`bubleydyer.nit` considered only if `"BubleyDyer"` generator is selected. Number of iterations in the Bubley-Dyer algorithm, if NULL (default) it is set as indicated in Bubley and Dyer (1999) depending on the value of `bubleydyer.precision` and the number of elements of the poset.

`bubleydyer.progressBar` logical that indicates whether to show a text progress bar or not

`degrees` to generate the lexicographic linear extensions of a product order, the poset `x` describes the dominance (e.g. relative importance) between ordinal variables and `degrees` is a numerical vector specifying the number of degrees of each variable, represented by in the poset.

Details

Argument `FUN` must be either a function or a list of functions, each one depending on a vector of characters representing the names of the elements of the poset.

If `degrees` is not `NULL` but a numerical vector as long as the number of elements in the poset, the poset elements are considered as ordinal variables. Therefore `degrees` represents their number of degrees that are represented as integer numbers starting from 0. In this case, `LEapply` generates the lexicographical linear extensions of the product order of the ordeinal variables. Its elements are called profiles and they are obtained by the combination of the degrees of variables separated by a dash. For details about lexicographical linear extensions and profiles see Fattore M, Arcagni A (2018). "A reduced posetic approach to the measurement of multidimensional ordinal deprivation." *Social Indicators Research*, **136**(3), 1053–1070..

Some functions are already implemented in the `C++` library and they can be called by their names. Currently, such functions are `"MutualRankingProbability"`, `"Separation"`, and `"AverageHeight"`.

Each function in `FUN` must return a numerical or logical matrix. Each function can depend on additional arguments that can be passed through `...`; such additinal arguments must be the same for all the functions in the list.

Argument `generator` specifies the linear extension generation algorithm. The available generators are `"AllLE"`, that produces all of the linear extensions of the input poset, and `"BubleyDyer"`, which samples uniformly from the set of linear extensions, through an MCMC algorithm (Bubley and Dyer 1999).

Value

The average values of the argument functions `FUN` over the set of linear extensions (or lexicographic ones if `degrees` argument is not `NULL`).

References

Bubley R, Dyer M (1999). "Faster random generation of linear extensions." *Discrete mathematics*, **201**(1-3), 81–88.

Fattore M (2016). "Partially ordered sets and the measurement of multidimensional ordinal deprivation." *Social Indicators Research*, **128**(2), 835–858.

Fattore M, Arcagni A (2018). "A reduced posetic approach to the measurement of multidimensional ordinal deprivation." *Social Indicators Research*, **136**(3), 1053–1070.

Habib M, Medina R, Nourine L, Steiner G (2001). "Efficient algorithms on distributive lattices." *Discrete Applied Mathematics*, **110**(2-3), 169–187.

See Also

poset

Examples

```
dom <- matrix(c(
"a", "b",
"c", "b",
"b", "d"
), ncol = 2, byrow = TRUE)
p <- poset(x = dom)

## Not run:
LEapply(
  x = p,
  FUN = "MutualRankingProbability",
  generator = "AllLE",
  degrees = c(3, 2, 3, 2)
)

a_rank_dist <- function(le) {
  return(matrix(le == "a"))
}
LEapply(x = p, FUN = a_rank_dist)
## End(Not run)
```

---

plot.Rcpp_POSet *Plotting the Hasse diagram of a poset*

Description

plot produces an igraph object and shows the Hasse diagram.

Usage

```
## S3 method for class 'Rcpp_POSet'
plot(
  x,
  vertex.color = rgb(1, 1, 1, 1),
  vertex.label = x$elements(),
  vertex.label.color = rgb(0, 0, 0, 1),
  vertex.label.family = "sans",
  edge.color = rgb(0, 0, 0, 1),
  edge.label = NA,
  edge.arrow.mode = "-",
```

```
  asp = 0,
  ...,
  equispaced = FALSE,
  show = TRUE
)
```

Arguments

x	an S4 object of class Rcpp_POSet, see poset for details.
vertex.color	argument of the plot.igraph function, see igraph.plotting for details.
vertex.label	argument of the plot.igraph function, see igraph.plotting for details.
vertex.label.color	argument of the plot.igraph function, see igraph.plotting for details.
vertex.label.family	argument of the plot.igraph function, see igraph.plotting for details.
edge.color	argument of the plot.igraph function, see igraph.plotting for details.
edge.label	argument of the plot.igraph function, see igraph.plotting for details.
edge.arrow.mode	argument of the plot.igraph function, see igraph.plotting for details.
asp	argument of the plot.igraph function, see igraph.plotting for details.
...	additional plotting parameters, see igraph.plotting for details.
equispaced	logical, if TRUE the nodes on the same level of the Hasse diagram are horizontally equispaced.
show	logical, if TRUE (default) the Hasse diagram is plotted.

Details

plot.Rcpp_POSet computes the cover relation and produces the corresponding Directed Acyclic Graph (DAG), as an igraph object, returned as invisible output. Function layout_with_sugiyama generates the DAG layout with edges oriented from top to bottom. When equispaced=TRUE, nodes on the same Hasse diagram level are horizontally equispaced.

The Hasse diagram is displayed by a call to plot.igraph (some default argument values are set to get a cleaner plot, by exploiting Hasse diagram properties.

Setting show = FALSE produces the igraph object, without showing the Hasse diagram.

Note that

Value

an igraph object.

See Also

poset, igraph, igraph.plotting

Examples

```
dom <- matrix(c(
  "a", "b",
  "c", "b",
  "b", "d"
), ncol = 2, byrow = TRUE)
p <- poset(x = dom)
hasse <- plot(p)
class(hasse)
```

`poset` *Generates a Partially Odered SET from the list of dominances*

Description

Function `poset` creates a poset from a dominance list. `x` argument is a two-column matrix, each row defines a pair of comparable elements, where the element in the first column is dominated by or coincide with the element in the second column. If the elements of `x` are numeric, they are first coerced to character and used as elements labels.

Usage

```
poset(x = NULL, elements = unique(as.character(x)))
```

Arguments

`x`	an object of class `matrix` with two columns, listing the dominances, by rows.
`elements`	a vector of characters listing all the labels of the elements.

Details

A partial order relation is reflexive, transitive and anti-symmetric. Given the dominance list provided by the user, the function produces the smallest poset comprising them (reflexive and transitive closure); in case the dominances provided by the user imply non-trivial cicles, violating anti-simmetry, the function returns an error.

By default `elements` is equal to all the different labels available in `x`. If some elements are incomparable, list all of the elements in `elements` or include self-comparabilities in `x`. Notice that antichains can be created in a simpler way, by function `antiChain`.

Value

an S4 object of class `Rcpp_POSet`; this class contains different C++ methods used by other functions of the package.

See Also

in the package are available functions that simply the creation of particular posets: `antiChain` to create a poset without comparabilities, `chain` to create a complete order.

Examples

```
dom <- matrix(c(
  "a", "b",
  "c", "b",
  "b", "d"
), ncol = 2, byrow = TRUE)
poset(x = dom)
poset(x = dom, elements = letters[1:5])
```

`print.Rcpp_POSet` *Method for the print function that shows the poset elements and comparabilities*

Description

`print` prints the list of poset elements and all of the strict dominances in it.

Usage

```
## S3 method for class 'Rcpp_POSet'
print(x, max = NULL, ...)
```

Arguments

`x`	an object of class `Rcpp_POSet`.
`max`	a non-null value for max specifies the approximate maximum number of entries to be printed. The default, NULL, uses `getOption("max.print")`: see that help page for more details.
`...`	further arguments passed to or from other methods.

Value

nothing

Examples

```
dom <- matrix(c(
  "a", "b",
  "c", "b",
  "b", "d"
), ncol = 2, byrow = TRUE)
p <- poset(x = dom)
print(p)
```

`print.summary_poset` *Method for the print function that shows the poset summary*

Description

`print` prints the poset summary.

Usage

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

Arguments

`x`	an object of class `summary_poset`.
`...`	further arguments passed to or from other methods.

Value

nothing

See Also

`summary.Rcpp_POSet`

Examples

```
dom <- matrix(c(
  "a", "b",
  "c", "b",
  "b", "d"
), ncol = 2, byrow = TRUE)
p <- poset(x = dom)
summary(p)

summary(chain(1:4))
summary(antiChain(LETTERS[1:5]))
```

productOrder *Product order between two posets*

Description

The function returns the product poset of two posets `X` and `Y`.

Usage

```
productOrder(x, y, sep = "-")

x %po% y
```

Arguments

`x`	an S4 object of class `Rcpp_POSet`, see `poset` for details.
`y`	an S4 object of class `Rcpp_POSet`, see `poset` for details.
`sep`	a character object indicating the separator to be used to paste profiles names.

Value

an S4 object of class `Rcpp_POSet`, see `poset` for details

References

Davey BA, Priestley HA (2002). *Introduction to lattices and order*. Cambridge university press.

See Also

`poset`

Examples

```
dom <- matrix(c(
  "a", "b",
  "c", "b",
  "b", "d"
), ncol = 2, byrow = TRUE)
p <- poset(x = dom)
q <- chain(1:3)
plot(p %po% q)
```

summary.Rcpp_POSet *Poset summary*

Description

Method of the summary function for objects of class Rcpp_POSet.

Usage

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

Arguments

object	an object of class Rcpp_POSet.
...	further arguments passed to or from other methods.

Value

An S3 object of class summary_poset listing and counting the poset elements, their strict dominances and their incomparabilities.

Examples

```
dom <- matrix(c(
  "a", "b",
  "c", "b",
  "b", "d"
), ncol = 2, byrow = TRUE)
p <- poset(x = dom)
summary(p)

summary(chain(1:4))
summary(antiChain(LETTERS[1:5]))
```

Index

%it% *(*intersection*)*, 4
%po% *(*productOrder*)*, 12

antiChain, 2, *3*, *9*

C++Object-class *(*poset*)*, 9
chain, *3*, 3, *9*

getOption, *10*

igraph, *7, 8*
igraph.plotting, *8*
intersection, 4

layout_with_sugiyama, *8*
LEapply, 5

plot.igraph, *8*
plot.Rcpp_POSet, 7
POSet *(*poset*)*, 9
poset, *3–5*, *7, 8*, 9, *12*
POSetR-package, 2
print.Rcpp_POSet, 10
print.summary_poset, 11
productOrder, 12

Rcpp_POSet, *13*
Rcpp_POSet-class *(*poset*)*, 9

summary, *13*
summary.Rcpp_POSet, *11*, 13