Package 'kstMatrix'

September 30, 2025

Type Package

Date 2025-09-30

Version 1.3-0

Title Basic Functions in Knowledge Space Theory Using Matrix Representation

Description Knowledge space theory by Doignon and Falmagne (1999) <doi:10.1007/978-3-642-58625-5> is a set- and order-theoretical framework, which proposes mathematical formalisms to operationalize knowledge structures in a particular domain. The 'kstMatrix' package provides basic functionalities to generate, handle, and manipulate knowledge structures and knowledge spaces. Opposed to the 'kst' package, 'kstMatrix' uses matrix representations for knowledge structures. Furthermore, 'kstMatrix' contains several knowledge spaces developed by the research group around Cornelia Dowling through querying experts.

Depends R (>= 4.4.0)

Suggests knitr, rmarkdown,

Imports stats, igraph, grDevices, sets, pks, tidyr

Maintainer Cord Hockemeyer <cord.hockemeyer@uni-graz.at>

License GPL-3

NeedsCompilation yes

Repository CRAN

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

VignetteBuilder knitr

Author Cord Hockemeyer [aut, cre],
Wai Wong [ctb]

Date/Publication 2025-09-30 12:10:02 UTC

Contents

---

cad *Knowledge spaces on AutoCAD knowledge*

Description

Bases of knowledge spaces on AutoCAD knowledge obtained from querying experts.

Usage

```
cad
```

Format

A list containing seven bases (cad1 to cad6, and cadmaj) in binary matrix form. Each matrix has 28 columns representing the different knowledge items and a varying number of rows containing the basis elements.

Details

Six experts were queried about prerequisite relationships between 28 AutoCAD knowledge items (Dowling, 1991; 1993). A seventh basis represents those prerequisite relationships on which the majority (4 out of 6) of the experts agree (Dowling & Hockemeyer, 1998).

References

Dowling, C. E. (1991). *Constructing Knowledge Structures from the Judgements of Experts.* Habilitationsschrift, Technische Universität Carolo-Wilhelmina, Braunschweig, Germany.

Dowling, C. E. (1993). Applying the basis of a knowledge space for controlling the questioning of an expert. *Journal of Mathematical Psychology, 37,* 21–48.

Dowling, C. E. & Hockemeyer, C. (1998). Computing the intersection of knowledge spaces using only their basis. In Cornelia E. Dowling, Fred S. Roberts, & Peter Theuns, editors, *Recent Progress in Mathematical Psychology,* pp. 133–141. Lawrence Erlbaum Associates Ltd., Mahwah, NJ.

See Also

Other Data: `fractions`, `readwrite`, `xpl`

---

fractions *Knowledge spaces on fractions*

Description

Bases of knowledge spaces on fractions obtained from querying experts.

Usage

```
fractions
```

Format

A list containing four bases (frac1 to frac3, and fracmaj) in binary matrix form. Each matrix has 77 columns representing the different knowledge items and a varying number of rows containing the basis elements.

Details

Three experts were queried about prerequisite relationships between 77 items on fractions (Baumunk & Dowling, 1997). A forth basis represents those prerequisite relationships on which the majority of the experts agree (Dowling & Hockemeyer, 1998).

References

Baumunk, K. & Dowling, C. E. (1997). Validity of spaces for assessing knowledge about fractions. *Journal of Mathematical Psychology, 41,* 99–105.

Dowling, C. E. & Hockemeyer, C. (1998). Computing the intersection of knowledge spaces using only their basis. In Cornelia E. Dowling, Fred S. Roberts, & Peter Theuns, editors, *Recent Progress in Mathematical Psychology,* pp. 133–141. Lawrence Erlbaum Associates Ltd., Mahwah, NJ.

See Also

Other Data: `cad`, `readwrite`, `xpl`

`kmassess` *Perform a probabilistic knowledge assessment*

Description

`kmassess` performs a probabilistic knowledge assessment for a given response vector, knowledge structure, and BLIM parameters.

Usage

```
kmassess(r, pks, questioning, update, beta, eta, zeta0, zeta1, threshold)
```

Arguments

`r`	Response pattern (binary vector)
`pks`	Probabilistic knowledge structure: a data frame with a probability distribution in the first columns and the structure matrix in the subsequent columns.
`questioning`	Questioning rule ("halfsplit" or "informative")
`update`	Update rule ("Bayesian" or "multiplicative")
`beta`	Careless error probabilities (vector)
`eta`	Lucky guess probabilities (vector)
`zeta0`	Vector of update parameters for wrong responses
`zeta1`	Vector of update parameters for correct responses
`threshold`	Probability threshold for stopping criterion

Details

kmassess implements the stochastic assessment procedures according to Doignon & Falmagne, 1999, chapter 10.

kmassess stops if the number of questions has reached twice the number of items.

Value

A list with the following elements:

state Diagnosed knowledge state (binary vector)

probs Resultng probability distribution

queried Sequence of items used in the assessment (list)

qtime Average time for finding a question

utime Average time for updating the probabilities

Background

Doignon & Falmagne (1985, 1999) proposed knowledge space theory originally with adaptive knowledge assessment in mind. The basic idea is to apply prerequisite relationships between items for reducing the number of problems to be posed to a learner in knowledge assessment.

Falmagne & Doignon (1988; Doignon & Falmange, 1999, chapte 10) proposed a class of stochastic procdures for such adaptive assessment which take into account that careless errors and lucky guesses may happen during the assessment by estimating a probability distribution over the knowledge structure. Such an assessment consists of three important parts

- Question rule
- Update rule
- Stopping criterion

For the **question rule,** they propose the *halfsplit* and the *infomrative* rules, implemented in kmassesshalfslit and kmassessinfomrative.

For the **update rule,** they again propose two possibilities there the *multiplicative rule* is a generalisation of the (classical) *Bayesian update rule* implemented here in kmassessmultiplicative and kmassessbayesian, respectively.

As **stopping criterion,** usually a threshold for the maximal probability for one knowledge state is used. It is strongly recommended to keep this larger than 0.5 in order to have one unequivocal resulting state (see also Hockemeyer, 2002).

Framework of assessment functions within the kstMatrix package::

The founding stones are the four aforementioned functions for finding suitable questions and for updating the probability estimates, respectively. They could also be used in an interactive system, e.g. a Shiny app, for "real" adaptive assessment.

The remaining thee assessment functions serve for mere simulation of adaptive assessment. kmassess takes, among others, a full response pattern as parameter and takes the responses for the selected questions from this vector. kmsassess is a simplified version where the update parameters (beta and eta for Bayesian or zeta0 and zeta1 for multiplicative update, respectively) are identical for all

items whereas they are item-specific in kmassess. Finally, kmassesssimulation takes a whole data set, i.e. a collection of response patterns, and does an assessment for each of these patterns. Its result is a data frame which should be suitable for further statistical evaluation, especially if it is called several times with variant parameters (e.g., structures, update parameters, update and question rules).

Both, kmsassess and kmassesssimulation call kmassess.

Problems:

In rare cases kmassess may flip forth and back between probability distributions resulting in an endless loop. Therefore, it stops after twice the number of items delivering a NULL result.

References

Doignon, J.-P. & Falmagne, J.-C. (1985). Spaces for the assessment of knowledge. *International Journal of Man-Machne-Studies, 23,* 175-196. doi:10.1016/S00207373(85)800316.

Doignon, J.-P. & Falmagne, J.-C. (1999). *Knowledge Spaces.* Springer Verlag, Berlin. doi:10.1007/9783642586255.

Falmagne, J.-C. & Doignon, J.-P. (1988). A class of stochastic procedures for the assessment of knowledge. *British Journal of Mathematical and Statistical Psychology, 41,* 1-23. doi:10.1111/j.20448317.1988.tb00884.x.

Hoxkemeyer, C. (2002). A comparison of non-deterministic procedures for the adaptive assessment of knowledge. *Psychlogische Beiträge, 44(4),* 495-503.

See Also

Other Knowledge assessment: kmassessbayesian(), kmassesshalfsplit(), kmassessinformative(), kmassessmentsimulation(), kmassessmultiplicative(), kmsassess()

Examples

```
kmassess(c(1, 1, 0, 0),
         cbind(as.data.frame(as.matrix(rep(1/9.0, 9), ncol=1)), xpl$space),
         "halfsplit",
         "Bayesian",
         rep(0.3, 4),
         rep(0.2, 4),
         NULL,
         NULL,
         0.55
        )
```

`kmassessbayesian` *Update probability distribution applying Bayesian update*

Description

`kmassessbayesian` updates a probability distribution over a knowledge structure according to the Bayesian update rule.

Usage

```
kmassessbayesian(probs, ks, beta, eta, question, response)
```

Arguments

`probs`	Probability distribution over the knowledge structure (vector)
`ks`	Binary matrix of the knowledge structure
`beta`	Vector of careless error probabilities
`eta`	Vector of lucky guess probabilities
`question`	Item that has been posed
`response`	Correctness of received response (0 or 1)

Value

Updated probability vector

See Also

Other Knowledge assessment: `kmassess()`, `kmassesshalfsplit()`, `kmassessinformative()`, `kmassessmentsimulation()`, `kmassessmultiplicative()`, `kmsassess()`

Examples

```
kmassessbayesian(c(0.02, 0.1, 0.07, 0.01, 0.4, 0.17, 0.07, 0.08, 0.08),
                 xpl$space,
                 rep(0.2,4),
                 rep(0.1,4),
                 3,
                 1
                )
```

kmassesshalfsplit *Determine next question for probabilistic knowledge assessment*

Description

kmassesshalfsplit determines the next question in a probabilistic assessment according to the halfsplit rule.

Usage

```
kmassesshalfsplit(probs, ks)
```

Arguments

probs	Probability distribution over the knowledge structure (vector)
ks	Binary matrix of the knowledge structure

Value

Number of the selected question

See Also

Other Knowledge assessment: kmassess(), kmassessbayesian(), kmassessinformative(), kmassessmentsimulation(
kmassessmultiplicative(), kmsassess()

Examples

```
kmassesshalfsplit(c(0.02, 0.1, 0.07, 0.01, 0.4, 0.17, 0.07, 0.08, 0.08),
                  xpl$space)
```

kmassessinformative *Determine next question for probabilistic knowledge assessment*

Description

kmassessinfmrative determines the next question in a probabiliststic assessment according to the informative rule.

Usage

```
kmassessinformative(probs, ks, update, beta, eta, zeta0, zeta1)
```

Arguments

probs	Probability distribution over the knowledge structure (vector)
ks	Binary matrix of the knowledge structure
update	Update rule ("Bayesian" or "multiplicative")
beta	Careless error probabilities (vector)
eta	Lucky guess probabilities (vector)
zeta0	Vector of update parameters for wrong responses
zeta1	Vector of update parameters for correct responses

Value

Number of the selected question

See Also

Other Knowledge assessment: kmassess(), kmassessbayesian(), kmassesshalfsplit(), kmassessmentsimulation(), kmassessmultiplicative(), kmsassess()

Examples

```
kmassessinformative(c(0.02, 0.1, 0.07, 0.01, 0.4, 0.17, 0.07, 0.08, 0.08),
                    xpl$space,
                    "Bayesian",
                    rep(0.3,4),
                    rep(0.2,4),
                    NULL,
                    NULL
                   )
```

---

kmassessmentsimulation

Simulate assessments for a set of response patterns

Description

kmassessmentsimulation does a probabilistic knowledge assessment for each response pattern in a data matrix and stores information about the assessment.

Usage

```
kmassessmentsimulation(
  respdata,
  ks,
  questioning,
  update,
  beta,
  eta,
  zeta0,
  zeta1,
  threshold
)
```

Arguments

respdata	Data matrix
ks	Knowledge structure
questioning	Question rule
update	Updating rule
beta	Careless error probability
eta	Lucky guess probability
zeta0	Update parameter for wrong responses
zeta1	Update parameter for correct responses
threshold	Stopping criterion

Details

kmassessmentsimulation applies the kmsassess function.

Value

Assessment data as data frame

See Also

Other Knowledge assessment: kmassess(), kmassessbayesian(), kmassesshalfsplit(), kmassessinformative(), kmassessmultiplicative(), kmsassess()

Examples

```
kmassessmentsimulation(
  xpl$data,
  xpl$space,
  "halfsplit",
  "multiplicative",
  NULL,
  NULL,
```

```
  5,
  5,
  0.55
)
```

`kmassessmultiplicative` *Update probability distribution applying multiplicative rule*

Description

`kmassessmultiplicative` updates a probability distribution on a knowledge structure according to the multiplicative rule.

Usage

```
kmassessmultiplicative(probs, ks, zeta0, zeta1, question, response)
```

Arguments

`probs`	Probability distribution over the knowledge structure (vector)
`ks`	Binary matrix of the knowledge structure
`zeta0`	Vector of update parameters for wrong responses
`zeta1`	Vector of update parameters for correct responses
`question`	Item that has been posed
`response`	Correctness of received response (0 or 1)

Value

Updated probability vector

See Also

Other Knowledge assessment: `kmassess()`, `kmassesbayesian()`, `kmassesshalfsplit()`, `kmassessinformative()`, `kmassessmentsimulation()`, `kmsassess()`

Examples

```
kmassessmultiplicative(c(0.02, 0.1, 0.07, 0.01, 0.4, 0.17, 0.07, 0.08, 0.08),
                xpl$space,
                rep(1.2,4),
                rep(2.1,4),
                3,
                1
               )
```

kmbasis *Compute the basis of a knowledge space*

Description

kmbasis returns a matrix representing the basis of a knowledge space. If x is a knowledge structure or an arbitrary family of sets kmreduction returns the basis of the smallest knowledge space containing x.

Usage

```
kmbasis(x)
```

Arguments

x	Binary matrix representing a knowledge space

Value

Binary matrix representing the basis of the knowledge space.

See Also

Other Different representations for knowledge spaces: kmsf2basis(), kmsurmisefunction(), kmsurmiserelation(), kmunionclosure()

Examples

```
kmbasis(xpl$space)
```

kmbasisdiagram *Plot the Hasse diagram of a basis stored as a matrix*

Description

kmbasisdiagram takes a matrix representing a basis and a color vector and draws a Hasse diagram. If the color vector is NULL the states are drawn in green.

Usage

```
kmbasisdiagram(struc, horizontal = TRUE, colors = NULL)
```

Arguments

struc	Binary matrix representing a basis
horizontal	Boolean defining orientation of the graph, default TRUE
colors	Color vector (default NULL)

See Also

Other Plotting knowledge structures: kmSRdiagram(), kmcolors(), kmhasse()

---

kmcolors *Determine a color vector based on probabilities*

Description

kmcolors takes a probabilty vector and a color palette and creates a color vector to be used with kmhasse.

Usage

```
kmcolors(prob, palette = cm.colors)
```

Arguments

prob	Probability vector
palette	Color palette (default = cm.colors)

See Also

Other Plotting knowledge structures: kmSRdiagram(), kmbasisdiagram(), kmhasse()

---

kmdist *Compute the distance between a data set and a knowledge structure*

Description

kmdist returns a named vector with the frequencies of distances between a set of response patterns and a knowledge structure. This vector can be used to compute, e.g., the Discrepancy Index (DI) or the Distance Agreement Coefficient (DA).

Usage

```
kmdist(data, struct)
```

Arguments

data	Binary matrix representing a set of response patterns
struct	Binary matrix representing a knowledge structure

Value

Distance distribution vector

See Also

Other Validating knowledge spaces: kmSRvalidate(), kmvalidate()

Examples

```
kmdist(xpl$data, xpl$space)
```

---

kmdoubleequal *Test two double numbers on equity with a certain tolerance*

Description

Test two double numbers on equity with a certain tolerance

Usage

```
kmdoubleequal(x, y, tol = sqrt(.Machine$double.eps))
```

Arguments

x	First double to compare
y	Second double to compare
tol	Tolerance optional)

Value

Boolean for (approximate) equity

See Also

Other Utilities: kmsymmsetdiff()

Examples

```
kmdoubleequal(0.5+0.5, 1)
```

kmeqreduction *Reduce a family of knowledge states with respect to item equivalence*

Description

kmeqreduction takes a family of knowledge states and returns its reduction to non-equivalent items.

Usage

```
kmeqreduction(x)
```

Arguments

x	Binary matrix

Value

Binary matrix reduced by equivalences

See Also

Other Properties of knowledge structures: kmiswellgraded(), kmnotions()

Examples

```
kmeqreduction(xpl$space)
```

kmfringe *Compute the fringe of a state within a knowledge structure*

Description

kmfringe computes the fringe of a state within a knowledge structure, i.e. the set of items by which the state differs from its neighbours.

Usage

```
kmfringe(state, struct)
```

Arguments

state	Binary vector representing a knowledge state
struct	Binary matrix representing a knowledge structure

Value

Binary vector representing the fringe

See Also

Other Neighbourhood & fringe: `kmneighbourhood()`

Examples

```
kmfringe(c(1,0,0,0), xpl$space)
```

`kmgenerate` *Generate a knowledge structure from a set of response patterns*

Description

`kmgenerate` returns a matrix representing a knowledge structure generated from data. It uses a simplistic approach: patterns with a frequency above a specified threshold are considered as knowledge states. If the specified threshold is 0 (default) a real threshold is computed as N (number of response patterns) divided by 2^|Q|. Please note that the number of response patterns should be much higher than the size of the power set of the item set Q. A factor of art least 10 is recommended. Currently, the number of items is limited to the number of bits in a C long minus one (i.e. 31 under Windows and 63 otherwise). But we would probably run into memory problems way earlier anyway.

Usage

```
kmgenerate(x, threshold = 0)
```

Arguments

`x`	Binary matrix representing a data set
`threshold`	Threshold for taking response patterns as knowledge states (default 0)

Value

Binary matrix representing the generated knowledge structure

Examples

```
kmgenerate(xpl$sim, 15)
```

kmhasse *Plot the Hasse diagram of a knowledge structure stored as a matrix*

Description

kmhasse takes a matrix representing a knowledge structure and a color vector and draws a Hasse diagram. If the color vector is NULL the states are drawn in green.

Usage

```
kmhasse(struc, horizontal = TRUE, colors = NULL)
```

Arguments

struc	Binary matrix representing a knowledge structure
horizontal	Boolean defining orientation of the graph, default TRUE
colors	Color vector (default NULL)

See Also

Other Plotting knowledge structures: kmSRdiagram(), kmbasisdiagram(), kmcolors()

kmiswellgraded *Check for wellgradedness of a knowledge structure*

Description

kmiswellgraded returns whether a knowledge structure is wellgraded.

Usage

```
kmiswellgraded(x)
```

Arguments

x	Binary matrix representing a knowledge space

Value

Logical value specifying whether x is wellgraded

References

Doignon, J.-P. & Falmagne, J.-C. (1999). *Knowledge Spaces.* Springer–Verlag, Berlin.

See Also

Other Properties of knowledge structures: `kmeqreduction()`, `kmnotions()`

Examples

```
kmiswellgraded(xpl$space)
```

`kmneighbourhood` *Compute the neighbourhod of a state within a knowledge structure*

Description

`kmneighbourhood` computes the neighbourhood of a state within a knowledge structure, i.e. the family of all other states with a symmetric set difference of 1.

Usage

```
kmneighbourhood(state, struct)
```

Arguments

`state`	Binary vector representing a knowledge state
`struct`	Binary matrix representing a knowledge structure

Value

Matrix containing the neighbouring states, one per row

See Also

Other Neighbourhood & fringe: `kmfringe()`

Examples

```
kmneighbourhood(c(1,1,0,0), xpl$space)
```

kmnneighbourhood *Compute the n-neighbourhod of a state within a knowledge structure*

Description

kmnneighbourhood computes the n-neighbourhood of a state within a knowledge structure, i.e. the family of all other states with a symmetric set difference maximal n.

Usage

```
kmnneighbourhood(state, struct, distance)
```

Arguments

state	Binary vector representing a knowledge state
struct	Binary matrix representing a knowledge structure
distance	Size of the n-neighbourhood

Value

Matrix containing the neighbouring states, one per row

Examples

```
kmnneighbourhood(c(1,1,0,0), xpl$space, 2)
```

kmnotions *Determine the notions of a knowledge structure*

Description

kmnotions returns a matrix representing the notions of a knowledge structure.

Usage

```
kmnotions(x)
```

Arguments

x	Binary matrix representing a knowledge structure

Value

Binary matrix representing notions in the knowledge structure

The matrix has a '1' in row 'i' and column 'j' if 'i' and 'j' belong to the same notion (i.e. are equivalent). It is a symmetric matrix with '1's in the main diagonal.

See Also

Other Properties of knowledge structures: `kmeqreduction()`, `kmiswellgraded()`

Examples

```
kmnotions(xpl$space)
```

`kmsassess` *Perform a simplified probabilistic knowledge assessment*

Description

`kmsassess` performs a simplified probabilistic knowledge assessment for a given response vector, knowledge structure, and BLIM parameters. It assumes an equal probability distribution over the knowledge structure as starting point and identical beta and eta values for all items.

Usage

```
kmsassess(r, ks, questioning, update, beta, eta, zeta0, zeta1, threshold)
```

Arguments

`r`	Response pattern (binary vector)
`ks`	Knowledge structure: a binary matrix
`questioning`	Questioning rule ("halfsplit" o "informative")
`update`	Update rule ("Bayesian" or "multiplicative")
`beta`	Careless error probability
`eta`	Lucky guess probability
`zeta0`	Update parameter for wrong responses
`zeta1`	Update parameter for correct responses
`threshold`	Probability threshold for stopping criterion

Details

`kmsassess` uses the `kmassess` function, so the explanations there hold also here.

Value

A list with the following elements:

state Diagnosed knowledge state (binary vector)

probs Resultng probability distribution

queried Sequence of items used in the assessment (list)

See Also

Other Knowledge assessment: kmassess(), kmassessbayesian(), kmassesshalfsplit(), kmassessinformative(), kmassessmentsimulation(), kmassessmultiplicative()

Examples

```
kmsassess(c(1,1,0,0), xpl$space, "halfsplit", "Bayesian", 0.3, 0.2, NULL, NULL, 0.55)
```

---

kmsf2basis *Derive a basis from a surmise function*

Description

kmsf2basis expects a surmise function data frame and returns the corresponding basis.

Usage

```
kmsf2basis(sf)
```

Arguments

sf	Surmise function

Value

Matrix representing the basis.

See Also

Other Different representations for knowledge spaces: kmbasis(), kmsurmisefunction(), kmsurmiserelation(), kmunionclosure()

---

kmsimulate *Simulate a set of response patterns according to the BLIM*

Description

kmsimulate returns a data set of n simulated response patterns based on the knowledge structure x given as a binary matrix. The simulation follows the BLIM (Basic Local Independence Model; see Doigon & Falmagne, 1999).

Usage

```
kmsimulate(x, n, beta, eta)
```

Arguments

x	Binary matrix representing a knowledge space
n	Number of simulated response patterns
beta	Careless error probability value or vector
eta	Lucky guess probability value or vector

Details

The beta and eta parameters must be either single numericals or vectors with a length identical to the number of rows in the x matrix. A mixture is possible.

The sample function used by kmsimulate might work inaccurately for knowledge structures x with 2^31 or more states.

Value

Binary matrix representing the simulated data set

References

Doignon, J.-P. & Falmagne, J.-C. (1999). *Knowledge Spaces.* Springer–Verlag, Berlin.

Examples

```
kmsimulate(xpl$space, 50, 0.2, 0.1)
kmsimulate(xpl$space, 50, c(0.2, 0.25, 0.15, 0.2), c(0.1, 0.15, 0.05, 0.1))
kmsimulate(xpl$space, 50, c(0.2, 0.25, 0.15, 0.2), 0)
```

---

kmSRdiagram *Plot the Hasse diagram of a basis stored as a matrix*

Description

kmSRdiagram takes a matrix representing a surmise relation and a color vector and draws a Hasse diagram. If the color vector is NULL the states are drawn in green.

Usage

```
kmSRdiagram(structure, horizontal = TRUE, colors = NULL)
```

Arguments

structure	Binary matrix representing a surmise relation
horizontal	Boolean defining orientation of the graph, default TRUE
colors	Color vector (default NULL)

See Also

Other Plotting knowledge structures: kmbasisdiagram(), kmcolors(), kmhasse()

---

kmSRvalidate *Validate a surmise relation against a data set*

Description

kmSRvalidate returns a list with two elements, Goodman & Kruskal's gamma value and the violational coefficient (VC).

Usage

```
kmSRvalidate(data, sr)
```

Arguments

data	Binary matrix representing a set of response patterns
sr	Binary matrix representing a surmise relation

Value

A list with two elements:

gamma Goodman & Kruskal's gamma index

VC Violational Coefficient

See Also

Other Validating knowledge spaces: kmdist(), kmvalidate()

Examples

```
kmSRvalidate(xpl$data, xpl$sr)
```

kmsurmisefunction *Compute the surmise function for a knowledge space or basis*

Description

kmsurmisefunction returns a data frame representing the surmise function for a knowledge space or basis. The rows of the data frame are ordered by item name.

Usage

```
kmsurmisefunction(x)
```

Arguments

x	Binary matrix representing a knowledge space or basis

Value

Data frame representing the surmise unction of x.

See Also

Other Different representations for knowledge spaces: kmbasis(), kmsf2basis(), kmsurmiserelation(), kmunionclosure()

Examples

```
kmsurmisefunction(xpl$space)
```

kmsurmiserelation *Compute the surmise relation of a quasi-ordinal knowledge space*

Description

kmsurmiserelation returns a matrix representing the surmise relation of a quasi-ordinal knowledge space. If x is a general knowledge space, a knowledge structure or an arbitrary family of sets, kmsurmiserelation returns the surmise relation of the smallest quasi-ordinal knowledge space containing x.

Usage

```
kmsurmiserelation(x)
```

Arguments

x	Binary matrix representing a quasi-ordinal knowledge space

Value

Binary matrix representing the surmise relation of the corresponding quasi-ordinal knowledge space

Note: The columns of the surmise relation matrix describe the minimal state for the respective item in the quasi-ordinal knowledge space.

See Also

Other Different representations for knowledge spaces: `kmbasis()`, `kmsf2basis()`, `kmsurmisefunction()`, `kmunionclosure()`

Examples

```
kmsurmiserelation(xpl$space)
```

`kmsymmsetdiff` *Compute the symmetric set difference between two sets*

Description

Compute the symmetric set difference between two sets

Usage

```
kmsymmsetdiff(x, y)

kmsetdistance(x, y)
```

Arguments

`x`	Binary vector representing a set
`y`	Binary vector representing a set

Value

kmsymmsetdiff: Symmetric set difference between 'x' and 'y'

kmsetdistance: Distance between the sets 'x' and 'y', i.e. the cardinality of the symmetric set difference

See Also

Other Utilities: `kmdoubleequal()`

Other Utilities: `kmdoubleequal()`

Examples

```
kmsymmsetdiff(c(1,0,0), c(1,1,0))

kmsetdistance(c(1,0,0), c(1,1,0))
```

`kmtrivial` *Create trivial knowledge spaces*

Description

These functions create trivial knowledge spaces of a given item number. The minimal space contains just the empty set and the full item set while the maximal space is equal to the power set.

Usage

```
kmminimalspace(noi)

kmmaximalspace(noi)
```

Arguments

`noi`	Number of items

Details

Please note that the computation time for creating large power sets can grow quite large easily.

Value

A binary matrix representing the respective knowledge space

Examples

```
kmminimalspace(5)
kmmaximalspace(5)
```

`kmunionclosure` *Close a family of sets under union*

Description

`kmunionclosure` returns a matrix representing a knowledge space. Please note that it may take quite some time for computing larger knowledge spaces.

Usage

```
kmunionclosure(x)
```

Arguments

`x`	Binary matrix representing a family of sets

Value

Binary matrix representing the corresponding knowledge space, i.e. the closure of the family under union including the empty set and the full set.

`kmunionclosure` implements the irredundant algorithm developed by Dowling (1993).

References

Dowling, C. E. (1993). On the irredundant construction of knowledge spaces. *Journal of Mathematical Psychology, 37,* 49–62.

See Also

Other Different representations for knowledge spaces: `kmbasis()`, `kmsf2basis()`, `kmsurmisefunction()`, `kmsurmiserelation()`

Examples

```
kmunionclosure(xpl$basis)
```

`kmvalidate` *Validate a knowledge structure against a data set*

Description

`kmvalidate` returns a list with three elements, a named vector (dist) with the frequencies of distances between a set of response patterns and a knowledge structure, the Discrepancy Index (DI), and the Distance Agreement Coefficient (DA).

Usage

```
kmvalidate(data, struct)
```

Arguments

`data`	Binary matrix representing a set of response patterns
`struct`	Binary matrix representing a knowledge structure

Value

A list with three elements:

dist Distance distribution vector

DI Discrepancy Index

DA Distance Agreement Coefficient

Warning

The DA computation can take quite some time for larger item sets as the power set has to be computed. For item sets with around 30 items or more, it may even crash the system due to huge memory requests.

See Also

Other Validating knowledge spaces: `kmSRvalidate()`, `kmdist()`

Examples

```
kmvalidate(xpl$data, xpl$space)
```

`readwrite` *Knowledge spaces on reading and writing abilities*

Description

Bases of knowledge spaces on reading/writing abilities obtained from querying experts.

Usage

```
readwrite
```

Format

A list containing four bases (rw1 to rw3, and rwmaj) in binary matrix form. Each matrix has 48 columns representing the different knowledge items and a varying number of rows containing the basis elements.

Details

Three experts were queried about prerequisite relationships between 48 items on reading and writing abilities (Dowling, 1991; 1993). A forth basis represents those prerequisite relationships on which the majority of the experts agree (Dowling & Hockemeyer, 1998).

References

Dowling, C. E. (1991). *Constructing Knowledge Structures from the Judgements of Experts.* Habilitationsschrift, Technische Universität Carolo-Wilhelmina, Braunschweig, Germany.

Dowling, C. E. (1993). Applying the basis of a knowledge space for controlling the questioning of an expert. *Journal of Mathematical Psychology, 37,* 21–48.

Dowling, C. E. & Hockemeyer, C. (1998). Computing the intersection of knowledge spaces using only their basis. In Cornelia E. Dowling, Fred S. Roberts, & Peter Theuns, editors, *Recent Progress in Mathematical Psychology,* pp. 133–141. Lawrence Erlbaum Associates Ltd., Mahwah, NJ.

See Also

Other Data: `cad`, `fractions`, `xpl`

xpl *Small example knowledge space*

Description

Basis and space matrix, surmise relation and surmise function of a small fictional knowledge space, and two data sets (data (7 patterns) and sim (500 patterns)to be used in examples. The latter was produced from the space with `kmsimulate()` with beta and eta values of 0.1.

Usage

```
xpl
```

Format

A list containing the basis, the space, the surmise relation, the surmise function, and the two data matrices data and sim.

See Also

Other Data: `cad`, `fractions`, `readwrite`

Index