Package 'pks'

September 16, 2026

Version 0.8-0

Date 2026-08-26

Title Probabilistic Knowledge Structures

Depends R (>= 4.1.0), stats, sets

Imports graphics

Suggests relations, Rgraphviz

Description Fitting and testing probabilistic knowledge structures, especially the basic local independence model (BLIM, Doignon & Flamagne, 1999) and the simple learning model (SLM), using the minimum discrepancy maximum likelihood (MDML) method (Heller & Wickelmaier, 2013 <doi:10.1016/j.endm.2013.05.145>).

License GPL (>= 2)

URL https://www.mathpsy.uni-tuebingen.de/wickelmaier/, https://codeberg.org/fwickel/pks

NeedsCompilation no

Author Florian Wickelmaier [aut, cre], Juergen Heller [aut], Julian Mollenhauer [aut], Paul Doerrbecker [aut], Pasquale Anselmi [ctb], Debora de Chiusole [ctb], Andrea Brancaccio [ctb], Luca Stefanutti [ctb]

Maintainer Florian Wickelmaier <wickelmaier@web.de>

Repository CRAN

Date/Publication 2026-09-16 02:40:02 UTC

Contents

blim 2
blimit 5

cdp 7
chess 9
conversion 11
delineate 12
DoignonFalmagne7 13
endm 14
getKFringe 15
gradedness 16
hsgeometry 17
ita 19
jacobian 20
plot.blim 21
predict.blim 22
print.blim 24
probability 25
residuals.blim 28
schoolarithm 29
simulate.blim 32
slm 34
Taagepera 36

Index **37**

`blim` *Basic Local Independence Models (BLIMs)*

Description

Fits a basic local independence model (BLIM) for probabilistic knowledge structures by minimum discrepancy maximum likelihood estimation.

Usage

```
blim(K, N.R, method = c("MD", "ML", "MDML"), R = as.binmat(N.R),
     P.K = rep(1/nstates, nstates),
     beta = rep(0.1, nitems), eta = rep(0.1, nitems),
     betafix = rep(NA, nitems), etafix = rep(NA, nitems),
     betaequal = NULL, etaequal = NULL,
     randinit = FALSE, incradius = 0,
     tol = 1e-07, maxiter = 10000, zeropad = 16)

blimMD(K, N.R, R = as.binmat(N.R),
       betafix = rep(NA, nitems), etafix = rep(NA, nitems),
       incrule = c("minimum", "hypblc1", "hypblc2"), m = 1)

## S3 method for class 'blim'
anova(object, ..., test = c("Chisq", "none"))
```

Arguments

K	a state-by-problem indicator matrix representing the knowledge structure. An element is one if the problem is contained in the state, and else zero.
N.R	a (named) vector of absolute frequencies of response patterns.
method	MD for minimum discrepancy estimation, ML for maximum likelihood estimation, MDML for minimum discrepancy maximum likelihood estimation.
R	a pattern-by-problem indicator matrix of unique response patterns. Per default inferred from the names of N.R.
P.K	the vector of initial parameter values for probabilities of knowledge states.
beta, eta	vectors of initial parameter values for probabilities of a careless error and a lucky guess, respectively.
betafix, etafix	vectors of fixed error and guessing parameter values; NA indicates a free parameter.
betaequal, etaequal	lists of vectors of problem indices; each vector represents an equivalence class: it contains the indices of problems for which the error or guessing parameters are constrained to be equal. (See Examples.)
randinit	logical, if TRUE then initial parameter values are sampled uniformly with constraints. (See Details.)
incradius	include knowledge states of distance from the minimum discrepant states less than or equal to incradius.
tol	tolerance, stopping criterion for iteration.
maxiter	the maximum number of iterations.
zeropad	the maximum number of items for which an incomplete N.R vector is completed and padded with zeros.
incrule	inclusion rule for knowledge states. (See Details.)
m	exponent for hyperbolic inclusion rules.
object	an object of class blim, typically the result of a call to blim.
test	should the p-values of the chi-square distributions be reported?
...	additional arguments passed to other methods.

Details

See Doignon and Falmagne (1999) for details on the basic local independence model (BLIM) for probabilistic knowledge structures.

Minimum discrepancy (MD) minimizes the number of expected response errors (careless errors or lucky guesses). Maximum likelihood maximizes the likelihood, possibly at the expense of inflating the error and guessing parameters. Minimum discrepancy maximum likelihood (MDML) maximizes the likelihood subject to the constraint of minimum response errors. See Heller and Wickelmaier (2013) for details on the parameter estimation methods.

If randinit is TRUE, initial parameter values are sampled uniformly with the constraint beta + eta < 1 (Weisstein, 2013) for the error parameters, and with sum(P.K) == 1 (Rubin, 1981) for the

probabilities of knowledge states. Setting `randinit` to `TRUE` overrides any values given in the `P.K`, `beta`, and `eta` arguments.

The degrees of freedom in the goodness-of-fit test are calculated as number of possible response patterns minus one or number of respondents, whichever is smaller, minus number of parameters.

`blimMD` uses minimum discrepancy estimation only. Apart from the hyperbolic inclusion rules, all of its functionality is also provided by `blim`. It may be removed in the future.

Value

An object of class `blim` having the following components:

`discrepancy`	the mean minimum discrepancy between response patterns and knowledge states.
`P.K`	the vector of estimated parameter values for probabilities of knowledge states.
`beta`	the vector of estimated parameter values for probabilities of a careless error.
`eta`	the vector of estimated parameter values for probabilities of a lucky guess.
`disc.tab`	the minimum discrepancy distribution.
`K`	the knowledge structure.
`N.R`	the vector of frequencies of response patterns.
`nitems`	the number of items.
`nstates`	the number of knowledge states.
`npatterns`	the number of response patterns.
`ntotal`	the number of respondents.
`nerror`	the number of response errors.
`npar`	the number of parameters.
`method`	the parameter estimation method.
`iter`	the number of iterations needed.
`loglik`	the log-likelihood.
`fitted.values`	the fitted response frequencies.
`goodness.of.fit`	the goodness of fit statistic including the likelihood ratio fitted vs. saturated model (G2), the degrees of freedom, and the p-value of the corresponding chi-square distribution. (See Details.)
`incradius`, `betafix`, `etafix`	See Arguments.

References

Doignon, J.-P., & Falmagne, J.-C. (1999). *Knowledge spaces*. Berlin: Springer.

Heller, J., & Wickelmaier, F. (2013). Minimum discrepancy estimation in probabilistic knowledge structures. *Electronic Notes in Discrete Mathematics*, **42**, 49–56. doi:10.1016/j.endm.2013.05.145

Rubin, D.B. (1981). The Bayesian bootstrap. *The Annals of Statistics*, **9**(1), 130–134. doi:10.1214/aos/1176345338

Weisstein, E.W. (2013, August 29). Triangle point picking. In *MathWorld – A Wolfram Web Resource*. Retrieved from https://mathworld.wolfram.com/TrianglePointPicking.html.

See Also

simulate.blim, plot.blim, residuals.blim, logLik.blim, delineate, jacobian, endm, probability, chess, predict.blim.

Examples

```
data(DoignonFalmagne7)
K   <- DoignonFalmagne7$K    # knowledge structure
N.R <- DoignonFalmagne7$N.R  # frequencies of response patterns

## Fit basic local independence model (BLIM) by different methods
blim(K, N.R, method = "MD")    # minimum discrepancy estimation
blim(K, N.R, method = "ML")    # maximum likelihood estimation by EM
blim(K, N.R, method = "MDML")  # MDML estimation

## Parameter restrictions: beta_a = beta_b = beta_d, beta_c = beta_e
##                         eta_a =  eta_b = 0.1
m1 <- blim(K, N.R, method = "ML",
           betaequal = list(c(1, 2, 4), c(3, 5)),
              etafix = c(0.1, 0.1, NA, NA, NA))
m2 <- blim(K, N.R, method = "ML")
anova(m1, m2)

## See ?endm, ?probability, and ?chess for further examples.
```

---

blimit *Basic Local Independence Model Identification Analysis*

Description

Tests the local identifiability of a basic local independence model (BLIM).

Usage

```
blimit(K, beta = NULL, eta = NULL, pi = NULL, file_name = NULL)
```

Arguments

K	a state-by-problem indicator matrix representing the knowledge structure. An element is one if the problem is contained in the state, and else zero.
beta, eta, pi	vectors of parameter values for probabilities of careless errors, lucky guesses, and knowledge states, respectively.
file_name	name of an output file.

Details

See Stefanutti et al. (2012) for details.

The `blimit` function has been adapted from code provided by Andrea Brancaccio, Debora de Chiusole, and Luca Stefanutti. It contains a function to compute the reduced row echelon form based on an implementation in the pracma package.

Value

A list having the following components:

`NItems`	the number of items.
`NStates`	the number of knowledge states.
`NPar`	the number of parameters.
`Rank`	the rank of the Jacobian matrix.
`NSD`	the null space dimension.
`RankBeta`, `RankEta`, `RankPi`, `RankBetaEta`, `RankBetaPi`, `RankEtaPi`	the rank of submatrices of the Jacobian.
`DiagBetaEta`, `DiagBetaPi`, `DiagEtaPi`, `DiagBetaEtaPi`	diagnostic information about specific parameter trade-offs.
`Jacobian`	the Jacobian matrix.
`beta`, `eta`, `pi`	the parameter values used in the analysis.

References

Stefanutti, L., Heller, J., Anselmi, P., & Robusto, E. (2012). Assessing the local identifiability of probabilistic knowledge structures. *Behavior Research Methods*, **44**(4), 1197–1211. doi:10.3758/s134280120187z

See Also

`blim`, `jacobian`.

Examples

```
K <- as.binmat(c("0000", "1000", "0100", "1110", "1101", "1111"))

set.seed(1234)
info <- blimit(K)
```

cdp *Competency Deletion Procedure (CDP)*

Description

The competency deletion procedure (CDP) reduces a set of competencies defined on a set of skills down to one of its possible reducts.

Usage

```
cdp(Competencies, CS, prefRel = c("NSP", "MBP", "MSP", "FSP"),
    DesignMat = NULL, verbose = FALSE)
```

Arguments

`Competencies`	a character vector representing the set of competencies which is to be reduced.
`CS`	a state-by-skill indicator matrix representing the competence structure. An element equals 1 if the skill is present in the state; otherwise, it equals 0.
`prefRel`	the deletion rule that determines the order in which competencies are deleted. It can either be a string defining one of the preference relations defined in Anselmi et al. (2024) – except for "LNP" –, or an object of the class `DeletionRule`, which is a character vector containing all collections of equally preferred competencies as elements, ordered from the most to the least preferred collection.
`DesignMat`	an indicator matrix with the dimensions `nrow(CS)` $\times$ `nrow(CS)`. An element equals 1 if the corresponding pair of two competence states shall be discerned during the procedure; otherwise, it equals 0. Alternatively `NULL` can be chosen instead.
`verbose`	an argument deciding if the values `delta`, `Competencies`, `CS` and `DesingMat` are printed in the output object.

Details

See Anselmi et al. (2022) for details on the competency deletion procedure (CDP) for competence structures.

See Anselmi et al. (2024) for details on the preference relations for the competency deletion procedure (CDP) for competence structures.

Using a manually defined preference relation for the argument `prefRel` requires a character vector with its class attribute set to `DeletionRule`. See examples.

If `NULL` is used for the argument `DesignMat`, all possible pairs of competence states (with respect to `CS`) are discerned.

Value

An object of class cdp having the following components:

- `reduct` the vector of the final set of competencies which is a reduct.
- `niter` the number of deletions/iterations needed.
- `ordered_deletions` the competencies which were deleted, ordered by the iteration in which they were deleted.
- `delta` a matrix of the discerned pairs of competence states in rows and the competencies from the reduct in columns. An entry in a row means that this pair of competence states can be discerned by the competency in which column is the entry.
- `pref_rel` the preference relation used as `DeletionRule` object.
- `pref_rel_method` the method used to create the preference relation.
- `Competencies`, `CS`, `DesignMat` See Arguments.

References

Anselmi, P., Heller, J., Stefanutti, L. & Robusto, E. (2022). Constructing, improving, and shortening tests for skill assessment. *Journal of Mathematical Psychology, 106,* 102621. https://doi.org/10.1016/j.jmp.2021.102621

Anselmi, P., Heller, J., Stefanutti, L., & Robusto, E. (2024). Constructing tests for skill assessment with competence-based test development. *British Journal of Mathematical and Statistical Psychology, 77*(3), 429-458. https://doi.org/10.1111/bmsp.12335

See Also

`delineate`.

Examples

```
## Create competence structure
CStr <- as.binmat(c("0000", "0100", "1100", "1010", "1110", "0111", "1111"))
rownames(CStr) <- as.pattern(CStr)

## Create competency set
T_ <- c("a", "b", "c", "ab", "ac", "ad", "bc", "bd", "cd", "abc", "abd",
        "acd", "bcd", "abcd")

## Create design matrices
DesignMat <- upper.tri(matrix(0, nrow(CStr), nrow(CStr))) + 0
colnames(DesignMat) <- rownames(DesignMat) <-
  as.pattern(CStr, useNames = TRUE)

DesignMat2 <- DesignMat
DesignMat2[1, ] <- c(0, 0, 0, 1, 1, 1, 1)
DesignMat2[3, ] <- c(0, 0, 0, 1, 1, 0, 0)
```

```
## Create manually defined preference relation
ManDef_PR <- c("a, ac, ad, abc, abd, abcd",
               "b, bc, cd, acd",
               "ab, bd, bcd",
               "c")
class(ManDef_PR) <- "DeletionRule"

## Classic examples from Anselmi et al. (2024)

## Basic version using preference relation "NSP"
cdp(Competencies = T_, CS = CStr, prefRel = "NSP")

## Larger output option and preference relation "FSP"
cdp(Competencies = T_, CS = CStr, prefRel = "FSP", verbose = TRUE)

## Classic complete design matrix and preference relation "MBP"
cdp(Competencies = T_, CS = CStr, prefRel = "MBP", DesignMat = DesignMat)

## Reduced design matrix and preference relation "MSP"
cdp(Competencies = T_, CS = CStr, prefRel = "MSP", DesignMat = DesignMat2)

## Manually defined preference relation
cdp(Competencies = T_, CS = CStr, prefRel = ManDef_PR)
```

`chess` *Responses to Chess Problems and Knowledge Structures*

Description

Held, Schrepp and Fries (1995) derive several knowledge structures for the representation of 92 responses to 16 chess problems. See Schrepp, Held and Albert (1999) for a detailed description of these problems.

Usage

```
data(chess)
```

Format

A list consisting of five components:

`dst1` a state-by-problem indicator matrix representing the knowledge structure DST1.

`dst3` the knowledge structure DST3.

`dst4` the knowledge structure DST4.

`N.R` a named integer vector. The names denote response patterns, the values denote their frequencies.

`R` a person-by-problem indicator matrix representing the responses. Column names `hdbgXX` and `grazYY` identify responses collected in Heidelberg and Graz, respectively.

Note

The graphs of the precedence relations for DST1 and DST4 in Held et. al (1995) contain mistakes that have been corrected. See examples.

Source

Held, T., Schrepp, M., & Fries, S. (1995). Methoden zur Bestimmung von Wissensstrukturen – eine Vergleichsstudie. *Zeitschrift fuer Experimentelle Psychologie*, **42**(2), 205–236.

References

Schrepp, M., Held, T., & Albert, D. (1999). Component-based construction of surmise relations for chess problems. In D. Albert & J. Lukas (Eds.), *Knowledge spaces: Theories, empirical research, and applications* (pp. 41–66). Mahwah, NJ: Erlbaum.

Examples

```
data(chess)
chess$dst1  # knowledge structure DST1

## Precedence relation (Held et al., 1995, p. 215) and knowledge space
P <- as.binmat(c("1111011101111001",   # s
                 # "0100000000000000",   # gs   mistake in Abb. 3
                   "0111010100111000",   # gs   correction
                   "0011010000011000",   # egs
                   "0011010000011000",   # eegs
                   "0000110000000000",   # cs
                   "0000010000000000",   # gcs
                   "0011011100111000",   # ts
                   "0011010100011000",   # ges
                   "1111111111111111",   # f
                   "0111010101111000",   # gf
                   "0011010000111000",   # gff
                   "0000000000010000",   # ggff
                   "0000000000001000",   # ggf
                   "0111011101111101",   # ff
                   "0111011101111011",   # tf
                   "0011010100111001"),  # tff
                 as.logical = TRUE)
dimnames(P) <- list("<" = colnames(chess$R), ">" = colnames(chess$R))
K <- rbind(0L, binary_closure(t(P)))
identical(sort(as.pattern(K)),
          sort(as.pattern(chess$dst1)))

blim(chess$dst1, chess$N.R)  # Tab. 1
```

conversion *Conversion between Representations of Responses or States*

Description

Converts between binary matrix and pattern representations of response patterns or knowledge states.

Usage

```
as.pattern(R, freq = FALSE, useNames = FALSE, as.set = FALSE,
           sep = "", emptyset = "{}", as.letters = NULL)

as.binmat(N.R, uniq = TRUE, col.names = NULL, as.logical = FALSE)

is.subset(R)
```

Arguments

R	an indicator matrix of response patterns or knowledge states.
N.R	either a (named) vector of absolute frequencies of response patterns; or a character vector of response patterns or knowledge states; or a set of sets representing the knowledge structure.
freq	logical, should the frequencies of response patterns be reported?
uniq	logical, if TRUE, only the unique response patterns are returned.
useNames	logical, return response patterns as combinations of item names.
as.set	logical, return knowledge states as set of sets.
sep	character to separate the item names.
emptyset	string representing the empty set if useNames is TRUE.
as.letters	deprecated, use useNames instead.
col.names	column names for the state or response matrix.
as.logical	logical, return logical matrix of states.

Value

as.pattern returns a vector of integers named by the response patterns if freq is TRUE, else a character vector. If as.set is TRUE, the return value is of class set.

as.binmat returns an indicator matrix. If as.logical is TRUE, it returns a logical matrix.

is.subset returns a logical incidence matrix of the subset relation among states.

See Also

blim, set in package sets.

Examples

```
data(DoignonFalmagne7)
K <- DoignonFalmagne7$K
as.pattern(K, freq = TRUE)
as.pattern(K)
as.pattern(K, useNames = TRUE)
as.pattern(K, as.set = TRUE)

N.R <- DoignonFalmagne7$N.R
dim(as.binmat(N.R))
dim(as.binmat(N.R, uniq = FALSE))

## Knowledge structure as binary matrix
as.binmat(c("000", "100", "101", "111"))
as.binmat(set(set(), set("a"), set("a", "c"), set("a", "b", "c")))
as.binmat(c("000", "100", "101", "111"), as.logical = TRUE)

## Subset relation incidence matrix
is.subset(K)

## Plotting the knowledge structure
if(requireNamespace("relations") &&
   requireNamespace("Rgraphviz")) {
  rownames(K) <- as.pattern(K, useNames = TRUE)
  plot(relations::as.relation(is.subset(K)), main = "")
}
```

`delineate` *Delineate a Knowledge Structure by a Skill Function*

Description

Computes the knowledge structure delineated by a skill function.

Usage

```
delineate(skillfun, itemID = 1)
```

Arguments

`skillfun`	a data frame or a matrix representing the skill function. It consists of an item indicator and a problem-by-skill indicator matrix.
`itemID`	index of the column in `skillfun` that holds the item indicator.

Details

The skill function (Q, S, μ) indicates for each item in Q which subsets of skills in S are required to solve the item. Thus, $\mu(q)$ is a set containing sets of skills. An item may have multiple entries in `skillfun`, each in a separate row identified by the same `itemID`.

See Doignon and Falmagne (1999, Chap. 4).

Value

A list of two components:

K	the knowledge structure delineated by the skill function.
classes	a list of equivalence classes of competence states; the members of these classes are mapped onto the same knowledge state by the problem function induced by the skill function μ.

References

Doignon, J.-P., & Falmagne, J.-C. (1999). *Knowledge spaces*. Berlin: Springer.

See Also

blim.

Examples

```
# Skill function
# mu(e) = {{s, t}, {s, u}},  mu(f) = {{u}}
# mu(g) = {{s}, {t}},        mu(h) = {{t}}
sf <- read.table(header = TRUE, text = "
  item s t u
     e 1 1 0
     e 1 0 1
     f 0 0 1
     g 1 0 0
     g 0 1 0
     h 0 1 0
")
delineate(sf)

## See ?probability for further examples.
```

DoignonFalmagne7 *Artificial Responses from Doignon and Falmagne (1999)*

Description

Fictitious data set from Doignon and Falmagne (1999, chap. 7). Response patterns of 1000 respondents to five problems. Each respondent is assumed to be in one of nine possible states of the knowledge structure K.

Usage

```
data(DoignonFalmagne7)
```

Format

A list consisting of two components:

`K` a state-by-problem indicator matrix representing the hypothetical knowledge structure. An element is one if the problem is contained in the state, and else zero.

`N.R` a named numeric vector. The names denote response patterns, the values denote their frequencies.

Source

Doignon, J.-P., & Falmagne, J.-C. (1999). *Knowledge spaces*. Berlin: Springer.

Examples

```
data(DoignonFalmagne7)
DoignonFalmagne7$K    # knowledge structure
DoignonFalmagne7$N.R  # response patterns
```

---

endm *Responses and Knowledge Structures from Heller and Wickelmaier (2013)*

Description

Knowledge structures and 200 artificial responses to four problems that are used to illustrate parameter estimation in Heller and Wickelmaier (2013).

Usage

```
data(endm)
```

Format

A list consisting of three components:

`K` a state-by-problem indicator matrix representing the true knowledge structure that underlies the model that generated the data.

`K2` a slightly misspecified knowledge structure.

`N.R` a named numeric vector. The names denote response patterns, the values denote their frequencies.

Source

Heller, J., & Wickelmaier, F. (2013). Minimum discrepancy estimation in probabilistic knowledge structures. *Electronic Notes in Discrete Mathematics*, **42**, 49–56. doi:10.1016/j.endm.2013.05.145

Examples

```
data(endm)
endm$K    # true knowledge structure
endm$K2   # misspecified knowledge structure
endm$N.R  # response patterns

## Generate data from BLIM based on K
blim0 <- list(
     P.K = setNames(c(.1, .15, .15, .2, .2, .1, .1), as.pattern(endm$K)),
    beta = rep(.1, 4),
     eta = rep(.1, 4),
       K = endm$K,
  ntotal = 200)
class(blim0) <- "blim"
simulate(blim0)

## Fit BLIM based on K2
blim1 <- blim(endm$K2, endm$N.R, "MD")
```

`getKFringe` *Outer and Inner Fringes of a Knowledge Structure*

Description

Returns the outer or inner fringe for each state in a knowledge structure.

Usage

```
getKFringe(K, nstates = nrow(K), nitems = ncol(K), outer = TRUE)
```

Arguments

`K`	a state-by-problem indicator matrix representing the knowledge structure. An element is one if the problem is contained in the state, and else zero.
`nstates`	the number of knowledge states in `K`.
`nitems`	the number of items in `K`.
`outer`	logical. If `TRUE` return outer fringe, else return inner fringe.

Details

The outer fringe of a knowledge state is the set of all items that can be learned from that state, such that adding an outer-fringe item to the state results in another state in `K`,

$$K^O = \{q \notin K | K \cup \{q\} \in \mathcal{K}\}.$$

The inner fringe of a knowledge state is the set of all items that have been learned most recently to reach that state, such that deleting an inner-fringe item from the state results in another state in `K`,

$$K^I = \{q \in K | K - \{q\} \in \mathcal{K}\}.$$

Value

A state-by-problem indicator matrix representing the outer or inner fringe for each knowledge state in K.

See Also

slm, simulate.blim.

Examples

```
data(DoignonFalmagne7)

## Which items can be learned from each state?
getKFringe(DoignonFalmagne7$K)

## Which items in each state have been recently learned?
getKFringe(DoignonFalmagne7$K, outer = FALSE)
```

---

gradedness *Forward-/Backward-Gradedness and Downgradability of a Knowledge Structure*

Description

Checks if a knowledge structure is

- forward- or backward-graded in any item;
- downgradable.

Usage

```
is.forward.graded(K)

is.backward.graded(K)

is.downgradable(K)
```

Arguments

K	a state-by-problem indicator matrix representing the knowledge structure. An element is one if the problem is contained in the state, and else zero. K should have non-empty colnames.

Details

A knowledge structure K is forward-graded in item q, if $S \cup \{q\}$ is in K for every state $S \in K$. A knowledge structure K is backward-graded in item q, if $S - \{q\}$ is in K for every state $S \in K$. See Spoto, Stefanutti, and Vidotto (2012).

A knowledge structure K is downgradable, if its inner fringe is empty only for a single state (the empty set). See Doignon and Falmagne (2015).

Value

For forward- and backward-gradedness, a named logical vector with as many elements as columns in `K`.

For downgradability, a single logical value.

References

Doignon, J.-P., & Falmagne, J.-C. (2015). Knowledge spaces and learning spaces. *arXiv*. doi:10.48550/arXiv.1511.06757

Spoto, A., Stefanutti, L., & Vidotto, G. (2012). On the unidentifiability of a certain class of skill multi map based probabilistic knowledge structures. *Journal of Mathematical Psychology*, **56**(4), 248–255. doi:10.1016/j.jmp.2012.05.001

See Also

`blim`, `jacobian`, `getKFringe`.

Examples

```
K <- as.binmat(c("0000", "1000", "1100", "1010", "0110", "1110", "1111"))
is.forward.graded(K)                  # forward-graded in a
is.backward.graded(K)                 # not backward-graded in a
is.downgradable(K)                    # not downgradable
all(K[, "a"] | getKFringe(K)[, "a"])  # every K or outer fringe contains a
```

---

hsgeometry *Responses to Geometry Problems and Knowledge Structures*

Description

Lakshminarayan (1995, 1996) presented high school geometry problems to 959 undergraduate students before and after a lesson. The problems deal with basic concepts in geometry, such as angles, lengths, circles, and polygons. Doignon and Falmagne (1999, chap. 8) refer to a subset of these data.

Usage

```
data(hsgeometry)
```

Format

Two lists, each consisting of three components:

`angles` a list with components `K`, `N.R`, and `changescore` for five problems mostly about angles.

`circles` a list with components `K`, `N.R`, and `changescore` for four problems mostly about circles.

`K` a state-by-problem indicator matrix representing the knowledge structure in Lakshminarayan (1995, 1996).

`N.R` a matrix. The row names denote response patterns, the column entries their frequencies before and after the lesson.

`changescore` a 3d table. For each problem, it contains the number of students who solved or did not solve the problem at pre- and posttest.

Note

Only those 800 responses used to estimate the model parameters are reported (the held-out sample is missing).

Thanks to Kamakshi Lakshminarayan, Cord Hockemeyer, and Mike Regenwetter for their help with finding the data.

Source

Lakshminarayan, K (1996). *A hybrid latent trait and latent class model of learning - Theoretical details and empirical application*. Technical report No. MBS 96-07. University of Califomia, Irvine.

References

Doignon, J.-P., & Falmagne, J.-C. (1999). *Knowledge spaces*. Berlin: Springer.

Lakshminarayan, K. (1995). *Theoretical and empirical aspects of some stochastic learning models*. Doctoral dissertation. University of California, Irvine.

Examples

```
data(hsgeometry)
dotchart(angles$N.R[, "posttest"], pch = 4,
         main = "High school geometry problems",
         xlab = "Response frequency (pre- [o] and posttest [x])")
points(angles$N.R[, "pretest"], 1:32)
mtext("Lakshminarayan (1995)", side = 3, line = 0.5)
```

`ita` *Item Tree Analysis (ITA)*

Description

Item tree analysis (ITA) on a set of binary responses.

Usage

```
ita(R, L = NULL, makeK = FALSE, search = c("local", "global"))
```

Arguments

`R`	a subject-by-problem indicator matrix representing the responses.
`L`	the threshold of violations acceptable for the precedence relation. If `NULL` (default), an optimal threshold is searched for.
`makeK`	should the corresponding knowledge structure be returned?
`search`	local (default) or global threshold search.

Details

ITA seeks to establish a precedence relation among a set of binary items. For each pair of items (p, q), it counts how often p is not solved if q is solved, which constitutes a violation of the relation. ITA searches for a threshold `L` for the maximum number of violations consistent with a (transitive) precedence relation. Its attempts to minimize the total discrepancy between `R` and `K`.

See van Leeuwe (1974) and Schrepp (1999) for details.

Value

An object of class `ita` having the following components:

`K`	the knowledge structure corresponding to the precedence relation.
`discrepancy`	the discrepancy between `R` and `K` (fit), between `K` and `R` (complexity), and their sum (total).
`transitiveL`	the vector of transitive thresholds.
`searchL`	either `NULL` or the method used for threshold search.
`L`	the selected or requested threshold.
`P`	the precedence matrix containing the number of violations.
`I`	the precedence relation as a logical incidence matrix at threshold `L`.

References

Schrepp, M. (1999). On the empirical construction of implications between bi-valued test items. *Mathematical Social Sciences*, **38**(3), 361–375. doi:10.1016/S01654896(99)000256

Van Leeuwe, J.F. (1974). Item tree analysis. *Nederlands Tijdschrift voor de Psychologie en haar Grensgebieden*, **29**(6), 475–483.

See Also

blim.

Examples

```
data(chess)

ita(chess$R)  # find (locally) optimal threshold L

i <- ita(chess$R, L = 6, makeK = TRUE)
identical(sort(as.pattern(i$K)),
          sort(as.pattern(chess$dst1)))

## Plotting the precedence relation
if(requireNamespace("relations") &&
   requireNamespace("Rgraphviz")) {
  plot(relations::as.relation(i$I))
}
```

---

jacobian *Jacobian Matrix for Basic Local Independence Model*

Description

Computes the Jacobian matrix for a basic local independence model (BLIM).

Usage

```
jacobian(object, P.K = rep(1/nstates, nstates),
         beta = rep(0.1, nitems), eta = rep(0.1, nitems),
         betafix = rep(NA, nitems), etafix = rep(NA, nitems))
```

Arguments

object	an object of class blim, typically the result of a call to blim.
P.K	the vector of parameter values for probabilities of knowledge states.
beta	the vector of parameter values for probabilities of a careless error.
eta	the vector of parameter values for probabilities of a lucky guess.
betafix, etafix	vectors of fixed error and guessing parameter values; NA indicates a free parameter.

Details

This is a draft version. It may change in future releases.

Value

The Jacobian matrix. The number of rows equals 2^(number of items) - 1, the number of columns equals the number of independent parameters in the model.

References

Heller, J. (2017). Identifiability in probabilistic knowledge structures. *Journal of Mathematical Psychology*, **77**, 46–57. doi:10.1016/j.jmp.2016.07.008

Stefanutti, L., Heller, J., Anselmi, P., & Robusto, E. (2012). Assessing the local identifiability of probabilistic knowledge structures. *Behavior Research Methods*, **44**(4), 1197–1211. doi:10.3758/s134280120187z

See Also

`blim`, `simulate.blim`, `gradedness`.

Examples

```
data(endm)
m <- blim(endm$K2, endm$N.R)

## Test of identifiability
J <- jacobian(m)
dim(J)
qr(J)$rank
```

---

plot.blim *Diagnostic Plot for Basic Local Independence Models*

Description

Plots BLIM residuals against fitted values.

Usage

```
## S3 method for class 'blim'
plot(x, xlab = "Predicted response probabilities",
     ylab = "Deviance residuals", ...)
```

Arguments

`x`	an object of class `blim`, typically the result of a call to `blim`.
`xlab`, `ylab`, `...`	graphical parameters passed to plot.

Details

The deviance residuals are plotted against the predicted response probabilities for each response pattern.

See Also

blim, residuals.blim.

Examples

```
## Compare MD and MDML estimation

data(DoignonFalmagne7)
blim1 <- blim(DoignonFalmagne7$K, DoignonFalmagne7$N.R, method="MD")
blim2 <- blim(DoignonFalmagne7$K, DoignonFalmagne7$N.R, method="MDML")

par(mfrow = 1:2)      # residuals versus fitted values
plot(blim1, main = "MD estimation",   ylim = c(-4, 4))
plot(blim2, main = "MDML estimation", ylim = c(-4, 4))
```

---

predict.blim *Predict Knowledge States from Basic Local Independence Models (BLIMs)*

Description

Predict knowledge state or state probabilities based on a fitted blim object given a response pattern.

Usage

```
## S3 method for class 'blim'
predict(object, newdata = NULL, type = c("state", "probs"),
        method = c("ML", "MD", "MDML"), quiet = FALSE,
        ties.method = c("min", "max", "random"), i.RK = NULL,
        incradius = object$incradius, as.pattern = TRUE, ...)
```

Arguments

object	an object of class blim, typically the result of a call to blim.
newdata	a character vector of response patterns with which to predict.
type	for each pattern, predict a knowledge state or the state probabilities.
method	how to compute the posterior state probabilities. (See Details.)
quiet	silence message when estimation and prediction methods differ.
ties.method	how to deal with a posteriori equally probable states.
i.RK	optional indicator matrix of states at minimum distance from response patterns.
incradius	see blim.
as.pattern	return a character vector via as.pattern or an indicator matrix.
...	further arguments passed to as.pattern.

Details

Predicted is the modal posterior state (`type = "state"`) or the posterior distribution (`type = "probs"`) of knowledge states given a response pattern. Depending on the `method` argument, the posterior distribution is defined as:

- Maximum likelihood (`method = "ML"`)

$$P(K|R)_{ML} = \frac{P(R|K)P(K)}{\sum_K P(R|K)P(K)} = \frac{P(R|K)P(K)}{P(R)}$$

- Minimum discrepancy (`method = "MD"`)

$$P(K|R)_{MD} = \frac{i_{RK}}{\sum_K i_{RK}}$$

- Minimum discrepancy ML (`method = "MDML"`)

$$P(K|R)_{MDML} = \frac{i_{RK} \cdot P(K|R)_{ML}}{\sum_K i_{RK} \cdot P(K|R)_{ML}}$$

where i_{RK} is a pattern-by-state indicator matrix that is one for each state K that is at minimum distance from pattern R.

Value

If `type = "state"`, a character vector of knowledge states (if `as.pattern = TRUE`) or a state-by-problem indicator matrix (if `as.pattern = FALSE`).

If `type = "probs"`, a matrix of posterior state probabilities.

See Also

`blim`, `slm`.

Examples

```
data(DoignonFalmagne7)
m <- blim(DoignonFalmagne7$K, DoignonFalmagne7$N.R)
predict(m)
predict(m, newdata = c("00100", "10100", "01111"), as.pattern = FALSE)

m <- slm(DoignonFalmagne7$K, DoignonFalmagne7$N.R)
predict(m, newdata = "00100")

data(endm)
m <- blim(endm$K, endm$N.R)
predict(m, type = "probs", method = "MD")
cbind(
  observed = names(m$N.R),
  min = predict(m, method = "MD", ties.method = "min"),
  max = predict(m, method = "MD", ties.method = "max"),
  rnd = predict(m, method = "MD", ties.method = "random")
) |> print(quote = FALSE)
```

print.blim *Print a blim Object*

Description

Prints the output of a blim model object.

Usage

```
## S3 method for class 'blim'
print(x, P.Kshow = FALSE, errshow = TRUE,
      digits=max(3, getOption("digits") - 2), ...)
```

Arguments

x	an object of class blim, typically the result of a call to blim.
P.Kshow	logical, should the estimated distribution of knowledge states be printed?
errshow	logical, should the estimates of careless error and lucky guess parameters be printed?
digits	a non-null value for digits specifies the minimum number of significant digits to be printed in values.
...	further arguments passed to or from other methods. None are used in this method.

Value

Returns the blim object invisibly.

See Also

blim.

Examples

```
data(DoignonFalmagne7)

blim1 <- blim(DoignonFalmagne7$K, DoignonFalmagne7$N.R)
print(blim1, showP.K = TRUE)
```

probability *Problems in Elementary Probability Theory*

Description

This data set contains responses to problems in elementary probability theory observed before and after some instructions (the so-called learning object) were given. Data were collected both in the lab and via an online questionnaire. Of the 1127 participants eligible in the online study, 649 were excluded because they did not complete the first set of problems (p101, ..., p112) or they responded too quickly or too slowly. Based on similar criteria, further participants were excluded for the second set of problems, indicated by missing values in the variables b201, ..., b212. Problems were presented in random order.

Participants were randomized to two conditions: an enhanced learning object including instructions with examples and a basic learning object without examples. Instructions were given on four concepts: how to calculate the classic probability of an event (pb), the probability of the complement of an event (cp), of the union of two disjoint events (un), and of two independent events (id).

The questionnaire was organized as follows:

Page 1 Welcome page.

Page 2 Demographic data.

Page 3 First set of problems.

Page 4 to 8 Instructions (learning object).

Page 9 Second set of problems.

Page 10 Feedback about number of correctly solved problems.

Usage

```
data(probability)
```

Format

A data frame with 504 cases and 68 variables:

- `case` a factor giving the case id, a five-digits code the fist digit denoting lab or online branch of the study, the last four digits being the case number.
- `lastpage` Which page of the questionnaire was reached before quitting? The questionnaire consisted of ten pages.
- `mode` a factor; `lab` or `online` branch of study.
- `started` a timestamp of class POSIXlt. When did participant start working on the questionnaire?
- `sex` a factor coding sex of participant.
- `age` age of participant.
- `educat` education as a factor with three levels: `1` secondary school or below; `2` higher education entrance qualification; `3` university degree.

- fos field of study. Factor with eight levels: ecla economics, business, law; else miscellaneous; hipo history, politics; lang languages; mabi mathematics, physics, biology; medi medical science; phth philosophy, theology; psco psychology, computer science, cognitive science.
- semester ordered factor. What semester are you in?
- learnobj a factor with two levels: enhan learning object enhanced with examples; basic learning object without examples.

The twelve problems of the first part (before the learning object):

- p101 A box contains 30 marbles in the following colors: 8 red, 10 black, 12 yellow. What is the probability that a randomly drawn marble is yellow? (Correct: 0.40)
- p102 A bag contains 5-cent, 10-cent, and 20-cent coins. The probability of drawing a 5-cent coin is 0.35, that of drawing a 10-cent coin is 0.25, and that of drawing a 20-cent coin is 0.40. What is the probability that the coin randomly drawn is not a 5-cent coin? (0.65)
- p103 A bag contains 5-cent, 10-cent, and 20-cent coins. The probability of drawing a 5-cent coin is 0.20, that of drawing a 10-cent coin is 0.45, and that of drawing a 20-cent coin is 0.35. What is the probability that the coin randomly drawn is a 5-cent coin or a 20-cent coin? (0.55)
- p104 In a school, 40% of the pupils are boys and 80% of the pupils are right-handed. Suppose that gender and handedness are independent. What is the probability of randomly selecting a right-handed boy? (0.32)
- p105 Given a standard deck containing 32 different cards, what is the probability of not drawing a heart? (0.75)
- p106 A box contains 20 marbles in the following colors: 4 white, 14 green, 2 red. What is the probability that a randomly drawn marble is not white? (0.80)
- p107 A box contains 10 marbles in the following colors: 2 yellow, 5 blue, 3 red. What is the probability that a randomly drawn marble is yellow or blue? (0.70)
- p108 What is the probability of obtaining an even number by throwing a dice? (0.50)
- p109 Given a standard deck containing 32 different cards, what is the probability of drawing a 4 in a black suit? (Responses that round to 0.06 were considered correct.)
- p110 A box contains marbles that are red or yellow, small or large. The probability of drawing a red marble is 0.70 (lab: 0.30), the probability of drawing a small marble is 0.40. Suppose that the color of the marbles is independent of their size. What is the probability of randomly drawing a small marble that is not red? (0.12, lab: 0.28)
- p111 In a garage there are 50 cars. 20 are black and 10 are diesel powered. Suppose that the color of the cars is independent of the kind of fuel. What is the probability that a randomly selected car is not black and it is diesel powered? (0.12)
- p112 A box contains 20 marbles. 10 marbles are red, 6 are yellow and 4 are black. 12 marbles are small and 8 are large. Suppose that the color of the marbles is independent of their size. What is the probability of randomly drawing a small marble that is yellow or red? (0.48)

The twelve problems of the second part (after the learning object):

- p201 A box contains 30 marbles in the following colors: 10 red, 14 yellow, 6 green. What is the probability that a randomly drawn marble is green? (0.20)

- p202 A bag contains 5-cent, 10-cent, and 20-cent coins. The probability of drawing a 5-cent coin is 0.25, that of drawing a 10-cent coin is 0.60, and that of drawing a 20-cent coin is 0.15. What is the probability that the coin randomly drawn is not a 5-cent coin? (0.75)
- p203 A bag contains 5-cent, 10-cent, and 20-cent coins. The probability of drawing a 5-cent coin is 0.35, that of drawing a 10-cent coin is 0.20, and that of drawing a 20-cent coin is 0.45. What is the probability that the coin randomly drawn is a 5-cent coin or a 20-cent coin? (0.80)
- p204 In a school, 70% of the pupils are girls and 10% of the pupils are left-handed. Suppose that gender and handedness are independent. What is the probability of randomly selecting a left-handed girl? (0.07)
- p205 Given a standard deck containing 32 different cards, what is the probability of not drawing a club? (0.75)
- p206 A box contains 20 marbles in the following colors: 6 yellow, 10 red, 4 green. What is the probability that a randomly drawn marble is not yellow? (0.70)
- p207 A box contains 10 marbles in the following colors: 5 blue, 3 red, 2 green. What is the probability that a randomly drawn marble is blue or red? (0.80)
- p208 What is the probability of obtaining an odd number by throwing a dice? (0.50)
- p209 Given a standard deck containing 32 different cards, what is the probability of drawing a 10 in a red suit? (Responses that round to 0.06 were considered correct.)
- p210 A box contains marbles that are green or red, large or small The probability of drawing a green marble is 0.40, the probability of drawing a large marble is 0.20. Suppose that the color of the marbles is independent of their size. What is the probability of randomly drawing a large marble that is not green? (0.12)
- p211 In a garage there are 50 cars. 15 are white and 20 are diesel powered. Suppose that the color of the cars is independent of the kind of fuel. What is the probability that a randomly selected car is not white and it is diesel powered? (0.28)
- p212 A box contains 20 marbles. 8 marbles are white, 4 are green and 8 are red. 15 marbles are small and 5 are large. Suppose that the color of the marbles is independent of their size. What is the probability of randomly drawing a large marble that is white or green? (0.15)

Further variables:

- time01, ..., time10 the time (in s) spent on each page of the questionnaire. In the lab branch of the study, participants started directly on Page 2.
- b101, ..., b112 the twelve problems of the first part coded as correct (1) or error (0).
- b201, ..., b212 the twelve problems of the second part coded as correct (1) or error (0).

Source

Data were collected by Pasquale Anselmi and Florian Wickelmaier at the Department of Psychology, University of Tuebingen, in February and March 2010.

Examples

```
data(probability)

## "Completer" sample
```

```
pb <- probability[!is.na(probability$b201), ]

## Response frequencies for first and second part
N.R1 <- as.pattern(pb[, sprintf("b1%.2i", 1:12)], freq = TRUE)
N.R2 <- as.pattern(pb[, sprintf("b2%.2i", 1:12)], freq = TRUE)

## Conjunctive skill function, one-to-one problem function
sf1 <- read.table(header = TRUE, text = "
  item cp id pb un
     1  0  0  1  0
     2  1  0  0  0
     3  0  0  0  1
     4  0  1  0  0
     5  1  0  1  0
     6  1  0  1  0
     7  0  0  1  1
     8  0  0  1  1
     9  0  1  1  0
    10  1  1  0  0
    11  1  1  1  0
    12  0  1  1  1
")

## Extended skill function
sf2 <- rbind(sf1, read.table(header = TRUE, text = "
  item cp id pb un
     2  0  0  0  1
     3  1  0  0  0
     6  0  0  1  1
     7  1  0  1  0
    12  1  1  1  0
"))

## Delineated knowledge structures
K1 <- delineate(sf1)$K
K2 <- delineate(sf2)$K

## After instructions, fit of knowledge structures improves
sapply(list(N.R1, N.R2), function(n) blim(K1, n)$discrepancy)
sapply(list(N.R1, N.R2), function(n) blim(K2, n)$discrepancy)
```

`residuals.blim` *Residuals for Basic Local Independence Models*

Description

Computes deviance and Pearson residuals for `blim` objects.

Usage

```
## S3 method for class 'blim'
residuals(object, type = c("deviance", "pearson"), ...)
```

Arguments

object	an object of class blim, typically the result of a call to blim.
type	the type of residuals which should be returned; the alternatives are: "deviance" (default) and "pearson".
...	further arguments passed to or from other methods. None are used in this method.

Details

See residuals.glm for details.

Value

A named vector of residuals having as many elements as response patterns.

See Also

blim, residuals.glm, plot.blim.

Examples

```
data(DoignonFalmagne7)
blim1 <- blim(DoignonFalmagne7$K, DoignonFalmagne7$N.R)

sum( resid(blim1)^2 )                # likelihood ratio G2
sum( resid(blim1, "pearson")^2 )     # Pearson X2
```

---

schoolarithm *Arithmetic Problems for Elementary and Middle School Students*

Description

The 23 fraction problems were presented to 191 first-level middle school students (about 11 to 12 years old). A subset of 13 problems is included in Stefanutti and de Chiusole (2017).

The eight subtraction problems were presented to 294 elementary school students and are described in de Chiusole and Stefanutti (2013).

Usage

```
data(schoolarithm)
```

Format

`fraction17` a person-by-problem indicator matrix representing the responses of 191 persons to 23 problems. The responses are classified as correct (0) or incorrect (1).

The 23 problems were:

- `p01` $\left(\frac{1}{3}+\frac{1}{12}\right):\frac{2}{9}=?$
- `p02` $\left(\frac{3}{2}+\frac{3}{4}\right)\times\frac{5}{3}-2=?$
- `p03` $\left(\frac{5}{6}+\frac{3}{14}\right)\times\left(\frac{19}{8}-\frac{3}{2}\right)=?$
- `p04` $\left(\frac{1}{6}+\frac{2}{9}\right)-\frac{7}{36}=?$
- `p05` $\frac{7}{10}+\frac{9}{10}=?$
- `p06` $\frac{8}{13}+\frac{5}{2}=?$
- `p07` $\frac{8}{12}+\frac{4}{15}=?$
- `p08` $\frac{2}{9}+\frac{5}{6}=?$
- `p09` $\frac{7}{5}+\frac{1}{5}=?$
- `p10` $\frac{2}{7}+\frac{3}{14}=?$
- `p11` $\frac{5}{9}+\frac{1}{6}=?$
- `p12` $\left(\frac{1}{12}+\frac{1}{3}\right)\times\frac{24}{15}=?$
- `p13` $2-\frac{3}{4}=?$
- `p14` $\left(4+\frac{3}{4}-\frac{1}{2}\right)\times\frac{8}{6}=?$
- `p15` $\frac{4}{7}+\frac{3}{4}=\frac{?}{28}$
- `p16` $\frac{5}{8}-\frac{3}{16}=\frac{?-?}{16}$
- `p17` $\frac{3}{8}+\frac{5}{12}=\frac{?\times3+?\times5}{24}$
- `p18` $\frac{2}{7}+\frac{3}{5}=\frac{5\times?+7\times?}{35}$
- `p19` $\frac{2}{3}+\frac{6}{9}=\frac{?}{9}=\frac{?}{?}$
- `p20` Least common multiple $lcm(6,8)=?$
- `p21` $\frac{7}{11}\times\frac{2}{3}=?$
- `p22` $\frac{2}{5}\times\frac{15}{4}=?$
- `p23` $\frac{9}{7}:\frac{2}{3}=?$

`subtraction13` is a data frame consisting of the following components:

`School` factor; school id.

`Classroom` factor; class room id.

`Gender` factor; participant gender.

`Age` participant age.

`R` a person-by-problem indicator matrix representing the responses of 294 persons to eight problems.

The eight problems were:

- `p1` $73-58$

- p2 $317 - 94$
- p3 $784 - 693$
- p4 $507 - 49$
- p5 $253 - 178$
- p6 $2245 - 418$
- p7 $156 - 68$
- p8 $3642 - 753$

Source

The data were made available by Debora de Chiusole, Andrea Brancaccio, and Luca Stefanutti.

References

de Chiusole, D., & Stefanutti, L. (2013). Modeling skill dependence in probabilistic competence structures. *Electronic Notes in Discrete Mathematics*, **42**, 41–48. doi:10.1016/j.endm.2013.05.144

Stefanutti, L., & de Chiusole, D. (2017). On the assessment of learning in competence based knowledge space theory. *Journal of Mathematical Psychology*, **80**, 22–32. doi:10.1016/j.jmp.2017.08.003

Examples

```
data(schoolarithm)

## Fraction problems used in Stefanutti and de Chiusole (2017)
R <- fraction17[, c(4:8, 10:11, 15:20)]
colnames(R) <- 1:13
N.R <- as.pattern(R, freq = TRUE)

## Conjunctive skill function in Table 1
sf <- read.table(header = TRUE, text = "
  item  a  b  c  d  e  f  g  h
     1  1  1  1  0  1  1  0  0
     2  1  0  0  0  0  0  1  1
     3  1  1  0  1  1  0  0  0
     4  1  1  0  0  1  1  1  1
     5  1  1  0  0  1  1  0  0
     6  1  1  1  0  1  0  1  1
     7  1  1  0  0  1  1  0  0
     8  1  1  0  0  1  0  1  1
     9  0  1  0  0  1  0  0  0
    10  0  1  0  0  0  0  0  0
    11  0  0  0  0  1  0  0  0
    12  1  1  0  0  1  0  1  1
    13  0  0  0  0  0  1  0  0
")
K <- delineate(sf)$K  # delineated knowledge structure
blim(K, N.R)

## Subtraction problems used in de Chiusole and Stefanutti (2013)
```

```
N.R <- as.pattern(subtraction13$R, freq = TRUE)

# Skill function in Table 1
# (f) mastering tens and hundreds; (g) mastering thousands; (h1) one borrow;
# (h2) two borrows; (h3) three borrows; (i) mastering the proximity of
# borrows; (j) mastering the presence of the zero; (k) mental calculation
sf <- read.table(header = TRUE, text = "
  item  f  g h1 h2 h3  i  j  k
     1  0  0  1  0  0  0  0  0
     2  1  0  1  0  0  0  0  0
     3  1  0  1  0  0  1  0  0
     4  1  0  1  1  1  0  1  0
     4  0  0  0  0  0  0  0  1
     5  1  0  1  1  1  1  0  0
     6  1  1  1  1  0  0  0  0
     7  1  0  1  1  1  1  0  0
     8  1  1  1  1  1  0  0  0
")
K <- delineate(sf)$K
blim(K, N.R)
```

`simulate.blim` *Simulate Responses from Basic Local Independence Models (BLIMs)*

Description

Simulates response frequencies from the distribution corresponding to a fitted `blim` model object. Alternatively, if sufficient information is provided, it generates response frequencies from scratch (see examples).

Usage

```
## S3 method for class 'blim'
simulate(object, nsim = 1, seed = NULL, ...,
         method = c("bernoulli", "multinomial"), zeropad = FALSE)
```

Arguments

`object`	an object of class `blim`, typically the result of a call to `blim`.
`nsim`	currently not used.
`seed`	currently not used.
`...`	further arguments passed to or from other methods. None are used in this method.
`method`	`bernoulli` (default) or `multinomial` sampling.
`zeropad`	logical, if `TRUE` non-sampled response patterns have a frequency of zero, else (default) they are omitted.

Details

Bernoulli sampling: Responses are simulated in two steps: First, a knowledge state is drawn with probability `P.K`. Second, responses are generated by applying `rbinom` with probabilities computed from the model object's `beta` and `eta` components. Faster with a large number of items.

Multinomial sampling: The probability of all response patterns is computed from the model object's parameters, and response frequencies are obtained from `rmultinom`. Faster with few items but a large number of respondents.

Value

A named vector of frequencies of response patterns.

See Also

`blim`, `endm`.

Examples

```
data(DoignonFalmagne7)

m1 <- blim(DoignonFalmagne7$K, DoignonFalmagne7$N.R)
simulate(m1)
simulate(m1, method = "multinomial", zeropad = TRUE)

## Parametric bootstrap for the BLIM
disc <- replicate(200, blim(m1$K, simulate(m1))$discrepancy)

hist(disc, col = "lightgray", border = "white", freq = FALSE, breaks = 20,
     main = "BLIM parametric bootstrap", xlim = c(.05, .3))
abline(v = m1$discrepancy, lty = 2)

## Parameter recovery for the SLM
m0 <- list( P.K = getSlmPK( g = rep(.8, 5),
                            K = DoignonFalmagne7$K,
                           Ko = getKFringe(DoignonFalmagne7$K)),
           beta = rep(.1, 5),
            eta = rep(.1, 5),
              K = DoignonFalmagne7$K,
         ntotal = 800)
class(m0) <- c("slm", "blim")

pars <- replicate(20, coef(slm(m0$K, simulate(m0), method = "ML")))
boxplot(t(pars), horizontal = TRUE, las = 1,
        main = "SLM parameter recovery")

## See ?endm for further examples.
```

slm *Simple Learning Models (SLMs)*

Description

Fits a simple learning model (SLM) for probabilistic knowledge structures by minimum discrepancy maximum likelihood estimation.

Usage

```
slm(K, N.R, method = c("MD", "ML", "MDML"), R = as.binmat(N.R),
    beta = rep(0.1, nitems), eta = rep(0.1, nitems),
    g = rep(0.1, nitems),
    betafix = rep(NA, nitems), etafix = rep(NA, nitems),
    betaequal = NULL, etaequal = NULL,
    randinit = FALSE, incradius = 0,
    tol = 1e-07, maxiter = 10000, zeropad = 16,
    checkK = TRUE)

getSlmPK(g, K, Ko)

## S3 method for class 'slm'
print(x, P.Kshow = FALSE, parshow = TRUE,
      digits=max(3, getOption("digits") - 2), ...)
```

Arguments

K	a state-by-problem indicator matrix representing the knowledge space. An element is one if the problem is contained in the state, and else zero.
N.R	a (named) vector of absolute frequencies of response patterns.
method	MD for minimum discrepancy estimation, ML for maximum likelihood estimation, MDML for minimum discrepancy maximum likelihood estimation.
R	a pattern-by-problem indicator matrix of unique response patterns. Per default inferred from the names of N.R.
beta, eta, g	vectors of initial values for the error, guessing, and solvability parameters.
betafix, etafix	vectors of fixed error and guessing parameter values; NA indicates a free parameter.
betaequal, etaequal	lists of vectors of problem indices; each vector represents an equivalence class: it contains the indices of problems for which the error or guessing parameters are constrained to be equal. (See Examples.)
randinit	logical, if TRUE then initial parameter values are sampled uniformly with constraints. (See Details.)
incradius	include knowledge states of distance from the minimum discrepant states less than or equal to incradius.

tol	tolerance, stopping criterion for iteration.
maxiter	the maximum number of iterations.
zeropad	the maximum number of items for which an incomplete N.R vector is completed and padded with zeros.
checkK	logical, if TRUE K is checked for well-gradedness.
Ko	a state-by-problem indicator matrix representing the outer fringe for each knowledge state in K; typically the result of a call to getKFringe.
x	an object of class slm, typically the result of a call to slm.
P.Kshow	logical, should the estimated distribution of knowledge states be printed?
parshow	logical, should the estimates of error, guessing, and solvability parameters be printed?
digits	a non-null value for digits specifies the minimum number of significant digits to be printed in values.
...	additional arguments passed to other methods.

Details

See Doignon and Falmagne (1999) for details on the simple learning model (SLM) for probabilistic knowledge structures. The model requires a well-graded knowledge space K.

An slm object inherits from class blim. See blim for details on the function arguments. The helper function getSlmPK returns the distribution of knowledge states P.K.

Value

An object of class slm and blim. It contains all components of a blim object. In addition, it includes:

g	the vector of estimates of the solvability parameters.

References

Doignon, J.-P., & Falmagne, J.-C. (1999). *Knowledge spaces*. Berlin: Springer.

See Also

blim, simulate.blim, getKFringe, is.downgradable

Examples

```
data(DoignonFalmagne7)
K   <- DoignonFalmagne7$K     # well-graded knowledge space
N.R <- DoignonFalmagne7$N.R   # frequencies of response patterns

## Fit simple learning model (SLM) by different methods
slm(K, N.R, method = "MD")    # minimum discrepancy estimation
slm(K, N.R, method = "ML")    # maximum likelihood estimation by EM
slm(K, N.R, method = "MDML")  # MDML estimation
```

```
## Compare SLM and BLIM
m1 <-  slm(K, N.R, method = "ML")
m2 <- blim(K, N.R, method = "ML")
anova(m1, m2)
```

`Taagepera` *Responses and Knowledge Structures from Taagepera et al. (1997)*

Description

Taagepera et al. (1997) applied knowledge space theory to specific science problems. The density test was administered to 2060 students, the conservation of matter test to 1620 students. A subtest of five items each is included here. The response frequencies were reconstructed from histograms in the paper.

Usage

```
data(Taagepera)
```

Format

Two lists, each consisting of two components:

`density97` a list with components `K` and `N.R` for the density test.

`matter97` a list with components `K` and `N.R` for the conservation of matter test.

`K` a state-by-problem indicator matrix representing the hypothetical knowledge structure. An element is one if the problem is contained in the state, and else zero.

`N.R` a named numeric vector. The names denote response patterns, the values denote their frequencies.

Source

Taagepera, M., Potter, F., Miller, G.E., & Lakshminarayan, K. (1997). Mapping students' thinking patterns by the use of knowledge space theory. *International Journal of Science Education*, **19**(3), 283–302. doi:10.1080/0950069970190303

Examples

```
data(Taagepera)
density97$K     # density test knowledge structure
density97$N.R   # density test response patterns
matter97$K      # conservation of matter knowledge structure
matter97$N.R    # conservation of matter response patterns
```

Index

∗ **datasets**
chess, 9
DoignonFalmagne7, 13
endm, 14
hsgeometry, 17
probability, 25
schoolarithm, 29
Taagepera, 36
∗ **models**
blim, 2
blimit, 5
cdp, 7
conversion, 11
delineate, 12
getKFringe, 15
gradedness, 16
ita, 19
jacobian, 20
plot.blim, 21
predict.blim, 22
print.blim, 24
residuals.blim, 28
simulate.blim, 32
slm, 34

angles *(*hsgeometry*)*, 17
anova.blim *(*blim*)*, 2
as.binmat *(*conversion*)*, 11
as.pattern, *22*
as.pattern *(*conversion*)*, 11

blim, 2, *6*, *11*, *13*, *17*, *20–24*, *29*, *32*, *33*, *35*
blimit, 5
blimMD *(*blim*)*, 2

cdp, 7
chess, *5*, 9
circles *(*hsgeometry*)*, 17
coef.blim *(*blim*)*, 2
coef.slm *(*slm*)*, 34
conversion, 11

delineate, *5*, *8*, 12
density97 *(*Taagepera*)*, 36
deviance.blim *(*blim*)*, 2
DoignonFalmagne7, 13

endm, *5*, 14, *33*

fraction17 *(*schoolarithm*)*, 29

getKFringe, 15, *17*, *35*
getSlmPK *(*slm*)*, 34
gradedness, 16, *21*

hsgeometry, 17

is.backward.graded *(*gradedness*)*, 16
is.downgradable, *35*
is.downgradable *(*gradedness*)*, 16
is.forward.graded *(*gradedness*)*, 16
is.subset *(*conversion*)*, 11
ita, 19

jacobian, *5*, *6*, *17*, 20

logLik.blim, *5*
logLik.blim *(*blim*)*, 2

matter97 *(*Taagepera*)*, 36

nobs.blim *(*blim*)*, 2

plot.blim, *5*, 21, *29*
predict.blim, *5*, 22
predict.slm *(*predict.blim*)*, 22
print.blim, 24
print.cdp *(*cdp*)*, 7
print.ita *(*ita*)*, 19
print.slm *(*slm*)*, 34
probability, *5*, 25

rbinom, *33*
residuals.blim, *5*, *22*, 28
residuals.glm, *29*
rmultinom, *33*

schoolarithm, 29
simulate.blim, *5*, *16*, *21*, 32, *35*
slm, *16*, *23*, 34
subtraction13 *(*schoolarithm*)*, 29

Taagepera, 36