Package ‘hmcdm’

January 25, 2023

Type Package

Title Hidden Markov Cognitive Diagnosis Models for Learning

Version 2.1.0

Description Fitting hidden Markov models of learning under the cognitive diagnosis framework.
The estimation of the hidden Markov diagnostic classification model,
the first order hidden Markov model, the reduced-reparameterized unified learning model,
and the joint learning model for responses and response times.
Chen, Y., Culpepper, S. A., Wang, S., & Douglas, J. (2018) <doi:10.1177/0146621617721250>.
Wang, S., Yang, Y., Culpepper, S. A., & Douglas, J. A. (2018) <doi:10.3102/1076998617719727>.
Wang, S., Zhang, S., Douglas, J., & Culpepper, S. (2018) <doi:10.1080/15366367.2018.1435105>.
Zhang, S., Douglas, J. A., Wang, S. & Culpepper, S. A. (2019) <doi:10.1007/978-3-030-05584-4_24>.

License GPL (>= 2)

URL `https://github.com/tmsalab/hmcdm`

BugReports `https://github.com/tmsalab/hmcdm/issues`

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0), stats (>= 3.0.0), bayesplot (>= 1.9.0),
rstantools (>= 1.0.0)

LinkingTo Rcpp, RcppArmadillo, progress

SystemRequirements C++11

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Susu Zhang [aut],
Shiyu Wang [aut],
Yinghan Chen [aut],
Sunbeom Kwon [aut, cre]

Maintainer Sunbeom Kwon <sunbeom2@illinois.edu>

Repository CRAN

Date/Publication 2023-01-25 21:20:05 UTC

R topics documented:

`hmcdm-package` *hmcdm: Hidden Markov Cognitive Diagnosis Models for Learning*

Description

Fitting hidden Markov models of learning under the cognitive diagnosis framework. The estimation of the hidden Markov diagnostic classification model, the first order hidden Markov model, the reduced-reparameterized unified learning model, and the joint learning model for responses and response times. Chen, Y., Culpepper, S. A., Wang, S., & Douglas, J. (2018) doi:10.1177/0146621617721250. Wang, S., Yang, Y., Culpepper, S. A., & Douglas, J. A. (2018) doi:10.3102/1076998617719727. Wang, S., Zhang, S., Douglas, J., & Culpepper, S. (2018) doi:10.1080/15366367.2018.1435105. Zhang, S., Douglas, J. A., Wang, S. & Culpepper, S. A. (2019) doi:10.1007/9783030055844_24.

Author(s)

Maintainer: Sunbeom Kwon <sunbeom2@illinois.edu>

Authors:

- Susu Zhang <szhan105@illinois.edu>
- Shiyu Wang <swang44@uga.edu>
- Yinghan Chen <yinghanc@unr.edu >

References

Wang, S., Yang, Y., Culpepper, S. A., & Douglas, J. A. (2018) doi:10.3102/1076998617719727 "Tracking Skill Acquisition With Cognitive Diagnosis Models: A Higher-Order, Hidden Markov Model With Covariates."

Chen, Y., Culpepper, S. A., Wang, S., & Douglas, J. (2018) doi:10.1177/0146621617721250 "A hidden Markov model for learning trajectories in cognitive diagnosis with application to spatial rotation skills."

Wang, S., Zhang, S., Douglas, J., & Culpepper, S. (2018) doi:10.1080/15366367.2018.1435105 "Using Response Times to Assess Learning Progress: A Joint Model for Responses and Response Times."

See Also

Useful links:

- https://github.com/tmsalab/hmcdm
- Report bugs at https://github.com/tmsalab/hmcdm/issues

`Design_array` *Design array*

Description

`Design_array` contains item administration information at all time points in the Spatial Rotation Learning Program.

Usage

```
Design_array
```

Format

An array of dimension N-by-J-by-L, containing each subject's item administration.

Details

The data object `"Design_array"` contains an array of dimension N-by-J-by-L indicating the items assigned (1/0) to each subject at each time point.

Author(s)

Shiyu Wang, Yan Yang, Jeff Douglas, and Steve Culpepper

Source

Spatial Rotation Learning Experiment at UIUC between Fall 2015 and Spring 2016.

`ETAmat` *Generate ideal response matrix*

Description

Based on the Q matrix and the latent attribute space, generate the ideal response matrix for each skill pattern

Usage

```
ETAmat(K, J, Q)
```

Arguments

`K`	An `int` of the number of attributes
`J`	An `int` of the number of items
`Q`	A J-by-K Q `matrix`

Value

A J-by-2^K ideal response `matrix`

Examples

```
Q = random_Q(15,4)
ETA = ETAmat(4,15,Q)
```

hmcdm *Gibbs sampler for learning models*

Description

Runs MCMC to estimate parameters of any of the listed learning models.

Usage

```
hmcdm(
  Response,
  Q_matrix,
  model,
  Design_array = NULL,
  Test_order = NULL,
  Test_versions = NULL,
  chain_length = 100L,
  burn_in = 50L,
  G_version = NA_integer_,
  theta_propose = 0,
  Latency_array = NULL,
  deltas_propose = NULL,
  R = NULL
)
```

Arguments

`Response`	An `array` of dichotomous item responses. t-th slice is an N-by-J matrix of responses at time t.
`Q_matrix`	A J-by-K Q-matrix.
`model`	A `charactor` of the type of model fitted with the MCMC sampler, possible selections are "DINA_HO": Higher-Order Hidden Markov Diagnostic Classification Model with DINA responses; "DINA_HO_RT_joint": Higher-Order Hidden Markov DCM with DINA responses, log-Normal response times, and joint modeling of latent speed and learning ability; "DINA_HO_RT_sep": Higher-Order Hidden Markov DCM with DINA responses, log-Normal response times, and separate modeling of latent speed and learning ability; "rRUM_indept": Simple independent transition probability model with rRUM responses "NIDA_indept": Simple independent transition probability model with NIDA responses "DINA_FOHM": First Order Hidden Markov model with DINA responses
`Design_array`	An `array` of dimension N-by-J-by-L indicating the items assigned (1/0) to each subject at each time point. Either 'Design_array' or both 'Test_order' & 'Test_versions' need to be provided to run HMCDM.
`Test_order`	Optional. A `matrix` of the order of item blocks for each test version.
`Test_versions`	Optional. A `vector` of the test version of each learner.

`chain_length`	An `int` of the MCMC chain length.
`burn_in`	An `int` of the MCMC burn-in chain length.
`G_version`	Optional. An `int` of the type of covariate for increased fluency (1: G is dichotomous depending on whether all skills required for current item are mastered; 2: G cumulates practice effect on previous items using mastered skills; 3: G is a time block effect invariant across subjects with different attribute trajectories)
`theta_propose`	Optional. A `scalar` for the standard deviation of theta's proposal distribution in the MH sampling step.
`Latency_array`	Optional. A `array` of the response times. t-th slice is an N-by-J matrix of response times at time t.
`deltas_propose`	Optional. A `vector` for the band widths of each lambda's proposal distribution in the MH sampling step.
`R`	Optional. A reachability `matrix` for the hierarchical relationship between attributes.

Value

A `list` of parameter samples and Metropolis-Hastings acceptance rates (if applicable).

Author(s)

Susu Zhang

Examples

```
output_FOHM = hmcdm(Y_real_array, Q_matrix, "DINA_FOHM", Design_array, 100, 30)
```

`inv_bijectionvector` *Convert integer to attribute pattern*

Description

Based on the bijective relationship between natural numbers and sum of powers of two, convert integer between 0 and 2^K-1 to K-dimensional attribute pattern.

Usage

```
inv_bijectionvector(K, CL)
```

Arguments

`K`	An `int` for the number of attributes
`CL`	An `int` between 0 and 2^K-1

Value

A vec of the K-dimensional attribute pattern corresponding to CL.

Examples

```
inv_bijectionvector(4,0)
```

`L_real_array` *Observed response times array*

Description

`L_real_array` contains the observed latencies of responses of all subjects to all questions in the Spatial Rotation Learning Program.

Usage

```
L_real_array
```

Format

An array of dimensions N-by-J-by-L. Each slice of the array is an N-by-J matrix, containing the subjects' response times in seconds to each item at time point l.

Author(s)

Shiyu Wang, Yan Yang, Jeff Douglas, and Steve Culpepper

Source

Spatial Rotation Learning Experiment at UIUC between Fall 2015 and Spring 2016.

`OddsRatio` *Compute item pairwise odds ratio*

Description

Based on a response matrix, calculate the item pairwise odds-ratio according do (n11*n00)/(n10*n01), where nij is the number of people answering both item i and item j correctly

Usage

```
OddsRatio(N, J, Yt)
```

Arguments

N	An int of the sample size
J	An int of the number of items
Yt	An N-by-J response matrix

Value

A J-by-J upper-triangular matrix of the item pairwise odds ratios

Examples

```
N = dim(Y_real_array)[1]
J = nrow(Q_matrix)
OddsRatio(N,J,Y_real_array[,,1])
```

---

pp_check.hmcdm *Graphical posterior predictive checks for hidden Markov cognitive diagnosis model*

Description

pp_check method for class hmcdm.

Usage

```
## S3 method for class 'hmcdm'
pp_check(object, plotfun = "dens_overlay", type = "total_score")
```

Arguments

object	a fitted model object of class "hmcdm".
plotfun	A character string naming the type of plot. The list of available plot functions include "dens_overlay", "hist", "stat_2d", "scatter_avg", "error_scatter_avg". The default function is "dens_overlay".
type	A character string naming the statistic to be used for obtaining posterior predictive distribution plot. The list of available types include "total_score", "item_mean", "item_OR", "latency_mean", and "latency_total". The default type is "total_score" which examines total scores of subjects. Type "item_mean" is related to the first order moment and examines mean scores of all the items included in the test. Type "item_OR" is related to the second order moment and examines odds ratios of all item pairs. Types "latency_mean" and "total_latency" are available only for hmcdm objects that include item response time information (i.e., hmcdm object fitted with "DINA_HO_RT" model).

Value

Plots for checking the posterior predictive distributions. The default `Plotfun` `"dens_overlay"` plots density of each dataset are overlaid with the distribution of the observed values.

References

Zhang, S., Douglas, J. A., Wang, S. & Culpepper, S. A. (2019) doi:10.1007/978-3-030-05584-4_24

See Also

`bayesplot::ppc_dens_overlay()` `bayesplot::ppc_stat()` `bayesplot::ppc_stat_2d()` `bayesplot::ppc_scatter_a`
`bayesplot::ppc_error_scatter_avg()`

Examples

```
output_FOHM = hmcdm(Y_real_array,Q_matrix,"DINA_FOHM",Design_array,1000,500)
library(bayesplot)
pp_check(output_FOHM)
pp_check(output_FOHM, plotfun="hist", type="item_mean")
```

`print.summary.hmcdm` *Summarizing Hidden Markov Cognitive Diagnosis Model Fits*

Description

`summary` method for class `"hmcdm"` or `"summary.hmcdm"`.

Usage

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

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

Arguments

`x`	an object of class `"hmcdm.summary"`.
`...`	further arguments passed to or from other methods.
`object`	a fitted model object of class `"hmcdm"`.

Value

The function `summary.hmcdm` computes and returns a `list` of point estimates of model parameters and model fit measures including DIC and PPP-values.

See Also

hmcdm()

Examples

```
output_FOHM = hmcdm(Y_real_array,Q_matrix,"DINA_FOHM",Design_array,1000,500)
summary(output_FOHM)
```

Q_list_g   *Generate a list of Q-matrices for each examinee.*

Description

Generate a list of length N. Each element of the list is a JxK Q_matrix of all items administered across all time points to the examinee, in the order of administration.

Usage

```
Q_list_g(Q_matrix, Design_array)
```

Arguments

Q_matrix	A J-by-K matrix, indicating the item-skill relationship.
Design_array	An N-by-J-by-L array indicating whether examinee n has taken item j at l time point.

Value

A list length of N. Each element of the list is a JxK Q_matrix for each examinee.

Examples

```
Q_examinee = Q_list_g(Q_matrix, Design_array)
```

`Q_matrix` *Q-matrix*

Description

`Q_matrix` contains the Q matrix of the items in the Spatial Rotation Learning Program.

Usage

```
Q_matrix
```

Format

A J-by-K matrix, indicating the item-skill relationship.

Author(s)

Shiyu Wang, Yan Yang, Jeff Douglas, and Steve Culpepper

Source

Spatial Rotation Learning Experiment at UIUC between Fall 2015 and Spring 2016.

`random_Q` *Generate random Q matrix*

Description

Creates a random Q matrix containing three identity matrices after row permutation

Usage

```
random_Q(J, K)
```

Arguments

`J`	An `int` that represents the number of items
`K`	An `int` that represents the number of attributes/skills

Value

A dichotomous `matrix` for Q.

Examples

```
random_Q(15,4)
```

`rOmega` *Generate a random transition matrix for the first order hidden Markov model*

Description

Generate a random transition matrix under nondecreasing learning trajectory assumption

Usage

```
rOmega(TP)
```

Arguments

`TP`	A 2^K-by-2^K dichotomous matrix of indicating possible transitions under the monotonicity assumption, created with the TPmat function

Value

A 2^K-by-2^K transition matrix, the (i,j)th element indicating the transition probability of transitioning from i-th class to j-th class.

Examples

```
K = ncol(Q_matrix)
TP = TPmat(K)
Omega_sim = rOmega(TP)
```

`sim_alphas` *Generate attribute trajectories under the specified hidden Markov models*

Description

Based on the learning model parameters, create cube of attribute patterns of all subjects across time. Currently available learning models are Higher-order hidden Markov DCM('HO_sep'), Higher-order hidden Markov DCM with learning ability as a random effect('HO_joint'), the simple independent-attribute learning model('indept'), and the first order hidden Markov model('FOHM').

Usage

```
sim_alphas(
  model,
  lambdas = NULL,
  thetas = NULL,
  Q_matrix = NULL,
  Design_array = NULL,
  taus = NULL,
  Omega = NULL,
  N = NA_integer_,
  L = NA_integer_,
  R = NULL,
  alpha0 = NULL
)
```

Arguments

`model`	The learning model under which the attribute trajectories are generated. Available options are: 'HO_joint', 'HO_sep', 'indept', 'FOHM'.
`lambdas`	A `vector` of transition model coefficients. With 'HO_sep' model specification, `lambdas` should be a length 4 `vector`. First entry is intercept of the logistic transition model, second entry is the slope of general learning ability, third entry is the slope for number of other mastered skills, fourth entry is the slope for amount of practice. With 'HO_joint' model specification, `lambdas` should be a length 3 `vector`. First entry is intercept of the logistic transition model, second entry is the slope for number of other mastered skills, third entry is the slope for amount of practice.
`thetas`	A length N `vector` of learning abilities of each subject.
`Q_matrix`	A J-by-K Q-matrix
`Design_array`	A N-by-J-by-L array indicating items administered to examinee n at time point l.
`taus`	A length K `vector` of transition probabilities from 0 to 1 on each skill
`Omega`	A 2^K-by-2^K `matrix` of transition probabilities from row pattern to column pattern
`N`	An `int` of number of examinees.
`L`	An `int` of number of time points.
`R`	A K-by-K dichotomous reachability `matrix` indicating the attribute hierarchies. The k,k'th entry of R is 1 if k' is prereq to k.
`alpha0`	Optional. An N-by-K `matrix` of subjects' initial attribute patterns.

Value

An N-by-K-by-L `array` of attribute patterns of subjects at each time point.

Examples

```
## HO_joint ##
N = nrow(Design_array)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = dim(Design_array)[3]
class_0 <- sample(1:2^K, N, replace = TRUE)
Alphas_0 <- matrix(0,N,K)
for(i in 1:N){
  Alphas_0[i,] <- inv_bijectionvector(K,(class_0[i]-1))
}
thetas_true = rnorm(N, 0, 1.8)
lambdas_true <- c(-2, .4, .055)
Alphas <- sim_alphas(model="HO_joint",
                    lambdas=lambdas_true,
                    thetas=thetas_true,
                    Q_matrix=Q_matrix,
                    Design_array=Design_array)

## HO_sep ##
N = dim(Design_array)[1]
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = dim(Design_array)[3]
class_0 <- sample(1:2^K, N, replace = L)
Alphas_0 <- matrix(0,N,K)
for(i in 1:N){
  Alphas_0[i,] <- inv_bijectionvector(K,(class_0[i]-1))
}
thetas_true = rnorm(N)
lambdas_true = c(-1, 1.8, .277, .055)
Alphas <- sim_alphas(model="HO_sep",
                     lambdas=lambdas_true,
                     thetas=thetas_true,
                     Q_matrix=Q_matrix,
                     Design_array=Design_array)

## indept ##
N = dim(Design_array)[1]
K = dim(Q_matrix)[2]
L = dim(Design_array)[3]
tau <- numeric(K)
for(k in 1:K){
  tau[k] <- runif(1,.2,.6)
}
R = matrix(0,K,K)
p_mastery <- c(.5,.5,.4,.4)
Alphas_0 <- matrix(0,N,K)
for(i in 1:N){
  for(k in 1:K){
    prereqs <- which(R[k,]==1)
```

```
      if(length(prereqs)==0){
        Alphas_0[i,k] <- rbinom(1,1,p_mastery[k])
      }
      if(length(prereqs)>0){
        Alphas_0[i,k] <- prod(Alphas_0[i,prereqs])*rbinom(1,1,p_mastery)
      }
    }
  }
  Alphas <- sim_alphas(model="indept", taus=tau, N=N, L=L, R=R)

  ## FOHM ##
  N = dim(Design_array)[1]
  K = ncol(Q_matrix)
  L = dim(Design_array)[3]
  TP <- TPmat(K)
  Omega_true <- rOmega(TP)
  class_0 <- sample(1:2^K, N, replace = L)
  Alphas_0 <- matrix(0,N,K)
  for(i in 1:N){
    Alphas_0[i,] <- inv_bijectionvector(K,(class_0[i]-1))
  }
  Alphas <- sim_alphas(model="FOHM", Omega = Omega_true, N=N, L=L)
```

`sim_hmcdm` *Simulate responses from the specified model (entire cube)*

Description

Simulate a cube of responses from the specified model for all persons on items across all time points. Currently available models are `DINA`, `rRUM`, and `NIDA`.

Usage

```
sim_hmcdm(
  model,
  alphas,
  Q_matrix,
  Design_array,
  itempars = NULL,
  r_stars = NULL,
  pi_stars = NULL,
  Svec = NULL,
  Gvec = NULL
)
```

Arguments

model	The cognitive diagnostic model under which the item responses are generated
alphas	An N-by-K-by-L array of attribute patterns of all persons across L time points
Q_matrix	A J-by-K of Q-matrix
Design_array	A N-by-J-by-L array indicating whether item j is administered to examinee i at l time point.
itempars	A J-by-2 mat of item parameters (slipping: 1st col, guessing: 2nd col).
r_stars	A J-by-K mat of item penalty parameters for missing skills.
pi_stars	A length J vector of item correct response probability with all requisite skills.
Svec	A length K vector of slipping probability in applying mastered skills
Gvec	A length K vector of guessing probability in applying mastered skills

Value

An array of item responses from the specified model of examinees across all time points.

Examples

```
## DINA ##
N = nrow(Design_array)
J = nrow(Q_matrix)
thetas_true = rnorm(N, 0, 1.8)
lambdas_true <- c(-2, .4, .055)
Alphas <- sim_alphas(model="HO_joint",
                    lambdas=lambdas_true,
                    thetas=thetas_true,
                    Q_matrix=Q_matrix,
                    Design_array=Design_array)
itempars_true <- matrix(runif(J*2,.1,.2), ncol=2)

Y_sim <- sim_hmcdm(model="DINA",Alphas,Q_matrix,Design_array,
                   itempars=itempars_true)

## rRUM ##
J = nrow(Q_matrix)
K = ncol(Q_matrix)
Smats <- matrix(runif(J*K,.1,.3),c(J,K))
Gmats <- matrix(runif(J*K,.1,.3),c(J,K))
r_stars <- Gmats / (1-Smats)
pi_stars <- apply((1-Smats)^Q_matrix, 1, prod)

Y_sim <- sim_hmcdm(model="rRUM",Alphas,Q_matrix,Design_array,
                   r_stars=r_stars,pi_stars=pi_stars)

## NIDA ##
K = ncol(Q_matrix)
Svec <- runif(K,.1,.3)
Gvec <- runif(K,.1,.3)
```

```
Y_sim <- sim_hmcdm(model="NIDA",Alphas,Q_matrix,Design_array,
                   Svec=Svec,Gvec=Gvec)
```

---

sim_RT *Simulate item response times based on Wang et al.'s (2018) joint model of response times and accuracy in learning*

Description

Simulate a cube of subjects' response times across time points according to a variant of the logNormal model

Usage

```
sim_RT(alphas, Q_matrix, Design_array, RT_itempars, taus, phi, G_version)
```

Arguments

alphas	An N-by-K-by-T `array` of attribute patterns of all persons across T time points
Q_matrix	A J-by-K Q-matrix for the test
Design_array	A N-by-J-by-L array indicating whether item j is administered to examinee i at l time point.
RT_itempars	A J-by-2 `matrix` of item time discrimination and time intensity parameters
taus	A length N `vector` of latent speed of each person
phi	A `scalar` of slope of increase in fluency over time due to covariates (G)
G_version	An `int` of the type of covariate for increased fluency (1: G is dichotomous depending on whether all skills required for current item are mastered; 2: G cumulates practice effect on previous items using mastered skills; 3: G is a time block effect invariant across subjects with different attribute trajectories)

Value

A cube of response times of subjects on each item across time

Examples

```
N = dim(Design_array)[1]
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = dim(Design_array)[3]
class_0 <- sample(1:2^K, N, replace = TRUE)
Alphas_0 <- matrix(0,N,K)
mu_thetatau = c(0,0)
Sig_thetatau = rbind(c(1.8^2,.4*.5*1.8),c(.4*.5*1.8,.25))
Z = matrix(rnorm(N*2),N,2)
```

```
thetatau_true = Z%*%chol(Sig_thetatau)
thetas_true = thetatau_true[,1]
taus_true = thetatau_true[,2]
G_version = 3
phi_true = 0.8
for(i in 1:N){
  Alphas_0[i,] <- inv_bijectionvector(K,(class_0[i]-1))
}
lambdas_true <- c(-2, .4, .055)
Alphas <- sim_alphas(model="HO_joint",
                     lambdas=lambdas_true,
                     thetas=thetas_true,
                     Q_matrix=Q_matrix,
                     Design_array=Design_array)
RT_itempars_true <- matrix(NA, nrow=J, ncol=2)
RT_itempars_true[,2] <- rnorm(J,3.45,.5)
RT_itempars_true[,1] <- runif(J,1.5,2)
ETAs <- ETAmat(K,J,Q_matrix)
L_sim <- sim_RT(Alphas,Q_matrix,Design_array,RT_itempars_true,taus_true,phi_true,G_version)
```

Test_order *Test block ordering of each test version*

Description

`Test_order` contains the item block ordering corresponding to each test module.

Usage

```
Test_order
```

Format

A L-by-L matrix, each row is the order of item blocks for that test version.

Details

Each row represents the test module number and shows the order of item blocks administered to a subject with the test module. For example, the first row is the order of item block administration (1-2-3-4-5) to subjects with test module 1.

Author(s)

Shiyu Wang, Yan Yang, Jeff Douglas, and Steve Culpepper

Source

Spatial Rotation Learning Experiment at UIUC between Fall 2015 and Spring 2016.

See Also

Test_versions

Test_versions *Subjects' test version*

Description

Test_versions contains each subject's test module in the Spatial Rotation Learning Program.

Usage

```
Test_versions
```

Format

A vector of length N, containing each subject's assigned test module.

Details

The data object "Test_versions" contains a vector of length N indicating the test module assigned to each subject. Each test module consists of multiple item blocks with different orders over L time points. The order of item blocks corresponding to each test module is presented in the data object "Test_order".

Author(s)

Shiyu Wang, Yan Yang, Jeff Douglas, and Steve Culpepper

Source

Spatial Rotation Learning Experiment at UIUC between Fall 2015 and Spring 2016.

See Also

Test_order

TPmat — *Generate monotonicity matrix*

Description

Based on the latent attribute space, generate a matrix indicating whether it is possible to transition from pattern cc to cc' under the monotonicity learning assumption.

Usage

```
TPmat(K)
```

Arguments

K	An `int` of the number of attributes.

Value

A 2^K-by-2^K dichotomous `matrix` of whether it is possible to transition between two patterns

Examples

```
TP = TPmat(4)
```

Y_real_array — *Observed response accuracy array*

Description

`Y_real_array` contains each subject's observed response accuracy (0/1) at all time points in the Spatial Rotation Learning Program.

Usage

```
Y_real_array
```

Format

An array of dimensions N-by-J-by-L. Each slice of the array is an N-by-J matrix, containing the subjects' response accuracy to each item at time point l.

Author(s)

Shiyu Wang, Yan Yang, Jeff Douglas, and Steve Culpepper

Source

Spatial Rotation Learning Experiment at UIUC between Fall 2015 and Spring 2016.

Index