Package ‘CDMLearning’

March 27, 2018

Type Package

Title Learning Models under Cognitive Diagnosis

Version 1.0

Date 2018-02-14

Maintainer Susu Zhang

<susu.zhang1992@gmail.com>

Description

Functions for fitting hidden Markov models of learning under the cognitive diagnosis framework. This package enables 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.

License GPL (>= 2)

Imports Rcpp (>= 0.12.14)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

Roxygen list(markdown = TRUE)

SystemRequirements C++11

Encoding UTF-8

R topics documented:

CDMLearning-package 2
ETAmat 3
Learning_fit 3
MCMC_learning 4
OddsRatio 6
point_estimates_learning 6
random_Q 7
rinvwish 8
rOmega 8
simDINA 9
simNIDA 9
simrRUM 10
simulate_alphas_FOHM 11
simulate_alphas_HO_joint 12

simulate_alphas_HO_sep 13
simulate_alphas_indept 14
sim_resp_DINA 15
sim_resp_NIDA 15
sim_resp_rRUM 16
sim_RT 17
TPmat 18

Index **19**

`CDMLearning-package` *Learning Models under Cognitive Diagnosis*

Description

Functions for fitting hidden Markov models of learning under the cognitive diagnosis framework. This package enables 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.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

Maintainer: Susu Zhang <susu.zhang1992@gmail.com>

Authors:

- Susu Zhang <`susu.zhang1992@gmail.com`>
- Shiyu Wang <`swang44@uga.edu`>
- Yinghan Chen <`yinghanc@unr.edu`>

References

Wang, S., Yang, Y., Culpepper, S. A., & Douglas, J. A. (2018). Tracking Skill Acquisition With Cognitive Diagnosis Models: A Higher-Order, Hidden Markov Model With Covariates. Journal of Educational and Behavioral Statistics, 1076998617719727.

Chen, Y., Culpepper, S. A., Wang, S., & Douglas, J. (2018). A hidden Markov model for learning trajectories in cognitive diagnosis with application to spatial rotation skills. Applied Psychological Measurement, 42(1), 5-23.

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)
```

Learning_fit *Model fit statistics of learning models*

Description

Obtain joint model's deviance information criteria (DIC) and posterior predictive item means, item response time means, item odds ratios, subject total scores at each time point, and subject total response times at each time point.

Usage

```
Learning_fit(output, model, Response_list, Q_list, test_order, Test_versions,
  Q_examinee = NULL, Latency_list = NULL, G_version = NA_integer_,
  R = NULL)
```

Arguments

output	A list of MCMC outputs, obtained from the MCMC_learning function
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
Response_list	A list of dichotomous item responses. t-th element is an N-by-Jt matrix of responses at time t.
Q_list	A list of Q-matrices. b-th element is a Jt-by-K Q-matrix for items in block b.
test_order	A matrix of the order of item blocks for each test version.
Test_versions	A vector of the test version of each learner.
Q_examinee	Optional. A list of the Q matrix for each learner. i-th element is a J-by-K Q-matrix for all items learner i was administered.
Latency_list	Optional. A list of the response times. t-th element is an N-by-Jt matrix of response times at time t.
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)
R	Optional. A reachability matrix for the hierarchical relationship between attributes.

Value

A list of DIC matrix, with deviance decomposed to that of the transition model, response model, response time model (if applicable), and joint model of random parameters, and posterior predictive item means, item odds ratios, item averaged response times, subjects' total scores at each time point, and subjects' total response times at each time point. Predicted values can be compared to the observed ones from empirical data.

Examples

```
FOHM_fit <- Learning_fit(output_FOHM,"DINA_FOHM",Y_sim_list,Q_list,test_order,Test_versions)
```

MCMC_learning — *Gibbs sampler for learning models*

Description

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

Usage

```
MCMC_learning(Response_list, Q_list, model, test_order, Test_versions,
  chain_length, burn_in, Q_examinee = NULL, Latency_list = NULL,
  G_version = NA_integer_, theta_propose = 0, deltas_propose = NULL,
  R = NULL)
```

Arguments

`Response_list`	A `list` of dichotomous item responses. t-th element is an N-by-Jt matrix of responses at time t.
`Q_list`	A `list` of Q-matrices. b-th element is a Jt-by-K Q-matrix for items in block b.
`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
`test_order`	A `matrix` of the order of item blocks for each test version.
`Test_versions`	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.
`Q_examinee`	Optional. A `list` of the Q matrix for each learner. i-th element is a J-by-K Q-matrix for all items learner i was administered.
`Latency_list`	Optional. A `list` of the response times. t-th element is an N-by-Jt matrix of response times at time t.
`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.
`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 = MCMC_learning(Y_sim_list,Q_list,"DINA_FOHM",test_order,Test_versions,10000,5000)
```

`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

```
OddsRatio(N,J,Y_sim)
```

`point_estimates_learning`
Obtain learning model point estimates

Description

Obtain EAPs of continuous parameters and EAP or MAP of the attribute trajectory estimates under the CDM learning models based on the MCMC output

Usage

```
point_estimates_learning(output, model, N, Jt, K, T, alpha_EAP = TRUE)
```

Arguments

output	A list of MCMC outputs, obtained from the MCMC_learning function
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
N	An int of number of subjects
Jt	An int of number of items in each block
K	An int of number of skills
T	An int of number of time points
alpha_EAP	A boolean operator (T/F) of whether to use EAP for alphas (if F: use most likely trajectory (MAP) for alphas)

Value

A list of point estimates of model parameters

Author(s)

Susu Zhang

Examples

```
point_estimates = point_estimates_learning(output_FOHM,"DINA_FOHM",N,Jt,K,T,alpha_EAP = T)
```

---

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)
```

`rinvwish` *Generate Random Inverse Wishart Distribution*

Description

Creates a random inverse wishart distribution when given degrees of freedom and a sigma matrix.

Usage

```
rinvwish(df, Sig)
```

Arguments

`df`	An `int` that represents the degrees of freedom. (> 0)
`Sig`	A `matrix` with dimensions m x m that provides Sigma, the covariance matrix.

Value

A `matrix` that is an inverse wishart distribution.

Author(s)

James J Balamuta

Examples

```
#Call with the following data:
riwishart(3, diag(2))
```

`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

Examples

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

simDINA — *Simulate DINA model responses (entire cube)*

Description

Simulate a cube of DINA responses for all persons on items across all time points

Usage

```
simDINA(alphas, itempars, ETA, test_order, Test_versions)
```

Arguments

`alphas`	An N-by-K-by-T `array` of attribute patterns of all persons across T time points
`itempars`	A J-by-2-by-T cube of item parameters (slipping: 1st col, guessin: 2nd col) across item blocks
`ETA`	A J-by-2^K-by-T `array` of ideal responses across all item blocks, with each slice generated with ETAmat function
`test_order`	A N_versions-by-T `matrix` indicating which block of items were administered to examinees with specific test version.
`Test_versions`	A length N `vector` of the test version of each examinee

Value

An `array` of DINA item responses of examinees across all time points

Examples

```
itempars_true <- array(runif(Jt*2*T,.1,.2), dim = c(Jt,2,T))

ETAs <- array(NA,dim = c(Jt,2^K,T))
for(t in 1:T){
  ETAs[,,t] <- ETAmat(K,Jt,Q_list[[t]])
}

Y_sim <- simDINA(Alphas,itempars_true,ETAs,test_order,Test_versions)
```

simNIDA — *Simulate NIDA model responses (entire cube)*

Description

Simulate a cube of NIDA responses for all persons on items across all time points

Usage

```
simNIDA(alphas, Svec, Gvec, Qs, test_order, Test_versions)
```

Arguments

alphas	An N-by-K-by-T `array` of attribute patterns of all persons across T time points
Svec	A length K `vector` of slipping probability in applying mastered skills
Gvec	A length K `vector` of guessing probability in applying mastered skills
Qs	A J-by-K-by-T cube of Q-matrices across all item blocks
test_order	A N_versions-by-T `matrix` indicating which block of items were administered to examinees with specific test version.
Test_versions	A length N `vector` of the test version of each examinee

Value

An `array` of NIDA item responses of examinees across all time points

Examples

```
data("Spatial_Rotation")
Svec <- runif(K,.1,.3)
Gvec <- runif(K,.1,.3)
Test_versions_sim <- sample(1:5,N,replace = T)
Y_sim = simNIDA(Alphas,Svec,Gvec,Qs,test_order,Test_versions_sim)
```

---

simrRUM — *Simulate rRUM model responses (entire cube)*

Description

Simulate a cube of rRUM responses for all persons on items across all time points

Usage

```
simrRUM(alphas, r_stars, pi_stars, Qs, test_order, Test_versions)
```

Arguments

alphas	An N-by-K-by-T `array` of attribute patterns of all persons across T time points
r_stars	A J-by-K-by-T cube of item penalty parameters for missing skills across all item blocks
pi_stars	A J-by-T `matrix` of item correct response probability with all requisite skills across blocks
Qs	A J-by-K-by-T cube of Q-matrices across all item blocks
test_order	A N_versions-by-T `matrix` indicating which block of items were administered to examinees with specific test version.
Test_versions	A length N `vector` of the test version of each examinee

Value

An `array` of rRUM item responses of examinees across all time points

Examples

```
data("Spatial_Rotation")
Smats <- array(runif(Jt*K*(T),.1,.3),c(Jt,K,(T)))
Gmats <- array(runif(Jt*K*(T),.1,.3),c(Jt,K,(T)))
r_stars <- array(NA,c(Jt,K,T))
pi_stars <- matrix(NA,Jt,(T))
for(t in 1:T){
  pi_stars[,,t] <- apply(((1-Smats[,,t])^Qs[,,t]),1,prod)
  r_stars[,,t] <- Gmats[,,t]/(1-Smats[,,t])
}
Test_versions_sim <- sample(1:5,N,replace = T)

Y_sim = simrRUM(Alphas,r_stars,pi_stars,Qs,test_order,Test_versions_sim)
```

`simulate_alphas_FOHM` *Generate attribute trajectories under the first order hidden Markov model*

Description

Based on the initial attribute patterns and probability of transitioning between different patterns, create cube of attribute patterns of all subjects across time.

Usage

```
simulate_alphas_FOHM(Omega, alpha0s, T)
```

Arguments

Omega	A 2^K-by-2^K `matrix` of transition probabilities from row pattern to column attern
alpha0s	An N-by-K `matrix` of subjects' initial attribute patterns.
T	An `int` of number of time points

Value

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

Examples

```
TP <- TPmat(K)
Omega_true <- rOmega(TP)
class_0 <- sample(1:2^K, N, replace = T)
Alphas_0 <- matrix(0,N,K)
for(i in 1:N){
  Alphas_0[i,] <- inv_bijectionvector(K,(class_0[i]-1))
}
Alphas <- simulate_alphas_FOHM(Omega_true, Alphas_0,T)
```

simulate_alphas_HO_joint *Generate attribute trajectories under the Higher-Order Hidden Markov DCM with latent learning ability as a random effect*

Description

Based on the initial attribute patterns and learning model parameters, create cube of attribute patterns of all subjects across time. General learning ability is regarded as a random intercept.

Usage

```
simulate_alphas_HO_joint(lambdas, thetas, alpha0s, Q_examinee, T, Jt)
```

Arguments

`lambdas`	A length 3 `vector` of transition model coefficients. 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.
`alpha0s`	An N-by-K `matrix` of subjects' initial attribute patterns.
`Q_examinee`	A length N `list` of Jt*K Q matrices across time for each examinee, items are in the order that they are administered to the examinee
`T`	An `int` of number of time points
`Jt`	An `int` of number of items in each block

Value

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

Examples

```
data("Spatial_Rotation")
class_0 <- sample(1:2^K, N, replace = T)
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]
for(i in 1:N){
  Alphas_0[i,] <- inv_bijectionvector(K,(class_0[i]-1))
}
lambdas_true <- c(-2, .4, .055)
Alphas <- simulate_alphas_HO_joint(lambdas_true,thetas_true,Alphas_0,Q_examinee,T,Jt)
```

simulate_alphas_HO_sep

Generate attribute trajectories under the Higher-Order Hidden Markov DCM

Description

Based on the initial attribute patterns and learning model parameters, create cube of attribute patterns of all subjects across time. General learning ability is regarded as a fixed effect and has a slope.

Usage

```
simulate_alphas_HO_sep(lambdas, thetas, alpha0s, Q_examinee, T, Jt)
```

Arguments

`lambdas`	A length 4 `vector` of transition model coefficients. 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.
`thetas`	A length N `vector` of learning abilities of each subject.
`alpha0s`	An N-by-K `matrix` of subjects' initial attribute patterns.
`Q_examinee`	A length N `list` of Jt*K Q matrices across time for each examinee, items are in the order that they are administered to the examinee
`T`	An `int` of number of time points
`Jt`	An `int` of number of items in each block

Value

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

Examples

```
data("Spatial_Rotation")
class_0 <- sample(1:2^K, N, replace = T)
Alphas_0 <- matrix(0,N,K)
thetas_true = rnorm(N)
for(i in 1:N){
  Alphas_0[i,] <- inv_bijectionvector(K,(class_0[i]-1))
}
lambdas_true = c(-1, 1.8, .277, .055)
Alphas <- simulate_alphas_HO_sep(lambdas_true,thetas_true,Alphas_0,Q_examinee,T,Jt)
```

simulate_alphas_indept *Generate attribute trajectories under the simple independent-attribute learning model*

Description

Based on the initial attribute patterns and probability of transitioning from 0 to 1 on each attribute, create cube of attribute patterns of all subjects across time. Transitions on different skills are regarded as independent.

Usage

```
simulate_alphas_indept(taus, alpha0s, T, R)
```

Arguments

taus	A length K `vector` of transition probabilities from 0 to 1 on each skill
alpha0s	An N-by-K `matrix` of subjects' initial attribute patterns.
T	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.

Value

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

Examples

```
data("Spatial_Rotation")
tau <- numeric(K)
for(k in 1:K){
  tau[k] <- runif(1,.2,.6)
}
R = matrix(0,K,K)
# Initial alphas
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 <- simulate_alphas_indept(tau,Alphas_0,T,R)
```

`sim_resp_DINA` *Simulate DINA model responses (single vector)*

Description

Simulate a single vector of DINA responses for a person on a set of items

Usage

```
sim_resp_DINA(J, K, ETA, Svec, Gvec, alpha)
```

Arguments

J	An `int` of number of items
K	An `int` of number of attributes
ETA	A `matrix` of ideal responses generated with ETAmat function
Svec	A length J `vector` of item slipping parameters
Gvec	A length J `vector` of item guessing parameters
alpha	A length K `vector` of attribute pattern of a person

Value

A length J `vector` of item responses

Examples

```
J = 10
K = 4
Q = random_Q(J,K)
ETA = ETAmat(K,J,Q)
s = runif(J,.1,.2)
g = runif(J,.1,.2)
alpha_i = c(1,0,0,1)
Y_i = sim_resp_DINA(J,K,ETA,s,g,alpha_i)
```

`sim_resp_NIDA` *Simulate NIDA model responses (single vector)*

Description

Simulate a single vector of NIDA responses for a person on a set of items

Usage

```
sim_resp_NIDA(J, K, Q, Svec, Gvec, alpha)
```

Arguments

J	An `int` of number of items
K	An `int` of number of attributes
Q	A J-by-K Q `matrix`
Svec	A length K `vector` of slipping probability in applying mastered skills
Gvec	A length K `vector` of guessing probability in applying mastered skills
alpha	A length K `vector` of attribute pattern of a person

Value

A length J `vector` of item responses

Examples

```
J = 10
K = 4
Q = random_Q(J,K)
Svec <- runif(K,.1,.3)
Gvec <- runif(K,.1,.3)
alpha_i = c(1,0,0,1)
Y_i = sim_resp_NIDA(J,K,Q,Svec,Gvec,alpha_i)
```

`sim_resp_rRUM` *Simulate rRUM model responses (single vector)*

Description

Simulate a single vector of rRUM responses for a person on a set of items

Usage

```
sim_resp_rRUM(J, K, Q, rstar, pistar, alpha)
```

Arguments

J	An `int` of number of items
K	An `int` of number of attributes
Q	A J-by-K Q `matrix`
rstar	A J-by-K `matrix` of item penalty parameters for missing requisite skills
pistar	length J `vector` of item correct response probability with all requisite skills
alpha	A length K `vector` of attribute pattern of a person

Value

A length J `vector` of item responses

Examples

```
J = 10
K = 4
Q = random_Q(J,K)
Smats <- matrix(runif(Jt*K,.1,.3),Jt,K)
Gmats <- matrix(runif(Jt*K,.1,.3),Jt,K)
r_stars <- matrix(NA,Jt,K)
pi_stars <- numeric(Jt)
for(t in 1:T){
  pi_stars <- apply(((1-Smats)^Q),1,prod)
  r_stars <- Gmats/(1-Smats)
}
alpha_i = c(1,0,0,1)
Y_i = sim_resp_rRUM(J,K,Q,r_stars,pi_stars,alpha_i)
```

---

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, RT_itempars, Qs, taus, phi, ETA, G_version, test_order,
  Test_versions)
```

Arguments

`alphas`	An N-by-K-by-T `array` of attribute patterns of all persons across T time points
`RT_itempars`	A J-by-2-by-T `array` of item time discrimination and time intensity parameters across item blocks
`Qs`	A J-by-K-by-T `cube` of Q-matrices across all item blocks
`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)
`ETA`	A J-by-2^K-by-T `array` of ideal responses across all item blocks, with each slice generated with ETAmat function
`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)
`test_order`	A N_versions-by-T `matrix` indicating which block of items were administered to examinees with specific test version.
`Test_versions`	A length N `vector` of the test version of each examinee

Value

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

Examples

```
class_0 <- sample(1:2^K, N, replace = T)
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 <- simulate_alphas_HO_joint(lambdas_true,thetas_true,Alphas_0,Q_examinee,T,Jt)
RT_itempars_true <- array(NA, dim = c(Jt,2,T))
RT_itempars_true[,2,] <- rnorm(Jt*T,3.45,.5)
RT_itempars_true[,1,] <- runif(Jt*T,1.5,2)
L_sim <- sim_RT(Alphas,RT_itempars_true,Qs,taus_true,phi_true,ETAs,G_version,test_order,Test_versions)
```

`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 attribtues.

Value

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

Examples

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

Index

∗Topic **package**
CDMLearning-package, 2

CDMLearning *(*CDMLearning-package*)*, 2
CDMLearning-package, 2

ETAmat, 3

Learning_fit, 3

MCMC_learning, 4

OddsRatio, 6

point_estimates_learning, 6

random_Q, 7
rinvwish, 8
rOmega, 8

sim_resp_DINA, 15
sim_resp_NIDA, 15
sim_resp_rRUM, 16
sim_RT, 17
simDINA, 9
simNIDA, 9
simrRUM, 10
simulate_alphas_FOHM, 11
simulate_alphas_HO_joint, 12
simulate_alphas_HO_sep, 13
simulate_alphas_indept, 14

TPmat, 18