Package ‘hmcdm’

October 13, 2022

Type Package

Title Hidden Markov Cognitive Diagnosis Models for Learning

Version 2.0.0

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 7.2.0

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 2022-08-29 09:20:08 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`

`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(
  Y_real_array,
  Q_matrix,
  model,
  Test_order,
  Test_versions,
  chain_length,
  burn_in,
  G_version = NA_integer_,
  theta_propose = 0,
  Latency_array = NULL,
  deltas_propose = NULL,
  R = NULL
)
```

Arguments

- `Y_real_array` 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
- `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.
- `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",Test_order,Test_versions,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 = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = nrow(Test_order)
Jt = J/L
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",Test_order,Test_versions,10000,5000)
library(bayesplot)
pp_check(output_FOHM)
pp_check(output_FOHM, plotfun="hist", type="item_mean")
```

Q_list   *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(Q_matrix, Test_order, Test_versions)
```

Arguments

Q_matrix	A J-by-K matrix, indicating the item-skill relationship.
Test_order	A TxT matrix, each row is the order of item blocks for that test version.
Test_versions	A vector of length N, containing each subject's test version.

Value

A list of length N. Each element of the list is a JxK matrix.

Examples

```
Q_examinee = Q_list(Q_matrix, Test_order, Test_versions)
```

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)
```

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-L `array` of attribute patterns of all persons across L time points
itempars	A J-by-2-by-L cube of item parameters (slipping: 1st col, guessing: 2nd col) across item blocks
ETA	A J-by-2^K-by-L `array` of ideal responses across all item blocks, with each slice generated with ETAmat function
Test_order	A N_versions-by-L `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

```
N = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = nrow(Test_order)
Jt = J/L
itempars_true <- array(runif(Jt*2*L,.1,.2), dim = c(Jt,2,L))

ETAs <- ETAmat(K,J,Q_matrix)
class_0 <- sample(1:2^K, N, replace = L)
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)
Q_examinee <- Q_list(Q_matrix, Test_order, Test_versions)
Alphas <- simulate_alphas_HO_joint(lambdas_true,thetas_true,Alphas_0,Q_examinee,L,Jt)
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, Q_matrix, Test_order, Test_versions)
```

Arguments

alphas	An N-by-K-by-L array of attribute patterns of all persons across L 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
Q_matrix	A J-by-K Q-matrix
Test_order	A N_versions-by-L 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

```
N = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = nrow(Test_order)
Jt = J/L
Svec <- runif(K,.1,.3)
Gvec <- runif(K,.1,.3)
Test_versions_sim <- sample(1:5,N,replace = L)
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,L,R)
Y_sim = simNIDA(Alphas,Svec,Gvec,Q_matrix,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_mat, pi_stars, Q_matrix, Test_order, Test_versions)
```

Arguments

alphas	An N-by-K-by-L `array` of attribute patterns of all persons across L time points
r_stars_mat	A J-by-K cube of item penalty parameters for missing skills across all item blocks
pi_stars	A Jt-by-L `matrix` of item correct response probability with all requisite skills across blocks
Q_matrix	A J-by-K of Q-matrix
Test_order	A N_versions-by-L `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

```
N = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = nrow(Test_order)
Jt = J/L
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 <- matrix(apply((1-Smats)^Q_matrix, 1, prod), nrow=Jt, ncol=L, byrow=L)
Test_versions_sim <- sample(1:5,N,replace = L)
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,L,R)
Y_sim = simrRUM(Alphas,r_stars,pi_stars,Q_matrix,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, L)
```

Arguments

`Omega`	A 2^K-by-2^K `matrix` of transition probabilities from row pattern to column pattern
`alpha0s`	An N-by-K `matrix` of subjects' initial attribute patterns.
`L`	An `int` of number of time points

Value

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

Examples

```
N = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = nrow(Test_order)
Jt = J/L
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 <- simulate_alphas_FOHM(Omega_true, Alphas_0,L)
```

---

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, L, 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
L	An `int` of number of time points
Jt	An `int` of number of items in each block

Value

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

Examples

```
N = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = nrow(Test_order)
Jt = J/L
class_0 <- sample(1:2^K, N, replace = L)
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)
Q_examinee <- Q_list(Q_matrix, Test_order, Test_versions)
Alphas <- simulate_alphas_HO_joint(lambdas_true,thetas_true,Alphas_0,Q_examinee,L,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, L, 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
L	An `int` of number of time points
Jt	An `int` of number of items in each block

Value

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

Examples

```
N = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = nrow(Test_order)
Jt = J/L
class_0 <- sample(1:2^K, N, replace = L)
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)
Q_examinee <- Q_list(Q_matrix, Test_order, Test_versions)
Alphas <- simulate_alphas_HO_sep(lambdas_true,thetas_true,Alphas_0,Q_examinee,L,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, L, 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.
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.

Value

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

Examples

```
N = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
L = nrow(Test_order)
Jt = J/L
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,L,R)
```

---

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,
  Q_matrix,
  taus,
  phi,
  ETAs,
  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
Q_matrix	A J-by-K Q-matrix for the test
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)
ETAs	A J-by-2^K matrix of ideal responses across all item blocks 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

```
N = length(Test_versions)
J = nrow(Q_matrix)
K = ncol(Q_matrix)
T = nrow(Test_order)
Jt = J/T
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)
Q_examinee <- Q_list(Q_matrix, Test_order, Test_versions)
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)
ETAs <- ETAmat(K,J,Q_matrix)
```

```
L_sim <- sim_RT(Alphas,RT_itempars_true,Q_matrix,taus_true,phi_true,ETAs,
G_version,Test_order,Test_versions)
```

---

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

Description

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

Usage

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

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

Arguments

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

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",Test_order,Test_versions,10000,5000)
summary(output_FOHM)
```

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