Package ‘regmhmm’

December 4, 2023

Type Package

Title 'regmhmm' Fits Hidden Markov Models with Regularization

Version 1.0.0

Date 2023-11-29

Maintainer Man Chong Leong <mc.leong26@gmail.com>

Description
Designed for longitudinal data analysis using Hidden Markov Models (HMMs). Tailored for applications in healthcare, social sciences, and economics, the main emphasis of this package is on regularization techniques for fitting HMMs. Additionally, it provides an implementation for fitting HMMs without regularization, referencing Zucchini et al. (2017, ISBN:9781315372488).

License GPL (>= 3)

Encoding UTF-8

Imports glmnet, glmnetUtils, MASS, Rcpp, stats

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.3

URL https://github.com/HenryLeongStat/regmhmm

BugReports https://github.com/HenryLeongStat/regmhmm/issues

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

NeedsCompilation yes

Author Man Chong Leong [cre, aut] (<https://orcid.org/0000-0003-3895-9527>)

Repository CRAN

Date/Publication 2023-12-04 16:40:05 UTC

R topics documented:

`backward` *Probability Calculation using the Backward Algorithm for Hidden Markov Models*

Description

Calculate the probability given parameters of a hidden Markov model utilizing the backward algorithm.

Usage

```
backward(delta, Y, A, B, X, family)
```

Arguments

`delta`	a vector of length S specifying the initial probabilities.
`Y`	a vector of observations of size T.
`A`	a matrix of size S x S specifying the transition probabilities.
`B`	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
`X`	a design matrix of size T x p.
`family`	the family of the response.

Value

A matrix of size S x T that is the backward probabilities in log scale.

Examples

```
# Example usage of the function
parameters_setting <- list()
parameters_setting$emis_mat <- matrix(NA, nrow = 2, ncol = 4)
parameters_setting$emis_mat[1, 1] <- 0.1
parameters_setting$emis_mat[1, 2] <- 0.5
parameters_setting$emis_mat[1, 3] <- -0.75
parameters_setting$emis_mat[1, 4] <- 0.75
parameters_setting$emis_mat[2, 1] <- -0.1
parameters_setting$emis_mat[2, 2] <- -0.5
parameters_setting$emis_mat[2, 3] <- 0.75
parameters_setting$emis_mat[2, 4] <- 1
parameters_setting$trans_mat <- matrix(NA, nrow = 2, ncol = 2)
parameters_setting$trans_mat[1, 1] <- 0.65
parameters_setting$trans_mat[1, 2] <- 0.35
parameters_setting$trans_mat[2, 1] <- 0.2
parameters_setting$trans_mat[2, 2] <- 0.8
parameters_setting$init_vec <- c(0.65, 0.35)
simulated_data <- simulate_HMM_data(
  seed_num = 1,
  p_noise = 7,
  N = 100,
  N_persub = 10,
  parameters_setting = parameters_setting
)
backward_C <- backward(
  delta = parameters_setting$init_vec,
  Y = simulated_data$y_mat[1, ],
  A = parameters_setting$trans_mat,
  B = parameters_setting$emis_mat,
  X = simulated_data$X_array[, 1:4, 1],
  family = "P"
)
```

---

compute_joint_state — *Posterior Joint Probability Calculation for Hidden States in a Hidden Markov Model*

Description

Calculate the posterior joint probability of hidden states given parameters of a hidden Markov model.

Usage

```
compute_joint_state(delta, Y, A, B, X, family)
```

Arguments

`delta`	a vector of length S specifying the initial probabilities.
`Y`	a vector of observations of size T.
`A`	a matrix of size S x S specifying the transition probabilities.
`B`	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
`X`	a design matrix of size T x p.
`family`	the family of the response.

Value

An array of size S x S x T that represents the posterior joint probability of hidden states.

Examples

```
# Example usage of the function
parameters_setting <- list()
parameters_setting$emis_mat <- matrix(NA, nrow = 2, ncol = 4)
parameters_setting$emis_mat[1, 1] <- 0.1
parameters_setting$emis_mat[1, 2] <- 0.5
parameters_setting$emis_mat[1, 3] <- -0.75
parameters_setting$emis_mat[1, 4] <- 0.75
parameters_setting$emis_mat[2, 1] <- -0.1
parameters_setting$emis_mat[2, 2] <- -0.5
parameters_setting$emis_mat[2, 3] <- 0.75
parameters_setting$emis_mat[2, 4] <- 1
parameters_setting$trans_mat <- matrix(NA, nrow = 2, ncol = 2)
parameters_setting$trans_mat[1, 1] <- 0.65
parameters_setting$trans_mat[1, 2] <- 0.35
parameters_setting$trans_mat[2, 1] <- 0.2
parameters_setting$trans_mat[2, 2] <- 0.8
parameters_setting$init_vec <- c(0.65, 0.35)
dat <- simulate_HMM_data(
  seed_num = 1,
  p_noise = 7,
  N = 100,
  N_persub = 10,
  parameters_setting = parameters_setting
)
compute_joint_state_get <- compute_joint_state(
    delta = parameters_setting$init_vec,
    Y = dat$y_mat[1, ],
    A = parameters_setting$trans_mat,
    B = parameters_setting$emis_mat,
    X = dat$X_array[, 1:4, 1],
    family = "P"
  )
```

compute_loglikelihood *Log-Likelihood Calculation for Hidden Markov Models (Forward Algorithm)*

Description

Calculate the log-likelihood given parameters of a hidden Markov model using the forward algorithm. This function aids in assessing the likelihood of the observed data under the specified model.

Usage

```
compute_loglikelihood(delta, Y, A, B, X, family)
```

Arguments

delta	a vector of length S specifying the initial probabilities.
Y	a vector of observations of size T.
A	a matrix of size S x S specifying the transition probabilities.
B	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
X	a design matrix of size T x p.
family	the family of the response.

Value

A value that is the likelihood in log scale.

Examples

```
# Example usage of the function
parameters_setting <- list()
parameters_setting$emis_mat <- matrix(NA, nrow = 2, ncol = 4)
parameters_setting$emis_mat[1, 1] <- 0.1
parameters_setting$emis_mat[1, 2] <- 0.5
parameters_setting$emis_mat[1, 3] <- -0.75
parameters_setting$emis_mat[1, 4] <- 0.75
parameters_setting$emis_mat[2, 1] <- -0.1
parameters_setting$emis_mat[2, 2] <- -0.5
parameters_setting$emis_mat[2, 3] <- 0.75
parameters_setting$emis_mat[2, 4] <- 1
parameters_setting$trans_mat <- matrix(NA, nrow = 2, ncol = 2)
parameters_setting$trans_mat[1, 1] <- 0.65
parameters_setting$trans_mat[1, 2] <- 0.35
parameters_setting$trans_mat[2, 1] <- 0.2
parameters_setting$trans_mat[2, 2] <- 0.8
parameters_setting$init_vec <- c(0.65, 0.35)
dat <- simulate_HMM_data(
```

```
  seed_num = 1,
  p_noise = 7,
  N = 100,
  N_persub = 10,
  parameters_setting = parameters_setting
)
llh_C <- compute_loglikelihood(
  delta = parameters_setting$init_vec,
  Y = dat$y_mat[1, ],
  A = parameters_setting$trans_mat,
  B = parameters_setting$emis_mat,
  X = dat$X_array[, 1:4, 1],
  family = "P"
)
```

`compute_state` *Posterior Probability Estimation for Hidden States in Hidden Markov Models*

Description

Calculate the posterior probability of hidden states given parameters of a hidden Markov model.

Usage

```
compute_state(delta, Y, A, B, X, family)
```

Arguments

`delta`	a vector of length S specifying the initial probabilities.
`Y`	a vector of observations of size T.
`A`	a matrix of size S x S specifying the transition probabilities.
`B`	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
`X`	a design matrix of size T x p.
`family`	the family of the response.

Value

A matrix of size S x T that represents the posterior probability of hidden states.

Examples

```
# Example usage of the function
parameters_setting <- list()
parameters_setting$emis_mat <- matrix(NA, nrow = 2, ncol = 4)
parameters_setting$emis_mat[1, 1] <- 0.1
parameters_setting$emis_mat[1, 2] <- 0.5
parameters_setting$emis_mat[1, 3] <- -0.75
parameters_setting$emis_mat[1, 4] <- 0.75
parameters_setting$emis_mat[2, 1] <- -0.1
parameters_setting$emis_mat[2, 2] <- -0.5
parameters_setting$emis_mat[2, 3] <- 0.75
parameters_setting$emis_mat[2, 4] <- 1
parameters_setting$trans_mat <- matrix(NA, nrow = 2, ncol = 2)
parameters_setting$trans_mat[1, 1] <- 0.65
parameters_setting$trans_mat[1, 2] <- 0.35
parameters_setting$trans_mat[2, 1] <- 0.2
parameters_setting$trans_mat[2, 2] <- 0.8
parameters_setting$init_vec <- c(0.65, 0.35)
simulated_data <- simulate_HMM_data(
  seed_num = 1,
  p_noise = 7,
  N = 100,
  N_persub = 10,
  parameters_setting = parameters_setting
)
compute_state_get <- compute_state(
  delta = parameters_setting$init_vec,
  Y = simulated_data$y_mat[1, ],
  A = parameters_setting$trans_mat,
  B = parameters_setting$emis_mat,
  X = simulated_data$X_array[, 1:4, 1],
 family = "P")
```

---

forward *Forward Algorithm for Probability Calculation in Hidden Markov Models*

Description

Calculate the probability given parameters of a hidden Markov model using the forward algorithm. This function is essential for estimating the likelihood of observing a particular sequence of observations in the context of a Hidden Markov Model (HMM).

Usage

```
forward(delta, Y, A, B, X, family)
```

Arguments

delta	a vector of length S specifying the initial probabilities.
Y	a vector of observations of size T.
A	a matrix of size S x S specifying the transition probabilities.
B	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
X	a design matrix of size T x p.
family	the family of the response.

Value

A matrix of size S x T that is the forward probabilities in log scale.

Examples

```
# Example usage of the function
parameters_setting <- list()
parameters_setting$emis_mat <- matrix(NA, nrow = 2, ncol = 4)
parameters_setting$emis_mat[1, 1] <- 0.1
parameters_setting$emis_mat[1, 2] <- 0.5
parameters_setting$emis_mat[1, 3] <- -0.75
parameters_setting$emis_mat[1, 4] <- 0.75
parameters_setting$emis_mat[2, 1] <- -0.1
parameters_setting$emis_mat[2, 2] <- -0.5
parameters_setting$emis_mat[2, 3] <- 0.75
parameters_setting$emis_mat[2, 4] <- 1
parameters_setting$trans_mat <- matrix(NA, nrow = 2, ncol = 2)
parameters_setting$trans_mat[1, 1] <- 0.65
parameters_setting$trans_mat[1, 2] <- 0.35
parameters_setting$trans_mat[2, 1] <- 0.2
parameters_setting$trans_mat[2, 2] <- 0.8
parameters_setting$init_vec <- c(0.65, 0.35)
dat <- simulate_HMM_data(
  seed_num = 1,
  p_noise = 7,
  N = 100,
  N_persub = 10,
  parameters_setting = parameters_setting
)
forward_C <- forward(
  delta = parameters_setting$init_vec,
  Y = dat$y_mat[1, ],
  A = parameters_setting$trans_mat,
  B = parameters_setting$emis_mat,
  X = dat$X_array[, 1:4, 1],
  family = "P"
)
```

forward_backward	*Probability Calculation in Hidden Markov Models using Forward-Backward Algorithm*

Description

Calculate the probability given parameters of a hidden Markov model using a combination of the forward and backward algorithms.

Usage

```
forward_backward(delta, Y, A, B, X, family)
```

Arguments

`delta`	a vector of length S specifying the initial probabilities.
`Y`	a vector of observations of size T.
`A`	a matrix of size S x S specifying the transition probabilities.
`B`	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
`X`	a design matrix of size T x p.
`family`	the family of the response.

Value

A list object with the following slots:

`log_alpha`	a matrix of size S x T that is the forward probabilities in log scale.
`log_beta`	a matrix of size S x T that is the backward probabilities in log scale.

Examples

```
# Example usage of the function
parameters_setting <- list()
parameters_setting$emis_mat <- matrix(NA, nrow = 2, ncol = 4)
parameters_setting$emis_mat[1, 1] <- 0.1
parameters_setting$emis_mat[1, 2] <- 0.5
parameters_setting$emis_mat[1, 3] <- -0.75
parameters_setting$emis_mat[1, 4] <- 0.75
parameters_setting$emis_mat[2, 1] <- -0.1
parameters_setting$emis_mat[2, 2] <- -0.5
parameters_setting$emis_mat[2, 3] <- 0.75
parameters_setting$emis_mat[2, 4] <- 1
parameters_setting$trans_mat <- matrix(NA, nrow = 2, ncol = 2)
parameters_setting$trans_mat[1, 1] <- 0.65
parameters_setting$trans_mat[1, 2] <- 0.35
parameters_setting$trans_mat[2, 1] <- 0.2
```

```
parameters_setting$trans_mat[2, 2] <- 0.8
parameters_setting$init_vec <- c(0.65, 0.35)
simulated_data <- simulate_HMM_data(
  seed_num = 1,
  p_noise = 7,
  N = 100,
  N_persub = 10,
  parameters_setting = parameters_setting
)
forward_backward_C <- forward_backward(
  delta = parameters_setting$init_vec,
  Y = simulated_data$y_mat[1, ],
  A = parameters_setting$trans_mat,
  B = parameters_setting$emis_mat,
  X = simulated_data$X_array[, 1:4, 1],
  family = "P"
)
```

`HMM` *Fitting Hidden Markov Models using Expectation-Maximization (EM) Algorithm*

Description

Fit Hidden Markov Models (HMMs) to the provided data using an iterative Expectation-Maximization (EM) algorithm. This method alternates between the E-step (Expectation) and M-step (Maximization) to iteratively optimize model parameters. The algorithm ensures convergence and provides robust estimates for latent state probabilities and transition probabilities, contributing to the accurate characterization of underlying patterns in the data.

Usage

```
HMM(delta, Y_mat, A, B, X_cube, family, trace, ...)
```

Arguments

`delta`	a vector of length S specifying the initial probabilities.
`Y_mat`	a matrix of observations of size N x T.
`A`	a matrix of size S x S specifying the transition probabilities.
`B`	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
`X_cube`	a design array of size T x p x N.
`family`	the family of the response.
`trace`	logical indicating if detailed output should be produced during the fitting process.
`...`	other arguments to be passed to the next method.

Value

A list object with the following slots:

`delta_hat`	the estimate of delta.
`A_hat`	the estimate of A.
`B_hat`	the estimate of B.
`log_likelihood`	the log-likelihood of the model.

Examples

```
# Example usage of the function
seed_num <- 1
p_noise <- 2
N <- 100
N_persub <- 10
parameters_setting <- list(
  init_vec = c(0.5, 0.5),
  trans_mat = matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE),
  emis_mat = matrix(c(1, 0.5, 0.5, 2), nrow = 2, byrow = TRUE)
)
simulated_data <- simulate_HMM_data(seed_num, p_noise, N, N_persub, parameters_setting)

init_start = c(0.5, 0.5)
trans_start = matrix(c(0.5, 0.5, 0.5, 0.5), nrow = 2)
emis_start = matrix(rep(1, 8), nrow = 2)

HMM_fit <- HMM(delta=as.matrix(init_start),
               Y_mat=simulated_data$y_mat,
               A=trans_start,
               B=emis_start,
               X_cube=simulated_data$X_array,
               family="P",
               eps=1e-4,
               trace = 0
)
```

`HMM_C_raw` *Fit Hidden Markov Model (HMM)*

Description

Employ this function to fit a Hidden Markov Model (HMM) to the provided data. It iteratively estimates model parameters using the EM algorithm.

Usage

```
HMM_C_raw(
  delta,
  Y_mat,
  A,
  B,
  X_cube,
  family,
  eps = 1e-05,
  eps_IRLS = 1e-04,
  N_iter = 1000L,
  max_N_IRLS = 300L,
  trace = 0L
)
```

Arguments

`delta`	a vector of length S specifying the initial probabilities.
`Y_mat`	a matrix of observations of size N x T.
`A`	a matrix of size S x S specifying the transition probabilities.
`B`	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
`X_cube`	a design array of size T x p x N.
`family`	the family of the response.
`eps`	convergence tolerance in the EM algorithm for fitting HMM.
`eps_IRLS`	convergence tolerance in the iteratively reweighted least squares step.
`N_iter`	the maximal number of the EM algorithm for fitting HMM.
`max_N_IRLS`	the maximal number of IRLS iterations.
`trace`	logical indicating if detailed output should be produced during the fitting process.

Value

A list object with the following slots:

`delta_hat`	the estimate of delta.
`A_hat`	the estimate of A.
`B_hat`	the estimate of B.
`log_likelihood`	the log-likelihood of the model.

Examples

```
# Example usage of the function
seed_num <- 1
p_noise <- 2
N <- 100
N_persub <- 10
parameters_setting <- list(
  init_vec = c(0.5, 0.5),
  trans_mat = matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE),
  emis_mat = matrix(c(1, 0.5, 0.5, 2), nrow = 2, byrow = TRUE)
)
simulated_data <- simulate_HMM_data(seed_num, p_noise, N, N_persub, parameters_setting)
init_start = c(0.5, 0.5)
trans_start = matrix(c(0.5, 0.5, 0.5, 0.5), nrow = 2)
emis_start = matrix(rep(1, 8), nrow = 2)
HMM_fit_raw <- HMM_C_raw(delta=as.matrix(init_start),
               Y_mat=simulated_data$y_mat,
               A=trans_start,
               B=emis_start,
               X_cube=simulated_data$X_array,
               family="P",
               eps=1e-4,
               trace = 0
)
```

HMM_one_step *Single EM Iteration for Fitting Hidden Markov Models (HMM)*

Description

Execute a single iteration of the Expectation-Maximization (EM) algorithm tailored for fitting Hidden Markov Models (HMMs).

Usage

```
HMM_one_step(
  delta,
  Y_mat,
  A,
  B,
  X_cube,
  family,
  eps_IRLS = 1e-04,
  max_N_IRLS = 300L
)
```

Arguments

`delta`	a vector of length S specifying the initial probabilities.
`Y_mat`	a matrix of observations of size N x T.
`A`	a matrix of size S x S specifying the transition probabilities.
`B`	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
`X_cube`	a design array of size T x p x N.
`family`	the family of the response.
`eps_IRLS`	convergence tolerance in the iteratively reweighted least squares step.
`max_N_IRLS`	the maximal number of IRLS iterations.

Value

A list object with the following slots:

`delta_hat`	the estimate of delta.
`A_hat`	the estimate of A.
`B_hat`	the estimate of B.
`log_likelihood`	the log-likelihood of the model.

Examples

```
# Example usage of the function
seed_num <- 1
p_noise <- 2
N <- 100
N_persub <- 10
parameters_setting <- list(
  init_vec = c(0.5, 0.5),
  trans_mat = matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE),
  emis_mat = matrix(c(1, 0.5, 0.5, 2), nrow = 2, byrow = TRUE)
)
simulated_data <- simulate_HMM_data(seed_num, p_noise, N, N_persub, parameters_setting)
init_start = c(0.5, 0.5)
trans_start = matrix(c(0.5, 0.5, 0.5, 0.5), nrow = 2)
emis_start = matrix(rep(1, 8), nrow = 2)
HMM_fit_raw_one_step <- HMM_one_step(delta=as.matrix(init_start),
               Y_mat=simulated_data$y_mat,
               A=trans_start,
               B=emis_start,
               X_cube=simulated_data$X_array,
               family="P")
```

`IRLS_EM` *Iterative Reweighted Least Squares for the EM algorithm*

Description

Iterative Reweighted Least Squares algorithm for optimizing the parameters in the M-step of the EM algorithm.

Usage

```
IRLS_EM(X, gamma, Y, beta, family, eps_IRLS, max_N)
```

Arguments

`X`	A design matrix of size n x p.
`gamma`	A vector of size n specifying the posterior probability of the hidden states.
`Y`	A vector of observations of size n.
`beta`	A vector of size p + 1 specifying the GLM parameters.
`family`	The family of the response.
`eps_IRLS`	convergence tolerance in the iteratively reweighted least squares step.
`max_N`	the maximal number of IRLS iterations.

Value

A vector representing the estimates of beta.

Examples

```
## Not run:
# Example usage of the function
IRLS_EM_one_step <- IRLS_EM_one(X,
                                gamma,
                                Y,
                                beta,
                                family)

## End(Not run)
```

print.HMM *Print Outputs from a Hidden Markov Model (HMM)*

Description

Display detailed summary outputs and relevant information derived from a Hidden Markov Model (HMM) object. This includes state-specific parameters, transition probabilities, log-likelihood, and other essential metrics, providing an overview of the fitted model.

Usage

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

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

Value

Return a invisible copy of "HMM" object

Examples

```
# Example usage of the function
seed_num <- 1
p_noise <- 2
N <- 100
N_persub <- 10
parameters_setting <- list(
  init_vec = c(0.5, 0.5),
  trans_mat = matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE),
  emis_mat = matrix(c(1, 0.5, 0.5, 2), nrow = 2, byrow = TRUE)
)
simulated_data <- simulate_HMM_data(seed_num, p_noise, N, N_persub, parameters_setting)

init_start = c(0.5, 0.5)
trans_start = matrix(c(0.5, 0.5, 0.5, 0.5), nrow = 2)
emis_start = matrix(rep(1, 8), nrow = 2)

HMM_fit <- HMM(delta=as.matrix(init_start),
               Y_mat=simulated_data$y_mat,
               A=trans_start,
               B=emis_start,
               X_cube=simulated_data$X_array,
               family="P",
               eps=1e-4,
```

```
                  trace = 0
)
print(HMM_fit)
```

rHMM *Fit Regularized Hidden Markov Models (rHMM) with Modified CCD*

Description

Utilize the modified Cyclic Coordinate Descent (CCD) algorithm to effectively fit a regularized Hidden Markov Model (rHMM).

Usage

```
rHMM(
  delta,
  Y_mat,
  A,
  B,
  X_cube,
  family,
  omega_cva = sqrt(sqrt(seq(0, 1, len = 5))),
  N_iter = 1000,
  eps = 1e-07,
  trace = 0
)
```

Arguments

delta	a vector of length S specifying the initial probabilities.
Y_mat	a matrix of observations of size N x T.
A	a matrix of size S x S specifying the transition probabilities.
B	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
X_cube	a design array of size T x p x N.
family	the family of the response.
omega_cva	a vector of omega values for the modified cyclical coordinate descent algorithm used for cross-validation.
N_iter	the maximal number of the EM algorithm for fitting HMM.
eps	convergence tolerance.
trace	logical indicating if detailed output should be produced during the fitting process.

Value

A list object with the following slots:

delta_hat	the estimate of delta.
A_hat	the estimate of A.
B_hat	the estimate of B.
log_likelihood	the log-likelihood of the model.
lambda	lambda from CV.
omega	omega from CV.

Examples

```
# Example usage of the function
seed_num <- 1
p_noise <- 2
N <- 100
N_persub <- 50
parameters_setting <- list(
  init_vec = c(0.5, 0.5),
  trans_mat = matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE),
  emis_mat = matrix(c(1, 0.5, 0.5, 2), nrow = 2, byrow = TRUE)
)
simulated_data <- simulate_HMM_data(seed_num, p_noise, N, N_persub, parameters_setting)

init_start = c(0.5, 0.5)
trans_start = matrix(c(0.5, 0.5, 0.5, 0.5), nrow = 2)
emis_start = matrix(rep(1, 8), nrow = 2)

rHMM_one_step <- rHMM(delta=as.matrix(init_start),
                               Y_mat=simulated_data$y_mat,
                               A=trans_start,
                               B=emis_start,
                               X_cube=simulated_data$X_array,
                               family="P",
                               omega_cva=sqrt(sqrt(seq(0, 1, len = 5))),
                               N_iter=10,
                               trace = 0)
```

---

rHMM_one_step *Single Iteration of EM Algorithm for Fitting Regularized Hidden Markov Model (rHMM)*

Description

Execute a single iteration of the Expectation-Maximization (EM) algorithm designed for fitting a regularized Hidden Markov Model (rHMM).

Usage

```
rHMM_one_step(
  delta,
  Y_mat,
  A,
  B,
  X_cube,
  family,
  omega_cva = sqrt(sqrt(seq(0, 1, len = 5))),
  trace = 0
)
```

Arguments

delta	a vector of length S specifying the initial probabilities.
Y_mat	a matrix of observations of size N x T.
A	a matrix of size S x S specifying the transition probabilities.
B	a matrix of size S x (p + 1) specifying the GLM parameters of the emission probabilities.
X_cube	a design array of size T x p x N.
family	the family of the response.
omega_cva	a vector of omega values for the modified cyclical coordinate descent algorithm used for cross-validation.
trace	logical indicating if detailed output should be produced during the fitting process.

Value

A list object with the following slots:

delta_hat	the estimate of delta.
A_hat	the estimate of A.
B_hat	the estimate of B.
log_likelihood	the log-likelihood of the model.
lambda	lambda from CV.
omega	omega from CV.

Examples

```
# Example usage of the function
seed_num <- 1
p_noise <- 2
N <- 100
N_persub <- 50
parameters_setting <- list(
```

```
  init_vec = c(0.5, 0.5),
  trans_mat = matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE),
  emis_mat = matrix(c(1, 0.5, 0.5, 2), nrow = 2, byrow = TRUE)
)
simulated_data <- simulate_HMM_data(seed_num, p_noise, N, N_persub, parameters_setting)

init_start = c(0.5, 0.5)
trans_start = matrix(c(0.5, 0.5, 0.5, 0.5), nrow = 2)
emis_start = matrix(rep(1, 8), nrow = 2)

rHMM_one_step <- rHMM_one_step(delta=as.matrix(init_start),
                               Y_mat=simulated_data$y_mat,
                               A=trans_start,
                               B=emis_start,
                               X_cube=simulated_data$X_array,
                               family="P",
                               omega_cva=sqrt(sqrt(seq(0, 1, len = 5))),
                               trace = 0)
```

simulate_HMM_data *Simulate Hidden Markov Model (HMM) Data*

Description

Generate synthetic HMM data for testing and validation purposes. This function creates a simulated dataset with specified parameters, including initial probabilities, transition probabilities, emission matrix, and noise covariates.

Usage

```
simulate_HMM_data(seed_num, p_noise, N, N_persub, parameters_setting)
```

Arguments

seed_num	Seed for reproducibility.
p_noise	Number of noise covariates.
N	Number of subjects.
N_persub	Number of time points per subject.
parameters_setting	A list containing the parameters for the HMM.

Value

A list containing the design matrix (X_array) and response variable matrix (y_mat).

Examples

```
seed_num <- 1
p_noise <- 2
N <- 100
N_persub <- 50
parameters_setting <- list(
  init_vec = c(0.5, 0.5),
  trans_mat = matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE),
  emis_mat = matrix(c(1, 0.5, 0.5, 2), nrow = 2, byrow = TRUE)
)
simulated_data <- simulate_HMM_data(seed_num, p_noise, N, N_persub, parameters_setting)
```

Index