Package ‘oHMMed’

April 19, 2024

Type Package

Title HMMs with Ordered Hidden States and Emission Densities

Version 1.0.2

Description Inference using a class of Hidden Markov models (HMMs) called 'oHMMed'(ordered HMM with emission densities <doi:10.1186/s12859-024-05751-4>): The 'oHMMed' algorithms identify the number of comparably homogeneous regions within observed sequences with autocorrelation patterns. These are modelled as discrete hidden states; the observed data points are then realisations of continuous probability distributions with state-specific means that enable ordering of these distributions. The observed sequence is labelled according to the hidden states, permitting only neighbouring states that are also neighbours within the ordering of their associated distributions. The parameters that characterise these state-specific distributions are then inferred. Relevant for application to genomic sequences, time series, or any other sequence data with serial autocorrelation.

License GPL-3

URL `https://github.com/LynetteCaitlin/oHMMed`, `https://lynettecaitlin.github.io/oHMMed/`

BugReports `https://github.com/LynetteCaitlin/oHMMed/issues`

Depends R (>= 3.5.0)

Imports cvms, ggmcmc, ggplot2, gridExtra, mistr, scales, stats, vcd

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Michal Majka [aut, cre] (<`https://orcid.org/0000-0002-7524-8014`>), Lynette Caitlin Mikula [aut] (<`https://orcid.org/0000-0002-0252-4014`>), Claus Vogl [aut] (<`https://orcid.org/0000-0002-3996-7863`>)

Maintainer Michal Majka <`michalmajka@hotmail.com`>

Repository CRAN

Date/Publication 2024-04-19 20:22:39 UTC

R topics documented:

`oHMMed-package` *oHMMed: HMMs with Ordered Hidden States and Emission Densities*

Description

Inference using a class of Hidden Markov models (HMMs) called 'oHMMed'(ordered HMM with emission densities doi:10.1186/s12859024057514): The 'oHMMed' algorithms identify the number of comparably homogeneous regions within observed sequences with autocorrelation patterns. These are modelled as discrete hidden states; the observed data points are then realisations of continuous probability distributions with state-specific means that enable ordering of these distributions. The observed sequence is labelled according to the hidden states, permitting only neighbouring states that are also neighbours within the ordering of their associated distributions. The parameters that characterise these state-specific distributions are then inferred. Relevant for application to genomic sequences, time series, or any other sequence data with serial autocorrelation.

Author(s)

Maintainer: Michal Majka <michalmajka@hotmail.com> (ORCID)

Authors:

- Lynette Caitlin Mikula <lynettecaitlin@gmail.com> (ORCID)
- Claus Vogl <claus.vogl@vetmeduni.ac.at> (ORCID)

References

Claus Vogl, Mariia Karapetiants, Burçin Yıldırım, Hrönn Kjartansdóttir, Carolin Kosiol, Juraj Bergman, Michal Majka, Lynette Caitlin Mikula. Inference of genomic landscapes using ordered Hidden Markov Models with emission densities (oHMMed). BMC Bioinformatics 25, 151 (2024). doi:10.1186/s12859024057514

See Also

Useful links:

- https://github.com/LynetteCaitlin/oHMMed
- https://lynettecaitlin.github.io/oHMMed/
- Report bugs at https://github.com/LynetteCaitlin/oHMMed/issues

---

`coef.hmm_mcmc_normal` *Extract Model Estimates*

Description

`coef` is a generic function which extracts model estimates from `mcmc_hmm_*` objects

Usage

```
## S3 method for class 'hmm_mcmc_normal'
coef(object, ...)

## S3 method for class 'hmm_mcmc_gamma_poisson'
coef(object, ...)
```

Arguments

`object`	an object of class inheriting from "`mcmc_hmm_*`"
`...`	not used

Value

Estimates extracted from MCMC HMM objects

Examples

```
coef(example_hmm_mcmc_normal)
coef(example_hmm_mcmc_gamma_poisson)
```

---

conf_mat *Calculate and Visualise a Confusion Matrix*

Description

A diagnostic function that tests the reliability of estimation procedures given the inferred transition rates

Usage

```
conf_mat(N, res, plot = TRUE)
```

Arguments

N	(numeric) number of simulations
res	(mcmc_hmm_*) simulated MCMC HMM model
plot	(logical) plot confusion matrix. By default TRUE

Details

First the data is simulated given the inferred model parameters and transition rates. Then posterior probabilities are calculated and states are inferred. Finally, the inferred states and simulated states are compared via confusion_matrix function.

Value

confusion_matrix

Examples

```
if (interactive()) {
  res <- conf_mat(100, example_hmm_mcmc_normal, plot = TRUE)
}
```

`convert_to_ggmcmc` *Converts MCMC Samples into* `ggmcmc` *Format*

Description

This helper function converts MCMC samples into `ggmcmc` format

Usage

```
convert_to_ggmcmc(
  x,
  pattern = c("mean", "sigma", "beta", "alpha", "pois_means", "T"),
  include_warmup = FALSE
)
```

Arguments

`x`	(mcmc_hmm_*) MCMC HMM object
`pattern`	(character) pattern(s) with model parameters to be included in the output
`include_warmup`	(logical) include warmup samples. By default `FALSE`

Details

By default, for a given model, all parameters are converted into ggmcmc format.

The parameter `pattern` can be used to extract specific parameters. For instance `pattern="mean"` extracts all mean parameters from a hmm_mcmc_normal model.

If a specific parameter is of interest it can be matched by an exact name: `pattern=c("mean[1]", "T[1,1]")`.

Value

data.frame compatible with functions from the ggmcmc package

Examples

```
# Convert all parameters (Normal model)
convert_normal_all <- convert_to_ggmcmc(example_hmm_mcmc_normal)
unique(convert_normal_all$Parameter)
head(convert_normal_all)
tail(convert_normal_all)

# Convert only means (Normal model)
convert_normal_means <- convert_to_ggmcmc(example_hmm_mcmc_normal,
                                           pattern = "mean")
unique(convert_normal_means$Parameter)

# Convert selected parameter (Normal model)
```

```
pattern_normal <- c("mean[1]", "sigma[1]", "T[1,1]")
convert_normal_param <- convert_to_ggmcmc(example_hmm_mcmc_normal,
                                          pattern = pattern_normal)
unique(convert_normal_param$Parameter)

# Convert all parameters (Poisson-Gamma model)
convert_pois_gamma_all <- convert_to_ggmcmc(example_hmm_mcmc_gamma_poisson)
unique(convert_pois_gamma_all$Parameter)
```

`eigen_system` *Calculate Eigenvalues and Eigenvectors*

Description

This helper function returns the eigenvalues in lambda and the left and right eigenvectors in forwards and backwards

Usage

```
eigen_system(mat)
```

Arguments

`mat`	(matrix) a square matrix

Value

a list with three elements:

- `lambda`: eigenvalues
- `forwards`: left eigenvector
- `backwards`: right eigenvector

Examples

```
mat_T0 <- rbind(c(1-0.01,0.01,0),
                c(0.01,1-0.02,0.01),
                c(0,0.01,1-0.01))
eigen_system(mat_T0)
```

`example_hmm_mcmc_gamma_poisson`
Example of a Simulated Gamma-Poisson Model

Description

Example of a Simulated Gamma-Poisson Model

Usage

```
example_hmm_mcmc_gamma_poisson
```

Format

hmm_mcmc_gamma_poisson object

Examples

```
# Data stored in the object
hist(example_hmm_mcmc_gamma_poisson$data,
     breaks = 50, xlab = "", main = "")

# Priors used in simulation
example_hmm_mcmc_gamma_poisson$priors

# Model
example_hmm_mcmc_gamma_poisson

summary(example_hmm_mcmc_gamma_poisson)
```

`example_hmm_mcmc_normal`
Example of a Simulated Normal Model

Description

Example of a Simulated Normal Model

Usage

```
example_hmm_mcmc_normal
```

Format

hmm_mcmc_normal object

Examples

```
# Data stored in the object
plot(density(example_hmm_mcmc_normal$data), main = "")

# Priors used in simulation
example_hmm_mcmc_normal$priors

# Model
example_hmm_mcmc_normal

summary(example_hmm_mcmc_normal)
```

`generate_random_T` *Generate a Random Transition Matrix*

Description

This helper function generates a transition matrix at random for testing purposes

Usage

```
generate_random_T(n = 3)
```

Arguments

`n`	(integer) dimension of a transition matrix

Details

Uniform random numbers $[0, 1]$ are used to fill the matrix. Rows are then normalized.

Value

random `n x n` transition matrix

Examples

```
mat_T <- generate_random_T(3)
mat_T

rowSums(mat_T)
```

`get_pi` *Get the Prior Probability of States*

Description

Calculate the prior probability of states that correspond to the stationary distribution of the transition matrix T

Usage

```
get_pi(mat_T = NULL)
```

Arguments

`mat_T`	(matrix) transition matrix

Details

It is assumed that the prior probability of states corresponds to the stationary distribution of the transition matrix T, denoted with π and its entries with $\pi_i = Pr(\theta_{l-1} = i)$.

Value

A numeric vector

Examples

```
T_mat <- rbind(c(1-0.01,0.01,0),
               c(0.01,1-0.02,0.01),
               c(0,0.01,1-0.01))
T_mat
get_pi(T_mat)
```

`hmm_mcmc_gamma_poisson` *MCMC Sampler sampler for the Hidden Markov with Gamma-Poisson emission densities*

Description

MCMC Sampler sampler for the Hidden Markov with Gamma-Poisson emission densities

Usage

```
hmm_mcmc_gamma_poisson(
  data,
  prior_T,
  prior_betas,
  prior_alpha = 1,
  iter = 5000,
  warmup = floor(iter/1.5),
  thin = 1,
  seed = sample.int(.Machine$integer.max, 1),
  init_T = NULL,
  init_betas = NULL,
  init_alpha = NULL,
  print_params = TRUE,
  verbose = TRUE
)
```

Arguments

`data`	(numeric) data
`prior_T`	(matrix) prior transition matrix
`prior_betas`	(numeric) prior beta parameters
`prior_alpha`	(numeric) a single prior alpha parameter. By default, `prior_alpha=1`
`iter`	(integer) number of MCMC iterations
`warmup`	(integer) number of warmup iterations
`thin`	(integer) thinning parameter. By default, `1`
`seed`	(integer) seed parameter
`init_T`	(matrix) `optional parameter`; initial transition matrix
`init_betas`	(numeric) `optional parameter`; initial beta parameters
`init_alpha`	(numeric) `optional parameter`; initial alpha parameter
`print_params`	(logical) `optional parameter`; print estimated parameters every iteration. By default, `TRUE`
`verbose`	(logical) `optional parameter`; print additional messages. By default, `TRUE`

Details

Please see supplementary information at doi:10.1186/s12859024057514 for more details on the algorithm.

For usage recommendations please see `https://github.com/LynetteCaitlin/oHMMed/blob/main/UsageRecommendations.pdf`.

Value

List with following elements:

- `data`: data used for simulation
- `samples`: list with samples
- `estimates`: list with various estimates
- `idx`: indices with iterations after the warmup period
- `priors`: prior parameters
- `inits`: initial parameters
- `last_iter`: list with samples from the last MCMC iteration
- `info`: list with various meta information about the object

References

Claus Vogl, Mariia Karapetiants, Burçin Yıldırım, Hrönn Kjartansdóttir, Carolin Kosiol, Juraj Bergman, Michal Majka, Lynette Caitlin Mikula. Inference of genomic landscapes using ordered Hidden Markov Models with emission densities (oHMMed). BMC Bioinformatics 25, 151 (2024). doi:10.1186/s12859024057514

Examples

```
# Simulate Poisson-Gamma data
N <- 2^10
true_T <- rbind(c(0.95, 0.05, 0),
                c(0.025, 0.95, 0.025),
                c(0.0, 0.05, 0.95))

true_betas <- c(2, 1, 0.1)
true_alpha <- 1

simdata_full <- hmm_simulate_gamma_poisson_data(L = N,
                                                mat_T = true_T,
                                                betas = true_betas,
                                                alpha = true_alpha)
simdata <- simdata_full$data
hist(simdata, breaks = 40, probability = TRUE,
     main = "Distribution of the simulated Poisson-Gamma data")
lines(density(simdata), col = "red")

# Set numbers of states to be inferred
n_states_inferred <- 3

# Set priors
prior_T <- generate_random_T(n_states_inferred)
prior_betas <- c(1, 0.5, 0.1)
prior_alpha <- 3

# Simmulation settings
iter <- 50
```

```
warmup <- floor(iter / 5) # 20 percent
thin <- 1
seed <- sample.int(10000, 1)
print_params <- FALSE # if TRUE then parameters are printed in each iteration
verbose <- FALSE # if TRUE then the state of the simulation is printed

# Run MCMC sampler
res <- hmm_mcmc_gamma_poisson(data = simdata,
                              prior_T = prior_T,
                              prior_betas = prior_betas,
                              prior_alpha = prior_alpha,
                              iter = iter,
                              warmup = warmup,
                              thin = thin,
                              seed = seed,
                              print_params = print_params,
                              verbose = verbose)
res

summary(res)# summary output can be also assigned to a variable

coef(res) # extract model estimates

# plot(res) # MCMC diagnostics
```

`hmm_mcmc_normal` *MCMC Sampler for the Hidden Markov Model with Normal emission densities*

Description

MCMC Sampler for the Hidden Markov Model with Normal emission densities

Usage

```
hmm_mcmc_normal(
  data,
  prior_T,
  prior_means,
  prior_sd,
  iter = 600,
  warmup = floor(iter/5),
  thin = 1,
  seed = sample.int(.Machine$integer.max, 1),
  init_T = NULL,
  init_means = NULL,
  init_sd = NULL,
  print_params = TRUE,
  verbose = TRUE
)
```

Arguments

`data`	(numeric) normal data
`prior_T`	(matrix) prior transition matrix
`prior_means`	(numeric) prior means
`prior_sd`	(numeric) a single prior standard deviation
`iter`	(integer) number of MCMC iterations
`warmup`	(integer) number of warmup iterations
`thin`	(integer) thinning parameter. By default, `1`
`seed`	(integer) `optional parameter`; seed parameter
`init_T`	(matrix) `optional parameter`; initial transition matrix
`init_means`	(numeric) `optional parameter`; initial means
`init_sd`	(numeric) `optional parameter`; initial standard deviation
`print_params`	(logical) `optional parameter`; print parameters every iteration. By default, `TRUE`
`verbose`	(logical) `optional parameter`; print additional messages. By default, `TRUE`

Details

Please see supplementary information at doi:10.1186/s12859024057514 for more details on the algorithm.

For usage recommendations please see `https://github.com/LynetteCaitlin/oHMMed/blob/main/UsageRecommendations.pdf`.

Value

List with following elements:

- `data`: data used for simulation
- `samples`: list with samples
- `estimates`: list with various estimates
- `idx`: indices with iterations after the warmup period
- `priors`: prior parameters
- `inits`: initial parameters
- `last_iter`: list with samples from the last MCMC iteration
- `info`: list with various meta information about the object

References

Claus Vogl, Mariia Karapetiants, Burçin Yıldırım, Hrönn Kjartansdóttir, Carolin Kosiol, Juraj Bergman, Michal Majka, Lynette Caitlin Mikula. Inference of genomic landscapes using ordered Hidden Markov Models with emission densities (oHMMed). BMC Bioinformatics 25, 151 (2024). doi:10.1186/s12859024057514

Examples

```
# Simulate normal data
N <- 2^10
true_T <- rbind(c(0.95, 0.05, 0),
                c(0.025, 0.95, 0.025),
                c(0.0, 0.05, 0.95))

true_means <- c(-5, 0, 5)
true_sd <- 1.5

simdata_full <- hmm_simulate_normal_data(L = N,
                                         mat_T = true_T,
                                         means = true_means,
                                         sigma = true_sd)
simdata <- simdata_full$data
hist(simdata,
     breaks = 40,
     probability = TRUE,
     main = "Distribution of the simulated normal data")
lines(density(simdata), col = "red")

# Set numbers of states to be inferred
n_states_inferred <- 3

# Set priors
prior_T <- generate_random_T(n_states_inferred)
prior_means <- c(-18, -1, 12)
prior_sd <- 3

# Simmulation settings
iter <- 50
warmup <- floor(iter / 5) # 20 percent
thin <- 1
seed <- sample.int(10000, 1)
print_params <- FALSE # if TRUE then parameters are printed in each iteration
verbose <- FALSE # if TRUE then the state of the simulation is printed

# Run MCMC sampler
res <- hmm_mcmc_normal(data = simdata,
                       prior_T = prior_T,
                       prior_means = prior_means,
                       prior_sd = prior_sd,
                       iter = iter,
                       warmup = warmup,
                       seed = seed,
                       print_params = print_params,
                       verbose = verbose)
res

summary(res) # summary output can be also assigned to a variable

coef(res) # extract model estimates
```

```
# plot(res) # MCMC diagnostics
```

hmm_simulate_gamma_poisson_data

Simulate data distributed according to oHMMed with gamma-poisson emission densities

Description

Simulate data distributed according to oHMMed with gamma-poisson emission densities

Usage

```
hmm_simulate_gamma_poisson_data(L, mat_T, betas, alpha)
```

Arguments

`L`	(integer) number of simulations
`mat_T`	(matrix) a square matrix with the initial state
`betas`	(numeric) `rate` parameter in `rgamma` for emission probabilities
`alpha`	(numeric) shape parameter in `rgamma` for emission probabilities

Value

Returns a list with the following elements:

- `data`: numeric vector with data
- `states`: an integer vector with "true" hidden states used to generate the data vector
- `pi`: numeric vector with prior probability of states

Examples

```
mat_T <- rbind(c(1-0.01, 0.01, 0),
               c(0.01, 1-0.02, 0.01),
               c(0, 0.01, 1-0.01))
L <- 2^7
betas <- c(0.1, 0.3, 0.5)
alpha <- 1

sim_data <- hmm_simulate_gamma_poisson_data(L = L,
                                            mat_T = mat_T,
                                            betas = betas,
                                            alpha = alpha)
hist(sim_data$data,
     breaks = 40,
     main = "Histogram of Simulated Gamma-Poisson Data",
     xlab = "")
sim_data
```

hmm_simulate_normal_data

Simulate data distributed according to oHMMed with normal emission densities

Description

Simulate data distributed according to oHMMed with normal emission densities

Usage

```
hmm_simulate_normal_data(L, mat_T, means, sigma)
```

Arguments

L	(integer) number of simulations
mat_T	(matrix) a square matrix with the initial state
means	(numeric) mean parameter in rnorm for emission probabilities
sigma	(numeric) sd parameter in rnorm for emission probabilities

Value

Returns a list with the following elements:

- data: numeric vector with data
- states: an integer vector with "true" hidden states used to generate the data vector
- pi: numeric vector with prior probability of states

Examples

```
mat_T0 <- rbind(c(1-0.01, 0.01, 0),
                c(0.01, 1-0.02, 0.01),
                c(0, 0.01, 1-0.01))
L <- 2^7
means0 <- c(-1,0,1)
sigma0 <- 1

sim_data <- hmm_simulate_normal_data(L = L,
                                     mat_T = mat_T0,
                                     means = means0,
                                     sigma = sigma0)

plot(density(sim_data$data), main = "Density of Simulated Normal Data")
sim_data
```

kullback_leibler_cont_appr

Calculate a Continuous Approximation of the Kullback-Leibler Divergence

Description

Calculate a Continuous Approximation of the Kullback-Leibler Divergence

Usage

```
kullback_leibler_cont_appr(p, q)
```

Arguments

p	(numeric) probabilities
q	(numeric) probabilities

Details

The continuous approximation of the Kullback-Leibler divergence is calculated as follows:

$$\frac{1}{n}\sum_{i=1}^{n}\left[\log(p_i)p_i - \log(q_i)p_i\right]$$

Value

Numeric vector

Examples

```
# Simulate n normally distributed variates
n <- 1000
dist1 <- rnorm(n)
dist2 <- rnorm(n, mean = 0, sd = 2)
dist3 <- rnorm(n, mean = 2, sd = 2)

# Estimate probability density functions
pdf1 <- density(dist1)
pdf2 <- density(dist2)
pdf3 <- density(dist3)

# Visualise PDFs
plot(pdf1, main = "PDFs", col = "red", xlim = range(dist3))
lines(pdf2, col = "blue")
lines(pdf3, col = "green")

# PDF 1 vs PDF 2
```

```
kullback_leibler_cont_appr(pdf1$y, pdf2$y)

# PDF 1 vs PDF 3
kullback_leibler_cont_appr(pdf1$y, pdf3$y)

# PDF 2 vs PDF 2
kullback_leibler_cont_appr(pdf2$y, pdf3$y)
```

kullback_leibler_disc *Calculate a Kullback-Leibler Divergence for a Discrete Distribution*

Description

Calculate a Kullback-Leibler Divergence for a Discrete Distribution

Usage

```
kullback_leibler_disc(p, q)
```

Arguments

p	(numeric) probabilities
q	(numeric) probabilities

Details

The Kullback-Leibler divergence for a discrete distribution is calculated as follows:

$$\sum_{i=1}^{n} p_i \log\left(\frac{p_i}{q_i}\right)$$

Value

Numeric vector

Examples

```
# Simulate n Poisson distributed variates
n <- 1000
dist1 <- rpois(n, lambda = 1)
dist2 <- rpois(n, lambda = 5)
dist3 <- rpois(n, lambda = 20)

# Generate common factor levels
x_max <- max(c(dist1, dist2, dist3))
all_levels <- 0:x_max

# Estimate probability mass functions
```

```
pmf_dist1 <- table(factor(dist1, levels = all_levels)) / n
pmf_dist2 <- table(factor(dist2, levels = all_levels)) / n
pmf_dist3 <- table(factor(dist3, levels = all_levels)) / n

# Visualise PMFs
barplot(pmf_dist1, col = "green", xlim = c(0, x_max))
barplot(pmf_dist2, col = "red", add = TRUE)
barplot(pmf_dist3, col = "blue", add = TRUE)

# Calculate distances
kullback_leibler_disc(pmf_dist1, pmf_dist2)
kullback_leibler_disc(pmf_dist1, pmf_dist3)
kullback_leibler_disc(pmf_dist2, pmf_dist3)
```

---

plot.hmm_mcmc_gamma_poisson

Plot Diagnostics for `hmm_mcmc_gamma_poisson` *Objects*

Description

This function creates a variety of diagnostic plots that can be useful when conducting Markov Chain Monte Carlo (MCMC) simulation of a gamma-poisson hidden Markov model (HMM). These plots will help to assess convergence, fit, and performance of the MCMC simulation

Usage

```
## S3 method for class 'hmm_mcmc_gamma_poisson'
plot(
  x,
  simulation = FALSE,
  true_betas = NULL,
  true_alpha = NULL,
  true_mat_T = NULL,
  true_states = NULL,
  show_titles = TRUE,
  log_statesplot = FALSE,
  ...
)
```

Arguments

`x`	(hmm_mcmc_gamma_poisson) HMM MCMC gamma-poisson object
`simulation`	(logical); default is `simulation=FALSE`, so the input data was empirical. If the input data was simulated, it must be set `simulation=TRUE`.
`true_betas`	(numeric) true betas. To be used if `simulation=TRUE`
`true_alpha`	(numeric) true alpha. To be used if `simulation=TRUE`
`true_mat_T`	(matrix) `optional parameter`; true transition matrix. To be used if `simulation=TRUE`

`true_states`	(integer) `optional parameter`; true states. To be used if `simulation=TRUE`
`show_titles`	(logical) if `TRUE` then titles are shown for all graphs. By default, `TRUE`
`log_statesplot`	(logical) if `TRUE` then log-statesplots are shown. By default, `FALSE`
`...`	not used

Value

Several diagnostic plots that can be used to evaluate the MCMC simulation of the gamma-poisson HMM

Examples

```
plot(example_hmm_mcmc_gamma_poisson)
```

`plot.hmm_mcmc_normal` *Plot Diagnostics for* `hmm_mcmc_normal` *Objects*

Description

This function creates a variety of diagnostic plots that can be useful when conducting Markov Chain Monte Carlo (MCMC) simulation of a normal hidden Markov model (HMM). These plots will help to assess convergence, fit, and performance of the MCMC simulation

Usage

```
## S3 method for class 'hmm_mcmc_normal'
plot(
  x,
  simulation = FALSE,
  true_means = NULL,
  true_sd = NULL,
  true_mat_T = NULL,
  true_states = NULL,
  show_titles = TRUE,
  ...
)
```

Arguments

`x`	(hmm_mcmc_normal) HMM MCMC normal object
`simulation`	(logical) `optional parameter`; default is `simulation=FALSE`, so the input data was empirical. If the input data was simulated, it must be set `simulation=TRUE`.
`true_means`	(numeric) `optional parameter`; true means. To be used if `simulation=TRUE`
`true_sd`	(numeric) `optional parameter`; true standard deviation. To be used if `simulation=TRUE`

`true_mat_T`	(matrix) `optional parameter`; true transition matrix. To be used if `simulation=TRUE`
`true_states`	(integer) `optional parameter`; true states. To be used if `simulation=TRUE`
`show_titles`	(logical) `optional parameter`; if `TRUE` then titles are shown for all graphs. By default, `TRUE`
`...`	not used

Value

Several diagnostic plots that can be used to evaluate the MCMC simulation of the normal HMM

Examples

```
plot(example_hmm_mcmc_normal)
```

`posterior_prob_gamma_poisson` *Forward-Backward Algorithm to Calculate the Posterior Probabilities of Hidden States in Poisson-Gamma Model*

Description

Forward-Backward Algorithm to Calculate the Posterior Probabilities of Hidden States in Poisson-Gamma Model

Usage

```
posterior_prob_gamma_poisson(data, pi, mat_T, betas, alpha)
```

Arguments

`data`	(numeric) Poisson data
`pi`	(numeric) prior probability of states
`mat_T`	(matrix) transition probability matrix
`betas`	(numeric) vector with prior rates
`alpha`	(numeric) prior scale

Details

Please see supplementary information at doi:10.1186/s12859024057514 for more details on the algorithm.

Value

List with the following elements:

- F: auxiliary forward variables
- B: auxiliary backward variables
- s: weights

Examples

```
mat_T <- rbind(c(1-0.01,0.01,0),
               c(0.01,1-0.02,0.01),
               c(0,0.01,1-0.01))
L <- 2^10
betas <- c(0.1, 0.3, 0.5)
alpha <- 1

sim_data <- hmm_simulate_gamma_poisson_data(L = L,
                                            mat_T = mat_T,
                                            betas = betas,
                                            alpha = alpha)
pi <- sim_data$pi
hmm_poison_data <- sim_data$data
hist(hmm_poison_data, breaks = 100)

# Calculate posterior probabilities of hidden states
post_prob <- posterior_prob_gamma_poisson(data = hmm_poison_data,
                                          pi = pi,
                                          mat_T = mat_T,
                                          betas = betas,
                                          alpha = alpha)
str(post_prob)
```

`posterior_prob_normal` *Forward-Backward Algorithm to Calculate the Posterior Probabilities of Hidden States in Normal Model*

Description

Forward-Backward Algorithm to Calculate the Posterior Probabilities of Hidden States in Normal Model

Usage

```
posterior_prob_normal(data, pi, mat_T, means, sdev)
```

Arguments

`data`	(numeric) normal data
`pi`	(numeric) prior probability of states
`mat_T`	(matrix) transition probability matrix
`means`	(numeric) vector with prior means
`sdev`	(numeric) prior standard deviation

Details

Please see supplementary information at doi:10.1186/s12859024057514 for more details on the algorithm.

Value

List with the following elements:

- `F`: auxiliary forward variables
- `B`: auxiliary backward variables
- `s`: weights

Examples

```
prior_mat <- rbind(c(1-0.05, 0.05, 0),
                   c(0.05, 1-0.1, 0.05),
                   c(0, 0.05, 1-0.05))

prior_means <- c(-0.1, 0.0, 0.1)
prior_sd  <- sqrt(0.1)
L <- 100

# Simulate HMM model based on normal data based on prior information
sim_data_normal <- hmm_simulate_normal_data(L = L,
                                            mat_T = prior_mat,
                                            means = prior_means,
                                            sigma = prior_sd)
pi <- sim_data_normal$pi
# pi <- get_pi(prior_mat)
hmm_norm_data <- sim_data_normal$data

# Calculate posterior probabilities of hidden states
post_prob <-  posterior_prob_normal(data = hmm_norm_data,
                                    pi = pi,
                                    mat_T = prior_mat,
                                    means = prior_means,
                                    sdev = prior_sd)
str(post_prob)
```

Index