Package 'betaHMM'

November 28, 2024

Type Package

Title A Hidden Markov Model Approach for Identifying Differentially Methylated Sites and Regions for Beta-Valued DNA Methylation Data

Version 1.2.0

Description A novel approach utilizing a homogeneous hidden Markov model. And effectively model untransformed beta values. To identify DMCs while considering the spatial. Correlation of the adjacent CpG sites.

License GPL-3

Encoding UTF-8

Imports stats, ggplot2, scales, methods, pROC, foreach, doParallel, parallel, cowplot, dplyr, tidyr, tidyselect, stringr, utils

Depends R (>= 4.3.0), SummarizedExperiment, S4Vectors, GenomicRanges

RoxygenNote 7.2.3

biocViews DNAMethylation, DifferentialMethylation, ImmunoOncology, BiomedicalInformatics, MethylationArray, Software, MultipleComparison, Sequencing, Spatial, Coverage, GeneTarget, HiddenMarkovModel, Microarray

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

VignetteBuilder knitr

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/betaHMM

git_branch RELEASE_3_20

git_last_commit 927ed74

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-28

Author Koyel Majumdar [cre, aut] (<https://orcid.org/0000-0001-6469-488X>), Romina Silva [aut], Antoinette Sabrina Perry [aut], Ronald William Watson [aut], Isobel Claire Gorley [aut] (<https://orcid.org/0000-0001-7713-681X>), Thomas Brendan Murphy [aut] (<https://orcid.org/0000-0002-5668-7046>), Florence Jaffrezic [aut], Andrea Rau [aut] (<https://orcid.org/0000-0001-6469-488X>)

Maintainer Koyel Majumdar <koyel.majumdar@ucdconnect.ie>

Contents

`annotatedData` *Accessors for the betaHMM package.*

Description

The accessor methods for accessing the betaHMMResults/ dmcResults/ dmrResults/ threshold_Results metadata.

Usage

```
annotatedData(object)

K(object)

N(object)

R(object)

A(object)

phi(object)
```

```
treatment_group(object)

llk(object)

tau(object)

hidden_states(object)

chromosome_number(object)

AUC(object)

uncertainty(object)

model_parameters(object)

threshold(object)

## S4 method for signature 'betaHMMResults'
annotatedData(object)

## S4 method for signature 'threshold_Results'
annotatedData(object)

## S4 method for signature 'RangedSummarizedExperiment'
K(object)

## S4 method for signature 'betaHMMResults'
K(object)

## S4 method for signature 'dmcResults'
K(object)

## S4 method for signature 'threshold_Results'
K(object)

## S4 method for signature '`NULL`'
K(object)

## S4 method for signature 'RangedSummarizedExperiment'
N(object)

## S4 method for signature 'betaHMMResults'
N(object)

## S4 method for signature 'dmcResults'
N(object)

## S4 method for signature '`NULL`'
N(object)

## S4 method for signature 'RangedSummarizedExperiment'
```

```
R(object)

## S4 method for signature 'betaHMMResults'
R(object)

## S4 method for signature 'dmcResults'
R(object)

## S4 method for signature '`NULL`'
R(object)

## S4 method for signature 'RangedSummarizedExperiment'
A(object)

## S4 method for signature 'betaHMMResults'
A(object)

## S4 method for signature '`NULL`'
A(object)

## S4 method for signature 'RangedSummarizedExperiment'
tau(object)

## S4 method for signature 'betaHMMResults'
tau(object)

## S4 method for signature '`NULL`'
tau(object)

## S4 method for signature 'RangedSummarizedExperiment'
treatment_group(object)

## S4 method for signature 'betaHMMResults'
treatment_group(object)

## S4 method for signature 'dmcResults'
treatment_group(object)

## S4 method for signature '`NULL`'
treatment_group(object)

## S4 method for signature 'RangedSummarizedExperiment'
llk(object)

## S4 method for signature 'betaHMMResults'
llk(object)

## S4 method for signature '`NULL`'
llk(object)

## S4 method for signature 'RangedSummarizedExperiment'
phi(object)
```

```
## S4 method for signature 'betaHMMResults'
phi(object)

## S4 method for signature 'threshold_Results'
phi(object)

## S4 method for signature '`NULL`'
phi(object)

## S4 method for signature 'RangedSummarizedExperiment'
hidden_states(object)

## S4 method for signature 'betaHMMResults'
hidden_states(object)

## S4 method for signature 'threshold_Results'
hidden_states(object)

## S4 method for signature '`NULL`'
hidden_states(object)

## S4 method for signature 'RangedSummarizedExperiment'
chromosome_number(object)

## S4 method for signature 'betaHMMResults'
chromosome_number(object)

## S4 method for signature 'dmrResults'
chromosome_number(object)

## S4 method for signature '`NULL`'
chromosome_number(object)

## S4 method for signature 'RangedSummarizedExperiment'
AUC(object)

## S4 method for signature 'dmcResults'
AUC(object)

## S4 method for signature '`NULL`'
AUC(object)

## S4 method for signature 'RangedSummarizedExperiment'
uncertainty(object)

## S4 method for signature 'dmcResults'
uncertainty(object)

## S4 method for signature '`NULL`'
uncertainty(object)
```

```
## S4 method for signature 'RangedSummarizedExperiment'
model_parameters(object)

## S4 method for signature 'threshold_Results'
model_parameters(object)

## S4 method for signature '`NULL`'
model_parameters(object)

## S4 method for signature 'RangedSummarizedExperiment'
threshold(object)

## S4 method for signature 'threshold_Results'
threshold(object)

## S4 method for signature '`NULL`'
threshold(object)
```

Arguments

object	a betaHMMResults/ dmcResults/ threshold_Results object.

Value

Output varies depending on the method.

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

annotation_data *MethylationEPIC manifest data.*

Description

A dataset containing a subset of the manifest data from the Illumina MethylationEPIC beadchip array. A subset of the complete dataset has been uploaded in the package for testing purpose. The complete dataset is available on GitHub.

Usage

```
data(annotation_data)
```

Format

A data frame with 100 rows and 9 columns.

- IlmnID: The unique identifier from the Illumina CG database, i.e. the probe ID.
- Genome_Build: The genome build referenced by the Infinium MethylationEPIC manifest.
- CHR: The chromosome containing the CpG (Genome_Build = 37).
- MAPINFO: The chromosomal coordinates of the CpG sites.
- UCSC_RefGene_Name: The target gene name(s), from the UCSC database. Note: multiple listings of the same gene name indicate splice variants.
- UCSC_RefGene_Accession: The UCSC accession numbers of the target transcripts. Accession numbers are in the same order as the target gene transcripts.
- UCSC_RefGene_Group: Gene region feature category describing the CpG position, from UCSC. Features are listed in the same order as the target gene transcripts.
- UCSC_CpG_Islands_Name: The chromosomal coordinates of the CpG Island from UCSC.
- Relation_to_UCSC_CpG_Island: The location of the CpG relative to the CpG island.

Value

A data frame containing the array design for Illumina's Human Methylation EPIC microarray for the chromosome 7. Based on the v1.0b2 version of the manifest file.

See Also

pca_methylation_data

AUC_DM_analysis *Area under the curve method for calculating dissimilarities between estimated distributions*

Description

The function is used to calculate the dissimilarity between the cumulative distributions estimated in each hidden state.

Usage

```
AUC_DM_analysis(M, N, R, K, tau, A, phi)
```

Arguments

M	Number of methylation states to be identified in a single DNA sample.
N	Number of patients or DNA sample replicates collected for each treatment group.
R	Number of treatment groups (For. eg: Benign and Tumour).
K	Number of hidden states estimated.
tau	The initial distribution for hidden states in the betaHMM model.
A	The transition matrix for hidden states in the betaHMM model.
phi	The shape parameters estimated for the observed data and the estimated hidden states in the betaHMM model.

Value

A dataframe returning the hidden state, the AUC value and the distributions which resulted in highest AUC value calculated for the corresponding hidden state.

backward *Backward algorithm*

Description

Computes the backward probabilities for K hidden states using the initial distribution, transition matrix and distribution parameters.

Usage

```
backward(probabilities, trained_params)
```

Arguments

probabilities	A matrix containing the marginal probability distribution of the K hidden states within the Markov model.
trained_params	A list containing the initialised shape parameters for the betaHMM model (initial distribution, transition matrix, shape parameters for the observed data).

Value

A list containing:

- log_beta - A matrix of dimension $C * K$ (where C is the number of CpG sites and K is the number of hidden states) containing the logarithmized backward probabilities.
- scaled_logL - The log-likelihood calculated using the backward probabilities.

`BaumWelch` *HMM parameter estimation using Baum-Welch algorithm*

Description

The function determines the parameters of a homogeneous beta hidden Markov model (betaHMM), wherein the Baum-Welch algorithm constitutes a variant of the EM (Estimation-Maximization) procedure.

Usage

```
BaumWelch(
  data,
  trained_params = NULL,
  K,
  M,
  N,
  R,
  seed = NULL,
  iterations = 100
)
```

Arguments

`data`	A dataframe of dimension $C \times (N \times R)$ containing methylation values for C CpG sites from R treatment groups each having N DNA samples.
`trained_params`	A list containing the initialized model parameters: • A - The transition matrix for the betaHMM model. • tau - The initial distribution for the betaHMM model. • phi - The shape parameters for the observation sequence data in the betaHMM model.
`K`	The number of hidden states identified using the betaHMM model.
`M`	Number of methylation states to be identified in a single DNA sample.
`N`	Number of DNA samples (patients/replicates) collected for each treatment group.
`R`	Number of treatment groups (For. eg: Benign and Tumour).
`seed`	Seed to allow for reproducibility (default = NULL).
`iterations`	Number of iterations for algorithm convergence.

Value

A list containing:

- A - The transition matrix estimated for the betaHMM model.
- tau - The initial distribution estimated for the betaHMM model.
- phi - The shape parameters estimated for the observed data in the betaHMM model.
- log_vec - A vector containing the log-likelihood values calculated for each iteration of the algorithm.
- z - A matrix of dimension $C \times K$ containing the conditional posterior probability of each CpG site belonging to each of the K hidden states.

`betaHMM` *HMM for beta valued DNA data*

Description

This is the primary user interface for the `betaHMM` function Generic S4 methods are implemented to eatimate the parameters of a homogeneous hidden Markov model for the beta valued DNA methylation data. The supported classes are `matrix`, `data.frame`, `RangedSummarizedExperiment` and `GRanges`. The output of `betaHMM` method is an S4 object of class `betaHMMResults`.

Usage

```
betaHMM(methylation_data, annotation_file, ...)

## S4 method for signature 'matrix,matrix'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'data.frame,data.frame'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
```

```
  iterations = 100,
  ...
)

## S4 method for signature
## 'RangedSummarizedExperiment,RangedSummarizedExperiment'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'GRanges,GRanges'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'RangedSummarizedExperiment,matrix'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'RangedSummarizedExperiment,data.frame'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
```

```
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'RangedSummarizedExperiment,GRanges'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'matrix,RangedSummarizedExperiment'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'data.frame,RangedSummarizedExperiment'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'data.frame,GRanges'
```

```
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'matrix,data.frame'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'matrix,GRanges'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'GRanges,matrix'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
```

```
  ...
)

## S4 method for signature 'GRanges,data.frame'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'GRanges,RangedSummarizedExperiment'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)

## S4 method for signature 'data.frame,matrix'
betaHMM(
  methylation_data,
  annotation_file,
  M = 3,
  N = 4,
  R = 2,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)
```

Arguments

`methylation_data`
: A dataframe of dimension $(C \times (N \times R)) + 1$ containing methylation values for C CpG sites from R treatment groups each having N DNA samples and the IlmnID for each CpG site. Maybe provided as a matrix or data.frame, GRanges or RangedSummarizedExperiment object.

annotation_file
: A dataframe containing the EPIC methylation annotation file. Maybe provided as a matrix or data.frame, GRanges or RangedSummarizedExperiment object.

...
: Extra arguments

M
: Number of methylation states to be identified in a single DNA sample.

N
: Number of DNA samples (patients/replicates) collected for each treatment group.

R
: Number of treatment groups (For. eg: Benign and Tumour).

treatment_group
: The names of each treatment groups/ conditions. If no value is passed then default values of sample names, e.g. Sample 1, Sample 2, etc are used as legend text (default = NULL).

parallel_process
: The 'TRUE' option results in parallel processing of the models for each chromosome for increased computational efficiency. The default option has been set as 'FALSE' due to package testing limitations.

seed
: Seed to allow for reproducibility (default = NULL).

iterations
: Number of iterations for algorithm convergence (default=100).

Value

An S4 object of class `betaHMMResults`, where conditional probabilities of each CpG site belonging to a hidden state is stored as a SimpleList of assay data, and the corresponding estimated model parameters, log-likelihood values, and most probable hidden state sequence for each chromosome are stored as metadata.

Author(s)

Koyel Majumdar

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)
```

```
## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

`betaHMMResults-class` *betaHMMResults object and constructor*

Description

`betaHMMResults` is a subclass of `RangedSummarizedExperiment`, used to store the betaHMM results as well as the annotated data useful for plotting.

Usage

```
betaHMMResults(SummarizedExperiment, annotatedData)
```

Arguments

`SummarizedExperiment`	a `RangedSummarizedExperiment` of `betaHMM` results
`annotatedData`	The annotated data passed as an input argument to the betaHMM package.

Details

This constructor function would not typically be used by "end users". This simple class extends the `RangedSummarizedExperiment` class of the SummarizedExperiment package to allow other packages to write methods for results objects from the betaHMM package. It is used by to wrap up the results table.

Value

a betaHMMResults object

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
```

```
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

---

betaHMMrun *The betaHMM model parameter estimation function*

Description

A homogeneous hidden Markov model for the beta valued DNA methylation data.

Usage

```
betaHMMrun(
  methylation_data,
  annotation_file,
  M,
  N,
  R,
  treatment_group = NULL,
  parallel_process = FALSE,
  seed = NULL,
  iterations = 100,
  ...
)
```

Arguments

methylation_data	A dataframe of dimension $(C \times (N \times R)) + 1$ containing methylation values for C CpG sites from R treatment groups each having N DNA samples and the IlmnID for each CpG site.
annotation_file	A dataframe containing the EPIC methylation annotation file.
M	Number of methylation states to be identified in a single DNA sample.
N	Number of DNA samples (patients/replicates) collected for each treatment group.
R	Number of treatment groups (For. eg: Benign and Tumour).
treatment_group	The names of each treatment groups/ conditions. If no value is passed then default values of sample names, e.g. Sample 1, Sample 2, etc are used as legend text (default = NULL).
parallel_process	The 'TRUE' option results in parallel processing of the models for each chromosome for increased computational efficiency. The default option has been set as 'FALSE' due to package testing limitations.

seed	Seed to allow for reproducibility (default = NULL).
iterations	Number of iterations for algorithm convergence (default=100).
...	Extra arguments

Details

The betaHMM function employs initially set parameters (utilizing a basic 3-state beta hidden Markov model) to estimate the parameters of the homogeneous hidden Markov model, adapted for beta-valued DNA methylation data, through implementation of the Baum-Welch algorithm. Subsequently, the derived parameters are utilized to ascertain the most probable sequence of hidden states using the Viterbi algorithm.

Value

The function returns an object of the betaHMMResults class which contains a SimpleList of assay data containing the posterior probability of each CpG site belonging to each of the K hidden states and the following values as metadata:

- K - The number of hidden states identified using the betaHMM model.
- C - The number of CpG sites analysed using the betaHMM model.
- N - The number of DNA samples corresponding to each treatment group analysed using the betaHMM model.
- R - The number of treatment groups analysed using the betaHMM model.
- A - The transition matrix estimated for the betaHMM model.
- tau - The initial distribution estimated for the betaHMM model.
- treatment_group - The names of the treatment groups/conditions analysed.
- phi - The shape parameters estimated for the observed data in the betaHMM model.
- llk - A vector containing the log-likelihood values calculated for each iteration of the algorithm.
- hidden_states - The vector containing the estimated hidden states for each CpG sites.

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)
```

```
# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

---

dmcResults-class *dmcResults object and constructor*

Description

dmcResults is a subclass of RangedSummarizedExperiment, used to store the DMCs identified.

Usage

```
dmcResults(SummarizedExperiment)
```

Arguments

SummarizedExperiment
: a RangedSummarizedExperiment of dmcResults results.

Details

This constructor function would not typically be used by "end users". This simple class extends the RangedSummarizedExperiment class of the SummarizedExperiment package to allow other packages to write methods for results objects from the dmc_identification function. It is used by to wrap up the results table.

Value

a dmcResults object

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
```

```
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

`dmc_identification` *DMC identification from estimated betaHMM model parameters*

Description

This is the primary user interface for the `dmc_identification` function. Generic S4 methods are implemented to identify the DMCs from the estimated betaHMM model parameters for each chromosome. The supported class is a `betaHMMResults` object. The output is an S4 object of class of `dmcResults`.

Usage

```
dmc_identification(betaHMM_object, ...)

## S4 method for signature 'betaHMMResults'
dmc_identification(
  betaHMM_object,
  AUC_threshold = 0.8,
  uncertainty_threshold = 0.2,
  ...
)
```

Arguments

`betaHMM_object`	An S4 object of class `betaHMMResults`
`...`	extra arguments
`AUC_threshold`	The threshold for AUC metric for each chromosome.
`uncertainty_threshold`	The threshold for uncertainty of belonging to a particular hidden state, for each chromosome.

Value

An S4 object of class `dmcResults`.

See Also

`betaHMM`

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

---

dmc_identification_run

DMC identification from estimated betaHMM model parameters

Description

The dissimilarities between the cumulative distributions calculated for each hidden state are determined through employment of the area-under-curve (AUC) technique. By incorporating user-defined threshold values for AUC alongside the associated uncertainties in membership within that hidden state, the aim is to pinpoint the most distinctively methylated states. This process facilitates the identification of CpGs that exhibit the most notable differential methylation, guided by the predefined threshold criteria.

Usage

```
dmc_identification_run(
  betaHMM_object,
  AUC_threshold = 0.8,
  uncertainty_threshold = 0.2,
  ...
)
```

Arguments

betaHMM_object	A betaHMMResults object.
AUC_threshold	The threshold for AUC metric for each chromosome.
uncertainty_threshold	The threshold for uncertainty of belonging to a particular hidden state, for each chromosome.
...	extra arguments

Value

The function returns an object of the dmcResults class which contains a SimpleList of assay data which contains the following values:

- CHR Chromosome number
- MAPINFO Mapinfo
- IlmnID IlmnID
- N*R columns containing methylation states
- hidden_state The assigned hidden_state
- DMC The value is 1 if the CpG is a DMC else 0.

The object contains the following values as the metadata:

- A list containing the AUC values for K hidden states for each chromosome and the conditions compared which resulted in the highest AUC value when more than 2 conditions are compared.
- A list containing the conditional probability values for K hidden states for each chromosome.
- The treatment group labels.
- K The number of hidden states estimated.
- N The number of DNA replicates/patients for each treatment group.
- R The number of treatment groups to be compared.

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)
```

```
# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

---

dmrResults-class *dmrResults object and constructor*

Description

dmrResults is a subclass of RangedSummarizedExperiment, used to store the DMRs identified.

Usage

```
dmrResults(SummarizedExperiment)
```

Arguments

SummarizedExperiment
: a dmrResults results.

Details

This constructor function would not typically be used by "end users". This simple class extends the RangedSummarizedExperiment class of the SummarizedExperiment package to allow other packages to write methods for results objects from the dmr_identification function. It is used by to wrap up the results table.

Value

a dmrResults object

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)
```

```
# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

`dmr_identification` *DMR identification from DMCs identified*

Description

This is the primary user interface for the `dmr_identification` function. Generic S4 methods are implemented to identify the DMRs from the DMCs identified in each chromosome. The supported classes are `data.frame` and `dmcResults` object. The output is an S4 object of class `dmrResults`.

Usage

```
dmr_identification(dmc_identification_object, ...)

## S4 method for signature 'dmcResults'
dmr_identification(dmc_identification_object, DMC_count = 2, ...)

## S4 method for signature 'matrix'
dmr_identification(dmc_identification_object, DMC_count = 2, ...)

## S4 method for signature 'data.frame'
dmr_identification(dmc_identification_object, DMC_count = 2, ...)
```

Arguments

`dmc_identification_object`	a `dmcResults` object or the assay data from the dmcResults.
`...`	extra arguments
`DMC_count`	The minimal number of consecutive CpGs in a DMR.

Value

An S4 object of class `dmrResults` where the CpG site information for each DMR is stored as a SimpleList of assay data and the chromosomes analysed by the model is stored as the metadata.

See Also

`betaHMM`

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

`dmr_identification_run` *DMR identification from DMCs identified*

Description

Function to identify the DMRs from the DMCs identified in each chromosome.

Usage

```
dmr_identification_run(
  dmc_identification_object,
  DMC_count = 2,
  parallel_process = FALSE,
  ...
)
```

Arguments

`dmc_identification_object`	a `dmcResults` object or the assay data from the dmcResults.
`DMC_count`	The minimal number of consecutive CpGs in a DMR.

parallel_process
: The 'TRUE' option results in parallel processing of the DMCs from each chromosome for increased computational efficiency. The default option has been set as 'FALSE' due to package testing limitations.

...
: extra arguments

Value

A dmrResults object containing a SimpleList of assay data containing the following data:

- start_CpG - The starting CpG site IlmnID in the particular DMR
- end_CpG - The ending CpG site IlmnID in the particular DMR
- DMR_size - Number of CPG sites identified in the DMR
- chr_dmr - The chromosome corresponding to the CpG sites in the DMR.
- map_start - MAPINFO of starting CpG site in the particular DMR
- map_end - MAPINFO of the ending CpG site in the particular DMR

The object also returns the chromosomes analysed by the betaHMM model as the metadata.

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

forward *Forward algorithm*

Description

Computes the forward probabilities for K hidden states using the initial distribution, transition matrix and distribution parameters.

Usage

```
forward(probabilities, trained_params)
```

Arguments

probabilities	A matrix containing the marginal probability distribution of the K states within the Markov model.
trained_params	A list containing the initialised shape parameters for the betaHMM model (initial distribution, transition matrix, shape parameters for the observed data).

Value

A list containing:

- log_alpha - A $C * K$ matrix (where C is the number of CpG sites and K is the number of hidden states) containing the logarithmized forward probabilities.
- scaled_logL - The log-likelihood calculated using the forward probabilities.

initialise_parameters *Initialising the betaHMM model parameters*

Description

Initialise the betaHMM model parameters.

Usage

```
initialise_parameters(data, M, N, R, seed = NULL)
```

Arguments

data	A dataframe of dimension $C \times (N \times R)$ containing methylation values for C CpG sites from R treatment groups each having N replicates or each collected from N patients.
M	Number of methylation states to be identified in a DNA sample.
N	Number of patients or DNA sample replicates collected for each treatment group.
R	Number of treatment groups (For. eg: Benign and Tumour).
seed	Seed to allow for reproducibility (default = NULL).

Details

Computes the shape parameters using the threshold_function to initialise the shape parameters of the model.

Value

A list containing:

- A - The transition matrix for the betaHMM model.
- tau - The initial distribution for the betaHMM model.
- phi - The shape parameters for the observation sequence data in the betaHMM model.

---

pca_methylation_data *Simulated DNA methylation data*

Description

A subset of the dataset containing beta methylation values from $R = 2$ sample types (Benign and Tumour), collected from $N = 4$ patients from the a prostate cancer study. The dataset contains methylation values corresponding to chromosome 7.

Usage

```
data(pca_methylation_data)
```

Format

A data frame with 38672 rows and 9 columns. The data contain no missing values.

- IlmnID: The unique identifier from the Illumina CG database, i.e. the probe ID.
- Benign_Patient_1: Methylation values from benign tissue from patient 1.
- Benign_Patient_2: Methylation values from benign tissue from patient 2.
- Benign_Patient_3: Methylation values from benign tissue from patient 3.
- Benign_Patient_4: Methylation values from benign tissue from patient 4.
- Tumour_Patient_1: Methylation values from tumor tissue from patient 1.
- Tumour_Patient_2: Methylation values from tumor tissue from patient 2.
- Tumour_Patient_3: Methylation values from tumor tissue from patient 3.
- Tumour_Patient_4: Methylation values from tumor tissue from patient 4.

Details

The array data were then normalized and and probes located outside of CpG sites and on the sex chromosome were filtered out. The CpG sites with missing values were removed from the resulting dataset. A subset of the complete dataset has been uploaded in the package for testing purposes. The complete dataset is available on GitHub.

Value

A data frame containing a subset of methylation data from real study.

See Also

annotation_data

---

plot *Visualize results from* betaHMM/ dmc_identification/ threshold_identification *functions.*

Description

Visualize results from betaHMM/ dmc_identification/ threshold_identification functions.

Usage

```
plot(x, ...)

## S4 method for signature 'betaHMMResults'
plot(
  x,
  chromosome = NULL,
  what = c("fitted density", "kernel density", "uncertainty"),
  treatment_group = NULL,
  AUC = NULL,
  uncertainty_threshold = 0.2,
  title = NULL,
  ...
)

## S4 method for signature 'dmcResults'
plot(
  x,
  start_CpG = NULL,
  end_CpG = NULL,
  treatment_group = NULL,
  N = NULL,
  title = NULL,
  ...
)

## S4 method for signature 'threshold_Results'
plot(x, plot_threshold = TRUE, title = NULL, ...)
```

Arguments

x	An object of class betaHMMResults/ dmcResults/ threshold_Results object.
...	Other graphics parameters.
chromosome	The chromosome number for which the plot is to be displayed.
what	The different plots that can be obtained are either 'fitted density','kernel density' or 'uncertainty' (default = 'fitted density').

`treatment_group`
: The names of the different treatment groups to be displayed in the plot.If no value is passed then the sample names estimated by the `betaHMM` function are used.

`AUC`
: The AUC values for that chromosome.

`uncertainty_threshold`
: The uncertainty threshold value used for DMC identification.

`title`
: The title that the user wants to display. If no title is to be displayed the default is 'NULL'.

`start_CpG`
: The IlmnID of starting CpG site when plotting the DMCs.

`end_CpG`
: The IlmnID of ending CpG site/ the total number of CpGs to be plotted excluing the starting CpG site when plotting the DMCs.

`N`
: The number of DNA samples corresponding to each treatment group analysed using the betaHMM model. If 'NULL', the value from `betaHMMResults` object is selected.

`plot_threshold`
: The "TRUE" option displays the threshold points in the graph for the 3 state betaHMM model (default = "FALSE").

Value

This function displays the following plots as requested by the user when analysing the `betaHMMResults` output:

- fitted density estimates - Plot showing the fitted density estimates of the clustering solution under the optimal model selected.
- kernel density estimates - Plot showing the kernel density estimates of the clustering solution under the optimal model selected.
- uncertainty - A boxplot showing the uncertainties in the hidden state estimation.

The function displays the DMCs and DMRs plot from the `dmcResults` object.

The function displays the plot for the estimated shape parameters and threshold for the methylation states in a single DNA treatment condition from the `threshold_Results` object.

Author(s)

Koyel Majumdar

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))
```

```
## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")

## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

`sample_annotation_file` *MethylationEPIC manifest data.*

Description

A dataset containing a subset of the manifest data from the Illumina MethylationEPIC beadchip array. A subset of the complete dataset has been uploaded in the package for testing purpose. The complete dataset is available on GitHub.

Usage

```
data(sample_annotation_file)
```

Format

A data frame with 100 rows and 9 columns.

- IlmnID: The unique identifier from the Illumina CG database, i.e. the probe ID.
- Genome_Build: The genome build referenced by the Infinium MethylationEPIC manifest.
- CHR: The chromosome containing the CpG (Genome_Build = 37).
- MAPINFO: The chromosomal coordinates of the CpG sites.
- UCSC_RefGene_Name: The target gene name(s), from the UCSC database. Note: multiple listings of the same gene name indicate splice variants.
- UCSC_RefGene_Accession: The UCSC accession numbers of the target transcripts. Accession numbers are in the same order as the target gene transcripts.
- UCSC_RefGene_Group: Gene region feature category describing the CpG position, from UCSC. Features are listed in the same order as the target gene transcripts.
- UCSC_CpG_Islands_Name: The chromosomal coordinates of the CpG Island from UCSC.
- Relation_to_UCSC_CpG_Island: The location of the CpG relative to the CpG island.

Value

A data frame containing the array design for Illumina's Human Methylation EPIC microarray for the simulated CpG sites. Based on the v1.0b2 version of the manifest file.

See Also

sample_methylation_file

---

sample_methylation_file

Simulated DNA methylation data

Description

A dataset containing simulated beta methylation values from $R = 2$ sample types (Benign and Tumour), collected from $N = 4$ patients.

Usage

```
data(sample_methylation_file)
```

Format

A data frame with 100 rows and 9 columns. The data contain no missing values.

- IlmnID: The unique identifier from the Illumina CG database, i.e. the probe ID.
- Benign_Patient_1: Methylation values from benign tissue from patient 1.
- Benign_Patient_2: Methylation values from benign tissue from patient 2.
- Benign_Patient_3: Methylation values from benign tissue from patient 3.
- Benign_Patient_4: Methylation values from benign tissue from patient 4.
- Tumour_Patient_1: Methylation values from tumor tissue from patient 1.
- Tumour_Patient_2: Methylation values from tumor tissue from patient 2.
- Tumour_Patient_3: Methylation values from tumor tissue from patient 3.
- Tumour_Patient_4: Methylation values from tumor tissue from patient 4.

Details

The array data were then normalized and and probes located outside of CpG sites and on the sex chromosome were filtered out. The CpG sites with missing values were removed from the resulting dataset. A subset of the complete dataset has been uploaded in the package for testing purposes. The complete dataset is available on GitHub.

Value

The data frame containing simulated methylation values.

See Also

sample_annotation_file

---

summary *Summarize results from the three betaHMM workflow functions*

Description

A function to summarize the betaHMMResults, dmcResults or dmrResults objects.

Usage

```
summary(object, ...)

## S4 method for signature 'betaHMMResults'
summary(object, ...)

## S4 method for signature 'dmcResults'
summary(object, ...)

## S4 method for signature 'dmrResults'
summary(object, ...)
```

Arguments

object	An object of class 'betaHMMResults' or 'dmcResults' or 'dmrResults'.
...	Additional arguments

Value

Summary of the 'betaHMMResults' or 'dmcResults' or 'dmrResults' object.

Author(s)

Koyel majumdar

See Also

betaHMM, dmc_identification, dmr_identification

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)

## read files
data(sample_methylation_file)
data(sample_annotation_file)
# Run betaHMM function
beta_out <- betaHMM(sample_methylation_file[1:50,],
                    sample_annotation_file[1:50,],
                    M = 3, N = 4, R = 2,iterations=2,
                    parallel_process = FALSE, seed = 12345,
                    treatment_group = c("Benign","Tumour"))

## Run dmc_identification function
dmc_out <- dmc_identification(beta_out)

# Run dmr_identification function
dmr_out <- dmr_identification(dmc_out, parallel_process = FALSE)

# Plot functions
# Get the AUC values calculated for each hidden state
AUC_chr <- AUC(dmc_out)

## plot the uncertainty for each hidden state
plot(beta_out, chromosome = "1", what = "uncertainty")
```

`threshold_identification` *HMM for beta valued DNA data for a single treatment condition*

Description

The supported classes are `matrix` and `data.frame`. The output of `threshold_identification` is an S4 object of class `threshold_Results`.

Usage

```
threshold_identification(object1, ...)

## S4 method for signature 'matrix'
threshold_identification(
  object1,
  package_workflow = TRUE,
  annotation_file = NULL,
  M = 3,
  N = 4,
  parameter_estimation_only = FALSE,
  seed = NULL,
  ...
)
```

```
## S4 method for signature 'data.frame'
threshold_identification(
  object1,
  package_workflow = TRUE,
  annotation_file = NULL,
  M = 3,
  N = 4,
  parameter_estimation_only = FALSE,
  seed = NULL,
  ...
)
```

Arguments

object1	Methylation data and IlmnID. Maybe provided as a matrix or dataframe.
...	Extra parameters.
package_workflow	Flag set to TRUE if method called from package workflow. If set to FALSE then the parameter annotation_file needs to be supplied to the function.
annotation_file	A dataframe containing the EPIC methylation annotation file.
M	Number of methylation states to be identified in a single DNA sample.
N	Number of DNA samples (patients/replicates) collected for each treatment group.
parameter_estimation_only	If only model parameters are to be estimated then value is TRUE else FALSE.
seed	Seed to allow for reproducibility (default = NULL).

Value

An S4 object of class `threshold_Results`.

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)
library(betaHMM)

## read files
data(sample_methylation_file)
head(sample_methylation_file)
data(sample_annotation_file)
head(sample_annotation_file)
##merge data
df=merge(sample_annotation_file[,c('IlmnID','CHR','MAPINFO')],
sample_methylation_file,by='IlmnID')

## sort data
df=df[order(df$CHR,df$MAPINFO),]
thr_out=threshold_identification(df[,c(1,4:7)],package_workflow=TRUE,M=3,4,
parameter_estimation_only=TRUE,seed=12345)
```

`threshold_identification_run` *Threshold indentification function*

Description

HMM for beta valued DNA data for a single treatment condition

Usage

```
threshold_identification_run(
  data,
  package_workflow = TRUE,
  annotation_file = NULL,
  M,
  N,
  parameter_estimation_only = FALSE,
  seed = NULL,
  ...
)
```

Arguments

`data`	Methylation data and IlmnID. Maybe provided as a matrix or dataframe.
`package_workflow`	Flag set to TRUE if method called from package workflow. If set to FALSE then the parameter annotation_file needs to be supplied to the function.
`annotation_file`	A dataframe containing the EPIC methylation annotation file.
`M`	Number of methylation states to be identified in a single DNA sample.
`N`	Number of DNA samples (patients/replicates) collected for each treatment group.
`parameter_estimation_only`	If only model parameters are to be estimated then value is TRUE else FALSE.
`seed`	Seed to allow for reproducibility (default = NULL).
`...`	Extra parameters.

Value

An S4 object of class `threshold_Results`, where conditional probabilities of each CpG site belonging to a one of the M methylation states is stored as a SimpleList of assay data, and the corresponding estimated model parameters, the thresholds and most probable hidden state sequence for each chromosome are stored as metadata.

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)
library(betaHMM)

## read files
```

```
data(sample_methylation_file)
head(sample_methylation_file)
data(sample_annotation_file)
head(sample_annotation_file)
##merge data
df=merge(sample_annotation_file[,c('IlmnID','CHR','MAPINFO')],
sample_methylation_file,by='IlmnID')

## sort data
df=df[order(df$CHR,df$MAPINFO),]
thr_out=threshold_identification(df[,c(1,4:7)],package_workflow=TRUE,M=3,4,
parameter_estimation_only=TRUE,seed=12345)
```

`threshold_Results-class` *threshold_Results object and constructor*

Description

`threshold_Results` is a subclass of `RangedSummarizedExperiment`, used to store the threshold_identification results as well as the annotated data useful for plotting.

Usage

```
threshold_Results(SummarizedExperiment, annotatedData)
```

Arguments

`SummarizedExperiment`	a `RangedSummarizedExperiment` of `threshold_Results` object.
`annotatedData`	The annotated data passed as an input argument to the `threshold_identification` function.

Details

This constructor function would not typically be used by "end users". This simple class extends the `RangedSummarizedExperiment` class of the SummarizedExperiment package to allow other packages to write methods for results objects from the `threshold_identification` function. It is used by to wrap up the results table.

Value

a `threshold_Results` object

Examples

```
## Use simulated data for the betaHMM workflow example
set.seed(12345)
library(betaHMM)

## read files
data(sample_methylation_file)
```

```
head(sample_methylation_file)
data(sample_annotation_file)
head(sample_annotation_file)
##merge data
df=merge(sample_annotation_file[,c('IlmnID','CHR','MAPINFO')],
sample_methylation_file,by='IlmnID')

## sort data
df=df[order(df$CHR,df$MAPINFO),]
thr_out=threshold_identification(df[,c(1,4:7)],package_workflow=TRUE,M=3,4,
parameter_estimation_only=TRUE,seed=12345)
```

`Viterbi` *Estimating the hidden states using Viterbi algorithm*

Description

The Viterbi algorithm is used to estimate the most probable sequence of the hidden states utilizing the betaHMM model parameters estimated by the Baum-Welch algorithm.

Usage

```
Viterbi(data, M, N, R, tau, A, phi, K)
```

Arguments

`data`	A dataframe of dimension $C \times (N \times R)$ containing methylation values for C CpG sites from R treatment groups each having N replicates or each collected from N patients.
`M`	Number of methylation states to be identified in a single DNA sample.
`N`	Number of DNA samples (patients/replicates) collected for each treatment group.
`R`	Number of treatment groups (For. eg: Benign and Tumour).
`tau`	The initial distribution for the betaHMM model.
`A`	The transition matrix for the betaHMM model.
`phi`	The shape parameters for the observation sequence data in the betaHMM model.
`K`	The number of hidden states identified using the betaHMM model.

Value

A vector containing the most probable sequence of the hidden states of the betaHMM model.

Index