

Package ‘STAN’

October 18, 2017

Version 2.4.0

Date 2016-05-30

Title The genomic STate ANnotation package

Author Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

Maintainer Benedikt Zacher <zacher@genzentrum.lmu.de>

Imports GenomicRanges, IRanges, S4Vectors, BiocGenerics, GenomeInfoDb, Gviz, Rsolnp

Depends methods, poilog, parallel

VignetteBuilder knitr

Suggests BiocStyle, gplots, knitr

Description Genome segmentation with hidden Markov models has become a useful tool to annotate genomic elements, such as promoters and enhancers. STAN (genomic STate ANnotation) implements (bidirectional) hidden Markov models (HMMs) using a variety of different probability distributions, which can model a wide range of current genomic data (e.g. continuous, discrete, binary). STAN de novo learns and annotates the genome into a given number of 'genomic states'. The 'genomic states' may for instance reflect distinct genome-associated protein complexes (e.g. 'transcription states') or describe recurring patterns of chromatin features (referred to as 'chromatin states'). Unlike other tools, STAN also allows for the integration of strand-specific (e.g. RNA) and non-strand-specific data (e.g. ChIP).

License GPL (>= 2)

biocViews HiddenMarkovModel, GenomeAnnotation, Microarray, Sequencing, ChIPSeq, RNASeq, ChipOnChip, Transcription

LazyLoad yes

NeedsCompilation yes

R topics documented:

STAN-package	2
bdHMM	3
bdHMM-class	4
binarizeData	5
c2optimize	5
call_dpoilog	6
data2Gviz	6
DimNames	7

DirScore	7
Emission	8
EmissionParams	9
example	9
fitHMM	10
flags	11
getAvgSignal	11
getLogLik	12
getPosterior	13
getSizeFactors	14
getViterbi	14
HMM	15
HMM-class	16
HMMEmission	17
HMMEmission-class	17
initBdHMM	18
initHMM	19
InitProb	19
LogLik	20
observations	21
pilot.hg19	21
runningMean	21
StateNames	22
trainRegions	22
Transitions	23
ucscGenes	23
viterbi2GRanges	24
viterbi2Gviz	24
yeastTF_databychrom_ex	25
yeastTF_SGDGenes	25
[,bdHMM,ANY,ANY-method	26
[,HMM,ANY,ANY-method	26
Index	27

 STAN-package

The genomic STate ANnotation package

Description

The genomic STate ANnotation package

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

References

Zacher, B. and Lidschreiber, M. and Cramer, P. and Gagneur, J. and Tresch, A. (2014): Annotation of genomics data using bidirectional hidden Markov models unveils variations in Pol II transcription cycle Mol. Syst. Biol. 10:768

bdHMM

*Create a bdHMM object***Description**

This function creates a bdHMM function.

Usage

```
bdHMM(initProb = numeric(), transMat = matrix(numeric(), ncol = 0, nrow =
  0), emission, nStates = numeric(), status = character(),
  stateNames = character(), dimNames = character(),
  transitionsOptim = "analytical", directedObs = integer(),
  dirScore = numeric())
```

Arguments

initProb	Initial state probabilities.
transMat	Transition probabilities
emission	Emission parameters as an HMM Emission object.
nStates	Number of states.
status	Status of the bdHMM. 'Initial' means that the model was not fitted yet. 'EM' means that the model was optimized using Expectation maximization.
stateNames	Indicates directinality of states. States can be forward (F1, F2, ..., Fn), reverse (R1, R2, ..., Rn) or unidirectional (U1, U2, ..., Um). Number of F and R states must be equal and twin states are indicated by integers in id (e.g. F1 and R1 and twins).
dimNames	Names of data tracks.
transitionsOptim	There are three methods to choose from for fitting the transitions. Bidirectional transition matrices (invariant under reversal of time and direction) can be fitted using c('rsolnp', 'analytical'). 'None' uses standard update formulas and the resulting matrix is not constrained to be bidirectional.
directedObs	An integer indicating which dimensions are directed. Undirected dimensions are 0. Directed observations must be marked as unique integer pairs. For instance c(0,0,0,0,0,1,1,2,2,3,3) contains 5 undirected observations, and three pairs (one for each direction) of directed observations.
dirScore	Directionality score of states of a fitted bdHMM.

See Also

[HMM Emission](#)

Examples

```
nStates = 5
stateNames = c('F1', 'F2', 'R1', 'R2', 'U1')
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
```

```

transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
myEmission = list(d1=HMMEmission(type='Gaussian', parameters=list(mu=means, cov=Sigma), nStates=length(means)
bdhmm = bdHMM(initProb=initProb, transMat=transMat, emission=myEmission, nStates=nStates, status='initial',

```

bdHMM-class	<i>This class is a generic container for bidirectional Hidden Markov Models.</i>
-------------	--

Description

This class is a generic container for bidirectional Hidden Markov Models.

Slots

initProb Initial state probabilities.

transMat Transition probabilities

emission Emission parameters as an HMMEmission object.

nStates Number of states.

status of the HMM. On of c('initial', 'EM').

stateNames State names.

dimNames Names of data tracks.

LogLik Log likelihood of a fitted HMM.

transitionsOptim There are three methods to choose from for fitting the transitions. Bidirectional transition matrices (invariant under reversal of time and direction) can be fitted using c('rsolnp', 'ipopt'). 'None' uses standard update formulas and the resulting matrix is not constrained to be bidirectional.

directedObs An integer indicating which dimensions are directed. Undirected dimensions are 0. Directed observations must be marked as unique integer pairs. For instance c(0,0,0,0,0,1,1,2,2,3,3) contains 5 undirected observations, and three pairs (one for each direction) of directed observations.

dirScore Directionality score of states of a fitted bdHMM.

Methods

[get elements from the bdHMM

See Also

[HMMEmission](#)

Examples

```

nStates = 5
stateNames = c('F1', 'F2', 'R1', 'R2', 'U1')
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
myEmission = list(d1=HMMemission(type='Gaussian', parameters=list(mu=means, cov=Sigma), nStates=length(means)

bdhmm = bdHMM(initProb=initProb, transMat=transMat, emission=myEmission, nStates=nStates, status='initial',

```

binarizeData

Binarize Sequencing data with the default ChromHMM binarization

Description

Binarize Sequencing data with the default ChromHMM binarization

Usage

```
binarizeData(obs)
```

Arguments

obs The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).

Value

Binarized observation sequences as a list.

Examples

```

data(trainRegions)
binData = binarizeData(trainRegions)

```

c2optimize

Optimize transitions

Description

The function is called from C++ to optimize transitions.

Usage

```
c2optimize(pars)
```

Arguments

pars Parameters for optimization.

Value

optimized transitions

call_dpoiilog *Calculate density of the Poisson-Log-Normal distribution.*

Description

Calculate density of the Poisson-Log-Normal distribution.

Usage

call_dpoiilog(x)

Arguments

x A vector c(n, mu, sigma), where n is the number of observed counts, mu the mean of the Log-Normal distribution and sigma its variance.

Value

Density of the Poisson-Log-Normal distribution.

Examples

call_dpoiilog(c(5, 2, 1))

data2Gviz *Convert data for plotting with Gviz*

Description

Convert data for plotting with Gviz

Usage

data2Gviz(obs, regions, binSize, gen, col = "black")

Arguments

obs The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).

regions GRanges object of the regions (e.g. chromosomes) stored in the viterbi path.

binSize The bin size of the viterbi path.

gen The genome id, e.g. hg19, hg38 for human.

col The color of the data tracks.

Value

A list containing the data tracks converted to Gviz objects for plotting.

DimNames	<i>Get dimNames of a (bd)HMM</i>
----------	----------------------------------

Description

This function returns the names of dimensions (data tracks).

Usage

```
DimNames(hmm)
```

Arguments

hmm An object of class HMM or bdHMM.

Value

A character vector

Examples

```
nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
hmm = HMM(dimNames="1", initProb=initProb, transMat=transMat, emission=HMMEmission(type='Gaussian', parameter=
DimNames(hmm)
```

DirScore	<i>Get directionality score of a bdHMM</i>
----------	--

Description

This function returns the directionality score of a bdHMM.

Usage

```
DirScore(bdhmm)
```

Arguments

bdhmm An object of class bdHMM.

Value

Directionality score of the bdHMM after model fitting.

Examples

```

data(example)
bdhmm_ex = initBdHMM(observations, nStates=3, method="Gaussian", directedObs=0)

# without flags
bdhmm_fitted_noFlags = fitHMM(observations, bdhmm_ex)
DirScore(bdhmm_fitted_noFlags)

# with flags
bdhmm_fitted_flags = fitHMM(observations, bdhmm_ex, dirFlags=flags)
DirScore(bdhmm_fitted_flags)

```

Emission

Get Emission functions of a (bd)HMM

Description

This function returns the Emission functions of a (bd)HMM.

Usage

```
Emission(hmm)
```

Arguments

hmm An object of class HMM or bdHMM.

Value

An object of class HMM Emission

See Also

[HMM Emission](#)

Examples

```

nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
hmm = HMM(initProb=initProb, transMat=transMat, emission=HMM Emission(type='Gaussian', parameters=list(mu=mea
Emission(hmm)

```

EmissionParams	<i>Get Emission parameters of a (bd)HMM.</i>
----------------	--

Description

This function returns the parameters of emission functions of a (bd)HMM object.

Usage

```
EmissionParams(hmm)
```

Arguments

hmm An object of class (bd)HMM.

Value

A list containing the parameters of the Emission functions.

See Also

[HMMEmission](#), [HMM](#), [bdHMM](#)

Examples

```
nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
hmm = HMM(initProb=initProb, transMat=transMat, emission=HMMEmission(type='Gaussian', parameters=list(mu=mea
EmissionParams(hmm)
```

example	<i>The data for the bdHMM example in the vignette and examples in the manual</i>
---------	--

Description

The data for the bdHMM example in the vignette and examples in the manual

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

fitHMM

*Fit a Hidden Markov Model***Description**

The function is used to fit (bidirectional) Hidden Markov Models, given one or more observation sequence.

Usage

```
fitHMM(obs=list(), hmm, convergence=1e-6, maxIters=1000, dirFlags=list(), emissionProbs=list(), e
```

Arguments

obs	The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).
hmm	The initial Hidden Markov Model. This is a HMM .
convergence	Convergence cutoff for EM-algorithm (default: 1e-6).
maxIters	Maximum number of iterations.
dirFlags	The flag sequence is needed when a bdHMM is fitted on undirected data (e.g.) ChIP only. It is a list of character vectors indication for each position its known directionality. U allows all states. F allows undirected states and states in forward direction. R allows undirected states and states in reverse direction.
emissionProbs	List of precalculated emission probabilities of emission function is of type 'null'.
effectiveZero	Transitions below this cutoff are analytically set to 0 to speed up computations.
verbose	logical for printing algorithm status or not.
nCores	Number of cores to use for computations.
incrementalEM	When TRUE, the incremental EM is used to fit the model, where parameters are updated after each iteration over a single observation sequence.
updateTransMat	Whether transitions should be updated during model learning, default: TRUE.
sizeFactors	Library size factors for Emissions PoissonLogNormal or NegativeBinomial as a length(obs) x ncol(obs[[1]]) matrix.

Value

A list containing the trace of the log-likelihood during EM learning and the fitted HMM model.

See Also

[HMM](#)

Examples

```
data(example)
hmm_ex = inithmm(observations, nStates=3, method="Gaussian")
hmm_fitted = fitHMM(observations, hmm_ex)
```

flags	<i>Pre-computed flag sequence for the 'example' data.</i>
-------	---

Description

Pre-computed flag sequence for the 'example' data.

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

getAvgSignal	<i>Compute average signal in state segmentation</i>
--------------	---

Description

Compute average signal in state segmentation

Usage

```
getAvgSignal(viterbi, obs, fct=mean)
```

Arguments

viterbi	A list containing the viterbi paths as factors. The output from getViterbi.
obs	The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).
fct	The averaging function, default: mean.

Value

A state x data track matrix containing the average signal.

Examples

```
data(yeastTF_databychrom_ex)
nStates = 6
dirobs = as.integer(c(rep(0,10), 1, 1))
bdhmm_gauss = initBdHMM(yeastTF_databychrom_ex, nStates, "Gaussian", directedObs=dirobs)
bdhmm_fitted_gauss = fitHMM(yeastTF_databychrom_ex, bdhmm_gauss)
viterbi_bdhmm_gauss = getViterbi(bdhmm_fitted_gauss, yeastTF_databychrom_ex)
avg_signal = getAvgSignal(viterbi_bdhmm_gauss, yeastTF_databychrom_ex)
```

`getLogLik`*Calculate log likelihood state distribution.*

Description

The function calculates log likelihood for one or more observation sequence.

Usage

```
getLogLik(hmm, obs = list(), emissionProbs = list(), dirFlags = list(), verbose = FALSE, nCores = 1)
```

Arguments

<code>hmm</code>	The Hidden Markov Model.
<code>obs</code>	The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).
<code>emissionProbs</code>	List of precalculated emission probabilities of emission function is of type 'null'.
<code>dirFlags</code>	The flag sequence is needed when a bdHMM is fitted on undirected data (e.g.) ChIP only. It is a list of character vectors indication for each position its known directionality. U allows all states. F allows undirected states and states in forward direction. R allows undirected states and states in reverse direction.
<code>verbose</code>	logical for printing algorithm status or not.
<code>nCores</code>	Number of cores to use for computations.
<code>sizeFactors</code>	Library size factors for Emissions PoissonLogNormal or NegativeBinomial as a <code>length(obs) x ncol(obs[[1]])</code> matrix.

Value

The log likelihood of the observations sequences, given the model.

See Also

[HMM](#)

Examples

```
data(example)
hmm_ex = inithMM(observations, nStates=3, method="Gaussian")
hmm_fitted = fithMM(observations, hmm_ex)
loglik = getLogLik(hmm_fitted, observations)
loglik
```

getPosterior	<i>Calculate posterior state distribution.</i>
--------------	--

Description

The function calculates posterior state probabilities for one or more observation sequence.

Usage

```
getPosterior(hmm, obs=list(), emissionProbs=list(), dirFlags=list(), verbose=FALSE, nCores=1, siz
```

Arguments

hmm	The Hidden Markov Model.
obs	The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).
emissionProbs	List of precalculated emission probabilities of emission function is of type 'null'.
dirFlags	The flag sequence is needed when a bdHMM is fitted on undirected data (e.g.) ChIP only. It is a list of character vectors indication for each position its known directionality. U allows all states. F allows undirected states and states in forward direction. R allows undirected states and states in reverse direction.
verbose	logical for printing algorithm status or not.
nCores	Number of cores to use for computations.
sizeFactors	Library size factors for Emissions PoissonLogNormal or NegativeBinomial as a length(obs) x ncol(obs[[1]]) matrix.

Value

A list containing for the observation sequences the posterior state (col) distribution at each position (row).

Examples

```
data(example)
hmm_ex = initWithM(observations, nStates=3, method="Gaussian")
hmm_fitted = fitHMM(observations, hmm_ex)
posterior = getPosterior(hmm_fitted, observations)
```

getSizeFactors	<i>Compute size factors</i>
----------------	-----------------------------

Description

Compute size factors

Usage

```
getSizeFactors(obs, celltypes)
```

Arguments

obs	The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).
celltypes	Indicates the cell type/tissue for each entry in obs.

Value

A celltype/tissue x data tracks matrix containing the size factors.

Examples

```
data(trainRegions)
celltypes = list("E123"=grep("E123", names(trainRegions)),
                "E116"=grep("E116", names(trainRegions)))
sizeFactors = getSizeFactors(trainRegions, celltypes)
sizeFactors
```

getViterbi	<i>Calculate the most likely state path</i>
------------	---

Description

Given a Hidden Markov Model, the function calculates the most likely state path (viterbi) for one or more observation sequence.

Usage

```
getViterbi(hmm, obs=list(), NAtol=5, emissionProbs=list(), verbose=FALSE, sizeFactors=matrix(1, n
```

Arguments

hmm	The initial Hidden Markov Model.
obs	The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).
NAto1	Successive positions having NAs longer than this threshold are masked in the viterbi path.
emissionProbs	List of precalculated emission probabilities of emission function is of type 'null'.
verbose	logical for printing algorithm status or not.
sizeFactors	Library size factors for Emissions PoissonLogNormal or NegativeBinomial as a length(obs) x ncol(obs[[1]]) matrix.

Value

A list containint the vterbi paths.

Examples

```
data(example)
hmm_ex = initWithM(observations, nStates=3, method="Gaussian")
hmm_fitted = fitHMM(observations, hmm_ex)
viterbi = getViterbi(hmm_fitted, observations)
```

HMM

Create a HMM object

Description

This function creates a HMM object.

Usage

```
HMM(initProb = numeric(), transMat = matrix(numeric(), ncol = 1, nrow = 1),
     emission, nStates = numeric(), status = character(),
     stateNames = character(), dimNames = character(), LogLik = numeric())
```

Arguments

initProb	Initial state probabilities.
transMat	Transition probabilities
emission	Emission parameters as an HMM Emission object.
nStates	Number of states.
status	of the HMM. On of c('initial', 'EM').
stateNames	State names.
dimNames	Names of data tracks.
LogLik	Log likelihood of a fitted HMM.

See Also[HMMEmission](#)**Examples**

```

nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
HMM(initProb=initProb, transMat=transMat, emission=HMMEmission(type='Gaussian', parameters=list(mu=means, c

```

HMM-class

*This class is a generic container for Hidden Markov Models.***Description**

This class is a generic container for Hidden Markov Models.

Slots

initProb Initial state probabilities.
transMat Transition probabilities
emission Emission parameters as an HMMEmission object.
nStates Number of states.
status of the HMM. On of c('initial', 'EM').
stateNames State names.
dimNames Names of data tracks.
LogLik Log likelihood of a fitted HMM.

Methods

[get elements from the HMM

See Also[HMMEmission](#)**Examples**

```

nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)

HMM(initProb=initProb, transMat=transMat, emission=HMMEmission(type='Gaussian', parameters=list(mu=means, c

```


HMM Emission

*Create a HMM Emission object***Description**

This function creates a HMM Emission object.

Usage

```
HMM Emission(type = character(), parameters = list(), nStates = numeric())
```

Arguments

type	The type of emission function c('Gaussian').
parameters	A list containing the the parameters for each state.
nStates	The number of states.

Examples

```
nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
HMM Emission(type='Gaussian', parameters=list(mu=means, cov=Sigma), nStates=length(means))
```

HMM Emission-class

*This class is a generic container for different emission functions of Hidden Markov Models.***Description**

This class is a generic container for different emission functions of Hidden Markov Models.

Slots

type	The type of emission function c('Gaussian').
parameters	A list containing the the parameters for each state.
dim	Number of dimensions.
nStates	The number of states.

Examples

```
nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
HMM Emission(type='Gaussian', parameters=list(mu=means, cov=Sigma), nStates=length(means))
```

initBdHMM	<i>Initialization of bidirectional hidden Markov models</i>
-----------	---

Description

Initialization of bidirectional hidden Markov models

Usage

```
initBdHMM(obs, nStates, method, directedObs = rep(0, ncol(obs[[1]])), sizeFactors = matrix(1, nrow =
```

Arguments

obs	The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).
nStates	The number of states.
method	Emission distribution of the model. One out of c("NegativeBinomial", "Poisson-LogNormal", "NegativeMultinomial", "ZINegativeBinomial", "Poisson", "Bernoulli", "Gaussian", "IndependentGaussian")
directedObs	Integer vector defining the directionality (or strand-specificity) of the data tracks. Undirected (non-strand-specific) data tracks (e.g. ChIP) are indicated by '0'. Directed (strand-specific) data tracks are indicated by increasing pairs of integers. For instance c(0,0,0,1,1,2,2): The first three data tracks are undirected, followed by two pairs of directed measurements.
sizeFactors	Library size factors for Emissions PoissonLogNormal or NegativeBinomial as a length(obs) x ncol(obs[[1]]) matrix.
sharedCov	If TRUE, (co-)variance of (Independent)Gaussian is shared over states. Only applicable to 'Gaussian' or 'IndependentGaussian' emissions. Default: FALSE.

Value

A HMM object.

Examples

```
data(example)
hmm_ex = initHMM(observations, nStates=3, method="Gaussian")
```

initHMM	<i>Initialization of hidden Markov models</i>
---------	---

Description

Initialization of hidden Markov models

Usage

```
initHMM(obs, nStates, method, sizeFactors = matrix(1, nrow = length(obs), ncol = ncol(obs[[1]])),
```

Arguments

obs	The observations. A list of one or more entries containing the observation matrix (numeric) for the samples (e.g. chromosomes).
nStates	The number of states.
method	Emission distribution of the model. One out of c("NegativeBinomial", "Poisson-LogNormal", "NegativeMultinomial", "ZINegativeBinomial", "Poisson", "Bernoulli", "Gaussian", "IndependentGaussian")
sizeFactors	Library size factors for Emissions PoissonLogNormal or NegativeBinomial as a length(obs) x ncol(obs[[1]]) matrix.
sharedCov	If TRUE, (co-)variance of (Independent)Gaussian is shared over states. Only applicable to 'Gaussian' or 'IndependentGaussian' emissions. Default: FALSE.

Value

A HMM object.

Examples

```
data(example)
hmm_ex = initHMM(observations, nStates=3, method="Gaussian")
```

InitProb	<i>Get initial state probabilities of a (bd)HMM</i>
----------	---

Description

This function returns the initial state probabilities of a (bd)HMM.

Usage

```
InitProb(hmm)
```

Arguments

hmm	An object of class HMM or bdHMM.
-----	----------------------------------

Value

The initial state probabilities as a numeric vector.

See Also

[HMM](#), [bdHMM](#)

Examples

```
nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
hmm = HMM(initProb=initProb, transMat=transMat, emission=HMMEmission(type='Gaussian', parameters=list(mu=mea
InitProb(hmm)
```

LogLik

Get stateNames of a (bd)HMM

Description

This function returns the Log-Likelihood of a (bd)HMM.

Usage

```
LogLik(hmm)
```

Arguments

`hmm` An object of class HMM or bdHMM.

Value

Log likelihood during model fitting.

Examples

```
data(example)
hmm_ex = initHMM(observations, nStates=3, method="Gaussian")
hmm_fitted = fitHMM(observations, hmm_ex)
LogLik(hmm_fitted)
```

observations	<i>Observation sequence for the 'example' data.</i>
--------------	---

Description

Observation sequence for the 'example' data.

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

pilot.hg19	<i>Genomic positions of processed signal for the Roadmap Epigenomics data set. Regions from the ENCODE pilot phase.</i>
------------	---

Description

Genomic positions of processed signal for the Roadmap Epigenomics data set. Regions from the ENCODE pilot phase.

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

runningMean	<i>Smooth data with running mean</i>
-------------	--------------------------------------

Description

Smooth data with running mean

Usage

```
runningMean(x, winHalfSize = 2)
```

Arguments

x	A vector with the data.
winHalfSize	The smoothing window half size.

Value

A vector containing the smoothed data.

Examples

```
data(trainRegions)
celltypes = list("E123"=grep("E123", names(trainRegions)),
                "E116"=grep("E116", names(trainRegions)))
sizeFactors = getSizeFactors(trainRegions, celltypes)
sizeFactors
```

StateNames	<i>Get stateNames of a (bd)HMM</i>
------------	------------------------------------

Description

This function returns the names of states.

Usage

```
StateNames(hmm)
```

Arguments

hmm An object of class HMM or bdHMM.

Value

A character vector

Examples

```
nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
hmm = HMM(stateNames=as.character(1:5), initProb=initProb, transMat=transMat, emission=HMMemission(type='Gauss'))
StateNames(hmm)
```

trainRegions	<i>Training regions for the Roadmap Epigenomics data set. Three ENCODE pilot regions with data from two cell lines.</i>
--------------	---

Description

Training regions for the Roadmap Epigenomics data set. Three ENCODE pilot regions with data from two cell lines.

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

Transitions	<i>Get transitions of a (bd)HMM</i>
-------------	-------------------------------------

Description

This function returns the transition matrix of a (bd)HMM.

Usage

```
Transitions(hmm)
```

Arguments

hmm An object of class HMM or bdHMM.

Value

The transitions as a nStates x nStates matrix.

See Also

[HMM](#), [bdHMM](#)

Examples

```
nStates = 5
means = list(4,11,4,11,-1)
Sigma = lapply(list(4,4,4,4,4), as.matrix)
transMat = matrix(1/nStates, nrow=nStates, ncol=nStates)
initProb = rep(1/nStates, nStates)
hmm = HMM(initProb=initProb, transMat=transMat, emission=HMMEmission(type='Gaussian', parameters=list(mu=mea
Transitions(hmm)
```

ucscGenes	<i>UCSC gene annotation for the Roadmap Epigenomics data set.</i>
-----------	---

Description

UCSC gene annotation for the Roadmap Epigenomics data set.

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

viterbi2GRanges	<i>Convert the viterbi path to a GRanges object</i>
-----------------	---

Description

Convert the viterbi path to a GRanges object

Usage

```
viterbi2GRanges(viterbi, regions, binSize)
```

Arguments

viterbi	A list containing the viterbi paths as factors. The output from getViterbi.
regions	GRanges object of the regions (e.g. chromosomes) stored in the viterbi path.
binSize	The bin size of the viterbi path.

Value

The viterbi path as GRanges object.

Examples

```
library(GenomicRanges)
data(yeastTF_databychrom_ex)
nStates = 6
dirobs = as.integer(c(rep(0,10), 1, 1))
bdhmm_gauss = initBdHMM(yeastTF_databychrom_ex, nStates, "Gaussian", directedObs=dirobs)
bdhmm_fitted_gauss = fitHMM(yeastTF_databychrom_ex, bdhmm_gauss)
viterbi_bdhmm_gauss = getViterbi(bdhmm_fitted_gauss, yeastTF_databychrom_ex)
yeastGRanges = GRanges(IRanges(start=1214616, end=1225008), seqnames="chrIV")
names(viterbi_bdhmm_gauss) = "chrIV"
viterbi_bdhmm_gauss_gr = viterbi2GRanges(viterbi_bdhmm_gauss, yeastGRanges, 8)
```

viterbi2Gviz	<i>Convert state segmentation for plotting with Gviz</i>
--------------	--

Description

Convert state segmentation for plotting with Gviz

Usage

```
viterbi2Gviz(viterbi, chrom, gen, from, to, statecols)
```


Arguments

viterbi	A list containing the viterbi paths as factors. The output from getViterbi.
chrom	The chromosome/sequence if to convert.
gen	The genome id, e.g. hg19, hg38 for human.
from	Genomic start position.
to	Genomic end position.
statecols	Named vector with state colors.

Value

A list containing the viterbi path converted to Gviz objects for plotting.

yeastTF_databychrom_ex

Processed ChIP-on-chip data for yeast TF example

Description

Processed ChIP-on-chip data for yeast TF example

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

yeastTF_SGDGenes

SGD annotation for the yeast TF example

Description

SGD annotation for the yeast TF example

Author(s)

Benedikt Zacher, Julia Ertl, Julien Gagneur, Achim Tresch

[,bdHMM,ANY,ANY-method

This function subsets a bdHMM object. Rows are interpreted as states, columns as dimensions of emissions.

Description

This function subsets a bdHMM object. Rows are interpreted as states, columns as dimensions of emissions.

Usage

```
## S4 method for signature 'bdHMM,ANY,ANY'
x[i, j, ..., drop = "missing"]
```

Arguments

x	A bidirectional hidden Markov model.
i	State ids to extract.
j	Emissions to extract.
...	...
drop	...

[,HMM,ANY,ANY-method

This function subsets an HMM object. Rows are interpreted as states, columns as dimensions of emissions.

Description

This function subsets an HMM object. Rows are interpreted as states, columns as dimensions of emissions.

Usage

```
## S4 method for signature 'HMM,ANY,ANY'
x[i, j, ..., drop = "missing"]
```

Arguments

x	A hidden Markov model.
i	State ids to extract.
j	Emissions to extract.
...	...
drop	...

Index

*Topic **data**

- example, 9
- flags, 11
- observations, 21
- pilot.hg19, 21
- trainRegions, 22
- ucscGenes, 23
- yeastTF_databychrom_ex, 25
- yeastTF_SGDGenes, 25

*Topic **package**

- STAN-package, 2
- .HMM (HMM-class), 16
- .HMMemission (HMMemission-class), 17
- .bdHMM (bdHMM-class), 4
- [, HMM, ANY, ANY, ANY-method (HMM-class), 16
- [, HMM, ANY, ANY-method, 26
- [, bdHMM, ANY, ANY, ANY-method (bdHMM-class), 4
- [, bdHMM, ANY, ANY-method, 26

- bdHMM, 3, 9, 20, 23
- bdHMM-class, 4
- binarizeData, 5

- c2optimize, 5
- call_dpoilog, 6

- data2Gviz, 6
- DimNames, 7
- DirScore, 7

- Emission, 8
- EmissionParams, 9
- example, 9

- fitHMM, 10
- flags, 11

- getAvgSignal, 11
- getLogLik, 12
- getPosterior, 13
- getSizeFactors, 14
- getViterbi, 14

- HMM, 9, 10, 12, 15, 20, 23

- HMM-class, 16
- HMMemission, 3, 4, 8, 9, 16, 17
- HMMemission-class, 17

- initBdHMM, 18
- initHMM, 19
- InitProb, 19

- LogLik, 20

- observations, 21

- pilot.hg19, 21

- runningMean, 21

- STAN-package, 2
- StateNames, 22

- trainRegions, 22
- Transitions, 23

- ucscGenes, 23

- viterbi2GRanges, 24
- viterbi2Gviz, 24

- yeastTF_databychrom_ex, 25
- yeastTF_SGDGenes, 25