# Package 'VanillaICE'

October 9, 2013

**Version** 1.22.0

**Title** A Hidden Markov Model for high throughput genotyping arrays

**Author**
Robert Scharpf <rscharpf@jhsph.edu>, Kevin Scharpf, and Ingo Ruczinski <ingo@jhsph.edu>

**Maintainer** Robert Scharpf <rscharpf@jhsph.edu>

**Depends** R (>= 2.14.0)

**Imports** stats, utils, methods, Biobase, oligoClasses (>= 1.21.12),lattice, IRanges (>= 1.13.22), grid, msm, iterators,foreach, GenomicRanges, matrixStats

**Suggests** genomewidesnp6Crlmm (>= 1.0.7), hapmapsnp6, RColorBrewer,genefilter, RSQLite, foreach, RUnit, pd.mapping50k.hind240,SNPchip (>= 2.5.7), doSNOW

**Enhances** DNAcopy, crlmm (>= 1.17.14)

**Description** Hidden Markov Models for characterizing chromosomal alterations in high throughput SNP arrays

**License** LGPL-2

**LazyLoad** yes

**Collate** AllGenerics.R AllClasses.R methods-AssayData.R
methods-CopyNumberSet.R methods-BeadStudioSet.R
methods-GRanges.R methods-SnpSet.R methods-Vit.R methods-Viterbi.R methods-Viterbi2.R
methods-BeadStudioSetList.R methods-oligoSetList.R
hmm-methods.R deprecated-functions.R genotype-functions.R
hmm-functions.R simulation-functions.R viterbi-functions.R
functions.R generator-functions.R utils.R zzz.R

**biocViews** Bioinformatics, CopyNumberVariants, SNP, GeneticVariability,Visualization

# R topics documented:



---

`BeadStudioSet` *Constructor for BeadStudioSet class*

---

## Description

Constructs an instance of BeadStudioSet from a list of files containing log R ratios and B allele frequencies.

## Usage

```
BeadStudioSet(filenames, lrr.colname = "Log.R.Ratio", baf.colname = "B.Allele", sep = "\t", header = TR
```

## Arguments

| | |
|---|---|
| `filenames` | character string providing the names of the BeadStudio files, including the complete path if not in the working directory. |
| `lrr.colname` | character string providing the column header for log R ratios |
| `baf.colname` | character string providing the column header for log R ratios |
| `sep` | field delimiter in the BeadStudio files. See `read.table` |
| `header` | logical: whether the files contain a header. |
| `colClasses` | See `read.table`. |
| `genome` | Character string indicating which genome build to use. Supported entries are "hg19" and "hg18". |
| `annotationPkg` | character string providing the name of the annotation package. |

| | |
|---|---|
| chromosome | integer vector indicating which chromosomes to include in the BeadStudioSet. E.g., 1:23 for autosomes and chromosome X |
| ... | Additional arguments to read.bsfiles. |

## Value

An object of class BeadStudioSet

## Author(s)

R. Scharpf

## See Also

read.bsfiles, BeadStudioSet

## Examples

```
path <- system.file("extdata", package="VanillaICE")
fname <- file.path(path, "LRRandBAF.txt")
bsSet <- BeadStudioSet(fname, annotationPkg="genomewidesnp6Crlmm", genome="hg19")
```

---

# BeadStudioSetList *Constructor for BeadStudioSetList class.*

---

## Description

Reads processed files containing log R Ratios and B allele frequencies and construct a BeadStudioList object.

## Usage

```
BeadStudioSetList(fnames, annotationPkg, genome = c("hg19", "hg18"), outdir = ldPath(), sampleIds, pher
```

## Arguments

| | |
|---|---|
| fnames | character vector containing the complete path to files containing log R ratios and BAFs. |
| annotationPkg | character string indicating the name of the annotation package. |
| genome | character string indicating which genome build. Supported entries are UCSC builds "hg19" and "hg18". |
| outdir | character string indicating where to store ff files for storing the log R ratios and B allele frequencies. Ignored if the ff package is not loaded. |
| sampleIds | character vector of sample identifiers. If missing, basename(fnames) is used. |
| phenoData | An AnnotatedDataFrame containing covariates for the samples. |

| | |
|---|---|
| byArm | Logical. Whether each element in the list should be a chromosome arm. If TRUE, centromere must be provided. (experimental) |
| centromere | data.frame containing start and stop coordinates of centromeres. |
| ... | Additional arguments passed to the initialization method for BeadStudioSetList. |

## Value

A BeadStudioSetList.

## Author(s)

R. Scharpf

## See Also

BeadStudioSet, BeadStudioSetList

## Examples

```
new("BeadStudioSetList")
```

---

copyNumberLimits *Constraints for updating the means for the copy number states in the hidden markov model.*

---

## Description

Constraints for updating the means for the copy number states in the hidden markov model.

## Usage

```
copyNumberLimits(is.log)
```

## Arguments

| | |
|---|---|
| is.log | logical: whether the copy number estimates are on the log scale |

## Details

Not indented to be called directly – used by packages that depend on VanillaICE.

## Value

A numeric vector of length 2 giving the lower and upper bounds for the copy number estimates.

## Author(s)

R. Scharpf

hmm-functions *HMM functions for oligoSnpSet and BeadStudioSet containers*

## Description

HMM functions for oligoSnpSet, BafLrrSet, SnpSet2 and BafLrrSetList containers. These functions are exported in the package's namespace to provide documentation of arguments that can be passed from the hmm method for these containers. The hmmBeadStudioSet2 function is always called when the object passed to the hmm method is a BeadStudioSet or BafLrrSet. The hmm method for oligoSnpSet objects will also call the hmmBafLrrSet2 function if if B allele frequencies (assay data element "baf") is included in the list of assay data elements. Otherwise, the hmm method for oligoSnpSet will call the hmmOligoSnpSet2 function.

## Usage

```
hmmBafLrrSet2(object, sampleIds, TAUP=1e10, tauMAX,cnStates=c(-2, -0.4, 0, 0, 0.4, 1), is.log=TRUE,...
hmmOligoSnpSet2(object, sampleIds, TAUP=1e10, tauMAX, cnStates = c(0,1,2,2,3,4), is.log=FALSE, ...)
hmmBafLrrSetList2(object, sampleIds, TAUP=1e10, tauMAX, cnStates=c(-2, -0.4, 0, 0, 0.4, 1), is.log=TRUE
```

## Arguments

| | |
|---|---|
| object | Any of the containers listed above. |
| sampleIds | A character vector indicating which samples in object to process. For large data, specifying sampleIds is much more efficient that subsetting the object prior to fitting the HMM. |
| TAUP | Scalar for the probability that the state of segment t is the same as the state at segment t-1. Larger values decrease the probability of transitioning to an different state. |
| tauMAX | Upper bound for the probability that the state at marker t is the same as the state at marker t-1. |
| cnStates | A vector of starting values (numeric) specifying the means of the Normal distribution assumed for latent copy numbers. The means must be specified for states homozygous deletion (zero copies), hemizygous deletion (1 copy), normal (2 copies), normal and no heterozygotes (2 copies), single copy duplication (3 copies), and two+ copy duplication (4+ copies). The starting values are updated via EM. |
| is.log | A logical indicating whether the copy number estimates are on the log scale. Note that the assay data elements in oligoSnpSet and BeadStudioSet should be represented as integers (copy number or relative copy number * 100). If is.log is TRUE, we assume that after division by 100 the assay data element containing the copy numbers (or relative copy numbers) is on the log-scale. The scale has implications on what is considered to be extreme. |
| ... | Additional arguments can be passed to viterbi2Wrapper. |

## Value

A GRanges or GRangesList object of the segmented data.

## Author(s)

R. Scharpf

## See Also

hmm-methods

---

hmm-methods *Hidden Markov Model methods*

---

## Description

Hidden Markov Model methods in package **VanillaICE**

## Methods

The hmm method is defined for several classes of containers of preprocessed and normalized SNP array data. The most common containers for use with genotyping platforms are the BeadStudioSet and oligoSnpSet classes. The primary difference between these two containers are the requirements for the assay data elements. A BeadStudioSet or BafLrrSet object (and the corresponding list classes) must have assay data elements "lrr" (log R ratios) and "baf" (B allele frequencies). As of version 1.18.0, all matrices stored in assay data are assumed to be integers. For copy number and relative copy number, the estimates should be scaled by 100. For B allele frequencies, the estimates should be scaled by 1000. The helper function integerMatrix in the oligoClasses package can be useful for the conversion. Genotype calls are optional for the BeadStudioSet and BafLrrSet objects. The BafListSet container can be generated as part of a pipeline to process data from either the Illumina or Affymetrix platforms. The oligoSnpSet object has required assay data elements "call" (genotype calls), "callProbability" (genotype confidence scores), "copyNumber", and "cnConfidence". As B allele frequencies are perhaps more informative than the genotype calls for distinguishing copy number states (particularly amplifications), an assay data element named "baf" can be included in the assay data for an oligoSnpSet object. The presence of a "baf" element in the assay data of an oligoSnpSet has implications on the particular HMM fit to identify the CNV boundaries (as discussed below).

A hidden Markov model for the BeadStudioSet class. The assay data are log R ratios and B allele frequencies. See hmmBeadStudioSet for additional arguments that can be passed through the ... operator.

signature(object = "BeadStudioSet", ...) signature(object = "SnpSet2", ...) A hidden Markov model for the SnpSet class. The assay data are diallelic genotype calls represented as integers (1=AA, 2=AB, 3=BB). See hmmSnpSet for additional arguments that can be passed through the ... operator.

`signature(object = "CNSet", ...)` A hidden Markov model for the `CNSet` class. The `CNSet` instance is first coerced to an object of class `oligoSnpSet` containing estimates of total copy number and B allele frequencies. See `hmmBeadStudioSet` for additional arguments that can be passed through the `...` operator. For large data sets, the initial coercion to the `oligoSnpSet` class can be very expensive in terms of I/O and require a large amount of RAM. Users with large data sets may prefer to coerce selected samples (e.g., the set of samples belonging to a given batch) to an `oligoSnpSet` object, and then fit the hmm on the `oligoSnpSet` object directly. This approach is illustrated in the `crlmmDownstream` vignette.

`signature(object = "CopyNumberSet", ...)` A hidden Markov model for the `CopyNumberSet` class. The assay data are estimates of total copy number. This method should not be used for arrays with genotype information as the genotypes / B allele frequencies are informative for copy number inference.

`signature(object = "oligoSnpSet", ...)` A hidden Markov model for the `oligoSnpSet` class. If "baf" is included among the assay data elements, the `hmmBeadStudioSet` HMM is implemented. Otherwise, the `hmmOligoSnpSet` is implemented.

`signature(object = "oligoSetList", ...)` The oligoSetList class is a container for genotypes, B allele frequencies (optional), and copy number organized by chromosome. Each element in the list class contains low-level summaries and phenotypic information for a single chromosome. The organization by chromosome facilitates parallelization of methods to identify copy number alterations. If B allele frequencies are included, the hmm fit to instance of this object is the same as the hmm fit to instances of a `BeadStudioSetList` object (the function `hmmBeadStudioSet` is fit to each element in the `oligoSetList` object).

`signature(object = "BeadStudioSetList", ...)` The only difference with `oligoSetList` is that the assayData for `BeadStudioSetList` objects must include B allele frequencies (B allele frequencies are optional in the `oligoSetList` class). The function `hmmBeadStudioSet` is fit to each element in the `BeadStudioSetList` object.

## See Also

`oligoSetList`, `BeadStudioSetList`, `BafLrrSetList` `hmmBafLrrSet2`, `hmmOligoSnpSet2`, `hmmSnpSet2` `hmmBafLrrSetList2`. For plotting copy number and B allele frequencies, see `xyplotLrrBaf`, `xypanelBaf`.

## Examples

```
library(oligoClasses)
library(IRanges)
data(oligoSetExample, package="oligoClasses")
oligoSet <- oligoSet[chromosome(oligoSet) == 1, ]
hmmResults <- hmm(oligoSet)
state(hmmResults[[1]])
##
## Plotting ranges:
##
if(require(SNPchip) && require(IRanges)){
## Plot the data for the second range with a blue
## border, and frame the region by 10 Mb on each side
## of the state boundary.
##
```

```
res <- hmmResults[[1]]
xyplot2(cn~x, oligoSet, range=res[2, ], frame=10e6,
       panel=xypanel, pch=21, cex=0.3,
       col.hom="royalblue", fill.hom="royalblue",
       col.het="red", fill.het="red", xlab="Mb",
       ylab=expression(log[2]("copy number")))
##  (Note that the formula cn~x is required at this time)
##
## Or, plot each range in its own panel with a frame
##  of 2e6 bases.  (Again, the formula is a standard format
##  with cn, x, range, and id the only allowed terms) Because
##  these are all the ranges from one individual's chromosome,
##  the ranges are overlapping The range 'in focus' is
##  demarcated by vertical blue lines
xyplot2(cn~x | range, oligoSet, range=res, frame=2e6,
       panel=xypanel,
       pch=21,
       cex=0.3,
       scales=list(x="free"),
       border="blue",
       col.hom="royalblue",
       col.het="salmon",
       col.np="grey",
       par.strip.text=list(cex=0.6),
       xlab="Mb",
       ylab=expression(log[2]("copy number")))
}
##---------------------------------------------------------------------------
## For an oligoSnpSet with B allele frequencies:
##---------------------------------------------------------------------------
path <- system.file("extdata", package="VanillaICE")
load(file.path(path, "oligosetForUnitTest.rda"))
## copy number estimates in this object are not on the log
## scale, so specify is.log=FALSE and provide the means for
## the latent copy number states.  IN addition we also specify
## an initial value and constraints for the probability that
## the BAF is an outlier
fit <- hmm(oligoset, is.log=FALSE, cnStates=c(0.5, 1.5, 2, 2, 2.5, 3.2),
   prOutlierBAF=list(initial=1e-4, max=1e-3, maxROH=1e-5))
##
##  For log R ratios, one could simply do
##  hmm(oligoset, prOutlierBAF=list(initial=1e-4, max=1e-3, maxROH=1e-5))
##
if(require(SNPchip)){
## plotting this data
          ## For plotting copy number and log R ratios for multiple genomic intervals, see xyplotLrrBaf
fit <- fit[[1]]
library(IRanges)
library(Biobase)
rect2 <- function(object){
col <- c("red", "red", "white", "grey70", "royalblue", "blue")
object <- object[state(object) !=3 , ]
object <- object[order(width(object), decreasing=TRUE), ]
```

```
rect(xleft=start(object)/1e6,xright=end(object)/1e6,
     ybottom=rep(0.7,length(object)),
     ytop=rep(1,length(object)),
     col=col[state(object)],
     border=col[state(object)])
}
par(las=1)
plot(position(oligoset)/1e6, copyNumber(oligoset)/100,
     pch=".", col="black",
     ylim=c(-1, 3), ylab="copy number", xlab="position (Mb)")
rescale <- function(x, l, u){
b <- 1/(u-l)
a <- l*b
(x+a)/b
}
b <- rescale(baf(oligoset)/1000, -1, 0)
rect2(fit)
##franges <- makeFeatureGRanges(oligoset)
##o <- subjectHits(findOverlaps(fit[4, ], franges))
points(position(oligoset)/1e6, b, pch=".", col="royalblue")
axis(side=4, at =c(-1, -0.5, 0), labels=c(0, 0.5, 1), col="blue")
text(10, 0.1, "BAF", col="blue")
}

##-----------------------------------------------------------------------------
        ##
## For a CNSet object (from the crlmm package):
        ##
##-----------------------------------------------------------------------------
library(oligoClasses)
library2(crlmm)
data(cnSetExample, package="crlmm")
## coerce to an object with log R ratios and B allele frequencies
oligosetlist <- OligoSetList(cnSetExample)
oligoset <- oligosetlist[[1]]
res <- hmm(oligoset, p.hom=0, prOutlierBAF=list(initial=1e-4, max=1e-1, maxROH=1e-3))
res <- res[[1]]
rd <- res[state(res)!=3 & numberProbes(res) >= 5, ]
elementMetadata(rd)$sampleId <- "NA19007"
if(require(lattice) && require(SNPchip)){
## a lattice display for multiple CNV calls ranges.
library(Biobase)
library(IRanges)
xyplotLrrBaf(rd, oligoset,
     frame=200e3,
     panel=xypanelBaf,
     cex=0.5,
     scales=list(x=list(relation="free"),
     y=list(alternating=1,
     at=c(-1, 0, log2(3/2), log2(4/2)),
     labels=expression(-1, 0, log[2](3/2), log[2](4/2)))),
     par.strip.text=list(cex=0.7),
     ylim=c(-3,1),
```

```
      col.hom="grey50",
      col.het="grey50",
      col.np="grey20",
    key=list(text=list(c(expression(log[2]("R ratios")), expression("B allele freqencies")),
       col=c("grey", "blue")), columns=2))
frange <- makeFeatureGRanges(oligoset)
i <- subjectHits(findOverlaps(rd[1,], frange))
b <- baf(oligoset)[i, 1]
b <- b/1000
hist(b, breaks=100)
}
```

---

`hmmResults` *Example output from hmm*

---

## Description

Example output from `hmm` method applied to simulated data.

## Usage

```
data(hmmResults)
```

## Format

A `RangedDataHMM` object.

## Details

The results of a 6-state HMM fit to simulated copy number and genotype data.

## See Also

`xyplot`, `hmm`

## Examples

```
data(hmmResults)
class(hmmResults)
```

hmmSnpSet *Function for fitting a HMM to SnpSet containers*

## Description

Function for fitting a HMM to SnpSet containers. This HMM uses only the genotypes to find regions of homozygosity. For copy number inference, see hmmBeadStudioSet and hmmOligoSnpSet.

## Usage

```
hmmSnpSet2(object, sampleIds, TAUP=1e10, tauMAX,normalIndex=1L,rohIndex=normalIndex+1L,S=2L,...)
hmmSnpSetIce(object, sampleIds, TAUP=1e10, tauMAX, ...)
```

## Arguments

| | |
|---|---|
| object | A SnpSet or SnpSet2 object. |
| sampleIds | A character vector indicating which samples in object to process. For large data, specifying sampleIds is much more efficient that subsetting the object prior to fitting the HMM. |
| TAUP | scalar for defining transition probabilities. Larger values of TAUP discourage jumps between states. |
| tauMAX | Upper bound for the probability that the state at marker t is the same as the state at marker t-1. |
| normalIndex | Index for state with typical rate of heterozygosity. |
| rohIndex | Index for state with homozygous genotypes. |
| S | Integer indicating number of states (typically 2). |
| ... | Additional arguments passed to viterbiForSnpSetIce |

## Value

A GRanges or GRangesList object containing the genomic intervals from the HMM annotated with the copy number states and number of probes per interval.

## Author(s)

R. Scharpf

## See Also

hmm, hmmBafLrrSet2

## icePlatforms *List platforms for which ICE option is supported.*

### Description

Lists platforms for which ICE option is supported.

### Usage

```
icePlatforms()
```

### Details

When procecessing genotypes with the **crlmm**, confidence scores for the diallelic genotype calls are available. One can estimate the emission probabilities for the crlmm diallelic genotypes using the confidence scores by setting the value of `ICE` to TRUE in the constructor for the `HmmOptionList` class. Currently, only certain platforms are supported for this option.

### Value

A character vector of the annotation packages that are supported for the ICE option

### Author(s)

R. Scharpf

### References

Scharpf, RB et al., 2008, Annals of Applied Statistics

### Examples

```
icePlatforms()
```

## oligoSetList-methods *Methods for oligoSetList class*

### Description

The oligoSetList class is a container for genotypes, B allele frequencies, and copy number organized by chromosome. Each element in the list class contains low-level summaries and phenotypic information for a single chromosome. The organization by chromosome facilitates parallelization of methods to identify copy number alterations.

## Methods

For each of the following methods, `object` is an instance of class `oligoSetList`.

`object[[i]]`:
: `i` must be an integer. Return a `oligoSnpSet` object for the ith element in the `oligoSetList` object.

`object[i]`:
: `i` can be a vector of integers. Returns an object of the same class with length equal to the length of the `i` vector.

dims(object):
: Return object dimensions

## See Also

`hmmBafLrrSetList2`

## Examples

```
library(oligoClasses)
library2(crlmm)
data(cnSetExample, package="crlmm")
## coerce to an object with log R ratios and B allele frequencies
oligosetlist <- OligoSetList(cnSetExample)
oligoset <- oligosetlist[[1]]
```

---

`read.bsfiles` *Read BeadStudio/GenomeStudio processed data.*

---

## Description

Read BeadStudio/GenomeStudio processed data and return an array of log R ratios and B allele frequencies.

## Usage

```
read.bsfiles(path = "", filenames, ext = "", row.names = 1, sep = "\t", lrr.colname = "Log.R.Ratio", baf.
```

## Arguments

| | |
|---|---|
| `path` | character: path to plain text files containing BeadStudio processed data |
| `filenames` | character: name of file(s) |
| `ext` | character: filename extension |
| `row.names` | As in read.table. By default, the first column is assumed to be the feature identifiers. |
| `sep` | As in read.table. |

| | |
|---|---|
| lrr.colname | character: used to grep for the log R ratios in the header. E.g., grep(lrr.colname, header) should return a length 1 vector, where header is a vector of the column labels. |
| baf.colname | character: used to grep for the B allele frequency in the header. E.g., grep(baf.colname, header) shoul return a length 1 vector, where header is a vector of the column labels. |
| drop | Logical: if TRUE, dimnames will not be returned |
| colClasses | Vector as in read.table. Note that if colClasses is not specified, the colClasses will be defined by reading in the first few rows. "NULL" will be assigned to all columns not containing B allele frequencies or log R ratios. |
| nrows | As in read.table. |
| ... | Additional arguments passed to read.table. |

## Value

A 3 dimensional array: features x statistic (lrr or baf) x sample

## Author(s)

R. Scharpf

## See Also

read.table

## Examples

```
path <- system.file("extdata", package="VanillaICE")
filename <- list.files(path, pattern="LRRandBAF", full.names=TRUE)
dat <- read.bsfiles(filenames=filename)
```

---

# rescale *Rescale a numeric vector*

---

## Description

Rescale a numeric vector

## Usage

```
rescale(x, l, u)
```

## Arguments

| | |
|---|---|
| x | a numeric vector |
| l | numeric: lower limit of rescaled x. |
| u | numeric: upper limit of rescaled x. |

### Details

Not intended to be called directly, but used in packages that depend on **VanillaICE**

### Value

`numeric` vector the same length as `x` with range [l, u].

### Author(s)

R. Scharpf

---

## robustSds *Calculate robust estimates of the standard deviation*

---

### Description

Uses the median absolute deviation (MAD) to calculate robust estimates of the standard deviation

### Usage

```
robustSds(x, takeLog = FALSE, ...)
```

### Arguments

| | |
|---|---|
| `x` | A matrix of copy number estimates. Rows are features, columns are samples. |
| `takeLog` | Whether to log-transform the copy number estimates before computing robust sds |
| `...` | additional arguments to `rowMedians` |

### Details

For matrices `x` with 4 or more samples, the row-wise MAD (SNP-specific sds) are scaled by sample MAD / median(sample MAD).

If the matrix has 3 or fewer samples, the MAD of the sample(s) is returned.

### Value

Matrix of standard deviations.

### Examples

```
data(locusLevelData, package="oligoClasses")
sds <- robustSds(locusLevelData[["copynumber"]]/100,
 takeLog=TRUE)
```

## rowMAD *Calculate the median absolute deviation for each row in a matrix.*

### Description

Calculate the median absolute deviation for each row in a matrix.

### Usage

```
rowMAD(x, y, ...)
```

### Arguments

| | |
|---|---|
| x | matrix |
| y | ignored |
| ... | Addition arguments to function mad. |

### Value

A numeric vector of median absolute deviations.

### Author(s)

R.Scharpf

### See Also

mad

## SetList-methods *BeadStudioSetList methods*

### Description

Methods for BeadStudioSetList objects

### Objects from the Class

Objects can be created by calls of the form new("BeadStudioSetList", assayDataList, logRRatio, BAF, featureData

## Methods

For the following methods, `object` can be a `BeadStudioSetList` or `oligoSetList` instance.

`object[i]`:
Returns an object of the same class as `object` with length equal to `length(i)`.

`object[[i]]`:
Returns a `BeadStudioSet` or a `oligoSnpSet` object, depending on whether the class of `object` is a `BeadStudioSetList` or an `oligoSetList`.

`object[[i]] <- value` :
Replaces the ith element of the `BafLrrSetList` object by `value`. The object `value` must be a `BafLrrSet` object.

`object$NAME`, `object$NAME <- value`:
Get or set values for for column NAME in `phenoData`. For the get method, `NAME` must be an element of `varLabels(object)`. `value` must be the same length as `ncol(object)`.

`hmm(object, ...)`: Fits HMM to BeadStudioSetList object. Additional arguments can be passed to `hmmBeadStudioSetList`.

`length(x)`:
Returns the number of elements in the list object.

## Author(s)

R. Scharpf

## See Also

`BeadStudioSetList`

## Examples

```
new("BeadStudioSetList")
```

---

`Viterbi-methods` *Methods for Viterbi objects*

---

## Description

Methods for Viterbi objects

## Methods

In the following methods, `object` is of class `Viterbi` or `Viterbi2`.

`emission(object)`: Accessor for the emission probabilities.

`viterbi2Wrapper` *Wrapper function for fitting the viterbi algorithm*

## Description

The viterbi algorithm, implemented in C, estimates the optimal state path as well as the forward and backward variables that are used for updating the mean and variances in a copy number HMM.

The function `viterbi2Wrapper` should not be called directly be the user. Rather, users should fit the HMM by passing an appropriate container to the method `hmm`. We document the `viterbi2Wrapper` arguments as several of the arguments can be modified from their default value when passed from the `hmm` method through the `...`. In particular, `nupdates`, `p.hom`, and `prOutlierBaf`.

## Usage

```
viterbi2Wrapper(index.samples, cnStates, prOutlierBAF=list(initial=1e-5, max=1e-3, maxROH=1e-5),
    p.hom=0.05,
    is.log,
    limits,
    normalIndex=3L,
    nupdates=10,
    tolerance=5,
    computeLLR=TRUE,
    returnEmission=FALSE,
    verbose=FALSE,
    grFun,
    matrixFun,
    snp.index,
    anyNP)
```

## Arguments

| | |
|---|---|
| `index.samples` | Index for the samples that are to be processed. |
| `cnStates` | `numeric` vector for the initial copy number state means. |
| `prOutlierBAF` | A list with elements 'initial', 'max', and 'maxROH' corresponding to the initial estimate of the probability that a B allele frequency (BAF) is an outlier, the maximum value for this parameter over states that do not involve homozygous genotypes, and the maximum value over states that assume homozygous genotypes. This parameter is experimental and could be used to fine tune the HMM for different platforms. For example, the BAFs for the Affy platform are typically more noisey than the BAFs for Illumina. One may want to set small values of these parameters for Illumina (e.g, 1e-5, 1e-3, and 1e-5) and larger values for Affy (e.g., 1e-3, 0.01, 1e-3). |
| `p.hom` | `numeric`: weight for observing homozygous genotypes. For value `0`, homozygous genotypes / B allele frequencies have the same emission probability in the 'normal' state as in the states hemizygous deletion and in copy-neutral region |

| | |
|---|---|
| | of homozygosity. Regions of homozygosity are common in normal genomes. For small values of `p.hom`, hemizygous deletions will only be called if the copy number estimates show evidence of a decrease from normal. |
| `is.log` | `logical`: Whether the copy number estimates in the `r` matrix are on the log-scale. |
| `limits` | `numeric` vector of length two specifying the range of the copy number estimates in `r`. Values of `r` outside of this range are truncated. See `copyNumberLimits`. |
| `normalIndex` | `integer` specifying the index for the normal state. Note that states must be ordered by the mean of the copy number state. E.g., state 1 is homozygous deletion (0 copies), state 2 is hemizygous deletion (1 copy), normal (2 copies), ... In a 6-state HMM, normalIndex should be 3. |
| `nupdates` | `integer` specifying the maximum number of iterations for reestimating the mean and variance for each of the copy number states. The number of iterations may be fewer than `nupdates` if the difference in the log-likelihood between successive iterations is less than `tolerance`. |
| `tolerance` | `numeric` value for indicating convergence of the log-likelihood. If the difference in the log-likelihood of the observed data given the HMM model at iteration i and i-1 is less than tolerance, no additional updates of model parameters using the EM algorithm is needed. |
| `computeLLR` | Logical. Whether to compute a log likelihood ratio (LLR) comparing the predicted state to the normal state. This is calculated post-hoc and is not precisely the likelihood estimated from the Viterbi algorithm. When `FALSE`, the LLR is not calculated and the algorithm is slightly faster. |
| `returnEmission` | Logical. If TRUE, an array of emission probabilities are returned. The dimensions of the array are SNPs, samples, and copy number states. |
| `verbose` | Logical. Whether to print some of the details of the processing. |
| `grFun` | An R function for coercing the state-path from the HMM to a GRanges object. Takes advantage of lexical scope. |
| `matrixFun` | An R function for subsetting the assay data (takes advantage of lexical scope). |
| `snp.index` | The SNP indices |
| `anyNP` | An indicator for whether any of the markers are nonpolymorphic, and therefore BAFs / genotypes are ignored |

## Details

This function is called from the `hmm` methods implemented in this package.

## Value

A `GRanges` object if `returnEmission` is FALSE. Otherwise, an array of emission probabilities is returned.

## Author(s)

R. Scharpf

# Index