Package ‘VanillaICE’

May 4, 2024

Version 1.66.0

Title A Hidden Markov Model for high throughput genotyping arrays

Description
Hidden Markov Models for characterizing chromosomal alteration in high throughput SNP arrays.

Date 2021-11-21

Depends R (>= 3.5.0), BiocGenerics (>= 0.13.6), GenomicRanges (>= 1.27.6), SummarizedExperiment (>= 1.5.3)

Imports MatrixGenerics, Biobase, S4Vectors (>= 0.23.18), IRanges (>= 1.14.0), oligoClasses (>= 1.31.1), foreach, matrixStats, data.table, grid, lattice, methods, GenomeInfoDb (>= 1.11.4), crlmm, tools, stats, utils, BSgenome.Hsapiens.UCSC.hg18

Suggests RUnit, human610quadv1bCrlmm


Enhances doMC, doMPI, doSNOW, doParallel, doRedis

License LGPL-2

LazyLoad yes

biocViews CopyNumberVariation

RoxygenNote 7.1.1

Encoding UTF-8

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

git_branch RELEASE_3_19

git_last_commit 7ff0406

git_last_commit_date 2024-04-30
Repository  Bioconductor 3.19
Date/Publication  2024-05-03
Author  Robert Scharpf [aut, cre]
Maintainer  Robert Scharpf <rscharpf@jhu.edu>

Contents

acf2 ................................................................. 3
ArrayViews-class .............................................. 4
baumWelchUpdate ............................................. 7
calculateEmission ............................................. 7
cnvFilter ....................................................... 8
cn_means ....................................................... 9
CopyNumScanParams-class .................................... 12
doUpdate ....................................................... 14
dropDuplicatedMapLocs ....................................... 14
dropSexChrom .................................................. 15
emission ....................................................... 15
emissionParam ................................................ 16
FilterParam-class ............................................ 16
filters .......................................................... 18
genotypes ..................................................... 18
getExampleSnpExperiment .................................... 19
getHmmParams ................................................ 20
HMM-class ..................................................... 20
hmm2 .......................................................... 21
HMMList ........................................................ 23
HMMList-class ................................................ 24
HmmParam ...................................................... 25
hmmResults .................................................... 26
HmmTrellisParam .............................................. 26
IdiogramParams .............................................. 27
IdiogramParams-class ....................................... 28
isHeterozygous .............................................. 29
LogLik .......................................................... 29
LogLik-class .................................................. 30
lrrFile ........................................................ 31
matrixOrNULL ................................................ 32
NA_filter ...................................................... 32
numberFeatures .............................................. 32
parsedPath .................................................... 33
parseSourceFile .............................................. 33
probability .................................................... 34
rescale ........................................................ 35
rowModes ....................................................... 35
segs ............................................................. 36
show,Viterbi-method ......................................... 36
Description

A wrapper for the function acf that returns the autocorrelation for the specified lag. Missing values are removed.

Usage

```r
acf2(x, lag = 10, 
```

Arguments

- **x**: numeric vector
- **lag**: integer
- **...**: additional arguments to acf

See Also

- acf
ArrayViews-class

ArrayViews class, constructor, and methods

Description

ArrayViews provides views to the low-level data – log R ratios, B allele frequencies, and genotypes that are stored in parsed files on disk, often scaled and coerced to an integer. Accessors to the low-level data are provided that extract the marker-level summaries from disk, rescaling when appropriate.

Usage

ArrayViews(
  class = "ArrayViews",
  colData,
  rowRanges = GRanges(),
  sourcePaths = character(),
  scale = 1000,
  sample_ids,
  parsedPath = getwd(),
  lrrFiles = character(),
  bafFiles = character(),
  gtFiles = character()
)

## S4 method for signature 'ArrayViews,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]

colnames(x) <- value

## S4 method for signature 'ArrayViews'
colnames(x, do.NULL = TRUE, prefix = "col")

## S4 method for signature 'ArrayViews'
x$name

## S4 replacement method for signature 'ArrayViews'
x$name <- value

## S4 method for signature 'ArrayViews'
show(object)

## S4 method for signature 'ArrayViews'
sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)

## S4 method for signature 'ArrayViews'
ncol(x)
## S4 method for signature 'ArrayViews'
nrow(x)

## S4 method for signature 'ArrayViews'
dim(x)

## S4 method for signature 'ArrayViews'
start(x)

### Arguments

- **class**: character string
- **colData**: DataFrame
- **rowRanges**: GRanges object
- **sourcePaths**: character string provide complete path to plain text source files (one file per sample) containing log R ratios and B allele frequencies
- **scale**: log R ratios and B allele frequencies can be stored as integers on disk to increase IO speed. If scale = 1, the raw data is not transformed. If scale = 1000 (default), the log R ratios and BAFs are multiplied by 1000 and coerced to an integer.
- **sample_ids**: character vector indicating how to name samples. Ignored if colData is specified.
- **parsedPath**: character vector indicating where parsed files should be saved
- **lrrFiles**: character vector of file names for storing log R ratios
- **bafFiles**: character vector of file names for storing BAFs
- **gtFiles**: character vector of file names for storing genotypes
- **x**: a ArrayViews object
- **i**: numeric vector or missing
- **j**: numeric vector or missing
- **...**: additional arguments to FUN
- **drop**: ignored
- **value**: a character-string vector
- **do.NULL**: ignored
- **prefix**: ignored
- **name**: character string indicating name in colData slot of ArrayViews object
- **object**: a ArrayViews object
- **X**: a ArrayViews object
- **FUN**: a function to apply to each column of X
- **simplify**: logical indicating whether result should be simplified
- **USE.NAMES**: whether the output should be a named vector
Slots

- colData  A character string
- rowRanges  A DataFrame. WARNING: The accessor for this slot is rowRanges, not rowRanges!
- index  A GRanges object
- sourcePaths  A character string providing complete path to source files (one file per sample) containing low-level summaries (Log R ratios, B allele frequencies, genotypes)
- scale  A length-one numeric vector
- parsedPath  A character string providing full path to where parsed files should be saved
- lrrFiles  character vector of filenames for log R ratios
- bafFiles  character vector of filenames for BAFs
- gtFiles  character vector of filenames for genotypes

See Also

CopyNumScanParams parseSourceFile

Examples

ArrayViews()
## From unit test
require(BSgenome.Hsapiens.UCSC.hg18)
require(data.table)
extdir <- system.file("extdata", package="VanillaICE", mustWork=TRUE)
features <- suppressWarnings(fread(file.path(extdir, "SNP_info.csv")))
fgr <- GRanges(paste0("chr", features$Chr), IRanges(features$Position, width=1),
isSnp=features["Intensity Only"]==0)
fgr <- SnpGRanges(fgr)
names(fgr) <- features["Name"]
bsgenome <- BSgenome.Hsapiens.UCSC.hg18
seqlevels(fgr, pruning.mode="coarse") <- seqlevels(bsgenome)[seqlevels(bsgenome) %in% seqlevels(fgr)]
seqinfo(fgr) <- seqinfo(bsgenome)[seqlevels(fgr),]
fgr <- sort(fgr)
files <- list.files(extdir, full.names=TRUE, recursive=TRUE, pattern="FinalReport")
ids <- gsub(".rds", "", gsub("FinalReport", "", basename(files)))
views <- ArrayViews(rowRanges=fgr,
sourcePaths=files,
sample_ids=ids)
lrrFile(views)
## view of first 10 markers and samples 3 and 5
views <- views[1:10, c(3,5)]
baumWelchUpdate

Function for updating parameters for emission probabilities

Description
This function is not meant to be called directly by the user. It is exported in the package NAMESPACE for internal use by other BioC packages.

Usage
baumWelchUpdate(param, assay_list)

Arguments
param A container for the HMM parameters
assay_list list of log R ratios and B allele frequencies

calculateEmission Calculate the emission probabilities for the 6-state HMM

Description
Given the data and an object containing parameters for the HMM, this function computes emission probabilities. This function is not intended to be called by the user and is exported for internal use by other BioC packages.

Usage
calculateEmission(x, param = EmissionParam())

Arguments
x list of low-level data with two elements: a numeric vector of log R ratios and a numeric vector of B allele frequencies
param parameters for the 6-state HMM

Value
A matrix of emission probabilities. Column correspond to the HMM states and rows correspond to markers on the array (SNPs and nonpolymorphic markers)

See Also
baumWelchUpdate
Filter the HMM-derived genomic ranges for copy number variants

**Description**

The HMM-derived genomic ranges are represented as a GRanges-derived object. `cnvFilter` returns a GRanges object using the filters stipulated in the `filters` argument.

**Usage**

```r
cnvFilter(object, filters = FilterParam())
cnvSegs(object, filters = FilterParam(state = c("1", "2", "5", "6")))
duplication(object, filters = FilterParam(state = c("5", "6")))
deletion(object, filters = FilterParam(state = c("1", "2")))
hemizygous(object, filters = FilterParam(state = "2"))
homozygous(object, filters = FilterParam(state = "1"))
```

```r
## S4 method for signature 'HMM'
cnvSegs(object, filters = FilterParam(state = as.character(c(1, 2, 5, 6))))
```

```r
## S4 method for signature 'HMMList'
segs(object)
```

```r
## S4 method for signature 'HMMList'
hemizygous(object)
```

```r
## S4 method for signature 'HMMList'
homozygous(object)
```

```r
## S4 method for signature 'HMMList'
duplication(object)
```

```r
## S4 method for signature 'HMMList'
cnvSegs(object, filters = FilterParam(state = as.character(c(1, 2, 5, 6))))
```

```r
## S4 method for signature 'HmmGRanges'
cnvSegs(object, filters = FilterParam(state = as.character(c(1, 2, 5, 6))))
```

```r
## S4 method for signature 'HMM'
cnvFilter(object, filters = FilterParam())
```

```r
## S4 method for signature 'HmmGRanges'
cnvSegs(object, filters = FilterParam(state = as.character(c(1, 2, 5, 6))))
```
**cn_means**

**Arguments**

object see showMethods(cnvFilter)

filters a FilterParam object

**See Also**

FilterParam

**Examples**

```r
data(snp_exp)
fit <- hmm2(snp_exp)
segs(fit) ## all intervals
cnvSegs(fit)
filter_param <- FilterParam(probability=0.95, numberFeatures=10, state=c("1", "2"))
cnvSegs(fit, filter_param)
filter_param <- FilterParam(probability=0.5, numberFeatures=2, state=c("1", "2"))
cnvSegs(fit, filter_param)
hemizygous(fit)
homozygous(fit)
duplication(fit)
```

---

**cn_means**

* A parameter class for computing Emission probabilities

**Description**

Parameters for computing emission probabilities for a 6-state HMM, including starting values for
the mean and standard deviations for log R ratios (assumed to be Gaussian) and B allele frequencies
(truncated Gaussian), and initial state probabilities.

This function is exported primarily for internal use by other BioC packages.

**Usage**

```r
cn_means(object)

cn_sds(object)

baf_means(object)

baf_sds(object)

baf_means(object) <- value

baf_sds(object) <- value

cn_sds(object) <- value
```
cn_means(object) <- value

EmissionParam(
  cn_means = CN_MEANS(),
  cn_sds = CN_SDS(),
  baf_means = BAF_MEANS(),
  baf_sds = BAF_SDS(),
  initial = rep(1/6, 6),
  EMupdates = 5L,
  CN_range = c(-5, 3),
  temper = 1,
  p_outlier = 1/100,
  modelHomozygousRegions = FALSE
)

EMupdates(object)

## S4 method for signature 'EmissionParam'
show(object)

Arguments

- **object**: see showMethods("EMupdates")
- **value**: numeric vector
- **cn_means**: numeric vector of starting values for log R ratio means (order is by copy number state)
- **cn_sds**: numeric vector of starting values for log R ratio standard deviations (order is by copy number state)
- **baf_means**: numeric vector of starting values for BAF means ordered. See example for details on how these are ordered.
- **baf_sds**: numeric vector of starting values for BAF means ordered. See example for details on how these are ordered.
- **initial**: numeric vector of initial state probabilities
- **EMupdates**: number of EM updates
- **CN_range**: the allowable range of log R ratios. Log R ratios outside this range are thresholded.
- **temper**: Emission probabilities can be tempered by emit^temper. This is highly experimental.
- **p_outlier**: probability that an observation is an outlier (assumed to be the same for all markers)
- **modelHomozygousRegions**: logical. If FALSE (default), the emission probabilities for BAFs are modeled from a mixture of truncated normals and a Unif(0,1) where the mixture probabilities are given by the probability that the SNP is heterozygous. See Details below for a discussion of the implications.
Details

The log R ratios are assumed to be emitted from a normal distribution with a mean and standard deviation that depend on the latent copy number. Similarly, the BAFs are assumed to be emitted from a truncated normal distribution with a mean and standard deviation that depends on the latent number of B alleles relative to the total number of alleles (A+B).

Value

numeric vector

Details

When `modelHomozygousRegions` is FALSE (the default in versions >= 1.28.0), emission probabilities for B allele frequencies are calculated from a mixture of a truncated normal densities and a Unif(0,1) density with the mixture probabilities given by the probability that a SNP is homozygous. In particular, let \( p \) denote a 6 dimensional vector of density estimates from a truncated normal distribution for the latent genotypes 'A', 'B', 'AB', 'AAB', 'ABB', 'AAAB', and 'ABBB'. The probability that a genotype is homozygous is estimated as

\[
prHom = \left( p["A"] + p["B"] \right) / \text{sum}(p)
\]

and the probability that the genotype is heterozygous (any latent genotype that is not 'A' or 'B') is given by

\[
prHet = 1 - prHom
\]

Since the density of a Unif(0,1) is 1, the 6-dimensional vector of emission probability at a SNP is given by

\[
emit = prHet \times p + (1 - prHet)
\]

The above has the effect of minimizing the influence of BAFs near 0 and 1 on the state path estimated by the Viterbi algorithm. In particular, the emission probability at homozygous SNPs will be virtually the same for states 3 and 4, but at heterozygous SNPs the emission probability for state 3 will be an order of magnitude greater for state 3 (diploid) compared to state 4 (diploid region of homozygosity). The advantage of this parameterization are fewer false positive hemizygous deletion calls. Log R ratios tend to be more sensitive to technical sources of variation than the corresponding BAFs/ genotypes. Regions in which the log R ratios are low due to technical sources of variation will be less likely to be interpreted as evidence of copy number loss if heterozygous genotypes have more 'weight' in the emission estimates than homozygous genotypes. The trade-off is that only states estimated by the HMM are those with copy number alterations. In particular, copy-neutral regions of homozygosity will not be called.

By setting `modelHomozygousRegions = TRUE`, the emission probabilities at a SNP are given simply by the \( p \) vector described above and copy-neutral regions of homozygosity will be called.
Examples

```r
ep <- EmissionParam()
 cn_means(ep)
 ep <- EmissionParam()
 cn_sds(ep)
 ep <- EmissionParam()
 baf_means(ep)
 ep <- EmissionParam()
 baf_sds(ep)
 ep <- EmissionParam()
 baf_means(ep) <- baf_means(ep)
 ep <- EmissionParam()
 baf_sds(ep) <- baf_sds(ep)
 ep <- EmissionParam()
 cn_sds(ep) <- cn_sds(ep)
 ep <- EmissionParam()
 cn_means(ep) <- cn_means(ep)
 ep <- EmissionParam()
 show(ep)
 cn_means(ep)
 cn_sds(ep)
 baf_means(ep)
 baf_sds(ep)
```

CopyNumScanParams-class

Parameters for parsing source files containing SNP-array processed data, such as GenomeStudio files for the Illumina platform

Description

Raw SNP array processed files have headers and variable labels that may depend the software, how the output files was saved, the software version, and other factors. The purpose of this container is to collect the parameters relevant for reading in the source files for a particular project in a single container. This may require some experimentation as the example illustrates. The function `fread` in the `data.table` package greatly simplifies this process.

Usage

```r
CopyNumScanParams(
 cnvar = "Log R Ratio",
 bafvar = "B Allele Freq",
 gtvar = c("Allele1 - AB", "Allele2 - AB"),
 index_genome = integer(),
 select = integer(),
 scale = 1000,
 row.names = 1L
)
```
## S4 method for signature 'CopyNumScanParams'
show(object)

### Arguments

- **cnvar**: length-one character vector providing name of variable for log R ratios
- **bafvar**: length-one character vector providing name of variable for B allele frequencies
- **gtvar**: length-one character vector providing name of variable for genotype calls
- **index_genome**: integer vector indicating which rows of the source files (e.g., GenomeStudio) to keep. By matching on a sorted GRanges object containing the feature annotation (see example), the information on the markers will also be sorted.
- **select**: integer vector specifying indicating which columns of the source files to import (see examples)
- **scale**: length-one numeric vector for rescaling the raw data and coercing to class integer. By default, the low-level data will be scaled and saved on disk as integers.
- **row.names**: length-one numeric vector indicating which column the SNP names are in
- **object**: a CopyNumScanParams object

### Slots

- **index_genome**: an integer vector
- **cnvar**: the column label for the log R ratios
- **bafvar**: the column label for the B allele frequencies
- **gtvar**: the column label(s) for the genotypes
- **scale**: length-one numeric vector indicating how the low-level data should be scaled prior to saving on disk
- **select**: numeric vector indicating which columns to read
- **row.names**: length-one numeric vector indicating which column the SNP names are in

### See Also

ArrayViews parseSourceFile

### Examples

CopyNumScanParams() ## empty container
doUpdate

Helper function to determine whether to update the HMM parameters via the Baum-Welch algorithm

Description
This function is not intended to be called directly by the user, and is exported only for internal use by other BioC packages.

Usage

doUpdate(param)

Arguments

param An object containing parameters for the HMM

See Also

HmmParam

dropDuplicatedMapLocs

Drop markers on the same chromosome having the same genomic coordinates

Description
If there are multiple markers on the same chromosome with the same annotated position, only the first is kept.

Usage

dropDuplicatedMapLocs(object)

Arguments

object a container for which the methods seqnames and start are defined

Value
an object of the same class with duplicated genomic positions removed
dropSexChrom

Examples

data(snp_exp)
g <- rowRanges(snp_exp)
## duplicate the first row
g[length(g)] <- g[1]
rowRanges(snp_exp) <- g
snp_exp2 <- dropDuplicatedMapLocs(snp_exp)

---

dropSexChrom  Filter sex chromosomes

Description

Removes markers on chromosomes X and Y.

Usage

dropSexChrom(object)

Arguments

object  an object for which the methods seqnames and rowRanges are defined.

Value

an object of the same class as the input

---

emission  Methods to set and get emission probabilities

Description

Get or set a matrix of emission probabilities. This function is exported primarily for internal use by other BioC packages.

Usage

emission(object)

emission(object) <- value

Arguments

object  see showMethods(emission)
value  a matrix of emission probabilities

Value

matrix
emissionParam

**Accessor for parameters used to compute emission probabilities**

**Description**

Parameters for computing emission probabilities include the starting values for the Baum Welch update and initial state probabilities.

**Usage**

```r
emissionParam(object)

emissionParam(object) <- value
```

**Arguments**

- `object`: an object of class `EmissionParam`
- `value`: an object of class `EmissionParam`

**Value**

`EmissionParam` instance

**Examples**

```r
hparam <- HmmParam()
emissionParam(hparam)
ep <- EmissionParam()
cn_means(ep) <- log2(c(.1/2, 1/2, 2/2, 2/2, 3/2, 4/2))
emissionParam(hparam) <- ep
```

---

**FilterParam-class**

**Container for the common criteria used to filtering genomic ranges**

**Description**

The maximum a posteriori estimate of the trio copy number state for each genomic range is represented in a `GRanges`-derived class. Ultimately, these ranges will be filtered based on the trio copy number state (e.g., denovo deletions), size, number of features (SNPs), or chromosome. `FilterParam` is a container for the parameters commonly used to filter the genomic ranges.
Usage

FilterParam(
  probability = 0.99,
  numberFeatures = 10,
  seqnames = paste0("chr", c(1:22, "X", "Y")),
  state = as.character(1:6),
  width = 1L
)

## S4 method for signature 'FilterParam'
probability(object)

## S4 method for signature 'FilterParam'
state(object)

## S4 method for signature 'FilterParam'
show(object)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>probability</td>
<td>minimum probability for the call</td>
</tr>
<tr>
<td>numberFeatures</td>
<td>minimum number of SNPs/nonpolymorphic features in a region</td>
</tr>
<tr>
<td>seqnames</td>
<td>the seqnames (character string or Rle to keep)</td>
</tr>
<tr>
<td>state</td>
<td>character: the HMM states to keep</td>
</tr>
<tr>
<td>width</td>
<td>the minimum width of a region</td>
</tr>
<tr>
<td>object</td>
<td>a FilterParam object</td>
</tr>
</tbody>
</table>

Slots

<table>
<thead>
<tr>
<th>Slot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>probability</td>
<td>a length-one numeric vector indicating the minimum posterior probability for the called state. Genomic intervals with posterior probabilities below probability will be filtered.</td>
</tr>
<tr>
<td>numberFeatures</td>
<td>a positive integer indicating the minimum number of features in a segment</td>
</tr>
<tr>
<td>seqnames</td>
<td>a character vector of seqnames to select (i.e., ‘chr1’ for only those intervals on chromosome 1)</td>
</tr>
<tr>
<td>width</td>
<td>positive integer indicating the minimal width of genomic intervals</td>
</tr>
<tr>
<td>state</td>
<td>character string indicating which hidden Markov model states to select</td>
</tr>
</tbody>
</table>

See Also

cnvFilter cnvSegs hmm2
Examples

```r
fp <- FilterParam()
width(fp)
numberFeatures(fp)
seqnames(fp)
## To select CNV segments for which
## - the CNV call has a 'posterior' probability of at least 0.95
## - the number of features is at least 10
## - the HMM states are 1 (homozygous deletion) or 2 (hemizygous deletion)
FilterParam(probability=0.95, numberFeatures=10, state=c("1", "2"))
```

## filters

Accessor for HMM filter parameters

**Description**

Accessor for HMM filter parameters

**Usage**

```r
filters(object)
```

**Arguments**

- `object` see `showMethods(filters)`

## genotypes

Accessor for SNP genotypes

**Description**

Extract SNP genotypes. Genotypes are assumed to be represented as integers: 1=AA, 2=AB, 3=BB.

**Usage**

```r
genotypes(object)
```

```r
## S4 method for signature 'ArrayViews'
lrr(object)
```

```r
## S4 method for signature 'ArrayViews'
baf(object)
```

```r
## S4 method for signature 'ArrayViews'
genotypes(object)
```
getExampleSnpExperiment

## S4 method for signature 'SnpArrayExperiment'
baf(object)

## S4 method for signature 'SnpArrayExperiment'
copyNumber(object)

## S4 method for signature 'SnpArrayExperiment'
lrr(object)

## S4 method for signature 'SnpArrayExperiment'
genotypes(object)

**Arguments**

object see showMethods("genotypes")

**See Also**

copyNumber

---

getExampleSnpExperiment

*Create an example SnpArrayExperiment from source files containing marker-level genomic data that are provided in this package*

---

**Description**

Create an example SnpArrayExperiment from source files containing marker-level genomic data that are provided in this package

**Usage**

getExampleSnpExperiment(bsgenome)

**Arguments**

bsgenome a BSgenome object

**Value**

A SnpArrayExperiment
getHmmParams  

Accessor for HMM model parameters

Description

Accessor for HMM model parameters

Usage

getHmmParams(object)

Arguments

object  

see showMethods(HmmParam)

Examples

hmm_object <- HMM()
getHmmParams(hmm_object)

HMM-class

Container for the segmented data and the 6-state HMM model parameters

Description

The constructor HMM creates and object of class HMM. Not typically called directly by the user.

Usage

HMM(
    granges = GRanges(),
    param = HmmParam(),
    posterior = matrix(),
    filters = FilterParam()
)
# S4 method for signature 'HMM'
state(object)

# S4 method for signature 'HMM'
show(object)

## Arguments
granges a GRanges object
param a HmmParam object
posterior matrix of posterior probabilities
filters an object of class FilterParam
object a HMM object

## Slots
granges a GRanges object
param a HmmParam object
posterior a matrix of posterior probabilities
filters a FilterParam object

## See Also
hmm2

## Examples

```
data(snp_exp)
hmm_list <- hmm2(snp_exp[,1])
resultsFirstSample <- hmm_list[[1]]
resultsFirstSample
HMM()
```

---

**hmm2**  

*Fit a 6-state HMM to log R ratios and B allele frequencies estimated from SNP arrays*

---

## Description

This function is intended for estimating the integer copy number from germline or DNA of clonal origin using a 6-state HMM. The states are homozygous deletion, hemizygous deletion, diploid copy number, diploid region of homozygosity, single copy gain, and two+ copy gain. Because heterozygous markers are more informative for copy number than homozygous markers and regions of homozygosity are common in normal genomes, we currently computed a weighted average of the BAF emission matrix with a uniform 0,1 distribution by the probability that the marker is heterozygous, thereby downweighting the contribution of homozygous SNPs to the likelihood. In addition
to making the detection of copy-neutral regions of homozygosity less likely, it also helps prevent confusing hemizygous deletions with copy neutral regions of homozygosity – the former would be driven mostly by the log R ratios. This is experimental and subject to change.

Usage

```r
hmm2(object, 
    emission_param = EmissionParam(), 
    transition_param = TransitionParam(), 
    ...
)
```

## S4 method for signature 'SnpArrayExperiment'

```r
hmm2(object, 
    emission_param = EmissionParam(), 
    transition_param = TransitionParam(), 
    ...
)
```

## S4 method for signature 'oligoSnpSet'

```r
hmm2(object, 
    emission_param = EmissionParam(), 
    transition_param = TransitionParam(), 
    ...
)
```

## S4 method for signature 'ArrayViews'

```r
hmm2(object, 
    emission_param = EmissionParam(), 
    transition_param = TransitionParam(), 
    tolerance = 2, 
    verbose = FALSE, 
    ...
)
```

Arguments

- `object` A `SnpArrayExperiment`
- `emission_param` A `EmissionParam` object
- `transition_param` A `TransitionParam` object
- `...` currently ignored
- `tolerance` length-one numeric vector. When the difference in the log-likelihood of the Viterbi state path between successive models (updated by Baum Welch) is less
than the tolerance, no additional model updates are performed.

**verbose**

logical. Whether to display messages indicating progress.

**Details**

The `hmm2` method allows parallelization across samples using the foreach paradigm. Parallelization is automatic when enabled via packages such as snow/doSNOW.

**Examples**

```r
tp <- TransitionParam()
TransitionParam(taup=1e12)
data(snp_exp)
emission_param <- EmissionParam(temper=1/2)
fit <- hmm2(snp_exp, emission_param)
unlist(fit)
cnvSegs(fit)
```

```r
## There is too little data to infer cnv reliably in this trivial example.
## To illustrate filtering options on the results, we select
## CNVs for which
## - the CNV call has a posterior probability of at least 0.5
## - the number of features is 2 or more
## - the HMM states are 1 (homozygous deletion) or 2 (hemizygous deletion)
fp <- FilterParam(probability=0.5, numberFeatures=2, state=c("1", "2"))
cnvSegs(fit, fp)
```

```r
## for parallelization
## Not run:
library(snow)
library(doSNOW)
cl <- makeCluster(2, type = "SOCK")
registerDoSNOW(cl)
fit <- hmm2(snp_exp, emission_param)
```

```r
## End(Not run)
```

**HMMList**

Constructor for `HMMList` class

**Description**

The constructor function for the `HMMList` class. The constructor is useful for representing a list of `HMM` objects.

**Usage**

```r
HMMList(object)
```

**Arguments**

- **object** a list. Each element of the list is in instance of the `HMM` class.
See Also

HMMList HMM hmm2

---

HMMList-class

Class, constructor, and methods for representing HMM results from multiple samples

Description

Each element of the HMMList contains the genomic intervals of the HMM segmentation (GRanges-derived object), parameters from the Baum-Welch, and a FilterParam object.

Usage

```r
## S4 method for signature 'HMMList'
show(object)
## S4 method for signature 'HMMList'
unlist(x, recursive = TRUE, use.names = TRUE)
```

Arguments

- `object`: a HMMList object
- `x`: a HMMList object
- `recursive`: logical; currently ignored
- `use.names`: logical; currently ignored

Slots

- `.Data`: a list. Each element of the list should be a HMM object.

See Also

HMM

Examples

```r
data(snp_exp)
fit <- hmm2(snp_exp)
class(fit)
identical(length(fit), ncol(snp_exp))
unlist(fit)
```
**HmmParam**

*Constructor for HmmParam class*

**Description**
Contains emission probabilities, parameters for emission probabilities, and transition probabilities required for computing the most likely state path via the Viterbi algorithm.

**Usage**

```r
HmmParam(
  emission = matrix(0, 0, 0),
  emission_param = EmissionParam(),
  transition = rep(0.99, nrow(emission)),
  chromosome = character(nrow(emission)),
  loglik = LogLik(),
  viterbi = Viterbi(),
  compute_posteriors = TRUE,
  verbose = FALSE
)
```

## S4 method for signature ' HmmParam' 

*show*(object)

## S4 method for signature ' HmmParam' 

*nrow*(x)

## S4 method for signature ' HmmParam' 

*ncol*(x)

**Arguments**

- **emission** A matrix of emission probabilities
- **emission_param** an object of class `EmissionParam`
- **transition** vector of transition probabilities whose length is N-1, where N is the number of markers. User should provide the probability that the state at marker j is the same as the state at marker j-1. It is assumed that the probability of transitioning to state_j from state_j-1 is the same for all states != state_j-1.
- **chromosome** character vector
- **loglik** an object of class `LogLik`
- **viterbi** an object of class `Viterbi`
- **compute_posteriors** logical
- **verbose** logical
- **object** a `HmmParam` object
- **x** a `HmmParam` object
Examples

HmmParam()

hmmResults  

Example output from the hidden markov model

Description

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

Format

a GRanges object

HmmTrellisParam  
Constructor for HmmTrellisParam class

Description

Constructor for HmmTrellisParam class

Usage

HmmTrellisParam(
    ylimits = list(c(0, 1), c(-3, 1)),
    expandfun = function(g) { width(g) * 50 }
)

Arguments

ylimits  
length-two list of the y-axis limits for B allele frequencies and log R ratios, respectively

expandfun  
a function that takes a length-one GRanges object as an argument and computes a width relative to the width of the GRanges object
IdiogramParams

Constructor for IdiogramParam objects

Description

Parameters for plotting idiograms

Usage

IdiogramParams(
  seqnames = character(),
  seqlengths = numeric(),
  unit = "kb",
  genome = "hg19",
  box = list(color = "blue", lwd = 1)
)

## S4 method for signature 'IdiogramParams,ANY'
plot(x, y, ...)

Arguments

- seqnames: length-one character vector providing chromosome name
- seqlengths: length-one numeric vector indicating size of chromosome
- unit: character string indicating unit for genomic position
- genome: character string indicating genome build
- box: a list of parameters for plotting the box around the part of the idiogram that is plotted
- x: an IdiogramParam object
- y: ignored
- ...: ignored

Value

IdiogramParam object
IdiogramParams-class  
Parameter class for plotting idiograms

Description
Parameter class for plotting idiograms

Usage
## S4 method for signature 'IdiogramParams'
show(object)

Arguments
object  an IdiogramParam object

Slots
seqnames  length-one character vector providing chromosome name
seqlengths  length-one numeric vector indicating size of chromosome
unit  character string indicating unit for genomic position (default is 'kb')
genome  character string indicating genome build
box  a list of parameters for plotting the box around the part of the idiogram that is plotted.

Examples
if(require(BSgenome.Hsapiens.UCSC.hg18) && require(grid)){
  si <- seqinfo(BSgenome.Hsapiens.UCSC.hg18)
  iparam <- IdiogramParams(seqnames="chr1",
                           genome="hg18",
                           seqlengths=seqlengths(si)["chr1"],
                           box=list(xlim=c(20e6L, 25e6L), color="blue", lwd=2))

  iparam
  idiogram <- plot(iparam)
  vp <- viewport(x=0.05, y=0.8, width=unit(0.9, "npc"), height=unit(0.2, "npc"),
                  name="vp1", just=c("left", "bottom"))
  grid.newpage()
pushViewport(vp)
  print(idiogram, vp=vp, newpage=FALSE)
}
isHeterozygous

Assess whether genotype is heterozygous based on BAFs

Description

Assess whether genotype is heterozygous based on BAFs

Usage

isHeterozygous(object, cutoff)

## S4 method for signature 'ArrayViews'
isHeterozygous(object, cutoff)

## S4 method for signature 'SnpArrayExperiment'
isHeterozygous(object, cutoff)

## S4 method for signature 'numeric'
isHeterozygous(object, cutoff)

## S4 method for signature 'matrix'
isHeterozygous(object, cutoff)

Arguments

object a SnpArrayExperiment or ArrayViews object containing BAFs, a matrix of BAFs, or a numeric vector of BAFs.
cutoff a length-two numeric vector providing the range of BAFs consistent with allelic heterozygosity

Examples

if(require("BSgenome.Hsapiens.UCSC.hg18")){
  bsgenome <- BSgenome.Hsapiens.UCSC.hg18
  snp_exp <- getExampleSnpExperiment(bsgenome)
  is_het <- isHeterozygous(snp_exp[, 1], c(0.4, 0.6))
  table(is_het)
}

LogLik

Constructor for LogLik class

Description

A container for the log likelihood of the Viterbi state path. Stores the log likelihood from successive updates of model parameters. When the difference between the log likelihoods at iteration i and i-1 is below the tolerance, no additional updates are performed.
Usage

LogLik(loglik = numeric(), tolerance = 1L)

Arguments

loglik length-one numeric vector for the log likelihood of the Viterbi state path
tolerance if the difference in the log-likelihood of the Viterbi state path after the Baum-Welch update is less than the specified tolerance, no additional Baum-Welch updates are required

See Also

LogLik
Description

Accessors for objects of class ArrayViews

Usage

lrrFile(object)

lrrFile(object) <- value

bafFile(object)

gtFile(object)

## S4 method for signature 'ArrayViews'
lrrFile(object)

## S4 replacement method for signature 'ArrayViews'
lrrFile(object) <- value

## S4 method for signature 'ArrayViews'
bafFile(object)

## S4 method for signature 'ArrayViews'
gtFile(object)

Arguments

object see showMethods("lrrFile")

value a character vector of filenames for the log R ratios

Examples

views <- ArrayViews(parsedPath=tempdir())
sourcePaths(views)
lrrFile(views)
bafFile(views)
gtFile(views)
numberFeatures

matrixOrNULL

A class allowing matrix or NULL objects

Description

Exported for internal use by other BioC packages

NA_filter

Remove SNPs with NAs in any of the low-level estimates

Description

Remove SNPs with NAs in any of the low-level estimates

Usage

NA_filter(x, i)

Arguments

x a container for SNP data (SnpArrayExperiment)
i integer vector to subset

Value

An object of the same class

numberFeatures

The number of SNP/nonpolymorphic probes contained in a genomic interval

Description

The number of SNP/nonpolymorphic probes contained in a genomic interval

Usage

numberFeatures(object)

Arguments

object see showMethods(numberFeatures)
parsedPath

Complete path to directory for keeping parsed files

Description
A character string indicating the complete path for storing parsed files.

Usage
parsedPath(object)

## S4 method for signature 'ArrayViews'
parsedPath(object)

Arguments
object a ArrayViews object

See Also
parseSourceFile ArrayViews
ArrayViews

parseSourceFile Function for parsing GenomeStudio files

Description
This function parses genome studio files, writing the low-level data for log R ratios, B allele frequencies, and genotypes to disk as integers (1 file per subject per data type).

Usage
parseSourceFile(object, param)

## S4 method for signature 'ArrayViews,CopyNumScanParams'
parseSourceFile(object, param)

Arguments
object An ArrayViews object
param An object of class CopyNumScanParams

See Also
ArrayViews ArrayViews CopyNumScanParams
Examples

```r
require(BSgenome.Hsapiens.UCSC.hg18)
bsgenome <- BSgenome.Hsapiens.UCSC.hg18
require(data.table)
extrdir <- system.file("extdata", package="VanillaICE", mustWork=TRUE)
features <- suppressWarnings(fread(file.path(extrdir, "SNP_info.csv")))

fgr <- GRanges(paste0("chr", features$Chr), IRanges(features$Position, width=1),
               isSnp=features["Intensity Only"]>=0)
fgr <- SnpGRanges(fgr)
names(fgr) <- features["Name"]
seqlevels(fgr) <- seqlevels(bsgenome)[seqlevels(bsgenome) %in% seqlevels(fgr)]
seqinfo(fgr) <- seqinfo(bsgenome)[seqlevels(fgr),]
fgr <- sort(fgr)
files <- list.files(extrdir, full.names=TRUE, recursive=TRUE, pattern="FinalReport")
views <- ArrayViews(rowRanges=fgr, sourcePaths=files, parsedPath=tempdir())
show(views)

## read the first file
dat <- fread(files[1], skip="[Data"]")
## information to store on the markers
select <- match(c("SNP Name", "Allele1 - AB", "Allele2 - AB",
                   "Log R Ratio", "B Allele Freq"), names(dat))
## which rows to keep in the MAP file. By matching on the sorted GRanges object
## containing the feature annotation, the low-level data for the log R ratios/
## B allele frequencies will also be sorted
index_genome <- match(names(fgr), dat["SNP Name"])
scan_params <- CopyNumScanParams(index_genome=index_genome, select=select)
## parse the source files
parseSourceFile(views, scan_params)
list.files(parsedPath(views))
## Inspecting source data through accessors defined on the views object
require(oligoClasses)
## log R ratios
r <- head(lrr(views))
## B allele frequencies
b <- head(baf(views))
g <- head(genotypes(views))
```

<table>
<thead>
<tr>
<th>probability</th>
<th>Accessor for probability filter</th>
</tr>
</thead>
</table>

Description

Accessor for probability filter
**rescale**

**Usage**

`probability(object)`

**Arguments**

- `object` a `FilterParam` object

---

**rescale** *Rescale a numeric vector*

**Description**

Rescale a numeric vector

**Usage**

`rescale(x, l, u)`

**Arguments**

- `x` numeric vector
- `l` lower limit of rescaled x
- `u` upper limit of rescaled x

---

**rowModes** *Robust statistics for matrices*

**Description**

Compute the column-wide or row-wise mode of numeric matrices

**Usage**

`rowModes(x)`

`colModes(x)`

`rowMAD(x, ...)`

**Arguments**

- `x` matrix
- `...` additional arguments to rowMedians
Value

numeric vector

See Also

mad
mad rowMedians

Examples

X <- matrix(rnorm(100), 10, 10)
rowMAD(X)

segs

Accessor for the HMM segments

Description

Accessor to obtain all segments from the HMM.

Usage

segs(object)

Arguments

object see showMethods(segs)

Value

a GRanges-derived object

show,Viterbi-method

Show method for objects of class Viterbi

Description

Show method for objects of class Viterbi

Usage

## S4 method for signature 'Viterbi'
show(object)

Arguments

object a Viterbi object
snpArrayAssays

Create an assays object from log R ratios and B allele frequencies

Description

This function is exported primarily for internal use by other BioC packages.

Usage

snpArrayAssays(cn = new("matrix"), baf = new("matrix"), ...)

Arguments

cn matrix of log R ratios
baf matrix of B allele frequencies
... additional matrices of the same dimension, such as SNP genotypes.

Examples

data(snp_exp, package="VanillaICE")
r <- lrr(snp_exp)
b <- baf(snp_exp)
sl <- snpArrayAssays(cn=r, baf=b)

SnpArrayExperiment-class

A RangedSummarizedExperiment-derived class of marker-level SNP array data for copy number inference

Description

Constructor for SnpArrayExperiment

Usage

SnpArrayExperiment(
  cn,
  baf,
  rowRanges = GRanges(),
  colData = DataFrame(),
  isSnp = logical(),
  ...
)

## S4 method for signature 'missing'
SnpArrayExperiment(
    cn,
    baf,
    rowRanges = GRanges(),
    colData = DataFrame(),
    isSnp = logical(),
    ...
)

## S4 method for signature 'matrix'
SnpArrayExperiment(
    cn,
    baf,
    rowRanges = GRanges(),
    colData = DataFrame(row.names = colnames(cn)),
    isSnp = logical(),
    ...
)

Arguments

- **cn**: matrix of copy number estimates (e.g., log R ratios)
- **baf**: matrix of B allele frequencies
- **rowRanges**: GRanges object for SNPs/nonpolymorphic markers
- **colData**: DataFrame containing sample-level covariates
- **isSnp**: logical vector indicating whether marker is a SNP
- **...**: additional arguments passed to SummarizedExperiment() constructor function

Examples

```r
## empty container
library(VanillaICE)
data(snp_exp, package="VanillaICE") # example

se <- SnpArrayExperiment(cn=lrr(snp_exp), baf=baf(snp_exp),
                          rowRanges=rowRanges(snp_exp))
```

### SnpExperiment

**Constructor for SnpArrayExperiment**

**Description**

A single-argument generic function to construct a SnpArrayExperiment.
SnpGRanges-class

Usage

SnpExperiment(object)

## S4 method for signature 'ArrayViews'
SnpExperiment(object)

Arguments

object see showMethods('SnpExperiment') for a list of supported objects

Examples

view <- ArrayViews()
SnpExperiment(view)

SnpGRanges-class An extension to GRanges for representing SNPs

Description

An extension to GRanges for representing SNPs

Constructor for SnpGRanges class

Usage

SnpGRanges(object = GRanges(), isSnp, ...)

## S4 method for signature 'missing'
SnpGRanges(object, isSnp)

## S4 method for signature 'GRanges'
SnpGRanges(object, isSnp)

Arguments

object A GRanges object

isSnp A logical vector. Each genomic interval in the GRanges container corresponds to a marker on the genotyping array. isSnp is FALSE for nonpolymorphic markers such as those included on the Affymetrix 6.0 chips.

... ignored

Slots

elementMetadata a SnpDataFrame
Examples

SnpGRanges()
g <- GRanges("chr1", IRanges(15L, 15L))
SnpGRanges(g, isSnp=TRUE)

---

snp_exp

An example SnpArrayExperiment

Description

A container for low-level summaries used for downstream copy number estimation, including log R ratios, B allele frequencies, and genotypes

Format

a SnpArrayExperiment object

---

sourcePaths

Accessor for file paths containing SNP-level summaries

Description

Files containing SNP-level summaries for log R ratios, B allele frequencies, and genotypes – one sample per subject – are required.

Usage

sourcePaths(object)

Arguments

object an ArrayViews object

Examples

sourcePaths(ArrayViews())
**Description**

Retrieve genomic location of SNPs

**Usage**

```r
## S4 method for signature 'oligoSnpSet'
start(x)
```

**Arguments**

- `x` a `oligoSnpSet` object

---

**state,HmmGRanges-method**

*Accessor for copy number state*

**Description**

Extract the copy number state for each genomic interval.

**Usage**

```r
## S4 method for signature 'HmmGRanges'
state(object)
```

**Arguments**

- `object` a `HmmGRanges` object
state-methods

**Accessor for the Viterbi state path**

### Description

The states are represented as integers: 1=homozygous deletion, 2=hemizygous deletion, 3=diploid normal heterozygosity, 4=diploid region of homozygosity, 5=single copy gain, 6=two or more copy gain.

### Usage

```r
## S4 method for signature 'Viterbi'
state(object)
```

### Arguments

- **object**  
  a Viterbi object

---

**sweepMode**  
*Sweep the modal log R ratio (by row or column) from a matrix of log R ratios*

### Description

This function simplifies the process of sweeping the modal log R ratio from the rows or columns of a *SnpArrayExperiment* object. It is most useful when a large number of samples (more than 10) are available and the dataset is a collection of germline samples. We assume that the samples are from a single batch and that the modal value will be a robust estimate of the mean log R ratio for diploid copy number. Variation in the modal estimates between markers is presumed to be attributable to probe effects (e.g., differences hybridization efficiency/PCR do to sequence composition). For sex chromosomes, one should apply this function separately to men and women and then recenter the resulting matrix according to the expected copy number.

### Usage

```r
sweepMode(x, MARGIN)
```

### Arguments

- **x**  
  see showMethods(sweepMode)

- **MARGIN**  
  integer indicating which margin (1=rows, 2=columns) to sweep the mode
threshold

Value

an object of the same class as x

Examples

data(snp_exp)
snp_exp_rowcentered <- sweepMode(snp_exp, 1)
snp_exp_colcentered <- sweepMode(snp_exp, 2)
x <- lrr(snp_exp)
x_rowcentered <- sweep(x, 1, rowModes(x))
all.equal(lrr(snp_exp_rowcentered), x_rowcentered)

threshold

Threshold numeric values

Description

Threshold numeric values according to user-specific limits. The thresholded values can also be jittered near the limits.

Usage

threshold(x, lim = c(-Inf, Inf), amount = 0)

Arguments

x numeric matrix or vector
lim limit at which to threshold entries in x
amount see jitter

See Also

jitter

Examples

x <- rnorm(1000, 0, 3)
y <- threshold(x, c(-5,5))
range(y)
**TransitionParam**  
*Constructor for TransitionParam class*

**Description**
Contains parameters for computing transition probabilities

**Usage**

```r
TransitionParam(taup = 1e+10, taumax = 1 - 5e+06)
```

## S4 method for signature 'TransitionParam'

```
show(object)
```

**Arguments**

- `taup`: length-one numeric vector
- `taumax`: The maximum probability that the current state is the same as the preceding state. See details
- `object`: a TransitionParam object

**Details**

Diagonal elements of the transition probability matrix are computed as $e^{-2*d/taup}$, where $d$ is the distance between markers $i$ and $i-1$ and $taup$ is typically in the range of $1e10$. This probability is constrained to be no larger than $taumax$. The probabilities on the off-diagonal elements are the same and are subject to the constraint that the rows of the transition probability matrix sum to 1.

**Examples**

```r
TransitionParam()
```

```
## higher values of taup make transitions between states less likely
TransitionParam(taup=1e12)
```

---

**updateHmmParams**  
*Run the Baum-Welch algorithm to update HMM parameters*

**Description**
This function is not intended to be called directly by the user. It is exported in the package NAMES-PACE for internal use by other BioC packages.
**Usage**

updateHmmParams(
  object,
  emission_param = EmissionParam(),
  transition_param = TransitionParam()
)

**Arguments**

- **object** a `SnpArrayExperiment` object
- **emission_param** a `EmissionParam` object
- **transition_param** a `TransitionParam` object

---

**VanillaICE**

*A hidden markov model for detection of germline copy number variants from arrays*

---

**viewports**

*Default viewports for plotting CNV data with lattice-style graphics*

---

**Description**

Default viewports for plotting CNV data with lattice-style graphics

**Usage**

```r
viewports()
```

**Value**

```r
list
```

**See Also**

`xyplotList` `xygrid`

**Examples**

```r
vps <- viewports()
```
xyplotList

Lattice-style plots for granges and SnpArrayExperiment objects

Description

Data for the graphic is generated by a call to grangesData.

Usage

xyplotList(granges, se, param = HmmTrellisParam())

## S4 method for signature 'HmmGRanges,SnpArrayExperiment'
xyplotList(granges, se, param = HmmTrellisParam())

## S4 method for signature 'GRangesList,SnpArrayExperiment'
xyplotList(granges, se, param = HmmTrellisParam())

xygrid(trellis_plot, viewports, granges)

Arguments

granges a HmmGRanges object
se a SnpArrayExperiment
param trellis parameters for plotting HMM
trellis_plot an object of class trellis
viewports a list of viewports as provided by the viewports function

See Also

viewports

Examples

if(require("BSgenome.Hsapiens.UCSC.hg18")){
  bsgenome <- BSgenome.Hsapiens.UCSC.hg18
  snp_exp <- getExampleSnpExperiment(bsgenome)
  seqlevels(snp_exp, pruning.mode="coarse") <- "chr22"
  fit <- hmm2(snp_exp)
  g <- reduce(hemizygous(fit), min.gapwidth=500e3)
  trellis_param <- HmmTrellisParam()
  fig <- xyplotList(g, snp_exp, trellis_param)
  vps <- viewports()
  xygrid(fig[[1]], vps, g)
}
Index

* **EmissionParam-methods**
  - cn_means, 9

* **datasets**
  - hmmResults, 26
  - snp_exp, 40

* **manip**
  - rescale, 35
  - '[', ArrayViews, ANY-method
    (ArrayViews-class), 4
  - [, ArrayViews, ANY, ANY, ANY-method
    (ArrayViews-class), 4
  - $[, ArrayViews, ANY-method
    (ArrayViews-class), 4
  - $<-, ArrayViews-method
    (ArrayViews-class), 4

acf, 3
acf2, 3
ArrayViews, 13, 33
ArrayViews (ArrayViews-class), 4
ArrayViews, numeric, numeric-method
    (ArrayViews-class), 4
ArrayViews-class, 4

baf, ArrayViews-method (genotypes), 18
baf, SnpArrayExperiment-method
    (genotypes), 18
baf_means (cn_means), 9
baf_means, ArrayViews-method
    (ArrayViews-class), 4
baf_means, EmissionParam-method
    (cn_means), 9
baf_means, HmmParam-method (cn_means), 9
baf_means<-, cn_means, 9
baf_means<-, EmissionParam, numeric-method
    (cn_means), 9
baf_sds (cn_means), 9

baf_sds, EmissionParam-method
    (cn_means), 9
baf_sds, HmmParam-method (cn_means), 9
baf_sds<-, cn_means, 9
baf_sds<-, EmissionParam, numeric-method
    (cn_means), 9
bafFile (lrrFile), 31
bafFile, ArrayViews-method (lrrFile), 31
baumWelchUpdate, 7
calculateEmission, 7
calculateEmission, list-method
    (calculateEmission), 7
calculateEmission, numeric-method
    (calculateEmission), 7
calculateEmission, RangedSummarizedExperiment-method
    (calculateEmission), 7
cn_means, 9
cn_means, EmissionParam-method
    (cn_means), 9
cn_means, HmmParam-method (cn_means), 9
cn_means<-, cn_means, 9
cn_means<-, EmissionParam, numeric-method
    (cn_means), 9
cn_sds (cn_means), 9
cn_sds, EmissionParam-method (cn_means), 9
cn_sds, HmmParam-method (cn_means), 9
cn_sds<-, cn_means, 9
cn_sds<-, EmissionParam, numeric-method
    (cn_means), 9
cnvFilter, 8, 17
cnvFilter, GRanges-method (cnvFilter), 8
cnvFilter, HmmM-method (cnvFilter), 8
cnvFilter, HmmML-list-method (cnvFilter), 8
cnvSegs, 17
cnvSegs (cnvFilter), 8
cnvSegs, HmmM-method (cnvFilter), 8
cnvSegs, HmmGRanges-method (cnvFilter), 8
cnvSegs, HmmMLlist-method (cnvFilter), 8
colModes (rowModes), 35
colnames (ArrayViews-class), 4
colnames, ArrayViews-method
  (ArrayViews-class), 4
colnames<-(ArrayViews-class), 4
colnames<-, ArrayViews-method
  (ArrayViews-class), 4
copyNumber, SnpArrayExperiment-method
  (genotypes), 18
CopyNumScanParams, 6, 33
CopyNumScanParams
  (CopyNumScanParams-class), 12
CopyNumScanParams-class, 12
deflection (cnvFilter), 8
deflection, HMM-method (cnvFilter), 8
dim, ArrayViews-method
  (ArrayViews-class), 4
doUpdate, 14
dropDuplicatedMapLocs, 14
dropSexChrom, 15
duplication (cnvFilter), 8
duplication, HMM-method (cnvFilter), 8
duplication, HMMList-method (cnvFilter), 8
edmission, 15
edmission, HmmParam-method (emission), 15
edmission<-(emission), 15
edmission<-, Hmm-method (emission), 15
edmission<-, HmmParam-method (emission), 15
EmissionParam, 16, 22, 45
EmissionParam (cn_means), 9
emissionParam, 16
emissionParam, Hmm-method
  (HmmParam), 16
emissionParam, HmmGRanges-method
  (emissionParam), 16
emissionParam, HmmParam-method
  (emissionParam), 16
EmissionParam, missing-method
  (cn_means), 9
EmissionParam, numeric-method
  (cn_means), 9
emissionParam<-(emissionParam), 16
emissionParam<-, HmmGRanges, EmissionParam-method
  (emissionParam), 16
emissionParam<-, HmmParam, EmissionParam-method
  (emissionParam), 16
EMupdates (cn_means), 9
EMupdates, EmissionParam-method
  (cn_means), 9
EMupdates, HmmParam-method (cn_means), 9
FilterParam, 9
FilterParam (FilterParam-class), 16
FilterParam-class, 16
filters, 18
filters, Hmm-method (filters), 18
filters, HmmParam-method (filters), 18
fread, 12
genotypes, 18
genotypes, ArrayViews-method
  (genotypes), 18
genotypes, SnpArrayExperiment-method
  (genotypes), 18
getExampleSnpExperiment, 19
gethmmParams, 20
gethmmParams, Hmm-method (gethmmParams), 20
gethmmParams, HmmParam-method
  (gethmmParams), 20
GRanges, 16
gtFile (lrrFile), 31
gtFile, ArrayViews-method (lrrFile), 31
hemizygous (cnvFilter), 8
hemizygous, Hmm-method (cnvFilter), 8
hemizygous, HMMList-method (cnvFilter), 8
HMM, 24
HMM (HMM-class), 20
HMM-class, 20
hmm2, 17, 21, 24
hmm2, ArrayViews-method (hmm2), 21
hmm2, oligoSnpsSet-method (hmm2), 21
hmm2, SnpArrayExperiment-method (hmm2), 21
HMMList, 23, 24
HMMList-class, 24
Hmmparam, 14, 25
Hmmparam, matrix-method (Hmmparam), 25
Hmmparam, missing-method (Hmmparam), 25
hmmResults, 26
homozygous (cnvFilter), 8
homozygous, HMM-method (cnvFilter), 8
homozygous, HMMList-method (cnvFilter), 8
IdiogramParams, 27
IdiogramParams-class, 28
isHeterozygous, 29
isHeterozygous, ArrayViews-method
(isHeterozygous), 29
isHeterozygous, matrix-method
(isHeterozygous), 29
isHeterozygous, numeric-method
(isHeterozygous), 29
isHeterozygous, SnpArrayExperiment-method
(isHeterozygous), 29
jitter, 43
length, LogLik-method (LogLik-class), 30
LogLik, 29, 30
LogLik-class, 30
lrr, ArrayViews-method (genotypes), 18
lrr, SnpArrayExperiment-method
(genotypes), 18
lrrFile, 31
lrrFile, ArrayViews-method (lrrFile), 31
lrrFile<-, ArrayViews-method (lrrFile), 31
mad, 36
matrixOrNULL, 32
matrixOrNULL-class (matrixOrNULL), 32
NA_filter, 32
NA_filter, character-method (NA_filter), 32
NA_filter, list-method (NA_filter), 32
NA_filter, numeric-method (NA_filter), 32
NA_filter, oligoSnpsSet-method
(NA_filter), 32
NA_filter, SnpArrayExperiment-method
(NA_filter), 32
ncol, ArrayViews-method
(ArrayViews-class), 4
ncol, HmmParam-method (HmmParam), 25
nrow, ArrayViews-method
(ArrayViews-class), 4
nrow, HmmParam-method (HmmParam), 25
numberFeatures, FilterParam-method
(numberFeatures), 32
numberFeatures, HMM-method
(numberFeatures), 32
numberFeatures, HmmGRanges-method
(numberFeatures), 32
parsedPath, 33
parsedPath, ArrayViews-method
(parsedPath), 33
parseSourceFile, 6, 13, 33, 33
parseSourceFile, ArrayViews, CopyNumScanParams-method
(parseSourceFile), 33
plot, IdiogramParams, ANY-method
(IdiogramParams), 27
plot, IdiogramParams-method
(IdiogramParams), 27
probability, 34
probability, FilterParam-method
(FilterParam-class), 16
rescale, 35
rowMAD (rowModes), 35
rowMedians, 36
rowModes, 35
sapply, ArrayViews-method
(ArrayViews-class), 4
segs, 36
segs, HMM-method (segs), 36
segs, HMMList-method (cnvFilter), 8
show, ArrayViews-method
(ArrayViews-class), 4
show, CopyNumScanParams-method
(CopyNumScanParams-class), 12
show, EmissionParam-method (cn_means), 9
show, FilterParam-method
(FilterParam-class), 16
show, Hmm-Method (Hmm-class), 20
show, HmmList-method (HmmList-class), 24
show, HmmParam-method (HmmParam), 25
show, IdiogramParams-method
(IdiogramParams-class), 28
show, LogLik-method (LogLik-class), 30
show, TransitionParam-method
(TransitionParam), 44
show, Viterbi-method, 36
snp_exp, 40
snpArrayAssays, 37
SnpArrayExperiment, 19, 22, 32, 45
SnpArrayExperiment
(SnpArrayExperiment-class), 37
SnpArrayExperiment, matrix-method
(SnpArrayExperiment-class), 37
SnpArrayExperiment, missing-method
(SnpArrayExperiment-class), 37
SnpArrayExperiment-class, 37
SnpExperiment, 38
SnpExperiment, ArrayViews-method
(SnpExperiment), 38
SnpGRanges (SnpGRanges-class), 39
SnpGRanges, GRanges-method
(SnpGRanges-class), 39
SnpGRanges, missing-method
(SnpGRanges-class), 39
SnpGRanges-class, 39
sourcePaths, 40
sourcePaths, ArrayViews-method
(sourcePaths), 40
start, ArrayViews-method
(ArrayViews-class), 4
start, oligoSnpSet-method, 41
state, FilterParam-method
(FilterParam-class), 16
state, HMM-method (HMM-class), 20
state, HmmGRanges-method, 41
state, Viterbi-method (state-methods), 42
state-methods, 42
sweepMode, 42
sweepMode, SnpArrayExperiment-method
(sweepMode), 42
threshold, 43
TransitionParam, 22, 44, 45
TransitionParam, missing-method
(TransitionParam), 44
TransitionParam, numeric-method
(TransitionParam), 44
unlist, HMMList-method (HMMList-class), 24
updateHmmParams, 44
VanillaICE, 45
viewports, 45, 46
xygrid, 45
xygrid (xyplotList), 46
xyplotList, 45, 46
xyplotList, GRangesList, SnpArrayExperiment-method
(xyplotList), 46
xyplotList, HmmGRanges, SnpArrayExperiment-method
(xyplotList), 46