Package 'ExomeDepth'

November 3, 2022

Type Package

Title Calls Copy Number Variants from Targeted Sequence Data

Version 1.1.16

Date 2022-10-14

Encoding UTF-8

Depends R (>= 3.4.0)

Imports Biostrings, IRanges, Rsamtools, GenomicRanges (>= 1.23.0), aod, VGAM (>= 0.8.4), methods, GenomicAlignments, dplyr, magrittr

Suggests knitr

VignetteBuilder knitr

Description
Calls copy number variants (CNVs) from targeted sequence data, typically exome sequencing experiments designed to identify the genetic basis of Mendelian disorders. The method is presented in details in Plagnol et al (2012) <https://pubmed.ncbi.nlm.nih.gov/22942019/>.

License GPL-3

RoxygenNote 7.2.0

NeedsCompilation yes

Author Vincent Plagnol [aut, cre],
Gerard Jungman [ctb] (Author of included GSL fragments),
Brian Gough [ctb] (Author of included GSL fragments),
Jorma O Tahtinen [ctb] (Author of included GSL fragments),
Gert Van den Eynde [ctb] (Author of included GSL fragments)

Maintainer Vincent Plagnol <vincent.plagnol@gmail.com>

Repository CRAN

Date/Publication 2022-11-03 16:00:05 UTC

R topics documented:

`AnnotateExtra,ExomeDepth-method`
AnnotateExtra

Description

Add annotations to a ExomeDepth object.

Usage

```
## S4 method for signature 'ExomeDepth'
AnnotateExtra(x, reference.annotation, min.overlap = 0.5, column.name)
```

Arguments

`x`	An `ExomeDepth` object.
`reference.annotation`	The list of reference annotations in GRanges format.
`min.overlap`	Numeric, defaults to 0.5. This defines the minimum fraction of the CNV call that is covered by the reference call to declare that there is a significant overlap.
`column.name`	The name of the column used to store the overlap (in the slot CNV.calls).

Details

This function takes annotations in the GRanges format and adds these to the CNV calls in the ExomeDepth object. Note that a recent version of `GenomicRanges (> 1.8.10)` is required. Otherwise the function will return a warning and not update the ExomeDepth object.

Value

An ExomeDepth object with the relevant annotations added to the CNVcalls slot.

Examples

```
data(ExomeCount)  #pick an example count file
small_count <- ExomeCount[1:100, ]  #reduce the size for speedy computations
## create a dummy test object
example_object <- new('ExomeDepth', test = small_count$Exome2, reference = small_count$Exome3)

## artifically create a couple of CNV calls for this test
example_object@CNV.calls <- data.frame(chromosome = c(1,7),
                                       start = c(108778622, 61286538),
                                       end = c(109000909,61296735))

data(Conrad.hg19)
print(example_object@CNV.calls)
example_object_annotated <- AnnotateExtra(x = example_object,
                              reference.annotation = Conrad.hg19.common.CNVs,
                              min.overlap = 0.1,
                              column.name = 'Conrad.hg19')
print(example_object_annotated@CNV.calls)
```

`CallCNVs,ExomeDepth-method`
CallCNVs

Description

Call CNV data from an ExomeDepth object.

Usage

```
## S4 method for signature 'ExomeDepth'
CallCNVs(
  x,
  chromosome,
  start,
  end,
  name,
  transition.probability = 1e-04,
  expected.CNV.length = 50000
)
```

Arguments

x	An ExomeDepth object
chromosome	Chromosome information for each exon (factor).
start	Start (physical position) of each exon (numeric, must have the same length as the chromosome argument).
end	End (physical position) of each exome (numeric, must have the same length as the chromosome argument).
name	Name of each exon (character or factor).
transition.probability	Transition probability of the hidden Markov Chain from the normal copy number state to either a deletion or a duplication. The default (0.0001) expect approximately 20 CNVs genome-wide.
expected.CNV.length	The expectation for the length of a CNV. This value factors into the Viterbi algorithm that is used to compte the transition from one state to the next, which depends on the distance between exons.

Details

Must be called on an ExomeDepth object and fits a hidden Markov model to the read depth data with three hidden states (normal, deletion, duplication). Likelihood data must have been pre-computed which should have been done by default when the ExomeDepth object was created.

Value

The same ExomeDepth object provided as input but with the slot CNVcalls containing a data frame with the output of the calling.

Examples

```
data(ExomeCount)
ExomeCount <- ExomeCount[1:500,] ## small for the purpose of this test
ref_counts <- ExomeCount$Exome2 + ExomeCount$Exome3 + ExomeCount$Exome4

## creates a simple ExomeDepth object
test_object <- new('ExomeDepth', test = ExomeCount$Exome1, reference = ref_counts)

## Call CNVs
called_object <- CallCNVs(x = test_object, transition.probability = 10^-4,
                          chromosome = GenomicRanges::seqnames(ExomeCount),
                          start = GenomicRanges::start(ExomeCount),
                          end = GenomicRanges::end(ExomeCount),
                          name = ExomeCount$names)


print(called_object@CNV.calls)
```

Conrad.hg19.common.CNVs *Conrad et al common CNVs*

Description

Positions of common CNV calls (detected in a panel of 42 sample) from the Conrad et al paper (Nature 2010). This is build hg19 of the human genome.

Format

A data frame with common CNV calls.

Source

Conrad et al, Origins and functional impact of copy number variation in the human genome, Nature 2010

count.everted.reads *Count the number of everted reads for a set of BAM files.*

Description

This is the ExomeDepth high level function that takes a GenomicRanges object, a list of indexed/sorted BAM files, and compute the number of everted reads in each of the defined bins.

Usage

```
count.everted.reads(
  bed.frame = NULL,
  bed.file = NULL,
  bam.files,
  index.files = bam.files,
  min.mapq = 20,
  include.chr = FALSE
)
```

Arguments

bed.frame	data.frame containing the definition of the regions. The first three columns must be chromosome, start, end.
bed.file	character file name. Target BED file with the definition of the regions. This file will only be used if no bed.frame argument is provided. No headers are assumed so remove them if they exist. Either a bed.file or a bed.frame must be provided for this function to run.

bam.files	character, list of BAM files to extract read count data from.
index.files	Optional character argument with the list of indexes for the BAM files, without the '.bai' suffix. If the indexes are simply obtained by adding .bai to the BAM files, this argument does not need to be specified.
min.mapq	numeric, minimum mapping quality to include a read.
include.chr	logical, if set to TRUE, this function will add the string 'chr' to the chromosome names of the target BED file.

Details

Everted reads are characteristic of the presence of duplications in a BAM files. This routine will parse a BAM files and the suggested use is to provide relatively large bins (for example gene based, and ExomeDepth has a genes.hg19 object that is appropriate for this) to flag the genes that contain such reads suggestive of a duplication. A manual check of the data using IGV is recommended to confirm that these reads are all located in the same DNA region, which would confirm the presence of a copy number variant.

Value

A data frame that contains the region and the number of identified reads in each bin.

Note

This function calls a lower level function called XXX that works on each single BAM file.

References

Medvedev et al (2009) <https://doi.org/10.1038/nmeth.1374> "Computational methods for discovering structural variation with next-generation sequencing"

See Also

getBAMCounts

Examples

```
data(genes.hg19)
bam_file <- system.file('extdata/minimum_1_25630000_25650000.bam',
                        package = 'ExomeDepth')
genes.hg19.TTC <- subset(genes.hg19, grepl(pattern = '^TTC34', genes.hg19[['name']]))
print(count.everted.reads (bed.frame = genes.hg19.TTC, bam.files = bam_file, min.mapq = 0))
print(count.everted.reads (bed.frame = genes.hg19.TTC, bam.files = bam_file, min.mapq = 35))
```

countBam.everted *Counts everted reads from a single BAM file*

Description

This is a utility function that is called by the higher level count.everted.reads. It processes each BAM file individually to generate the count data.

Usage

```
countBam.everted(bam.file, granges, index = bam.file, min.mapq = 1)
```

Arguments

bam.file	BAM file that needs to be parsed
granges	Genomic Ranges object with the location of the bins for which we want to count the everted reads.
index	Index for the BAM files.
min.mapq	Minimum mapping quality to include reads.

Details

Most users will not use this function, and it will only be called by the higher level count.everted.reads. Nevertheless it may be useful on its own in some cases.

Value

A list with the number of reads in each bin.

See Also

count.everted.reads

Examples

```
data(genes.hg19)
bam_file <- system.file('extdata/minimum_1_25630000_25650000.bam',
                        package = 'ExomeDepth')
genes.hg19.small <- subset(genes.hg19, grepl(pattern = '^TTC34|^RHD', genes.hg19[['name']]))
my_range <- GenomicRanges::GRanges(paste0(genes.hg19.small$chromosome, ":",
                       genes.hg19.small$start, '-', genes.hg19.small$end))
print(my_range)
print(countBam.everted (granges = my_range, bam.file = bam_file, min.mapq = 0))
```

`countBamInGRanges.exomeDepth` *Compute read count data from BAM files.*

Description

Parses a BAM file and count reads that are located within a target region defined by a GenomicRanges object.

Usage

```
countBamInGRanges.exomeDepth(
  bam.file,
  index = bam.file,
  granges,
  min.mapq = 1,
  read.width = 1
)
```

Arguments

`bam.file`	BAM file to be parsed
`index`	Index of the BAM file, without the '.bai' suffix.
`granges`	Genomic ranges object defining the bins
`min.mapq`	Minimum read mapping quality (Phred scaled).
`read.width`	For single end reads, an estimate of the frament size. For paired reads, the fragment size can be directly computed from the paired alignment and this value is ignored.

Details

Largely derived from its equivalent function in the exomeCopy package.

Value

A GRanges object with count data.

Examples

```
minimum_bam_file <- system.file('extdata/minimum_1_25630000_25650000.bam',
                                package = 'ExomeDepth')

data(exons.hg19)
exons.hg19.RHD <- subset(exons.hg19, grepl(pattern = '^RHD', exons.hg19[['name']]))
my_range <- GenomicRanges::GRanges(paste0(exons.hg19.RHD$chromosome, ":",
                       exons.hg19.RHD$start, '-', exons.hg19.RHD$end))
```

```
print(countBamInGRanges.exomeDepth(bam.file = minimum_bam_file,
                                   granges = my_range))
```

C_hmm *C_hmm*

Description

Implements the hidden Markov model (Viterbi algorithm) using a C routine

Value

A list with two objects: the first contains the optimum Viterbi path and the second the actual CNV calls

---

ExomeCount *Example dataset for ExomeDepth*

Description

An example dataset of 4 exome samples, chromosome 1 only.

Format

A data frame with 25592 observations on the following 9 variables:

- chromosome, Character vector with chromosome names (only chromosome 1 in that case)
- start, start of exons
- end, end of exons
- exons, character name of exons
- camfid.032KA_sorted_unique.bam
- camfid.033ahw_sorted_unique.bam
- camfid.034pc_sorted_unique.bam
- camfid.035if_sorted_unique.bam
- GC, a numeric vector with the GC content

Source

Dataset generated in collaboration with Sergey Nejentsev, University of Cambridge.

ExomeDepth-class *Class* ExomeDepth

Description

A class to hold the read count data that is used by ExomeDepth to call CNVs.

Objects from the Class

Objects can be created by calls of the form `new("ExomeDepth", data = NULL, test, reference, formula = 'cbind(test, reference) ~ 1', subset.for.speed = NULL)`. `data` is optional and is only used if the `formula` argument refers to covariates (in which case these covariates must be included in the data frame). `test` and `reference` refer to the read count data for the test and reference samples.

Critically, it is not required to store the positions of the DNA fragments that led to the test and reference counts. That is only required for the function `TestCNV`. If this is of use, a GRanges object can be provided using the argument `positions`.

Creating a ExomeDepth object will automatically fit the beta-binomial model (using routines from the `aod` package) and compute the likelihood for the three copy number states (normal, deletion and duplication).

References

A robust model for read count data in exome sequencing experiments and implications for copy number variant calling, Plagnol et al 2012

See Also

`?select.reference.set` `?CallCNVs`

Examples

```
showClass("ExomeDepth")
```

exons.hg19 *Positions of exons on build hg19 of the human genome*

Description

Exon position extracted from the ensembl database version 71.

Format

A data frame with 192,379 observations on the following 4 variables:

- chromosome, a factor with levels 1, 2 3 4, 5 6 7 8 9, 10 11 12 13 14 15 16 17 18 19 2 20 21 22
- start a numeric vector
- end a numeric vector
- name A character vector of names for the exon(s)

Source

Ensemble database version 71.

`exons.hg19.X` *Positions of exons on build hg19 of the human genome and on chromosome X*

Description

Exon position extracted from the ensembl database version 61 and on chromosome X only.

Format

A data frame of exons with the following 4 variables:

- chromosome, a factor with levels X, Y.
- start Numeric.
- end Numeric.
- name Character names for the exons.

Source

Ensemble database version 71.

genes.hg19 *Positions of genes on build hg19 of the human genome*

Description

Exon position extracted from the ensembl database version 71.

Format

A data frame with 18,033 observations on the following 4 variables:

- chromosome, a factor with levels 1, 2 3 4, 5 6 7 8 9, 10 11 12 13 14 15 16 17 18 19 2 20 21 22
- start a numeric vector
- end a numeric vector
- name A character vector of names for the exon(s)

Source

Ensemble database version 71.

get.power.betabinom *Estimate the power to compare two beta-binomial distributions.*

Description

A power study useful in the context of ExomeDepth.

Usage

```
get.power.betabinom(
  size,
  my.phi,
  my.p,
  my.alt.p,
  theory = FALSE,
  frequentist = FALSE,
  limit = FALSE
)
```

Arguments

size	Number of samples from the beta-binomial distribution.
my.phi	Over-dispersion parameter.
my.p	Expected p under the null.
my.alt.p	Expected p under the alternative.
theory	logical, should a theoretical limit (large sample size) be used? Defaults to FALSE.
frequentist	logical, should a frequentist version be used? Defaults to FALSE.
limit	logical, should another large sample size limit be used? Defaults to FALSE.

Value

An expected Bayes factor.

Examples

```
get.power.betabinom(size = 200, my.phi = 0.1, my.p = 0.2, my.alt.p = 0.6)
get.power.betabinom(size = 200, my.phi = 0.1, my.p = 0.2, my.alt.p = 0.2)
```

---

getBamCounts *Get count data for multiple exomes*

Description

Essentially a wrapper for the accessory function countBamInGRanges which only considers a single BAM file at a time.

Usage

```
getBamCounts(
  bed.frame = NULL,
  bed.file = NULL,
  bam.files,
  index.files = bam.files,
  min.mapq = 20,
  read.width = 300,
  include.chr = FALSE,
  referenceFasta = NULL
)
```

Arguments

bed.frame	data.frame containing the definition of the regions. The first three columns must be chromosome, start, end.
bed.file	character file name. Target BED file with the definition of the regions. This file will only be used if no bed.frame argument is provided. No headers are assumed so remove them if they exist. Either a bed.file or a bed.frame must be provided for this function to run.
bam.files	character, list of BAM files to extract read count data from.
index.files	Optional character argument with the list of indexes for the BAM files, without the '.bai' suffix. If the indexes are simply obtained by adding .bai to the BAM files, this argument does not need to be specified.
min.mapq	numeric, minimum mapping quality to include a read.
read.width	numeric, maximum distance between the side of the target region and the middle of the paired read to include the paired read into that region.
include.chr	logical, if set to TRUE, this function will add the string 'chr' to the chromosome names of the target BED file.
referenceFasta	character, file name for the reference genome in fasta format. If available, GC content will be computed and added to the output.

Details

This function is largely a copy of a similar one available in the exomeCopy package.

Value

A GenomicRanges object that stores the read count data for the BAM files listed as argument.

Author(s)

Vincent Plagnol

References

exomeCopy R package.

Examples

```
data(exons.hg19)
exons.hg19.RHD <- subset(exons.hg19, grepl(pattern = '^RHD', exons.hg19[['name']]))
minimum_bam_file <- system.file('extdata/minimum_1_25630000_25650000.bam',
                                package = 'ExomeDepth')
my_counts <- getBamCounts(bed.frame = exons.hg19.RHD,
                          bam.files = minimum_bam_file)
print(my_counts)
```

`get_loglike_matrix` *get_loglike_matrix*

Description

Computes the loglikelihood matrix for the three states and each exon

Value

A likelihood matrix with the states as rows and the exons as columns

`initialize,ExomeDepth-method`
ExomeDepth initialization tool

Description

Builds an exomeDepth object from test and reference vectors

Usage

```
## S4 method for signature 'ExomeDepth'
initialize(
  .Object,
  data = NULL,
  test,
  reference,
  formula = "cbind(test, reference) ~ 1",
  phi.bins = 1,
  prop.tumor = 1,
  subset.for.speed = NULL,
  positions = GenomicRanges::GRanges(),
  verbose = TRUE
)
```

Arguments

`.Object`	ExomeDepth object
`data`	Data frame containing potential covariates.
`test`	Numeric, vector of counts for the test sample.
`reference`	Numeric, vector of counts for the reference sample.
`formula`	Linear model to be used when fitting the data.

phi.bins	Numeric, defaults to 1. Number of discrete bins for the over-dispersion parameter phi, depending on read depth. Do not modify this parameter for the standard use of ExomeDepth.
prop.tumor	Numeric, defaults to 1. For the somatic variant calling, this assesses the proportion of the test sample data originating from the tumour. Do not modify this parameter for the standard use of ExomeDepth.
subset.for.speed	Numeric, defaults to NULL. If non-null, this sets the number of data points to be used for an accelerated fit of the data.
positions	Optional GRanges argument specifying the positions of the exons (or DNA regions) where the reads were counted for test and reference.
verbose	Logical, controls the output level.

Value

An ExomeDepth object, which contains the CNV calls after running a Viterbi algorithm.

Examples

```
data(ExomeCount)  #pick an example count file
small_count <- ExomeCount[1:100, ]  #reduce the size for speedy computations

## remove exons without data below
small_count <- small_count[ small_count$Exome2 + small_count$Exome3 > 0, ]

example_object <- new('ExomeDepth', test = small_count$Exome2,
                                    reference = small_count$Exome3,
                                    formula = 'cbind(test, reference) ~ 1')
print(example_object)
print( mean(example_object@expected)) ## proportion of reads expected to match the test set
```

---

plot-methods *Plotting function for ExomeDepth objects*

Description

Plot function for the ExomeDepth class

Usage

```
## S4 method for signature 'ExomeDepth,ANY'
plot(
  x,
  sequence,
  xlim,
  ylim = NULL,
```

```
  count.threshold = 10,
  ylab = "Observed by expected read ratio",
  xlab = "",
  type = "b",
  pch = "+",
  with.gene = FALSE,
  col = "red",
  ...
)
```

Arguments

x	ExomeDepth object
sequence	character, Name of the sequence/chromosome of the region to plot (for example "chr5" would be typical)
xlim	numeric of size 2, start and end position of the region to plot
ylim	numeric of size 2, range for the y-axis
count.threshold	numeric, minimum number of reads in the reference set to display a point in the plot
ylab	Defaults to "
xlab	Defaults to "
type	Defaults to 'b'
pch	Defaults to '+'
with.gene	Logical, defaults to FALSE, Should the gene information (obtained from the annotation data) be plotted under the read depth plot?
col	character, Colour for the line displaying the read depth ratio for each exon
...	Additional arguments to be passed to the base plot function

---

qbetabinom *Quantile for betabin function*

Description

Quantile function for the betabinomal distribution using the p/phi parameterisation.

Usage

```
qbetabinom(p, size, phi, prob)
```

Arguments

p	Point of the distribution from which one is looking for the quantile
size	Sample size of the random variable
phi	Over-dispersion parameter
prob	Mean probability of the binomial distribution

Details

Filling a gap in the VGAM package.

Value

A real number corresponding to the quantile p.

Author(s)

Vincent Plagnol

See Also

VGAM R package.

Examples

```
qbetabinom(p = 0.2, size = 50, phi = 0.4, prob = 0.3)
qbetabinom(p = 0.2, size = 50, phi = 0.1, prob = 0.8)
```

---

qbetabinom.ab *Quantile function for the beta-binomial distribution*

Description

Standard qbetabinomial.ab function which is missing from the VGAM package.

Usage

```
qbetabinom.ab(p, size, shape1, shape2)
```

Arguments

p	Mean value of the beta-binomial distribution.
size	Size of the beta-binomial.
shape1	First parameter of the beta distribution for p.
shape2	Second parameter of the beta distribution for p.

Value

A quantile of the distribution.

See Also

VGAM package.

Examples

```
qbetabinom.ab(p = 0.5, size = 50, shape1 = 0.2, shape2 = 0.25)
```

`select.reference.set` *Combine multiple samples to optimize the reference set in order to maximise the power to detect CNV.*

Description

The power to detect copy number variant (CNVs) from targeted sequence data can be maximised if the most appropriate set of sequences is used as reference. This function is designed to combine multiple reference exomes in order to build the best reference set.

Usage

```
select.reference.set(
  test.counts,
  reference.counts,
  bin.length = NULL,
  n.bins.reduced = 0,
  data = NULL,
  formula = "cbind(test, reference) ~ 1",
  phi.bins = 1
)
```

Arguments

`test.counts`	Read count data for the test sample (numeric, typically a vector of integer values).
`reference.counts`	Matrix of read count data for a set of additional samples that can be used as a comparison point for the test sample.
`bin.length`	Length (in bp) of each of the regions (often exons, but not necessarily) that were used to compute the read count data (i.e. what is provided in the argument test.counts of this function). If not provided all bins are assumed to have equal length.
`n.bins.reduced`	This optimization function can be slow when applied genome-wide. For the purpose of building the reference sample, it is not necessary to use the full data. The number provided by this argument specifies the number of regions (typically exons) that will be sub-sampled (using a grid) to optimise the referenceset. I find that 10,000 is largely sufficient for exome data.
`data`	Defaults to NULL: A data frame of covariates that can be included in the model.
`formula`	Defaults to 'cbind(test, reference) ~ 1'. This formula will be used to fit the read count data. Covariates present in the data frame (for example GC content) can be included in the right hand side of the equation'. If covariates are provided they must be provided as arguments (in the data frame "data").
`phi.bins`	Numeric integer (typically 1, 2, or 3) that specifies the number of windows where the over-dispersion parameter phi can vary. It defaults to 1, i.e. a single over-dispersion parameter, independently of read depth.

Value

`reference.choice`
: character: list of samples selected as optimum reference set.

`summary.stats`
: A data frame summarizing the output of this computation, including expected Bayes factor, Rs statistic (see reference for explanation) for multiple choices of reference set.

Examples

```
data(ExomeCount)
ref_counts <- matrix(data = c(ExomeCount$Exome2, ExomeCount$Exome3, ExomeCount$Exome4),
                     ncol = 3, byrow = FALSE)
colnames(ref_counts) <- c("Ex1", "Ex2", "Ex3")

select.reference.set(test.counts = ExomeCount$Exome1[1:200],
                     reference.counts = ref_counts[1:200,])
```

---

somatic.CNV.call *somatic.CNV.call*

Description

Call somatic variants between healthy and disease tissues.

Usage

```
somatic.CNV.call(normal, tumor, prop.tumor = 1, chromosome, start, end, names)
```

Arguments

`normal`	Read count data (numeric vector) for the normal tissue.
`tumor`	Read count data (numeric vector) for the tumor.
`prop.tumor`	Proportion of the tumour DNA in the tumour sample (between 0 and 1, and less than 1 if there is normal tissue in the tumor sample).
`chromosome`	Chromosome information for the bins.
`start`	Start position of each bin (typically in bp).
`end`	End position of each bin.
`names`	Names for each bin (tyically exon names but any way to track the bins will do).

Details

Use read depth data from targeted sequencing experiments to call CNV between a tumor and matched healthy tissue. This is an experimental function at this stage.

Value

An ExomeDepth object with CNV calls.

Note

Absolutely experimental, not the main function from the package.

TestCNV,ExomeDepth-method
TestCNV

Description

Computes the Bayes Factor in favour of a CNV defined by position and type.

Usage

```
## S4 method for signature 'ExomeDepth'
TestCNV(x, chromosome, start, end, type)
```

Arguments

x	ExomeDepth object
chromosome	Character, chromosome name.
start	Numeric, start of the tested CNV
end	Numeric, end of the tested CNV
type	Character, must be either deletion or duplication.

Value

A single numeric value that is the log likelihood ratio in favour of a call present at this location.

Examples

```
data(ExomeCount)
ExomeCount <- ExomeCount[1:500,] ## small for the purpose of this test
ref_counts <- ExomeCount$Exome2 + ExomeCount$Exome3 + ExomeCount$Exome4

## creates a simple ExomeDepth object
## Note that I include the positions here (GRanges format)
## This is necessary for TestCNV to work
test_object <- new('ExomeDepth', test = ExomeCount$Exome1,
                                 reference = ref_counts,
                                 positions = ExomeCount)

## A positive control first
TestCNV (test_object, chromosome = 'chr1', start = 1387428, end = 1405539, type = 'deletion')
```

```
## And a region without evidence of call
TestCNV (test_object, chromosome = 'chr1', start = 987428, end = 1005539, type = 'deletion')
```

`viterbi.hmm` *Computes the Viterbi path for a hidden markov model*

Description

Estimates the most likely path for a hidden Markov Chain using the maximum likelihood Viterbi algorithm. The code assumes 3 states (normal, deletion and duplication). It is also setup for the first and last exons to be at position 0 (i.e. normal).

Usage

```
viterbi.hmm(transitions, loglikelihood, positions, expected.CNV.length)
```

Arguments

`transitions`	Transition matrix
`loglikelihood`	numeric matrix containing the loglikelihood of the data under the possible states
`positions`	Positions of the exons
`expected.CNV.length`	Expected length of CNV calls, which impacts the transition matrix between CNV states.

Details

Standard forward-backward Viterbi algorithm using a precomputed matrix of likelihoods.

Value

A list with the two slots ‘Viterbi.path‘ and ‘calls‘.

Examples

```
transitions <- matrix(data = 1/3, ncol = 3, nrow = 3)
loglikelihood <- matrix(c(rep(c(0, -10, -10), 3),
                          rep(c(-10, -10, 0), 3),
                          rep(c(-10, 0, -10), 4)), nrow = 3)
## note the final 0 state, enforced by the code
viterbi.hmm(transitions, t(loglikelihood), positions = 1:10, expected.CNV.length = 1)

## Now we cannot transition out of 0 and should have no call
transitions <- matrix(c(1, 0, 0, 0, 1, 0, 0, 0, 1), ncol = 3)

## we can check that no call is made
viterbi.hmm(transitions, t(loglikelihood), positions = 1:10, expected.CNV.length = 1)
```

Index