Package 'h5vc'

October 8, 2014

Type Package

Title Managing alignment tallies using a hdf5 backend

Version 1.4.0

Author Paul Theodor Pyl

Maintainer Paul Theodor Pyl <pyl@embl.de>

Description This package contains functions to interact with tally data from NGS experiments that is stored in HDF5 files. For detail see the webpage at http://www.ebi.ac.uk/~pyl/h5vc.

License GPL (>= 3)

VignetteBuilder knitr

Depends grid, gridExtra, ggplot2

Imports rhdf5, reshape, IRanges, Biostrings, Rsamtools, methods,GenomicRanges, abind, BiocParallel, BatchJobs, h5vcData

Suggests knitr, locfit, BSgenome.Hsapiens.UCSC.hg19, bit64

R topics documented:

`h5vc-package` *Managing alignment tallies using a hdf5 backend*

Description

This package contains functions to interact with tally data from NGS experiments that is stored in HDF5 files. For detail see vignettes shipped with this package.

Details

Package:	h5vc
Type:	Package
Version:	1.0.4
Date:	2013-10-11
License:	GPL (>= 3)

This package is desgned to facilitate the analysis of genomics data through tallies stored in a HDF5 file. Within a HDF5 file the tally is simply a table of bases times genomic positions listing for each position the count of each base observed as a mismatch in the sample at any given position. Strand and sample are additional dimension in this array, which leads to a 4D-array called 'Counts'. The total coverage is stored in a separate array of 3 dimensions (Sample x Strand x Genomic Position) called 'Coverages', there is a 3 dimensional 'Deletions' array and a 1D-vector encoding the reference base ('Reference'). Those 4 arrays are stored as datasets within a HDF5 tally file in which the group-structure of the tally file encodes for the organisatorial levels of 'Study' and 'Chromosome'. For details on the layout of HDF5 files visit (http://www.hdfgroup.org), a short description is given in the vignettes.

Creating those HDF5 tally files can be accomplished from within R or through a Python script that will generate a tally file from a set of .bam files. The workflow is described in the vignettes `h5vc.creating.tallies` and `h5vc.creating.tallies.within.R`.

Author(s)

Paul Pyl Maintainer: Paul Pyl pyl@embl.de

applyTallies *Preparing the results of tallyBAM for writing to an HDF5 tally file*

Description

This function tallies a set of bam files and prepares the data for writing to an HDF5 tally file.

Usage

```
applyTallies( bamfiles, chrom, start, stop, q=25, ncycles = 0, max.depth=1000000, prepForHDF5 = TRUE, re
```

Arguments

bamfiles	A character vector of filenames of the bam files that should be tallies. Note that for writing to an HDF5 file the order of this vector must match the order of the Column field in the sampledata object that corresponds to the dataset - see setSampleData for details.
prepForHDF5	Boolean flag to specify whether the data shall be structured for compatibility with the HDF5 tally file format. See the details section of this manual page.
reference	A DNAString object containing the reference sequence corresponding to the region that is described in the counts array – if this is NULL a consensus vote will be used to estimate the reference at any given position, this means you cannot detect variants with AF >= 0.5 anymore
stratifyDeletions	Boolean flag to specify if deletion counts should be stratified by sequencing cycle or not - default is FALSE - note that dataset in the tally file must have compatible dimensions.
chrom	Chromosome in which to tally
start	First position of the tally
stop	Last position of the tally
q	quality cut-off for considering a base call
ncycles	number of sequencing cycles form the front and back of the read that should be considered unreliable - used for stratifying the nucleotide counts
max.depth	only tally a position if there are less than this many reads overlapping it - can prevent long runtimes in unreliable regions
BPPARAM	BPPARAM object to be passed to the bplapply call used to apply along the filenames - see BiocParallel documentation for details

Details

This is a wrapper function for applying tallyBAM to a set of bam files specified in the bamfiles argument. If prepForHDF5 is not true the result is equivalent to calling tallyBAM with lapply on the file names, otherwise the resulting data structure has the same layout as the return value of h5readBlock and can be written to an HDF5 tally file directly. The order or samples along the sample dimension is the same as the order of the file names (i.e. the order of the bamfiles argument).

Value

A list with slots containing the Counts,Coverages,Deletions and Reference datasets for the given sample if prepForHDF5 is true, a list of 3D-arrays (Nucleotide x Strand x Position) otherwise.

Author(s)

Paul Pyl

Examples

```
library(h5vc)
files <- c("NRAS.AML.bam","NRAS.Control.bam")
bamFiles <- file.path( system.file("extdata", package = "h5vcData"), files)
chrom = "1"
startpos <- 115247090
endpos <- 115259515
theData <- applyTallies( bamFiles, chr = chrom, start = startpos, stop = endpos, ncycles = 10 )
str(theData)
```

---

batchTallies *Tallying bam files in parallel using* BatchJobs *on high performance compute clusters (HPC)*

Description

These function tally a set of bam files in blocks spanning a specified region and write the results to an HDF5 tally file; uses BatchJobs for parallel computation on HPCs

Usage

```
batchTallyParam(
  bamFiles,
  destination,
  group,
  chrom, start, stop,
  blocksize = 100000,
  registryDir = tempdir(),
 resources = list("queue" = "research-rh6", "memory"="4000", "ncpus"="4", walltime="90:00"),
  q=25, ncycles = 0, max.depth=1000000,
  reference = NULL,
  sleep = 5
)

batchTallies( confList = batchTallyParam() )

rerunBatchTallies( confList, tryCollect = TRUE )

collectTallies(blocks, confList, registries )
```

Arguments

bamFiles	A character vector of filenames of the bam files that should be tallies. Note that for writing to an HDF5 file the order of this vector must match the order of the Column field in the sampledata object that corresponds to the dataset - see setSampleData for details.
reference	A DNAString object containing the reference sequence corresponding to the region that is to be tallied – if this is NULL a consensus vote will be used to estimate the reference at any given position, this means you cannot detect variants with AF >= 0.5 anymore – especially when tallying more than one bamFile you really should specify this
destination	Filename of the HDF5 tally file that will be written to – this needs to contain all the groups and datasets already – see prepareTallyFile for details
group	Location within the tally file where the data will be written – e.g. "/ExampleStudy/22"
chrom	Chromosome in which to tally
start	First position of the tally
stop	Last position of the tally
q	quality cut-off for considering a base call
ncycles	number of sequencing cycles form the front and back of the read that should be considered unreliable - used for stratifying the nucleotide counts
max.depth	only tally a position if there are less than this many reads overlapping it - can prevent long runtimes in unreliable regions
blocksize	Size of the blocks in bases that the tallying will be performed in, this influences the number of jobs send to the cluster
registryDir	Directory in which the registries created by BatchJobs wil be held, this can be temporary since we delete them when we are done
resources	A named list specifying the resource requirements of the cluster jobs, this must contain names for the fields specified in the cluster configuration file – see the documentation of BatchJobs for details
confList	A configuration list as returned by a call to batchTallyParam()
sleep	Number of seconds to sleep before checking if blocks are finshed, increase this if you have large blocks and find the output of batchTallies to verbose
tryCollect	Boolean flag specifying whether the rerunBatchTallies function should try to collect data from the specified registries before re-submitting.
blocks	data.frame defining blocks to tally in, result of a cal to defineBlocks
registries	A list mapping registry IDs to the work paths of the corresponding registries

Details

This is a wrapper function for applying tallyBAM to a set of bam files specified in the bamFiles argument. The order or samples along the sample dimension is the same as the order of the file names (i.e. the order of the bamfiles argument). The function uses BatchJobs to dispatch tallying in blocks along the genome to a HPC and collects the results and writes them into the HDF5 tally file specified in the destination parameter.

rerunBatchTallies can be used to re-submit failed blocks.

collectTallies can be used to manually collect tally data from the registries created by batchTallies

Value

None – prints progress messages along the way.

Author(s)

Paul Pyl

Examples

```
## Not run:
library(h5vc)
files <- c("NRAS.AML.bam","NRAS.Control.bam")
bamFiles <- file.path( system.file("extdata", package = "h5vcData"), files)
chrom = "1"
startpos <- 115247090
endpos <- 115259515
batchTallies( batchTallyParam(bamFiles, chrom, startpos, endpos) )

## End(Not run)
```

---

callVariants *Variant calling*

Description

These functions implement various attempts at variant calling.

Usage

```
callVariantsPaired( data, sampledata, cl = vcConfParams() )
callDeletionsPaired( data, sampledata, cl = vcConfParams() )

vcConfParams(
  minStrandCov = 5,
  maxStrandCov = 200,
  minStrandAltSupport = 2,
  maxStrandAltSupportControl = 0,
  minStrandDelSupport = 2,
  maxStrandDelSupportControl = 0,
  minStrandCovControl = 5,
  maxStrandCovControl = 200,
  bases = 5:8,
  returnDataPoints = FALSE,
  annotateWithBackground = FALSE,
  mergeCalls = FALSE,
  mergeAggregator = mean,
  pValueAggregator = max
)
```

Arguments

data	A list with elements Counts (a 4d integer array of size [1:12, 1:2, 1:k, 1:n]), Coverage (a 3d integer array of size [1:2, 1:k, 1:n]), Deletions (a 3d integer array of size [1:2, 1:k, 1:n]), Reference (a 1d integer vector of size [1:n]) – see Details.
sampledata	A data.frame with k rows (one for each sample) and columns Type, Column and (SampleGroup or Patient). The tally file should contain this information as a group attribute, see getSampleData for an example.
cl	A list with parameters used by the variant calling functions. Such a list can be produced, for instance, by a call to vcConfParams.
minStrandCov	Minimum coverage per strand in the case sample.
maxStrandCov	Maximum coverage per strand in the case sample.
minStrandCovControl	Minimum coverage per strand in the control sample.
maxStrandCovControl	Maximum coverage per strand in the control sample.
minStrandAltSupport	Minimum support for the alternative allele per strand in the case sample. This should be 1 or higher.
maxStrandAltSupportControl	Maximum support for the alternative allele per strand in the control sample. This should usually be 0.
minStrandDelSupport	Minimum support for the deletion per strand in the case sample. This should be 1 or higher.
maxStrandDelSupportControl	Maximum support for the deletion per strand in the control sample. This should usually be 0.
bases	Indices for subsetting in the bases dimension of the Counts array, 5:8 extracts only those calls made in the middle one of the sequencing cycle bins.
returnDataPoints	Boolean flag to specify that a GRanges object with the variant calls should be returned. If returnDataPoints == FALSE only the variant positions are returned.
annotateWithBackground	Boolean flag to specify that the background mismatch / deletion frequency estimated from all control samples in the cohort should be added to the output. A simple binomial test will be performed as well. Only usefull if returnDataPoints == TRUE
mergeCalls	Boolean flag to specify that adjacent calls should be merged where appropriate (used by callDeletionsPaired). Only usefull applied if returnDataPoints == TRUE
mergeAggregator	Aggregator function for merging adjacent calls, defaults to mean, which means that a deletion larger than 1bp will be annotated with the means of the counts and coverages

pValueAggregator
: Aggregator function for combining the p-values of adjacent calls when merging, defaults to max. Is only applied if annotateWithBackground == TRUE

Details

data is a list of datasets which has to at least contain the Counts and Coverages for variant calling respectively Deletions for deletion calling. This list will usually be generated by a call to the h5dapply function in which the tally file, chromosome, datasets and regions within the datasets would be specified. See ?h5dapply for specifics.

vcConfParams is a helper function that builds a set of variant calling parameters as a list. This list is provided to the calling functions e.g. callVariantsPaired and influences their behavior.

callVariantsPaired implements a simple pairwise variant callign approach applying the filters specified in cl, and might additionally computes an estimate of the background mismatch rate (the mean mismatch rate of all samples labeled as 'Control' in the sampledata and annotate the calls with p-values for the binom.test of the observed mismatch counts and coverage at each of the samples labeled as 'Case'.

callDeletionsPaired implements an essential identical approach as callVariantsPaired but works on deletion counts per genomic position instead of mismatches.

Value

The result is either a list of positions with SNVs / deletions or a data.frame containing the calls themselves which might contain annotations. Adjacent calls might be merged and calls might be annotated with p-values depending on configuration parameters.

When the configuration parameter returnDataPoints is FALSE the functions return the positions of potential variants as a list containing one integer vector of positions for each sample, if no positions were found for a sample the list will contain NULL instead. In the case of returnDatapoints == TRUE the functions return either NULL if no poisitions were found or a data.frame with the following slots:

Chrom
: The chromosome the potential variant / deletion is on

Start
: The starting position of the variant / deletion

End
: The end position of the variant / deletions (equal to Start for SNVs and single basepair deletions)

Sample
: The Case sample in which the variant was observed

altAllele
: The alternate allele for SNVs (skipped for deletions, would be "-")

refAllele
: The reference allele for SNVs (skipped for deletions since the tally file might not contain all the information necessary to extract it)

caseCountFwd
: Support for the variant in the Case sample on the forward strand

caseCountRev
: Support for the variant in the Case sample on the reverse strand

caseCoverageFwd
: Coverage of the variant position in the Case sample on the forward strand

caseCoverageRev
: Coverage of the variant position in the Case sample on the reverse strand

controlCountFwd
: Support for the variant in the Control sample on the forward strand

controlCountRev
: Support for the variant in the Control sample on the reverse strand

controlCoverageFwd
: Coverage of the variant position in the Control sample on the forward strand

controlCoverageRev
: Coverage of the variant position in the Control sample on the reverse strand

If the annotateWithBackground option is set the following extra columns are returned

backgroundFrequencyFwd
: The averaged frequency of mismatches / deletions at the position of all samples of type Control on the forward strand

backgroundFrequencyRev
: The averaged frequency of mismatches / deletions at the position of all samples of type Control on the reverse strand

pValueFwd
: The p.value of the test binom.test(caseCountFwd, caseCoverageFwd, p = backgroundFrequencyF

pValueRev
: The p.value of the test binom.test(caseCountRev, caseCoverageRev, p = backgroundFrequencyF

The function callDeletionsPaired merges adjacent single-base deletion calls if the option mergeCalls is set to TRUE, in that case the counts and coverages (e.g. caseCountFwd) are aggregated using the function supplied in the mergeAggregator option of the configuration list (defaults to mean) and the p-values pValueFwd and pValueFwd (if annotateWithBackground is TRUE), are aggregated using the function supplied in the pValueAggregator option (defaults to max).

Author(s)

Paul Pyl

Examples

```
library(h5vc) # loading library
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
position <- 29979629
windowsize <- 1000
vars <- h5dapply( # Calling Variants
  filename = tallyFile,
  group = "/ExampleStudy/16",
  blocksize = 500,
  FUN = callVariantsPaired,
  sampledata = sampleData,
  cl = vcConfParams(returnDataPoints=TRUE),
  names = c("Coverages", "Counts", "Reference"),
  range = c(position - windowsize, position + windowsize)
)
vars <- do.call( rbind, vars ) # merge the results from all blocks by row
vars # We did find a variant
dels <- h5dapply( # Calling Deletions
```

```
  filename = tallyFile,
  group = "/ExampleStudy/16",
  blocksize = 500,
  FUN = callDeletionsPaired,
  sampledata = sampleData,
  cl = vcConfParams(returnDataPoints=TRUE),
  names = c("Coverages", "Deletions", "Reference"),
  range = c(position - windowsize, position + windowsize)
)
dels <- do.call( rbind, dels ) # merge the results from all blocks by row
dels # unfortunately this example dataset does not contain a deletion here
```

`callVariantsFisher` *Paired variant calling using fisher tests*

Description

This function implements a simple paired variant calling strategy based on the fisher test

Usage

```
callVariantsPairedFisher(data, sampledata, pValCutOff = 0.05, minCoverage = 5, mergeDels = TRUE, mergeA
```

Arguments

`data`	A `list` with elements `Counts` (a 4d `integer` array of size [1:12, 1:2, 1:k, 1:n]), `Coverage` (a 3d `integer` array of size [1:2, 1:k, 1:n]), `Reference` (a 1d `integer` vector of size [1:n]) – see Details.
`sampledata`	A `data.frame` with `k` rows (one for each sample) and columns `Type`, `Column` and (`Group` or `Patient`). The tally file should contain this information as a group attribute, see `getSampleData` for an example.
`pValCutOff`	Maximum allowed p-Value for the fisher test on contingency matrix `matrix(c(caseCounts, caseCovera`
`minCoverage`	Required coverage in both sample for a call to be made
`mergeDels`	Boolean flag specifying whether adjacent deletions should be merged
`mergeAggregator`	Which function to use for aggregating the values associated with adjacent deletions that are being merged

Details

`data` is a list which has to at least contain the `Counts`, `Coverages` and `Reference` datasets. This list will usually be generated by a call to the `h5dapply` function in which the tally file, chromosome, datasets and regions within the datasets would be specified. See `h5dapply` for specifics.

`callVariantsPairedFisher` implements a simple pairwise variant callign approach based on using the `fisher.test` on the following contingency matrix:

caseSupport	caseCoverage - caseSupport
conttrolSupport	controlCoverage - controlSupport

The results are filtered by `pValCutOff` and `minCoverage`.

Value

The return value is a `data.frame` with the following slots:

`Chrom`	The chromosome the potential variant is on
`Start`	The starting position of the variant
`End`	The end position of the variant
`Sample`	The `Case` sample in which the variant was observed
`refAllele`	The reference allele
`altAllele`	The alternate allele
`caseCount`	Support for the variant in the `Case` sample
`caseCoverage`	Coverage of the variant position in the `Case` sample
`controlCount`	Support for the variant in the `Control` sample
`controlCoverage`	Coverage of the variant position in the `Control` sample
`pValue`	The `p.value` of the `fisher.test`

Author(s)

Paul Pyl

Examples

```
library(h5vc) # loading library
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
position <- 29979629
windowsize <- 2000
vars <- h5dapply( # Calling Variants
  filename = tallyFile,
  group = "/ExampleStudy/16",
  blocksize = 1000,
  FUN = callVariantsPairedFisher,
  sampledata = sampleData,
  pValCutOff = 0.1,
  names = c("Coverages", "Counts", "Reference"),
  range = c(position - windowsize, position + windowsize),
  verbose = TRUE
)
vars <- do.call(rbind, vars)
vars
```

callVariantsSinle *Single sample variant calling*

Description

A simple single sample variant calling function (calling SNVs and deletions)

Usage

```
callVariantsSingle( data, sampledata, samples = sampledata$Sample, errorRate = 0.001, minSupport = 2, m
```

Arguments

data	A list with elements Counts (a 4d integer array of size [1:12, 1:2, 1:k, 1:n]), Coverage (a 3d integer array of size [1:2, 1:k, 1:n]), Deletions (a 3d integer array of size [1:2, 1:k, 1:n]), Reference (a 1d integer vector of size [1:n]) – see Details.
sampledata	A data.frame with k rows (one for each sample) and columns Column and (Sample. The tally file should contain this information as a group attribute, see getSampleData for an example.
samples	The samples on which variants should be called, by default all samples specified in sampledata are used
errorRate	The expected error rate of the sequencing technology that was used, for illumina this should be 1/1000
minSupport	minimal support required for a position to be considered variant
minAF	minimal allelic frequency for an allele at a position to be cosidered a variant
minStrandSupport	minimal per-strand support for a position to be considered variant
mergeDels	Boolean flag to specify that adjacent deletion calls should be merged
aggregator	Aggregator function for merging statistics of adjacent deletion calls, defaults to mean, which means that a deletion larger than 1bp will be annotated with the means of the counts and coverages etc.

Details

data is a list of datasets which has to at least contain the Counts and Coverages for variant calling respectively Deletions for deletion calling (if Deletions is not present no deletion calls will be made). This list will usually be generated by a call to the h5dapply function in which the tally file, chromosome, datasets and regions within the datasets would be specified. See h5dapply for specifics.

callVariantsSingle implements a simple single sample variant callign approach for SNVs and deletions (if Deletions is a dataset present in the data parameter. The function applies three essential filters to the provided data, requiring:

- minSupport total support for the variant at the position - minStrandSupport support for the variant on each strand - an allele freqeuncy of at least minAF (for pure diploid samples this can be set relatively high, e.g. 0.3, for calling potentially homozygous variants a value of 0.8 or higher might be used)

Calls are annotated with the p-Value of a binom.test of the present support and coverage given the error rate provided in the errorRate parameter, no filtering is done on this annotation.

Adjacent deletion calls are merged based in the value of the mergeDels parameter and their statistics are aggregated with the function supplied in the aggregator parameter.

Value

This function returns a data.frame containing annotated calls with the following slots:

Chrom	The chromosome the potential variant / deletion is on
Start	The starting position of the variant / deletion
End	The end position of the variant / deletions (equal to Start for SNVs and single basepair deletions)
Sample	The sample in which the variant was called
altAllele	The alternate allele for SNVs (deletions will have a "-" in that slot)
refAllele	The reference allele for SNVs (deletions will have the deleted sequence here as extracted from the Reference dataset, if the tally file contains a sparse representation of the reference, i.e. only positions with mismatches show a reference value the missing values are substituted with "N"'s. It is strongly suggested to write the whole reference into the tally file prior to deletion calling - see writeReference for details)
SupFwd	Support for the variant in the sample on the forward strand
SupRev	Support for the variant in the sample on the reverse strand
CovFwd	Coverage of the variant position in the sample on the forward strand
CovRev	Coverage of the variant position in the sample on the reverse strand
AF_Fwd	Allele frequency of the variant in the sample on the forward strand
AF_Rev	Allele frequency of the variant in the sample on the reverse strand
Support	Total Support of the variant - i.e. SupFwd + SupRev
Coverage	Total Coverage of the variant position - i.e. CovFwd + CovRev
AF	Total allele frequency of the variant, i.e. Support / Coverage
fBackground	Background frequency of the variant in all samples but the one the variant is called in
pErrorFwd	Probablity of the observed support and coverage given the error rate on the forward strand
pErrorRev	Probablity of the observed support and coverage given the error rate on the reverse strand
pError	Probablity of the observed support and coverage given the error rate on both strands combined
pStrand	p-Value of a fisher.test on the contingency matrix matrix(c(CovFwd,CovRev,SupFwd,SupRev), nro at this position - low values could indicate strand bias

Author(s)

Paul Pyl

Examples

```
library(h5vc) # loading library
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
position <- 29979629
windowsize <- 1000
vars <- h5dapply( # Calling Variants
  filename = tallyFile,
  group = "/ExampleStudy/16",
  blocksize = 500,
  FUN = callVariantsSingle,
  sampledata = sampleData,
  names = c("Coverages", "Counts", "Reference", "Deletions"),
  range = c(position - windowsize, position + windowsize)
)
vars <- do.call( rbind, vars ) # merge the results from all blocks by row
vars # We did find a variant
```

Coverage *Coverage analysis*

Description

Functions to do analyses based on coverage

Usage

```
binnedCoverage( data, sampledata, gccount = FALSE )
```

Arguments

data	A list with element Coverage (a 3d integer array of size [1:2, 1:k, 1:n])
sampledata	A data.frame with k rows (one for each sample) and columns Type, Column and (SampleGroup or Patient). The tally file should contain this information as a group attribute, see getSampleData for an example.
gccount	Boolean flag to specify whether the gc count of the bin should be reported as well, Reference must be a slot in the data object

Details

Explanations:

This computes the per sample coverage in a given bin (determined by the width of data). This feature is not implemented yet!

Value

Returns a `data.frame` with columns containing the coverage with the current bin for all samples provided in `sampledata`. The binsize is determined by the `blocksize` argument given to `h5dapply` when this function is run directly on a tally file.

Author(s)

Paul Pyl

Examples

```
# loading library and example data
library(h5vc)
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
sampleData <- getSampleData( tallyFile, "/ExampleStudy/22" )
data <- h5dapply( # extractting coverage binned at 1000 bases
  filename = tallyFile,
  group = "/ExampleStudy/22",
  blocksize = 1000,
  FUN = binnedCoverage,
  sampledata = sampleData,
  gccount = TRUE,
  names = c( "Coverages", "Reference" ),
  range = c(38900000,39000000)
)
data <- do.call(rbind, data)
rownames(data) <- NULL
head(data)
```

`geom_h5vc` *geom_h5vc*

Description

Plotting function that returns a `ggplot2` layer representing the specified dataset for the specified samples in the region `[positon - windowsize, position + windowsize]`.

Usage

```
geom_h5vc( data, sampledata, samples=sampledata$Sample, windowsize, position, dataset, ... )
```

Arguments

`data`	The data to be plotted. Returned by `h5dapply`. Must be centered on `position`, extend by `windowsize` in each direction and contain a slot named like the `dataset` argument
`sampledata`	The sampledata for the cohort represented by `data`. Returned by `getSampleData`

samples	A character vector listing the names of samples to be plotted, defaults to all samples as described in sampledata
windowsize	Size of the window in which to plot on each side. The total interval that is plotted will be [position-windowsize,position+windowsize]
position	The position at which the plot shall be centered
dataset	The slot in the data argument that should be plotted
...	Paramteters to be passed to the internally used geom_rect, see geom_rect for details

Details

Creates a ggplot layer centered on position using the specified dataset from list data, annotating it with sample information provided in the data.frame sampledata and showing all samples listed in sample. The resulting plot uses ggplot2's geom_rect to draw boxes representing the values from dataset. The x-axis is the position and will span the interval [positon - windowsize, position + windowsize]. The x-axis is centered at 0 and additional layers to be added to the plot should be centered at 0 also.

Ths function allows for fast creation of overview plots similar to mismatchPlot (without the stacking of tracks). The example below shows how one can create a plot showing the coverage and number of mismatches per position (but not the alternative allele) for a given region.

Value

A ggplot layer object containing the plot of the specified dataset, this can be used like any other ggplot layer, i.e. it may be added to another plot.

Author(s)

Paul Pyl

Examples

```
# loading library and example data
library(h5vc)
library(ggplot2)
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
position <- 29979629
windowsize <- 30
samples <- sampleData$Sample[sampleData$Patient == "Patient8"]
data <- h5dapply(
  filename = tallyFile,
  group = "/ExampleStudy/16",
 blocksize = windowsize * 3, #choose blocksize larger than range so that all needed data is collected as one block
  names = c("Coverages", "Counts", "Deletions"),
  range = c(position - windowsize, position + windowsize)
)[[1]]
# Summing up all mismatches irrespective of the alternative allele
data$CountsAggregate = colSums(data$Counts)
# Simple overview plot showing number of mismatches per position
```

```
p <- ggplot() +
geom_h5vc( data=data, sampledata=sampleData, windowsize = 35, position = 500, dataset = "Coverages", fill = "gray'
geom_h5vc( data=data, sampledata=sampleData, windowsize = 35, position = 500, dataset = "CountsAggregate", fill =
 facet_wrap( ~ Sample, ncol = 2 )
 print(p)
```

---

getSampleData *Reading and writing sample data from / to a tally file*

Description

These functions allow reading and writing of sample data to the HDF5-based tally files. The sample data is stored as group attribute.

Usage

```
getSampleData( filename, group )
setSampleData( filename, group, sampleData, largeAttributes = FALSE, stringSize = 64 )
```

Arguments

filename	The name of a tally file
group	The name of a group within that tally file, e.g. /ExampleStudy/22
sampleData	A data.frame with k rows (one for each sample) and columns Type, Column and (SampleGroup or Patient. Additional column will be added as well but are not required.)
largeAttributes	HDF5 limits the size of attributes to 64KB, if you have many samples setting this flag will write the attributes in a separate dataset instead. getSampleData is aware of this and automatically chooses the dataset-stored attributes if they are present
stringSize	Maximum length for string attributes (number of characters) - default of 64 characters should be fine for most cases; This has to be specified since we do not support variable length strings as of now.

Details

The returned data.frame contains information about the sample ids, sample columns in the sample dimension of the dataset. The type of sample must be one of c("Case","Control") to be used with the provided SNV calling function. Additional relevant per-sample information may be stored here.

Note that the following columns are required in the sample data where the rows represent samples in the cohort:

Sample: the sample id of the corresponding sample

Column: the index within the genomic position dimension of the corresponding sample, be aware that getSampleData and setSampleData automatically add / remove 1 from this value since internally the tally files store the dimension 0-based whereas within R we count 1-based.

Patient the patient id of the corresponding sample

Type the type of sample

Value

sampledata	A data.frame with k rows (one for each sample) and columns Type, Column and (SampleGroup or Patient).

Author(s)

Paul Pyl

Examples

```
# loading library and example data
library(h5vc)
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
sampleData
# modify  the sample data
sampleData$AnotherColumn <- paste( sampleData$Patient, "Modified" )
# write to tallyFile
setSampleData( tallyFile, "/ExampleStudy/16", sampleData )
# re-load and check if it worked
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
sampleData
```

---

h5dapply *h5dapply*

Description

This is the central function of the h5vc package, allows an apply operation along common dimensions of datasets in a tally file.

Usage

```
h5dapply( filename, group, blocksize, FUN = function(x) x, ... , names, dims, range, samples = NULL, samp
```

Arguments

filename	The name of a tally file to process
group	The name of a group in that tally file
blocksize	The size of the blocks in which to process the data (integer)

FUN	The function to apply to each block, defaults to `function(x) x`, which returns the data as is (a list of arrays)
...	Further parameters to be handed over to FUN
names	The names of the datasets to extract, e.g. `c("Counts","Coverages")` - optional (defaults to all datasets)
dims	The dimension to apply along for each dataset in the same order as names, these should correspond to compatible dimensions between the datsets. - optional (defaults to the genomic position dimension)
range	The range along the specified dimensions which should be processed, this allows for limiting the apply to a specific region or set of samples, etc. - optional (defaults to the whole chromosome)
samples	Character vector of sample names - must match contents of sampleData stored in the tallyFile
sampleDimMap	A list mapping dataset names to their respective sample dimensions - default provides values for "Counts", "Coverages", "Deletions" and "Reference"
verbose	Boolean flag that controls the amount of messages being printed by h5dapply
BPPARAM	BPPARAM object to be passed to the bplapply call used to apply FUN to the blocks - see BiocParallel documentation for details; if this is NULL a normal lapply will be used instead of bplapply.

Details

This function applys parameter FUN to blocks along a specified axis within the tally file, group and specified datasets. It creates a list of arrays (one for each dataset) and processes that list with the function FUN.

This is by far the most essential and powerful function within this package since it allows the user to execute their own analysis functions on the tallies stored within the HDF5 tally file.

The supplied function FUN must have a parameter data or ... (the former is the expected behaviour), which will be supplied to FUN from h5dapply for each block. This structure is a list with one slot for each dataset specified in the names argument to h5dapply containing the array corresponding to the current block in the given dataset. Furthemore the slot h5dapplyInfo is reserved and contains another list with the following content:

Blockstart is an integer specifying the starting position of the current block (in the dimension specified by the dims argument to h5dapply)

Blockend is an integer specifying the end position of the current block (in the dimension specified by the dims argument to h5dapply)

Datasets Contains a data.frame as it is returned by h5ls listing all datasets present in the other slots of data with their group, name, dimensions, number of dimensions (DimCount) and the dimension that is used for splitting into blocks (PosDim)

Group contains the name of the group as specified by the group argument to h5dapply

Value

A list with one entry per block, which is the result of applying FUN to the datasets specified in the parameter names within the block.

Author(s)

Paul Pyl

Examples

```
# loading library and example data
library(h5vc)
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
# check the available samples and sampleData
print(sampleData)
data <- h5dapply( #extracting coverage using h5dapply
  filename = tallyFile,
  group = "/ExampleStudy/16",
  blocksize = 1000,
  FUN = function(x) rowSums(x$Coverages),
  names = c( "Coverages" ),
  range = c(29000000,29010000),
  verbose = TRUE
  )
coverages <- do.call( rbind, data )
colnames(coverages) <- sampleData$Sample[order(sampleData$Column)]
coverages
#Subsetting by Sample
sampleData <- sampleData[sampleData$Patient == "Patient5",]
data <- h5dapply( #extracting coverage using h5dapply
  filename = tallyFile,
  group = "/ExampleStudy/16",
  blocksize = 1000,
  FUN = function(x) rowSums(x$Coverages),
  names = c( "Coverages" ),
  range = c(29000000,29010000),
  samples = sampleData$Sample,
  verbose = TRUE
  )
coverages <- do.call( rbind, data )
colnames(coverages) <- sampleData$Sample[order(sampleData$Column)]
coverages
```

---

h5readBlock *h5readBlock*

Description

A simple access function for extracting a single block of data from a tally file, use h5dapply for applying functions on multiple blocks / extracting multiple blocks form a tally file.

Usage

```
h5readBlock( filename, group, names, dims, range, samples = NULL, sampleDimMap = .sampleDimMap, verbose
```

Arguments

filename	The name of a tally file to process
group	The name of a group in that tally file
names	The names of the datasets to extract, e.g. c("Counts","Coverages") - optional (defaults to all datasets)
dims	The dimension in which the block shall be extracted for each dataset in the same order as names, these should correspond to compatible dimensions between the datsets. - optional (defaults to the genomic position dimension)
range	The range along the specified dimensions which should be extracted
samples	Character vector of sample names - must match contents of sampleData stored in the tallyFile
sampleDimMap	A list mapping dataset names to their respective sample dimensions - default provides values for "Counts", "Coverages", "Deletions" and "Reference"
verbose	Boolean flag that controls the amount of messages being printed by h5dapply

Details

This function extracts a block along the dimensions specified in dims (default: genomic position) from the datasets specified in names and returns it. The block is defined by the parameter range.

The function returns a list with one slot for each dataset specified in the names argument to containing the array corresponding to the specified block in the given dataset. Furthemore the slot h5dapplyInfo is reserved and contains another list with the following content:

Blockstart is an integer specifying the starting position of the current block (in the dimension specified by the dims argument to h5dapply)

Blockend is an integer specifying the end position of the current block (in the dimension specified by the dims argument to h5dapply)

Datasets Contains a data.frame as it is returned by h5ls listing all datasets present in the other slots of data with their group, name, dimensions, number of dimensions (DimCount) and the dimension that is used for splitting into blocks (PosDim)

Group contains the name of the group as specified by the group argument to h5dapply

Value

A list with one entry per dataset and an additional slot h5dapplyInfo containing auxiliary information.

Author(s)

Paul Pyl

Examples

```
library(h5vc) # loading the library
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
data <- h5readBlock( #extracting coverage, deletions and reference using h5dreadBlock
  filename = tallyFile,
  group = "/ExampleStudy/16",
  names = c( "Coverages", "Deletions", "Reference" ),
  range = c(29000000,29010000),
  verbose = TRUE
)
str(data)
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
#Subsetting by Sample
sampleData <- sampleData[sampleData$Patient == "Patient8",]
data <- h5readBlock( #extracting coverage, deletions and reference using h5dreadBlock
  filename = tallyFile,
  group = "/ExampleStudy/16",
  names = c( "Coverages", "Deletions", "Reference" ),
  range = c(29000000,29010000),
  samples = sampleData$Sample,
  verbose = TRUE
)
str(data)
```

---

helpers *helper functions*

Description

These functions are helpers for dealing with tally data stored in HDF5 files.

Usage

```
formatGenomicPosition( x, unit = "Mb", divisor = 1000000, digits = 3,
nsmall = 1 )
encodeDNAString( ds )
defineBlocks( start, stop, blocksize )
getChromSize( tallyFile, group, dataset = "Reference", posDim = 1 )
```

Arguments

x	Numerical genomic position
unit	Which unit to convert the position to
divisor	divisor corresponding to the unit, i.e. 'Mb' -> 1e6, 'Kb' -> 1e3
digits	number of digits to keep
nsmall	nsmall parameter to the format function

ds	A DNAString object to be encoded in the HDF5 tally file specific encoding of nucleotides.
start	first position
stop	last position
blocksize	size of blocks
tallyFile	Tally file to work on
group	Group within tallyFile that we want to find the chromosome size for
dataset	Datset to extract chromosome size from - default is "Reference"
posDim	Which dimension of the dataset describes the genomic position

Details

formatGenomicPosition: Helps formatting genomic positions for annotating axes in mismatch plots etc.

encodeDNAString: This translates a DNAString object into a comaptible encoding that can be written to a HDF5 based tally file in the Reference dataset. Since the Python script for generating tallies only sets the Reference dataset in positions where mismatches exists updating the Reference dataset becomes necessary if one would like to perform analysis involving sequence context (GC-bias, mutationSpectrum, etc.)

defineBlocks: This function returns a data.frame with the columns Start and End for blocks of size blocksize spanning the interval [start, stop].

getChromSize: This function is a helper to quickly look-up the chromosome size of a given group and tally file.

Value

formatGenomicPosition: formatted genomic position, e.g. "123.4 Mb"

encodeDNAString: A numeric vector encoding the nucleotide sequence provided in ds according to the scheme c("A"=0,"C"=1,"G"=2,"T"=3).

defineBlocks: A data.frame with the columns Start and End for blocks of size blocksize spanning the interval [start, stop].

getChromSize: Returns a numeric that is the size of the chromosome.

Author(s)

Paul Pyl

Examples

```
formatGenomicPosition(123456789)
library(Biostrings)
lapply( DNAStringSet( c("simple"="ACGT", "movie"="GATTACA") ), encodeDNAString )
getChromSize( system.file("extdata", "example.tally.hfs5", package="h5vcData"), "/ExampleStudy/16" )
```

`mergeTallies` *Merging the prepared results from multiple bam file tallies into one block that can be written to the HDF5 tally file*

Description

This function merges a set of tallies that have been processed with `prepareForHDF5` into one block of data.

Usage

```
mergeTallies( tallies )
```

Arguments

`tallies`	A list of prepared talies, i.e. a list of lists with slots for the datasets "Counts", "Coverage", "Deletions" and "Reference" in each sub-list

Details

This function merges tallies from a set of bam files / samples, note that the order of samples in the sample column will be the same as the order of samples in the provided list, so ake sure this matches your sampledata.

Value

A list with slots containing the `Counts`,`Coverages`,`Deletions` and `Reference` datasets for the samples given in `tallies`. Each of the slots contains an array with the contents of the provided sub-lists merged along the "sample" axis. The `Reference` slot os filled from the first element of `tallies` and it is up to the user to make sure that the tallies provided for merging have compatible references.

Author(s)

Paul Pyl

Examples

```
library(h5vc)
files <- c("NRAS.AML.bam","NRAS.Control.bam")
bamFiles <- file.path( system.file("extdata", package = "h5vcData"), files)
chrom = "1"
startpos <- 115247090
endpos <- 115259515
theData <- lapply( bamFiles, function(bamf){ tallyBAM(bamf, chrom, startpos, endpos) } )
str(theData)
theMergedData <- mergeTallies( lapply( theData, prepareForHDF5 ) )
str(theMergedData)
```

`mergeTallyFiles` *Merging multiple tally files into one*

Description

Function to merge multiple tally files by genomic position (i.e. gluing samples together)

Usage

```
mergeTallyFiles( inputFiles, destFile, destGroup, blockSize = 1e6, sampleDims = c(), positionDims = c()
```

Arguments

`inputFiles`	A list mapping input file names to the groups within them from which the data shall be taken (e.g. "example.tally.hfs5" -> "/ExampleStudy/16")
`destFile`	Name of the file that should be created
`destGroup`	Group within `destFile` that will hold the merged data
`blockSize`	Size of the blocks in bases that the merging will be performed in
`sampleDims`	List mapping dataset names to their respective sample dimension, e.g. "Counts" -> 2 - has the standard datasets included by default
`positionDims`	List mapping dataset names to their respective position dimension, e.g. "Counts" -> 4 - has the standard datasets included by default

Details

This function merges tally data from a list of tally files into a new destination file.

Value

None – prints progress messages along the way.

Author(s)

Paul Pyl

Examples

```
## Not run:
mergeTallyFiles{ # merging a file to itself, i.e. "doubling" it
  list(
    "example.tally.hfs5" = "/ExampleStudy/16",
    "example.tally.hfs5" = "/ExampleStudy/16"
  ),
  "test.merge.hfs5",
  "/MergedStudy/16"}

## End(Not run)
```

mismatchPlot *mismatchPlot*

Description

Plotting function that returns a ggplot2 object representing the mismatches and coverages of the specified samples in the specified region.

Usage

```
mismatchPlot( data, sampledata, samples=sampledata$Sample, windowsize, position, plotReference = TRUE,
```

Arguments

data	The data to be plotted. Returned by h5dapply. Must be centered on position and extend by windowsize in each direction
sampledata	The sampledata for the cohort represented by data. Returned by getSampleData
samples	A character vector listing the names of samples to be plotted, defaults to all samples as described in sampledata
windowsize	Size of the window in which to plot on each side. The total interval that is plotted will be [position-windowsize,position+windowsize]
position	The position at which the plot shall be centered
plotReference	This boolean flag specifies if a reference track should be plotted, only takes effect if there is a slot named Reference in the data object passed to the function
refHeight	Height of the reference track in coverage units (default of 8 = reference track is as high as 8 reads coverage would be in the plot of a sample.)

Details

Creates a plot centered on position using the coverage and mismatch counts stored in data, annotating it with sample information provided in the data.frame sampledata and showing all samples listed in sample.

The plot has the genomic position on the x-axis (centered around position spanning windowsize bases up- and downstream). The y-axis encodes values where positive values are on the forward strand and negative values on the reverse. The coverage is shown in grey, deletions in purple and the mismatches in the colors specified in the legend. Note that for each possible mismatch there is an additional color for low-quality counts (coming from the first and last sequencing cycles), so e.g. C is filled dark red and C_lq light red.

Value

A ggplot object containing the mismatch plot, this can be used like any other ggplot object, i.e. additional layers and styles my be applied by simply adding them to the plot.

Author(s)

Paul Pyl

Examples

```
# loading library and example data
library(h5vc)
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
sampleData <- getSampleData( tallyFile, "/ExampleStudy/16" )
position <- 29979628
windowsize <- 30
samples <- sampleData$Sample[sampleData$Patient == "Patient8"]
data <- h5readBlock(
  filename = tallyFile,
  group = "/ExampleStudy/16",
  names = c("Coverages", "Counts", "Deletions", "Reference"),
  range = c(position - windowsize, position + windowsize)
)
p <- mismatchPlot(
  data = data,
  sampledata = sampleData,
  samples = samples,
  windowsize = windowsize,
  position = position
)
print(p)
```

---

mutationSpectra *Mutation spectrum analyses*

Description

These functions help in analyses of mutation spectra

Usage

```
mutationSpectrum( variantCalls, tallyFile, study, context = 1 )
```

Arguments

variantCalls	A data.frame object that can be the output of a call to a callVariantsPaired or callDeletionsPaired function. The following columns are required: - altAllele - refAllele - Sample - Start - End - Chrom
tallyFile	filename of a tally file matching the variant calls
study	the study id used in the tally file
context	An integer specifying the size of the context that should be considered (i.e. the length of the prefix and suffix of the variant call)

Details

This function takes a set of variant calls (SNVs/Deletions) and a tallyFile as well as a context size and tabulates the number of observed mutations stratified by type (refAllele->altAllele) and sequence context (i.e. the prefix and suffix of size `context` around the variant position in the genome)

`bases` serves to map character representations to numeric encoding of bases

`variantCalls` is an example dataset of variant calls created by running `callVariantsPaired` on the `example.tally.hfs5` file.

Value

A table listing the counts of mutations stratified by allele, sequence context and sample.

Author(s)

Paul Pyl

Examples

```
library(h5vc)
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
data( "example.variants", package = "h5vcData" )
head( mutationSpectrum( variantCalls, tallyFile, "/ExampleStudy" ) )
```

---

plotMutationSpectrum *Plotting a mutation spectrum*

Description

This function generates a mutation spectrum plot from a mutation spectrum returned by a call to `mutationSPectrum`

Usage

```
plotMutationSpectrum( ms, plotCounts = TRUE )
```

Arguments

`ms`	A mutation spectrum as returned by `mutationSpectrum`
`plotCounts`	Boolean flag specifying whether `ms` contains one row per variant (default) or already contains summarized counts per type of mutation

Details

The plot is inspired by the one shown in figure 1b of `Signatures of mutational processes in human cancer -- Alexand`

Value

A ggplot object containing the mutation spectrum plot

Author(s)

Paul Pyl

Examples

```
library(h5vc)
tallyFile <- system.file( "extdata", "example.tally.hfs5", package = "h5vcData" )
data( "example.variants", package = "h5vcData" )
plotMutationSpectrum( mutationSpectrum( variantCalls, tallyFile, "/ExampleStudy" ) )
```

---

prepareForHDF5 *Preparing the results of tallyBAM for writing to an HDF5 tally file*

Description

This function prepares the resulting array of a call to `tallyBAM` for writing to an HDF5 tally file.

Usage

```
prepareForHDF5( counts, reference = NULL, stratifyDeletions = FALSE )
```

Arguments

`counts`	An array as produced by a call to `tallyBAM`
`reference`	A DNAString object containing the reference sequence corresponding to the region that is described in the counts array – if this is `NULL` a consensus vote will be used to estimate the reference at any given position, this means you cannot detect variants with AF >= 0.5 anymore
`stratifyDeletions`	Boolean flag to specify if deletion counts should be stratified by sequencing cycle or not - default is `FALSE` - note that dataset in the tally file must have compatible dimensions.

Details

This function performs the neccessary transformation to the `array` returned by `tallyBAM` to be compatible with the HDF5 tally file data structure.

Value

A list with slots containing the `Counts`,`Coverages`,`Deletions` and `Reference` datasets for the given sample.

Author(s)

Paul Pyl

Examples

```
library(h5vc)
files <- c("NRAS.AML.bam","NRAS.Control.bam")
bamFiles <- file.path( system.file("extdata", package = "h5vcData"), files)
chrom = "1"
startpos <- 115247090
endpos <- 115259515
theData <- lapply( bamFiles, function(bamf){
  tallyBAM( file = bamf, chr = chrom, start = startpos, stop = endpos, ncycles = 10 )
})
theData <- lapply( theData, prepareForHDF5 )
str(theData)
```

---

prepareTallyFile *prepareTallyFile*

Description

Functions for preparing an HDF5 file for storing tally data and / or modifying an existing file

Usage

```
prepareTallyFile( filename, study, chrom, chromlength, nsamples, maxsamples = nsamples, chunkSize = 500
resizeCohort( filename, study, chrom, newNumberOfSamples, dimmap = .sampleDimMap, force = FALSE )
```

Arguments

filename	Filename of the HDF5 file that should store the tallies
study	Study identifier which will be used in structuring the file
chrom	Chromosome for which the structure should be generated
chromlength	The length of the chromosom, this will be the size of genomic position dimension
nsamples	Number of samples that will be stored in the file
maxsamples	Maximum Number of samples that can be stored in the file, this relatesto the maxdim property of HDF5 datasets, which is used to specify possible re-sizing of datasets after creation - see http://www.hdfgroup.org for details
chunkSize	The size of the chunks used in HDF5 storage, this is specified along the genomic position dimension, by default chunks will always be all data from all samples with the given width along the genomic position dimension
compressionLevel	Compression level to use in the HDF5 file, defaults to 9 (highest), use lower numbers to improve access time at the cost of disk space usage

newNumberOfSamples
: New cohort size, this must be smaller than the value of maxsamples that was provided when the file was created

dimmap
: A list mapping dataset names to the dimension in which the samples are stored (e.g. "Counts" -> 2)

force
: Boolean parameter that controls whether a shrinking operation (i.e. newNumberOfSamples is smaller than the current number of samples) should be performed or throw an error. Shrinking will result in data loss.

Details

prepareTallyFile prepares (and creates if neccessary) an HDF5 file for storing the datasets that are associated with a tally. It creates the required groups and datasets (filled with 0's). resizeCohortResizes the datasets to a new number of samples, this is limited by the value of maxsamples that was provided in the initial call to prepareTallyFile

Value

Returns TRUE on success

Author(s)

Paul Pyl

Examples

```
prepareTallyFile( file.path( tempdir(), "test.tally.hfs5" ), "SomeStudy", "ChromosomeB", 1e6, 20 )
```

---

tallyBAM *tallyBAM*

Description

Function for creating tallies from bam files.

Usage

```
tallyBAM(file, chr, start, stop, q=25, ncycles = 0, max.depth=1000000, verbose=FALSE, reference = NULL)
```

Arguments

file
: filename of the BAM file that should be tallies

chr
: Chromosome in which to tally

start
: First position of the tally

stop
: Last position of the tally

q
: quality cut-off for considering a base call

ncycles	number of sequencing cycles form the front and back of the read that should be considered unreliable
max.depth	only tally a position if there are less than this many reads overlapping it - can prevent long runtimes in unreliable regions
verbose	should additional information be printed
reference	DNAString object holding the reference sequence of the region being tallies, if this is NULL (the default) the raw tally is returned, otherwise prepareForHDF5 is called with the raw tally and the reference and the prepared tally is returned instead

Details

This function tallies nucleotides and deletion counts in the specified region of a given BAM file. The results can be processed with the prepareForHDF5 function.

This function was adapted from the bam2R function provided by the deepSNV package.

Value

An array object with dimensions [stop - start + 1, 15, 2] which represent positions times nucleotides (4 bases + deletions times three for early, middle and late sequencing cycles) times strands.

Author(s)

Paul Pyl

Examples

```
library(h5vc)
files <- c("NRAS.AML.bam","NRAS.Control.bam")
bamFiles <- file.path( system.file("extdata", package = "h5vcData"), files)
chrom = "1"
startpos <- 115247090
endpos <- 115259515
theData <- lapply( bamFiles, function(bamf){
  tallyBAM( file = bamf, chr = chrom, start = startpos, stop = endpos, ncycles = 10 )
})
str(theData)
print(theData[[1]][,,9491]) #position 9491 of the pileup
```

---

writeReference *Filling the Reference dataset in a tally file from a DNAString*

Description

Function to fill the Reference dataset of a tally file from a DNAString object

Usage

```
writeReference( tallyFile, group, dnastring, blocksize = 1000000, verbose = TRUE )
```

Arguments

tallyFile	filename of a tally file matching the variant calls
group	The group that the Reference dataset is located in
dnastring	A DNAString object containing the new reference sequence
blocksize	The size of blocks in which to process the reference (higher values imply higher memory consumption)
verbose	Boolean flag to specify if diagnostic messages should be printed

Details

This function takes a tally file, a location within it (the group argument) and a reference sequence as a DNAString object, encodes the reference in the appropriate way and writes it to the location in the tally file in blocks of size specified in blocksize. The reference will be written to a dataset with the path paste(group, "Reference", sep = "/") within the tally file. The dataset itself must exists and have the correct dimensions to hold the sequence specified in dnastring.

Value

Returns TRUE on success.

Author(s)

Paul Pyl

Examples

```
library(h5vc)
library(rhdf5)
library(Biostrings)
filename = file.path(tempdir(), "write.ref.test.hfs5")
prepareTallyFile(filename=filename,study="SomeStudy",chrom="Foo",chromlength=8,nsamples=1)
writeReference(filename, group = "/SomeStudy/Foo", dnastring = DNAString("GATTACCA"))
h5dump(filename)$SomeStudy$Foo$Reference
```

Index