Package 'h5vc'

April 5, 2014

Type Package

Title Managing alignment tallies using a hdf5 backend

Version 1.0.6

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

Imports rhdf5, ggplot2, reshape, bit64, IRanges, GenomicRanges,Biostrings

Suggests knitr, locfit, deepSNV, BSgenome.Hsapiens.UCSC.hg19, h5vcData

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

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 identival 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
```

`Coverage` *Coverage analysis*

Description

Functions to do analyses based on coverage

Usage

```
binnedCoverage( data, sampledata )
```

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.

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,
  names = c( "Coverages" ),
  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` *Extracting sample data from 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 )
```

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.)

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, verbose = FALSE)
```

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)
verbose	Boolean flag that controls the amount of messages being printed by h5dapply

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
```

`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 )
```

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.

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.)

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)`.

Author(s)

Paul Pyl

Examples

```
formatGenomicPosition(123456789)
library(Biostrings)
lapply( DNAStringSet( c("simple"="ACGT", "movie"="GATTACA") ), encodeDNAString )
```

`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 )
```

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

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 <- 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]]
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" ) )
```

Index