Package 'VariantTools'

March 26, 2013

Type Package

Title Tools for Working with Genetic Variants

Version 1.0.1

Author Michael Lawrence, Jeremiah Degenhardt, Robert Gentleman

Maintainer Michael Lawrence <michafla@gene.com>

Description Tools for Tools for detecting, filtering, calling,comparing and plotting variants.

Depends IRanges (>= 1.15.44), GenomicRanges (>= 1.9.52),VariantAnnotation (>= 1.3.20), methods

Imports IRanges, Rsamtools (>= 1.9.31), GenomicRanges, BiocGenerics,Biostrings, parallel, gmapR (>= 0.99.28), GenomicFeatures,VariantAnnotation, methods

Suggests RUnit, LungCancerLines (>= 0.0.6)

biocViews Genetics, GeneticVariability, HighThroughputSequencing

License Artistic-2.0

LazyLoad yes

R topics documented:

callSampleSpecificVariants 2
callVariants 3
qaVariants 5
tallyVariants 6
variantFilter 7
variantGR2Vcf 9

Index **11**

callSampleSpecificVariants
Call Sample-Specific Variants

Description

Calls sample-specific variants by comparing case and control variants from paired samples, starting from the BAM files or unfiltered tallies. For example, these variants would be considered somatic mutations in a tumor vs. normal comparison.

Usage

```
## S4 method for signature 'BamFile,BamFile'
callSampleSpecificVariants(case, control,
  tally.param, ...)
## S4 method for signature 'character,character'
callSampleSpecificVariants(case, control, ...)
## S4 method for signature 'GenomicRanges,GenomicRanges'
callSampleSpecificVariants(case,
  control, control.cov, qa.filters = VariantQAFilters(),
  calling.filters = VariantCallingFilters(), ...)
SampleSpecificVariantFilters(control, control.cov, calling.filters,
                             power = 0.8, p.value = 0.01)
```

Arguments

case	The BAM file for the case, or the raw tallies as output by tallyVariants.
control	The BAM file for the control, or the raw tallies as output by tallyVariants.
tally.param	Parameters controlling the variant tallying step, as typically constructed by VariantTallyParam.
qa.filters	Filters to use in the QA process, typically generated by VariantQAFilters.
calling.filters	Filters to use for the initial, single-sample calling against reference, typically constructed by VariantCallingFilters.
...	For a BAM file, arguments to pass down to the GenomicRanges method. For the GenomicRanges method, arguments to pass down to SampleSpecificVariantFilters, except for control.cov, control.called, control.raw and lr.filter.
control.cov	The coverage for the control sample.
power	The power cutoff, beneath which a variant will not be called case-specific, due to lack of power in control.
p.value	The binomial p-value cutoff for determining whether the control frequency is sufficiently extreme (low) compared to the case frequency. A p-value below this cutoff means that the variant will be called case-specific.

Details

For each sample, the variants are tallied (when the input is BAM), QA filtered (case only), called and determined to be sample-specific. The callSampleSpecificVariants function is fairly high-level, but it still allows the user to override the parameters and filters for each stage of the process. See VariantTallyParam, VariantQAFilters, VariantCallingFilters and SampleSpecificVariantFilters.

It is safest to pass a BAM file, so that the computations are consistent for both samples. The GenomicRanges method is provided mostly for optimization purposes, since tallying the variants over the entire genome is time-consuming. For small gene-size regions, performance should not be a concern.

This is the algorithm that determines whether a variant is specific to the case sample:

1. Filter out all case calls that were also called in control. The callSampleSpecificVariants function does **not** apply the QA filters when calling variants in control. This prevents a variant from being called specific to case merely due to questionable data in the control.
2. For the remaining case calls, calculate whether there was sufficient power in control under the likelihood ratio test, for a variant present at the p.lower frequency. If that is below the power cutoff, discard it.
3. For the remaining case calls, test whether the control frequency is sufficient extreme (low) compared to the case frequency, under the binomial model. The null hypothesis is that the frequencies are the same, so if the test p-value is above p.value, discard the variant. Otherwise, the variant is called case-specific.

Value

A tally GRanges with the case-specific variants (such as somatic mutations).

Author(s)

Michael Lawrence, Jeremiah Degenhardt

Examples

```
bams <- LungCancerLines::LungCancerBamFiles()
tally.param <- VariantTallyParam(gmapR::TP53Genome(),
                                 readlen = 100L,
                                 high_base_quality = 23L,
                                 which = gmapR::TP53Which())
callSampleSpecificVariants(bams$H1993, bams$H2073, tally.param)
```

---

callVariants *Call Variants*

Description

Calls variants from either a BAM file or a tally GRanges object. The variants are first filtered with qaVariants, and the remaining candidates are called using a binomial likelihood ratio test: P(D|p=p.lower) / P(D|p=p.error) > 1.

Usage

```
## S4 method for signature 'BamFile'
callVariants(x, tally.param,
                                 qa.filters = VariantQAFilters(),
                                 calling.filters = VariantCallingFilters(...),
                                 ...)
## S4 method for signature 'character'
callVariants(x, ...)
```

```
## S4 method for signature 'GenomicRanges'
callVariants(x,
    calling.filters = VariantCallingFilters(...),
    ...)
VariantCallingFilters(read.count = 2L, p.lower = 0.2, p.error = 1/1000,
                      use.high.qual = TRUE)
```

Arguments

x	Either a path to an indexed bam, a BamFile object, or a GRanges as returned by tallyVariants.
tally.param	Parameters controlling the variant tallying step, as typically constructed by VariantTallyParam.
qa.filters	Filters used in the QA step, see VariantQAFilters.
calling.filters	Filters used in the calling step, typically constructed with VariantCallingFilters, see arguments listed below.
	...Arguments for VariantCallingFilters, listed below.
read.count	Require at least this many reads with the alternate base. By default, this is a sanity check to prevent calling a variant at a position with a single read.
p.lower	The lower bound on the binomial probability for a true variant.
p.error	The binomial probability for a sequencing error (default is reasonable for Illumina data).
use.high.qual	Whether to use the high quality counts in the likelihood ratio test.
...	Arguments to pass to VariantCallingFilters.

Value

For callVariants, a GRanges of the called variants (the tallies that pass the QA and calling filters).

For VariantCallingFilters, a FilterRules object with the filters for calling the variants (presumably after the QA filters have been applied).

Author(s)

Michael Lawrence, Jeremiah Degenhardt

Examples

```
bams <- LungCancerLines::LungCancerBamFiles()
tally.param <- VariantTallyParam(gmapR::TP53Genome(),
                                 readlen = 100L,
                                 high_base_quality = 23L,
                                 which = gmapR::TP53Which())

## simple usage
variants <- callVariants(bams$H1993, tally.param)

## customize
qa.filters <- VariantQAFilters(fisher.strand.p.value = 1e-4)
calling.filters <- VariantCallingFilters(read.count = 3L)
callVariants(bams$H1993, tally.param, qa.filters, calling.filters)
```

qaVariants *QA Filtering of Variants*

Description

Filters a tally GRanges through a series of simple checks for strand and cycle (read position) biases.

Usage

```
qaVariants(x, qa.filters = VariantQAFilters(...), ...)
VariantQAFilters(cycle.count = 2L, fisher.strand.p.value = 1e-3)
```

Arguments

x	A tally GRanges as output by tallyVariants.
qa.filters	The filters used for the QA process, typically constructed with VariantQAFilters, see arguments below.
...	Arguments passed to VariantQAFilters, listed below.
cycle.count	Minimum number of unique cycles for the alternate base.
fisher.strand.p.value	p-value cutoff for the Fisher's Exact Test for strand bias (+/- counts, alt vs. ref). Any variants with p-values below this cutoff are discarded.

Details

There are currently three QA filters:

- Alternate base was read at a minimum (2) number of unique cycles. This avoids false positives from one aberrant cycle.
- Fisher's Exact Test for strand bias, using the +/- counts, alt vs. ref. If the null is rejected, the variant is discarded.
- If the tallies contain cycle bin counts, the variant must have at least one count in the middle bins (those not at the start or end). We trust the internal cycles more.

Prior to the QA checks, the variants are passed through a simple sanity filter that discards positions where reference has an N.

Value

For qaVariants, a tally GRanges of the variants that pass the QA checks.

For VariantQAFilters, a FilterRules object with the QA and sanity filters.

Author(s)

Michael Lawrence and Jeremiah Degenhardt

Examples

```
bams <- LungCancerLines::LungCancerBamFiles()
tally.param <- VariantTallyParam(gmapR::TP53Genome(),
                                 readlen = 100L,
                                 high_base_quality = 23L,
                                 which = gmapR::TP53Which())
tally.variants <- tallyVariants(bams$H1993, tally.param)
qaVariants(tally.variants, fisher.strand.p.value = 1e-4)
```

---

tallyVariants *Tally the positions in a BAM file*

Description

Tallies the bases, qualities and read positions for every genomic position in a BAM file. By default, this only returns the positions for which an alternate base has been detected. The typical usage is to pass a BAM file, the genome, the (fixed) readlen and (if the variant calling should consider quality) an appropriate high_base_quality cutoff. Passing a which argument allows computing on only a subregion of the genome.

Usage

```
## S4 method for signature 'BamFile'
tallyVariants(x, param = VariantTallyParam(...), ...,
                                 mc.cores = getOption("mc.cores", 2))
## S4 method for signature 'character'
tallyVariants(x, ...)
VariantTallyParam(genome, readlen = NA,
                  cycle_flank_width = 10L,
                  cycle_breaks = flankingCycleBreaks(readlen,
                  cycle_flank_width),
                  high_base_quality = 0L,
                  minimum_mapq = 13L,
                  variant_strand = 1L, ignore_query_Ns = TRUE,
                  ...)
```

Arguments

x	An indexed BAM file, either a path or a BamFile object.
param	The parameters for the tallying process, as a BamTallyParam, typically constructed with VariantTallyParam, see arguments below.
...	For tallyVariants, arguments to pass to VariantTallyParam, listed below. For VariantTallyParam, arguments to pass to BamTallyParam.
genome	The genome, either a GmapGenome or something coercible to one.
readlen, cycle_flank_width	If cycle_breaks is missing, these two arguments are used to generate a cycle_breaks for three bins, with the two outside bins having cycle_flank_width. If readlen is NA, cycle_breaks is not generated.
cycle_breaks	The breaks used for tabulating the cycles (read positions) at each position. If this information is included (not NULL), qaVariants will use it during filtering.

high_base_quality
: The cutoff for whether a base is counted as high quality. By default, callVariants will use the high quality counts in the likelihood ratio test. Note that bam_tally will shift your quality scores by 33 no matter what type they are. If Illumina (pre 1.8) this will result in a range of 31-71. If Sanger/Illumina1.8 this will result in a range of 0-40/41. The default counts all bases as high quality. We typically use 56 for old Illumina, 23 for Sanger/Illumina1.8.

minimum_mapq
: Minimum MAPQ of a read for it to be included in the tallies. This depend on the aligner; the default is reasonable for gsnap.

variant_strand
: On how many strands must an alternate base be detected for a position to be returned. Highly recommended to set this to at least 1 (otherwise, the result is huge and includes many uninteresting reference rows).

ignore_query_Ns
: Whether to ignore N calls in the reads. Usually, there is no reason to set this to FALSE. If it is FALSE, beware of low quality datasets returning enormous results.

mc.cores
: The number of cores to use when parallelizing over the chromosomes.

Value

For tallyVariants, the tally GRanges.

For VariantTallyParam, an object with parameters suitable for variant calling.

Author(s)

Michael Lawrence, Jeremiah Degenhardt

Examples

```
tally.param <- VariantTallyParam(gmapR::TP53Genome(),
                                 readlen = 100L,
                                 high_base_quality = 23L,
                                 which = gmapR::TP53Which())
bams <- LungCancerLines::LungCancerBamFiles()
raw.variants <- tallyVariants(bams$H1993, tally.param)
```

---

variantFilter

A function to call variants from nextgen sequening data.Works on a tally GRanges object as produced by tally2GR in gmapR package

---

Description

This function takes a GRanges object as generated by tally2GR in gmapR and filters the variants based on a set of options resulting in a set of 'called' variants.

Usage

```
variantFilter(granges, useQual = FALSE, pval = 0.05, readCount = 2,
cycleCount = 1, lrt = FALSE, lrt_p = 0.01, lr = TRUE, lower = 0.2,
error = 1/1000, fet_2x2 = TRUE, read_pos = NULL)
```

Arguments

granges	Variants GRanges object as produced by tally2GR in the gmapR package
useQual	Boolean flag to turn on or off filtering by base quality. If on, the LR or LRT will be evaluated on only the high-quality bases
readCount	minimum number of alternate bases needed to call a variant. The default value is 2
cycleCount	minimum number of cycles that the variant base must be seen in to call. The default value is 1.
pval	pval to be used in filtering during the Fisher's exact test
lrt	Boolean flag to choose whether to use the LRT. If this is set to true, lr must be set to false.
lrt_p	p-value threshold to use when the LRT method is used.
lr	Boolean flag to turn on the LR method.
lower	Used in the LR method this is the lowest frequency you with to have power to detect.
error	assumed error rate from the sequencing data (eg 1/1000)
read_pos	gives the name of the bin to filter on for the read position filter
fet_2x2	Boolean flag to turn on or off the Fisher's exact test of strand usage bias in the variants

Value

raw_granges	GRanges object of the raw variants. This is all positions with 1 or more non-reference bases in the bam file.
filtered_granges	GRanges object of the filtered (called) variants.
N_nucleotide_rejects	number of positions removed due to the non-ref base being an N
low_count_rejects	number of positions removed due to having too few non-reference bases (too few is defined as less than 2 by default)
frequency_rejects	number of positions removed due to either failing the LRT or having frequency below the threshold set in the LR
fisher_test_rejects_2by2	number of positions removed for failing the Fisher's exact test on the strand usage of the variant.
read_pos_rejects	number of positions removed for failing the read positional filter this will be 0 if read_pos is NULL

Author(s)

Jeremiah Degenhardt

variantGR2Vcf *Create a VCF for some variants*

Description

Creates a VCF object from a variant/tally GRanges. This can then be output to a file using writeVcf. The flavor of VCF is specific for calling variants, not genotypes; see below.

Usage

```
variantGR2Vcf(x, sample.id, project = NULL)
```

Arguments

x	The variant/tally GRanges.
sample.id	Unique ID for the sample in the VCF.
project	Description of the project/experiment; will be included in the VCF header.

Details

A variant GRanges has an element for every unique combination of position and alternate base. A VCF object, like the file format, has a row for every position, with multiple alternate alleles collapsed within the row. This is the fundamental difference between the two data structures. We feel that the GRanges is easier to manipulate for filtering tasks, while VCF is obviously necessary for communication with external databases and tools.

Normally, despite its name, VCF is used for communicating *genotype* calls. We are calling *variants*, not genotypes, so we have extended the format accordingly.

Here is the mapping in detail:

- The rowData is formed by dropping the metadata columns from the GRanges.
- The colData consists of a single column, "Samples", with a single row, set to 1 and named sample.id.
- The exptData has an element "header" with element "reference" set to the seqlevels(x) and element "samples" set to sample.id. This will also include the necessary metadata for describing our extensions to the format.
- The fixed table has the "REF" and "ALT" alleles, with "QUAL" and "FILTER" set to NA.
- The geno list has six matrix elements, all with a single column. The first is the mandatory "GT" element, the genotype, which we set to NA. Then there is "AR" (list matrix with the read count for each ALT), "RR" (integer matrix with the reference read count), "DP" (integer matrix with the total read count), "AAP" (list matrix of 0/1 flags for whether each ALT was present in the data), and "RAP" (integer matrix of 0/1 flags for whether the REF was present).

Value

A VCF object.

Author(s)

Michael Lawrence, Jeremiah Degenhardt

Examples

```
example(callVariants)
vcf <- variantGR2Vcf(variants, "H1993", "example")
## Not run:
writeVcf(vcf, "H1993.vcf", index = TRUE)

## End(Not run)
```

Index

BamTallyParam, *6*

callSampleSpecificVariants, 2
callSampleSpecificVariants,BamFile,BamFile-method
(callSampleSpecificVariants), 2
callSampleSpecificVariants,character,character-method
(callSampleSpecificVariants), 2
callSampleSpecificVariants,GenomicRanges,GenomicRanges-method
(callSampleSpecificVariants), 2
callVariants, 3, *7*
callVariants,BamFile-method
(callVariants), 3
callVariants,character-method
(callVariants), 3
callVariants,GenomicRanges-method
(callVariants), 3

FilterRules, *4, 5*

GmapGenome, *6*
gsnap, *7*

qaVariants, *3*, 5, *6*

SampleSpecificVariantFilters
(callSampleSpecificVariants), 2

tallyVariants, *2*, *4, 5*, 6
tallyVariants,BamFile-method
(tallyVariants), 6
tallyVariants,character-method
(tallyVariants), 6

VariantCallingFilters, *2*
VariantCallingFilters (callVariants), 3
variantFilter, 7
variantGR2Vcf, 9
VariantQAFilters, *2*, *4*
VariantQAFilters (qaVariants), 5
VariantTallyParam, *2*, *4*
VariantTallyParam (tallyVariants), 6
VCF, *9*

writeVcf, *9*