Package 'gmapR'

October 15, 2025

Maintainer Michael Lawrence <michafla@gene.com>

License Artistic-2.0

Title An R interface to the GMAP/GSNAP/GSTRUCT suite

Type Package

Author Cory Barr, Thomas Wu, Michael Lawrence

Description GSNAP and GMAP are a pair of tools to align short-read data written by Tom Wu. This package provides convenience methods to work with GMAP and GSNAP from within R. In addition, it provides methods to tally alignment results on a per-nucleotide basis using the bam_tally tool.

Version 1.50.0

Depends R (>= 2.15.0), methods, GenomeInfoDb (>= 1.1.3), GenomicRanges (>= 1.31.8), Rsamtools (>= 1.31.2)

Imports S4Vectors (>= 0.17.25), IRanges (>= 2.13.12), BiocGenerics (>= 0.25.1), rtracklayer (>= 1.39.7), GenomicFeatures (>= 1.31.3), Biostrings, VariantAnnotation (>= 1.25.11), tools, Biobase, BSgenome, GenomicAlignments (>= 1.15.6), BiocParallel, BiocIO

Suggests RUnit, BSgenome.Dmelanogaster.UCSC.dm3, BSgenome.Scerevisiae.UCSC.sacCer3, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg19.knownGene, BSgenome.Hsapiens.UCSC.hg19, LungCancerLines

Collate GmapBamReader-class.R GmapGenomeDirectory-class.R GmapGenome-class.R GmapSnpDirectory-class.R GmapSnps-class.R GmapParam-class.R GsnapParam-class.R GsnapOutput-class.R GmapOutput-class.R atoiindex-command.R iit-format.R BamTallyParam-class.R bam_tally-command.R cmetindex-command.R get-genome-command.R gmap-command.R gmap_build-command.R gsnap-command.R iit_store-command.R info.R snpindex-command.R system.R test_gmapR_package.R makeGmapGenomePackage.R TP53Genome.R utils.R asSystemCall.R

biocViews Alignment

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

git_branch RELEASE_3_21
git_last_commit 0b5ffd1
git_last_commit_date 2025-04-15
Repository Bioconductor 3.21
Date/Publication 2025-10-15

Contents

`BamTallyParam-class` *Class* "BamTallyParam"

Description

A `BamTallyParam` object stores parameters for `bam_tally`. The function of the same name serves as its constructor.

Usage

```
BamTallyParam(genome, which = GRanges(),
              desired_read_group = NULL,
              minimum_mapq = 0L,
              concordant_only = FALSE, unique_only = FALSE,
              primary_only = FALSE, ignore_duplicates = FALSE,
              min_depth = 0L, variant_strand = 0L, variant_pct = 0,
              ignore_query_Ns = FALSE,
              indels = FALSE, min_softclip = 0L, max_softclip = 0L,
              exon_iit = NULL, IIT_BPPARAM = NULL,
              xs = FALSE, read_pos = FALSE,
              min_base_quality = 0L, noncovered = FALSE, nm = FALSE)
```

Arguments

genome	A GmapGenome object, or something coercible to one.
which	A IntegerRangesList or something coercible to one that limits the tally to that range or set of ranges. By default, the entire genome is processed.
desired_read_group	The name of the read group to which to limit the tallying; if not NULL, must be a single, non-NA string.
minimum_mapq	Minimum mapping quality for a read to be counted at all.
concordant_only	Consider only what gnsap calls "concordant" alignments.
unique_only	Consider only the uniquly mapped reads.
primary_only	Consider only primary pairs.
ignore_duplicates	Whether to ignore the reads flagged as PCR/optical duplicates.
min_depth	The minimum number of reads overlapping a position for it to be counted.
variant_strand	The number of strands on which a variant must be seen for it to be counted. This means that a value of 0 will report reference alleles in addition to variants. A value of 1 will report only positions where a variant was seen on at least one strand, and 2 requires the variant be seen on both strands. Setting this to 1 is a good way to save resources.
variant_pct	The minimum alternate allele fraction for a variant to be reported for a strand.
ignore_query_Ns	Whether to ignore the N base pairs when counting. Can save a lot of resources when processing low quality data.
indels	Whether to return indel counts. The ref and alt columns in the returned VRanges conform to VCF conventions; i.e., the first base upstream is included. The range always spans the sequence in ref; so e.g. a deletion extends one nt upstream of the actual deleted sequence.
min_softclip, max_softclip	Minimum and maximum length of soft clips that are considered for counting. Soft-clipping is often useful (for GSNAP at least) during alignment, and it should be preserved in the output. However, soft clipping can preferentially occur in regions of discordance with the reference, and if those clipped regions are ignored during counting, the allele fraction is misestimated.
exon_iit	An object which indicates the exons to be used for tallying codons (a character value indicating an existing .iit file, a GRangesList of exons by gene or a TxDb object from which to make such a GRangesList) or NULL indicating no codon-level tallying should be done.
IIT_BPPARAM	A BiocParallelParam object to use when generating the iit file from an R object. Ignored if exon_iit is a character vector or NULL
xs	Whether to tabulate reads by XS tag, the aligner's best guess about the strand of transcription.
read_pos	Whether to tabulate by read position.

min_base_quality
: Minimum base quality cutoff. Calls of lower quality are not counted, except in the total raw depth.

noncovered
: Whether to report zero tallies, where there is no coverage.

nm
: Whether to tally by NM tag, the number of mismatches for a read.

See Also

bam_tally

---

bam_tally-methods *Per-position Alignment Summaries*

Description

Given a set of alignments, for each position in the genome output counts for the reference allele and all alternate alleles. Often used as a precursor to detecting variants. Indels will be supported soon.

Usage

```
## S4 method for signature 'BamFile'
bam_tally(x, param, ...)
## S4 method for signature 'character'
bam_tally(x, param, ...)
variantSummary(x, read_pos_breaks = NULL,
               keep_ref_rows = FALSE, read_length = NA_integer_,
               high_nm_score = NA_integer_)
```

Arguments

x
: a BamFile object or string path to a BAM file to read

param
: The BamTallyParam object with parameters for the tally operation.

read_pos_breaks
: The breaks, like those passed to cut for aggregating the per-read position counts. If NULL, no per-cycle counts are returned.

keep_ref_rows
: Whether to keep the rows describing only the reference calls, i.e., where ref and alt are the same. These are useful when one needs the reference counts even when there are no alts at that position.

read_length
: The expected read length. If the read length is NA, the MDFNE (median distance from nearest end) statistic will NOT be calculated.

high_nm_score
: The value at which an NM value is considered high.

...
: Arguments that override settings in param.

Value

The bam_tally function returns an opaque pointer to a C-level data structure with the class "TallyIIT". Currently, the only operation applicable to this object is variantSummary.

The variantSummary function returns a VRanges, with a range for each position that passed the filters. The depth columns correspond to the counts after quality filtering (except for indels, for which there is no quality filtering). The following elementMetadata columns are also present:

- n.read.pos — The number of unique read positions for the alt allele.
- n.read.pos.ref — The number of unique read positions for the ref allele.
- raw.count.total — The total number of reads at that position, including reference and all alternates.
- count.plus — The number of positive strand reads for the alternate allele, NA for the reference allele row.
- count.plus.ref — The number of positive strand reads for the reference allele.
- count.minus — The number of negative strand reads for the alternate allele, NA for the reference allele row.
- count.minus.ref — The number of negative strand reads for the reference allele.
- count.del.plus — The plus strand deletion count over the position.
- count.del.minus — The minus strand deletion count over the position.
- read.pos.mean — Mean read position for the alt allele.
- read.pos.mean.ref — Mean read position for the ref allele.
- read.pos.var — Variance in the read positions for the alt allele.
- read.pos.var.ref — Variance in the read positions for the ref allele.
- mdfne — Median distance from nearest end for the alt allele.
- mdfne.ref — Median distance from nearest end for the ref allele.
- count.high.nm — The number of alt reads with an NM value at or above the high_nm_score cutoff.
- count.high.nm.ref — The number of ref reads with an NM value at or above the high_nm_score cutoff.

If codon counting was enabled, there will be a column giving the codon strand: codon.strand.

If the xs parameter was TRUE, there will be four additional columns giving the counts by aligner-determined strand: count.xs.plus, count.xs.plus.ref, count.xs.minus, and count.xs.minus.ref.

An additional column is present for each bin formed by the read_pos_breaks parameter, with the read count for that bin.

Author(s)

Michael Lawrence

See Also

tallyVariants in the VariantTools package provides a high-level wrapper for this functionality.

---

cmetindex *Call the cmetindex command*

Description

Call the GMAP cmetindex command to build an index suitable for alignment of bisulfite-treated DNA, by allowing for C->T and G->A differences.

Usage

```
cmetindex(db, use_snps = NULL)
```

Arguments

db	The GmapGenome object
use_snps	A GmapSnps object for generating a SNP-tolerant index

Author(s)

Michael Lawrence

Examples

```
## Not run:
    library(BSgenome.Dmelanogaster.UCSC.dm3)
    flyGG <- GmapGenome(Dmelanogaster, create = TRUE)
    cmetindex(flyGG)

## End(Not run)
```

---

directory *Get the Path to the Location on Disk from a gmapR Class*

Description

Many objects in gmapR represent data stored on disk. The directory accessor will return this directory.

Usage

```
directory(x)
```

Arguments

x	A GmapGenome or GmapSnps object

Value

a character vector

GmapGenome-class *Class* "GmapGenome"

Description

The GmapGenome class represents a genome that has been indexed for use with the GMAP suite of tools. It is typically used as a parameter to the functions gsnap and bam_tally. This class also provides the means to index new genomes, from either a FASTA file or a BSgenome object. Genome indexes are typically stored in a centralized directory on the file system and are identified by a string key.

Constructor

GmapGenome(genome, directory = GmapGenomeDirectory(create = create), name = genomeName(genome), create = FALSE, ...):

Creates a GmapGenome corresponding to the genome argument, which may be either a string identifier of the genome within directory, a FastaFile or DNAStringSet of the genome sequence, or a BSgenome object.

The genome index is stored in directory argument, which may be either a GmapGenomeDirectory object, or a string path.

The name argument is the actual key used for storing the genome index within directory. If genome is a string, it is taken as the key. If a FastaFile, it is the basename of the file without the extension. If a BSgenome, it is the providerVersion. Otherwise, the name must be specified. If create is TRUE, the genome index is created if one with that name does not already exist. This obviously only works if genome actually contains the genome sequence.

The first example below gives the typical and recommended usage when implementing a reproducible analysis.

Extracting Genomic Sequence

getSeq(x, which = seqinfo(x)): Extracts the genomic sequence for each region in which (something coercible to GRanges). The result is a character vector for now. This is implemented in C and is very efficient. The default for which will retrieve the entire genome.

Coercion

as(object, "DNAStringSet"): Extracts the entire sequence of the genome as a DNAStringSet. One consequence is that this comes possible with rtracklayer: export(object, "genome.fasta").

Accessors

`path(object)`: returns the path to the directory containing the genome index files.

`directory(x)`: returns the `GmapGenomeDirectory` that is the parent of the directory containing the index files for this genome.

`genome(x)`: gets the name of this genome.

`seqinfo(x)`: gets the `Seqinfo` for this genome; only sequence names and lengths are available.

Author(s)

Michael Lawrence

Examples

```
## Not run:
library(BSgenome.Dmelanogaster.UCSC.dm3)
flyGG <- GmapGenome(Dmelanogaster, create = TRUE)

## access system-wide genome using a key
flyGG <- GmapGenome(genome = "dm3")

which <- seqinfo(flyGG)["chr4"]
firstchr <- getSeq(flyGG, which)

genome(which) <- "hg19"
## should throw an error
try(getSeq(flyGG, which))

##create a GmapGenome from a FASTA file
fa <- system.file("extdata/hg19.p53.fasta", package="gmapR")
fastaFile <- rtracklayer::FastaFile(fa)
gmapGenome <- GmapGenome(fastaFile, create=TRUE)

## End(Not run)
```

GmapGenomeDirectory-class

Class "GmapGenomeDirectory"

Description

The GmapGenomeDirectory class stores a path to a directory containing a one or more genome-specific subdirectories, each represented by a `GmapGenome`. Inside those directories are the files that the GMAP suite of tools uses for alignment, tallying, and other operations. This class is typically used to create a `GmapGenome` object. The default directory is `~/.local/share/gmap`, following the freedesktop.org XDG standard.

Constructor

`GmapGenomeDirectory(path = getDefaultGmapGenomePath(), create = FALSE)`: Creates an object pointing to the directory at `path`, creating it if it does not yet exist and `create` is `TRUE`.

Methods

`path(object)`: gets the path to the genome directory.

`genome(x)`: gets the names of the genomes in the directory.

Author(s)

Michael Lawrence

See Also

`GmapGenome-class`

Examples

```
gmapGenomePath <- file.path(getwd(), "newGmapGenomeDirectory")
gmapGenomeDirectory <- GmapGenomeDirectory(gmapGenomePath, create = TRUE)
```

GmapSnpDirectory-class

Class "GmapSnpDirectory"

Description

This class represents a directory containig one or more sets of SNPs, each corresponding to a genome. These SNP databases enable SNP-tolerant alignment with GMAP and GSNAP. If the underlying files have not been created, this class provides a means to do so.

Methods

[[<- `signature(x = "GmapSnpDirectory", i = "ANY", j = "ANY")`: ...

length `signature(x = "GmapSnpDirectory")`: ...

names `signature(x = "GmapSnpDirectory")`: ...

path `signature(object = "GmapSnpDirectory")`: ...

Author(s)

Michael Lawrence

GmapSnps-class *Class* "GmapSnps"

Description

This class represents a set of SNPs (single nucleotide polymorphisms) for use with GMAP and GSNAP (typically for SNP-tolerant alignment.)

Usage

```
GmapSnps(snps, directory, name = snps, create = FALSE, ...)
```

Arguments

snps	A path to a VCF file
directory	The directory to create the IIT files used by GMAP and GSNAP
name	If provided, the name to give the database of SNPs. If not provided, defauts to the snps argument.
create	If the directory provided in the directory argument does not exist, create it.
...	Additional arguments to be passed to the SNPs replacement method.

Objects from the Class

##TODO: doc these args Objects can be created by calls of the form GmapSnps(snps, directory, name, create).

Accessors

name(x): returns the name of the GmapSnps object

directory(x): returns the GmapGenomeDirectory that is the parent of the directory containing the index files for this GmapSnps object.

Methods

directory signature(x = "GmapSnps"): ...

Author(s)

Michael Lawrence

gmap_build-methods *Build Gmap/Gsnap Genome*

Description

Construct the IIT (interval index tree) needed from the GMAP suite of tools to run from a genome file. IIT files are an oligomer index and what allow GMAP and GSNAP to efficiently lookup interval information for fast genomic mapping. Fast and SNP-tolerant detection of complex variants and splicing in short reads offers an depth explication of IIT files and their use in GMAP and GSNAP.

Arguments

d	genome name
D	destination directory for installation (defaults to gmapdb directory specified at configure time
k	k-mer value for genomic index (allowed: 12..15, default 14)
S	do not order chromosomes in numeric/alphabetic order, but use order in FASTA file(s)
g	files are gzipped, so need to gunzip each file first

Methods:

`signature(x = "ANY", genome = "GmapGenome")`

`signature(x = "character", genome = "GmapGenome")`

`signature(x = "DNAStringSet", genome = "GmapGenome")`

Examples

```
## Not run: flyGG <- GmapGenome(genome = "dm3",
                    directory = ggd)
gmap_build(x=Dmelanogaster, genome=flyGG)

## End(Not run)
```

gsnap-methods *Align a Set of Reads Using the GSNAP Aligner*

Description

Given a set of alignments, align them to a genome using the GSNAP algorithm. The GSNAP algorithm contains a number of features making it a very high quality algorithm for dealing with short reads and those from RNA-seq data in particular. Via the `GsnapParam` class and the `gsnap` function, R users are given complete control over GSNAP.

Usage

```
## S4 method for signature 'character,character_OR_NULL,GsnapParam'
gsnap(input_a, input_b, params,
                  output = file.path(getwd(),
                                     file_path_sans_ext(basename(input_a),
                                                        TRUE)),
                  consolidate = TRUE, ...)
```

Arguments

input_a	A path to the FASTA file containing reads to align against a GmapGenome object. If the sequencing data is single-end, this is the only FASTA file used as input.
input_b	If provided, a path to the FASTA file containing the second set of reads from paired-end sequencing data.
params	A GsnapParam object to configure the behavior of GSNAP.
output	The output path for the GSNAP alignments. The results will be saved in dirname(output). If split_output in params is TRUE, basename(output) is used as the common stem for the multiple output files. Otherwise, the results are saved to a single SAM file, its path formed by adding the "sam" extention to output.
consolidate	If GSNAP is run with multiple worker threads, each thread will output its own set of files. If consolidate is set to TRUE, these files will be merged. The default is TRUE.
...	Additional arguments to pass to GSNAP not specifically supported by the gmapR package.

Value

A GsnapOutput class.

Author(s)

Michael Lawrence

GsnapOutput-class *Class* "GsnapOutput"

Description

A GsnapOutput object stores locations of data output by the GSNAP alignment algorithm.

Objects from the Class

GsnapOutput objects are created from the gsnap function, though the function GsnapOutput can also be used as a constructor.

Coercion

In the code snippets below, x is a GsnapOutput object.

`as(x, BamFile)`, `as(x, BamFileList)`:
Returns either a BamFile or BamFileList object containing paths to the output of GSNAP.

`asBam(x)`:
converts all gsnap SAM files to BAM files and creates the .bai index files.

Author(s)

Michael Lawrence

See Also

gsnap

GsnapParam-class *Class* "GsnapParam"

Description

A GsnapParam object stores parameters for gsnap. The function of the same name serves as its constructor.

Usage

```
GsnapParam(genome, unique_only = FALSE, molecule = c("RNA", "DNA"),
           max_mismatches = NULL,
           suboptimal_levels = 0L, mode = "standard",
           snps = NULL,
           npaths = if (unique_only) 1L else 100L,
           quiet_if_excessive = unique_only, nofails = unique_only,
           split_output = !unique_only,
           novelsplicing = FALSE, splicing = NULL,
           nthreads = 1L, part = NULL, batch = "2",
           terminal_threshold = if (molecule == "DNA") 1000L else 2L,
           gmap_mode = if (molecule == "DNA") "none" else
                       "pairsearch,terminal,improve",
           clip_overlap = FALSE, ...)
```

Arguments

genome	A GmapGenome object to align against
unique_only	Whether only alignments with a unique match should be output. The default is FALSE.

`molecule` The type of molecule sequenced; used to determine appropriate parameter defaults.

`max_mismatches` The maximum number of mismatches to allow per alignment. If NULL, then the value defaults to ((readlength + 2) / 12 - 2))

`suboptimal_levels`
Report suboptimal hits beyond best hit. The default is 0L.

`mode` The alignment mode. It can be "standard", "cmet-stranded", "cmet-nonstranded", "atoi-stranded", or "atoi-nonstranded". The default is "standard".

`snps` If not NULL, then a GmapSnps object. Provided SNPs will not count as mismatches.

`npaths` The maximum number of paths to print.

`quiet_if_excessive`
If more than maximum number of paths are found, then no alignment from the read will be in the output.

`nofails` Exclude failed alignments from output

`split_output` Basename for multiple-file output, separately for nomapping, halfmapping_uniq, halfmapping_mult, unpaired_uniq, unpaired_mult, paired_uniq, paired_mult, concordant_uniq, and concordant_mult results (up to 9 files, or 10 if –fails-as-input is selected, or 3 for single-end reads)

`novelsplicing` Logical indicating whether to look for novel splicing events. FALSE is the default.

`splicing` If not NULL, a GmapSplices object. NULL is the default.

`nthreads` The number of worker threads gsnap should use to align.

`part` If not NULL, then process only the i-th out of every n sequences e.g., 0/100 or 99/100 (useful for distributing jobs to a computer farm). If NULL, then all sequences are processed. NULL is the default.

`batch` This argument allows control over gsnap's memory mapping and allocation. The default is mode 2. Mode 0: {offsets=allocate, positions=mmap, genome=mmap}, Mode 1: {offsets=allocate, positions=mmap & preload,genome=mmap}, Mode 2: {offsets=allocate, positions=mmap & preload,genome=mmap & preload}, Mode 3: {offsets=allocate, positions=allocate,genome=mmap & preload}, Mode 4: {offsets=allocate, positions=allocate,genome=allocate}

`...` Additional parameters for gsnap. See gsnap's full documentation for those available.

`terminal_threshold`
If this number of mismatches is exceeded, GSNAP will attempt to align from one of the sequence, eventually giving up and discarding the rest of the sequence. This is called a "terminal alignment". By setting this to a high value, we have effectively disabled it for DNA, since terminal alignments were motivated by splicing alignment problems and other special cases.

`gmap_mode` Specifies the GMAP pipeline executed when GSNAP delegates to GMAP (a Smith-Waterman aligner) in difficult cases. We have disabled this for DNA, since such difficult cases are only anticipated in the context of splicing or complex rearrangements.

clip_overlap — Whether to equally clip paired ends that overlap each other (due to the fragment length being shorter than 2X the read length). This can be important for getting accurate counts from bam_tally.

See Also

gsnap

---

internals *gmapR2 internals*

Description

Internal methods, etc, that need an alias but are not intended for public use, at least not yet.

---

makeGmapGenomePackage *Function to create a GmapGenome package from a GmapGenome object*

Description

A GmapGenome object is required to align reads using the GSNAP or GMAP algorithms. The makeGmapGenomePackage function allows users to save a particular GmapGenome object in an R package.

Usage

```
makeGmapGenomePackage(gmapGenome, version, maintainer, author,
destDir = ".", license = "Artistic-2.0", pkgName)
```

Arguments

gmapGenome	A GmapGenome object.
version	The version number of this package.
maintainer	The maintainer of the package. The string must contain a valid email address.
author	The author of the package
destDir	The path that the new GmapGenome package should be created at.
license	The package's license (and its version)
pkgName	The name the package should have. Though free form, names of the form GmapGenome.Organism.Source.Build are recommended. E.g., GmapGenome.Hsapiens.UCSC.hg19

Author(s)

Cory Barr

See Also

GmapGenome

Examples

```
## Not run:
library(gmapR)

if (!require(BSgenome.Dmelanogaster.UCSC.dm3)) {
  library(BiocManager)
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
  library(BSgenome.Dmelanogaster.UCSC.dm3)
}

gmapGenomePath <- file.path(getwd(), "flyGenome")
if (file.exists(gmapGenomePath)) unlink(gmapGenomePath, recursive=TRUE)
ggd <- GmapGenomeDirectory(gmapGenomePath, create = TRUE)
gmapGenome <- GmapGenome(genome=Dmelanogaster,
                         directory = ggd,
                         name = "dm3",
                         create = TRUE)

makeGmapGenomePackage(gmapGenome=gmapGenome,
                      version="0.1.0",
                      maintainer="<your.name@somewhere.com>",
                      author="Your Name",
                      destDir=".",
                      license="Artistic-2.0",
                      pkgName="GmapGenome.Dmelanogaster.UCSC.dm3")


## End(Not run)
```

TP53Genome *Demo genome around TP53*

Description

Returns a GmapGenome object consisting of the UCSC hg19 sequence centered on the region of the TP53 gene, with 1 Mb flanking sequence on each side. This is intended as a test/demonstration genome and can be used, e.g., in conjunction with the LungCancerLines data package.

Usage

```
TP53Genome()
TP53Which()
```

Value

For TP53Genome, a GmapGenome object. If this is the first time the user has run this function, a side-effect will be the generation of an on-disk genome index, under the name "TP53_demo_VERSION" in the default genome directory, where VERSION is the version of the TxDb package providing the bounds of the P53 gene.

For TP53Which, a GRanges of the extents of the TP53 gene, translated to the space of TP53Genome.

Author(s)

Michael Lawrence, Cory Barr

Examples

```
TP53Genome()
```

Index