

An Introduction to the GenomicAlignments Package

Hervé Pagès

May 29, 2026

Contents

1	Introduction	1
2	<i>GAlignments</i>: Genomic Alignments	1
2.1	Load a 'BAM' file into a <i>GAlignments</i> object	2
2.2	Simple accessor methods	3
2.3	More accessor methods	3
3	<i>GAlignmentPairs</i>: Pairs of Genomic Alignments	4
4	<i>GAlignmentsList</i>: Groups of Genomic Alignments	4
5	Session Information	4

1 Introduction

The *GenomicAlignments* package serves as the foundation for representing genomic alignments within the Bioconductor project. In the Bioconductor package hierarchy, it builds upon the *GenomicRanges* (infrastructure) package and provides support for many Bioconductor packages.

This package defines three classes: *GAlignments*, *GAlignmentPairs*, and *GAlignmentsList*, which are used to represent genomic alignments, pairs of genomic alignments, and groups of genomic alignments.

The *GenomicAlignments* package is available at bioconductor.org and can be downloaded via `BiocManager::install`:

```
> if (!require("BiocManager"))
+   install.packages("BiocManager")
> BiocManager::install("GenomicAlignments")

> library(GenomicAlignments)
```

2 *GAlignments*: Genomic Alignments

The *GAlignments* class which is a container for storing a set of genomic alignments. The class is intended to support alignments in general, not only those coming from a 'Binary Alignment Map' or 'BAM' files. Also alignments with gaps in the reference sequence (a.k.a. *gapped alignments*) are supported which, for example, makes the class suited for storing junction reads from an RNA-seq experiment.

More precisely, a *GAlignments* object is a vector-like object where each element describes an *alignment*, that is, how a given sequence (called *query* or *read*, typically short) aligns to a reference sequence (typically long).

As shown later in this document, a *GAlignments* object can be created from a 'BAM' file. In that case, each element in the resulting object will correspond to a record in the file. One important thing to note though is that not all

the information present in the BAM/SAM records is stored in the object. In particular, for now, we discard the query sequences (SEQ field), the query ids (QNAME field), the query qualities (QUAL), the mapping qualities (MAPQ) and any other information that is not needed in order to support the basic set of operations described in this document. This also means that multi-reads (i.e. reads with multiple hits in the reference) don't receive any special treatment i.e. the various SAM/BAM records corresponding to a multi-read will show up in the *GAlignments* object as if they were coming from different/unrelated queries. Also paired-end reads will be treated as single-end reads and the pairing information will be lost. This might change in the future.

2.1 Load a 'BAM' file into a *GAlignments* object

First we use the `readGAlignments` function from the *GenomicAlignments* package to load a toy 'BAM' file into a *GAlignments* object:

```
> library(GenomicAlignments)
> aln1_file <- system.file("extdata", "ex1.bam", package="Rsamtools")
> aln1 <- readGAlignments(aln1_file)
> aln1
```

GAlignments object with 3271 alignments and 0 metadata columns:

	seqnames	strand	cigar	qwidth	start	end
	<Rle>	<Rle>	<character>	<integer>	<integer>	<integer>
[1]	seq1	+	36M	36	1	36
[2]	seq1	+	35M	35	3	37
[3]	seq1	+	35M	35	5	39
[4]	seq1	+	36M	36	6	41
[5]	seq1	+	35M	35	9	43
...
[3267]	seq2	+	35M	35	1524	1558
[3268]	seq2	+	35M	35	1524	1558
[3269]	seq2	-	35M	35	1528	1562
[3270]	seq2	-	35M	35	1532	1566
[3271]	seq2	-	35M	35	1533	1567

	width	njunc
	<integer>	<integer>
[1]	36	0
[2]	35	0
[3]	35	0
[4]	36	0
[5]	35	0
...
[3267]	35	0
[3268]	35	0
[3269]	35	0
[3270]	35	0
[3271]	35	0

seqinfo: 2 sequences from an unspecified genome

```
> length(aln1)
```

```
[1] 3271
```

3271 ‘BAM’ records were loaded into the object.

Note that `readGAlignments` would have discarded any ‘BAM’ record describing an unaligned query (see description of the `<flag>` field in the SAM Format Specification ¹ for more information). The reader interested in tracking down these events can always use the `scanBam` function but this goes beyond the scope of this document.

2.2 Simple accessor methods

There is one accessor per field displayed by the `show` method and it has the same name as the field. All of them return a vector or factor of the same length as the object:

```
> head(seqnames(aln1))

factor-Rle of length 6 with 1 run
  Lengths:    6
  Values  : seq1
Levels(2): seq1 seq2

> seqlevels(aln1)

[1] "seq1" "seq2"

> head(strand(aln1))

factor-Rle of length 6 with 1 run
  Lengths: 6
  Values  : +
Levels(3): + - *

> head(cigar(aln1))

[1] "36M" "35M" "35M" "36M" "35M" "35M"

> head(qwidth(aln1))

[1] 36 35 35 36 35 35

> head(start(aln1))

[1]  1  3  5  6  9 13

> head(end(aln1))

[1] 36 37 39 41 43 47

> head(width(aln1))

[1] 36 35 35 36 35 35

> head(njunc(aln1))

[1] 0 0 0 0 0 0
```

2.3 More accessor methods

[coming soon...]

¹<http://samtools.sourceforge.net/SAM1.pdf>

3 *GAlignmentPairs*: Pairs of Genomic Alignments

[coming soon...]

4 *GAlignmentsList*: Groups of Genomic Alignments

[coming soon...]

5 Session Information

All of the output in this vignette was produced under the following conditions:

```
> sessionInfo()

R version 4.6.0 (2026-04-24)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.4 LTS

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblas-p0.3.26.so; LAPACK version 3.12.0

locale:
 [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8      LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8  LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8     LC_NAME=C
 [9] LC_ADDRESS=C             LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Etc/UTC
tzcode source: system (glibc)

attached base packages:
[1] stats4      stats      graphics  grDevices  utils      datasets
[7] methods     base

other attached packages:
[1] pasillaBamSubset_0.51.0
[2] GenomicAlignments_1.49.0
[3] SummarizedExperiment_1.43.0
[4] Biobase_2.73.1
[5] MatrixGenerics_1.25.0
[6] matrixStats_1.5.0
[7] Rsamtools_2.29.0
[8] Biostrings_2.81.2
[9] XVector_0.53.0
[10] GenomicRanges_1.65.0
[11] IRanges_2.47.2
[12] S4Vectors_0.51.3
[13] Seqinfo_1.3.0
[14] BiocGenerics_0.59.6
[15] generics_0.1.4
[16] RNAseqData.HNRNPC.bam.chr14_0.51.0
```

```
[17] BiocStyle_2.41.0
```

```
loaded via a namespace (and not attached):
```

```
[1] Matrix_1.7-5          jsonlite_2.0.0      compiler_4.6.0
[4] BiocManager_1.30.27  crayon_1.5.3        bitops_1.0-9
[7] parallel_4.6.0        jquerylib_0.1.4     BiocParallel_1.47.0
[10] yaml_2.3.12          fastmap_1.2.0       lattice_0.22-9
[13] R6_2.6.1             S4Arrays_1.13.0     knitr_1.51
[16] DelayedArray_0.39.3  maketools_1.3.2     bslib_0.11.0
[19] rlang_1.2.0          cachem_1.1.0        xfun_0.57
[22] sass_0.4.10          sys_3.4.3           SparseArray_1.13.2
[25] cli_3.6.6            grid_4.6.0          digest_0.6.39
[28] lifecycle_1.0.5      evaluate_1.0.5      cigarillo_1.3.0
[31] codetools_0.2-20     buildtools_1.0.0    abind_1.4-8
[34] rmarkdown_2.31       tools_4.6.0         htmltools_0.5.9
```