Using rbambools package

Wolfgang Kaisers, CBiBs HHU Dusseldorf

April 25, 2013

1 What this package is made for

BAM files are a important and powerful file format in Bioinformatics. This package pursues several objectives:

- Provide a technical (reading and writing) access to BAM files from within R.
- Give an authentic representation of the informational structure inside BAM files as programming interface.
- Provide a fast, C-based access to special (cumulative) aspects of the stored information.

These objectives transform into three implementational layers:

- The samtools C-library (written by Heng Li).
- C-based align and align-gap container.
- A R S4 class library.

The samtools library is (almost) a copy of the library originally written by Heng Li. All reading and writing transactions are done via samtools. There is C-code which handle align data for whole ranges and C-code for accumulation of information about splice-sites from gapped aligns.
The R-part of the code contains objects which communicate directly with samtools for reading and writing files, managing of file-header data, managing data for single aligns and functions which transform align data into data.frame format. Then there are objects that calculate and keep align-gap information for whole BAM-files and to summarize align-gap data over several BAM-files.
Align-gaps are emphasized here because they are highly informative representations of genomic splice-sites in RNA-seq data.

2 SAM file format

Data in BAM files is compressed and optionally indexed data in SAM file format. The current definition of the SAM file format can be found here:

`http://samtools.sourceforge.net/SAM1.pdf`.

BAM files contain sequence alignment data which is the result of potentially incomplete matching sequence snippets to a reference sequence. In practice the snippets are DNA sequences which come from short read sequencing of DNA or RNA extracted from a biological probe and the reference sequence is a genome reference. Usually one BAM file contains align data from one biological probe where the read number is in the magnitude of 100 million reads. The size of the corresponding compressed files is in the range of 10 Gbyte. A very important feature of BAM files is that sorted BAM files can be indexed and indexed files allow random access. This allows very fast access to aligns that are located in arbitrary regions of the reference genome.

BAM files are divided in a header section and an alignment section.

2.1 The header section

The header section contains the following information:

Tag	Description	Information
HD	Header line	Format version and sorting
SQ	Reference sequence dictionary	Indexed reference sequences (Chromosomes)
RG	Read group	Sequencing technology
PG	Program	Alignment program
CO	Comment	

There are accessor functions for reading and writing the listed fields. The header section is stored and retrieved as a tag delimited string. The Information is processed in the same way. The objects parse and compose these strings from and to object slots which then can be accessed via script code.

2.1.1 The reference sequence dictionary

The reference sequence dictionary section contains a list of reference sequences (usually chromosomes). Off the six fields (declared in the SAM file format specification) usually only two are used:

Tag	Description
SN	Reference sequence name
LN	Reference sequence length

The reference sequence dictionary section misses an index entry (refid) which is used in alignment structures and is described below (˜2.2.1).

2.2 The alignment section

The alignment section contains a series of align datasets. Each align describes the coordinates of the identified sequence matches in the reference sequence. The information for each align basically consists of:

Field	Content
QNAME	Align name (read identifier)
RNAME	Reference sequence identifier
POS	Mapping position: 1-based
CIGAR	Matching type string
FLAG	A set of bitwise flags.

As example: A query sequence that matches on chromosome 1 at start position 1000 (1-based!) will have entries: RNAME=0 (where 0 is the ID of chr1) and POS=1000. "1-based" position notation means that the coordinate of the first character in the reference sequence is 1 (not 0).

2.2.1 The RNAME identifier: refid

Although RNAME associates with a textual entry, usually this field contains a number which identifies a sequence in the header section. To make things complicated, RNAME is a "0-based" sequential identifier which is not explicitly included in the "Reference sequence dictionary" (SQ). So, RNAME=0 means the first SQ entry and the "0" is not present in the header. We call this missing value *refid* throughout this document and there are functions in this package that automatically generate and use this id. The refid value is used by the `samtools` library as sequence identifier in align-structures and for defining ranges in index based random access.

3 Object types inside rbamtools package

The description of object types in this section starts with reading and writing access to BAM files, proceeds to objects which elementary data inside BAM files and ends with the description of more complex containers.

3.1 Reading and writing access

Immediate reading and writing access is provided by `bamReader` and `bamWriter` Objects.

3.2 bamReader

An object of class bamReader is constructed and returnd by the function `bamReader` in the following way:

```
> library(rbamtools)
> bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
> # Open bam file
> reader<-bamReader(bam)
```

An opened bamReader can be used to access the BAM header section and to read aligns sequenitally. `bamReader` can also be used to sort and index BAM files.

Sorting large BAM files requires some time and produces intermediate files. So the recommended way of sorting large BAM files is to use the samtools command line version. Sorting BAM files within R can be done with:

```
> bamSort(reader,prefix="my_sorted",byName=FALSE,maxmem=1e+9)
```

Sorted BAM files can be indexed. Indexing results in a second file which is usually named as the BAM file itself with an added suffix ".bai". An index file can be created with:

```
> create.index(reader,idx_filename="index_file_name.bai")
```

Omitting the `idx_filename` argument results in adding the ".bai" suffix to the filename of the BAM file which is then automatically located in the same directory as the BAM file itself:

```
> create.index(reader)
```

The creation of indexes for large BAM files (10 GB) takes some minutes time but can readily be done with this routine and of course has to be done only once per file.
Index files must be loaded before they can be used:

```
> idx<- system.file("extdata", "accepted_hits.bam.bai", package="rbamtools")
> load.index(reader,idx)
```

The reader object can be checked for for loaded index with:

```
> index.initialized(reader)
```

```
[1] TRUE
```

A shortcut for opening a BAM file and loading the "standard" index at the same time is:

```
> reader<-bamReader(bam,idx=TRUE)
```

3.3 bamWriter

For creation of a `bamWriter` object, a `bamHeader` and a filename must be given. The most convenient way of obaining a `bamHeaer` class is retrieving from an opened `bamReader` object.
Aligns can be written to a BAM file either from single instances of `bamAlign`'s or from whole `bamRange` objects.

3.4 Tabled reference sequences: `getRefData`

A data.frame with the reference sequences contained in the BAM header can be obtained with:

```
> getRefData(reader)
```

```
  ID    SN        LN
1  0  chr1 249250621
2  1 chr10 135534747
```

The returned data.frame contains in the first column (ID) the mentioned refid˜2.2.1 value which is not part of the header but uses as identifier for aligns and ranges.

4 Elementary data structures

The content of BAM files can be divided in `header` section and `alignment` section.

4.1 Structures for header section

The complete header information (in binary representation) can be retrieved from a BAM file with the function `getHeader`. An object of this type is needed for creation of a `bamWriter` object. In order to get Access to the data itself, the binary data has to be converted into a string representation which is maintained inside an object of class `bamHeaderText`:

```
> header<-getHeader(reader)
> htxt<-getHeaderText(header)
```

The header section is divided into several seqments (as described above) with data tags that describe the origin of the contained alignments. For each segment there is a class which can be be obtained by calling the appropriate function on a `bamHeaderText` object:

Segment ID	Description	S4 class	Retrieving function
HD	The header line	headerLine	headerLine
SQ	Reference sequence dictionary	refSeqDict	refSeqDict
RG	Read group		
PG	Program	headerProgram	header Program
CO	Comment		

4.2 Structures for alignment section

The alignment section in BAM files is a series of alignment (align) records. The data inside of each record is represented by a `bamAlign` object.

5 Complex and cumulative container

5.1 Align lists for specific reference regions: bamRange

`bamRange` objects manage a list of `bamAlign`'s. As BAM files usually contain alignment results against a reference-genome, `bamRange` objects contain list of all aligns that match between a given start and stop position on a given chromosome. Region coordinates are thereby defined by a refid˜2.2.1 and a start and stop position.

5.1.1 Reading bamRange from bamReader

In order to create a bamRange object, an index-initialized `bamReader` object and a numeric coordinates-vector of length three are passed to the `bamRange` function.

There are several ways to provide the coordinates for which the aligns are to be retrieved. The first way is to specify a circumscribed genomic region (e.g. where

a gene of interest is located). The names for the coordinates are not required and only added for explanational purposes:

```
> coords<-c(0,899000,900000)
> names(coords)<-c("refid","start","stop")
> range<-bamRange(reader,coords)
> size(range)

[1] 28
```

The second way is to specify coordinates for a whole reference sequence (chromosome). As can be seen from the output of the `getRefData` function, the coordinates for the whole first chromosome should be given as:

```
> getRefData(reader)

  ID    SN        LN
1  0  chr1 249250621
2  1 chr10 135534747

> coords<-c(0,0,249250621)
> names(coords)<-c("refid","start","stop")
> range<-bamRange(reader,coords)
> size(range)

[1] 738
```

The function `getRefCoords` is used here as shortcut:

```
> coords<-getRefCoords(reader,"chr1")
> coords

[1]         0         0 249250621

> range<-bamRange(reader,coords)
> size(range)

[1] 738
```

5.1.2 Accessing aligns in bamReader

`bamReader` objects keep a list of `bamAlign` objects. The objects can sequentially accessed or a data.frame with the align data can be retrieved.

5.1.3 Writing aligns to bamWriter