Filtering read alignments in BAM format

Tonatiuh Peña-Centeno
University of Greifswald

January 25, 2012

Abstract

This note describes the operation of filterBam, a C++ program that makes use of Bamtools API in order to filter alignment files stored in BAM format.

1 Main features

filterBam is a C++ code that cleans alignment files in BAM format that is based on filterPSL, a Perl routine written by Prof. Mario Stanke for PSL files. filterBAM includes the following filtering options:

- Screens out unmapped alignments.
- Screens out alignments that do not comply with a pre-defined coverage level (default=80%).
- Screens out alignments that do not comply with a pre-defined percentage of identity (default=92%).
- Screeens out alignments whose insert gaps do not comply with a pre-defined distance (default=10).
- Filters in two modalities: single and paired-end reads, expecting paired-queries to be given the suffixes "/1", "/2" or ".f", ".r".
- When in paired-read mode, it optionally writes to separate files a prospective list of common target genes and of pairedness coverage.

2 Single reads

filterBam has been

3 NOTES:

This document makes reference to the SAM/BAM format specification of **?**.

4 Bamtools

Bamtools is a C++ wrapper API of the more well-known Samtools software. The latest version of Bamools is 2.0 and is available on the website

https://github.com/pezmaster31/bamtools/downloads

5 Test data

We have generated a

6 Compilation

Make sure to link with the "-lz" and "-libbamtools.a" flags on; where -lz refers to the ZLIB library, and libbamtools.a to the static bamtools library included in the software distribution. An example of how to compile and link follows:

g++ -I**$BAMTOOLS**/include -g -std=c++0x -c filterBam.cc -o filterBam.o

g++ -g -std=c++0x filterBam.o -o filterBam **$BAMTOOLS**/lib/libbamtools.a -lz

where **$BAMTOOLS** is the path where Bamtools was installed.

Note that the flag "-std=c++0" has been used given that some of the functionalities of the filter require some of the newest features of GNU's g++ compiler. This and future versions of the software have been tested on Ubuntu's g++ version 4.4.3.

7 How to run

A run that will let pass most, if not all, readings:

./filterBam input.bam output.bam –minCover 0 –minId 0 –insertLimit 10000000 –nointrons

Note: that all options are provided at the very end.

8 Coverage, percent of identity and insert length

The coverage is computed as the sum of the alignment matches (sequence matches or mismatches) and the insertions to the reference. Both figures, alignment matches and insertions to the reference, correspond to CIGAR string operations M and I, respectively. Thus the following is done

$$\text{coverage} = \frac{\sum \text{CIGAR}\,(M, I)}{qLength} \tag{1}$$

An approximation to the percentage of identity is given by computing the query length and subtracting the so-called edit distance to the reference (tag "NM" in SAM jargon), i.e.

$$\text{percId} = \frac{qLength - \text{Tag}(NM)}{qLength} \tag{2}$$

The length of inserts is estimated by summing CIGAR operations "M" and "I", which correspond to alingment matches and deletions from the reference. In other words, we do the following

$$\text{InsertSize} = \frac{\sum \text{CIGAR}\,(D, I)}{qLength} \tag{3}$$