Read Naming Format Specification (draft)

Karel Břinda Valentina Boeva Gregory Kucherov

April 8, 2015

Abstract

This document provides a standard for naming simulated Next-Generation Sequencing (NGS) reads in order to make read simulators and mapper evaluation tools generally inter-compatible.

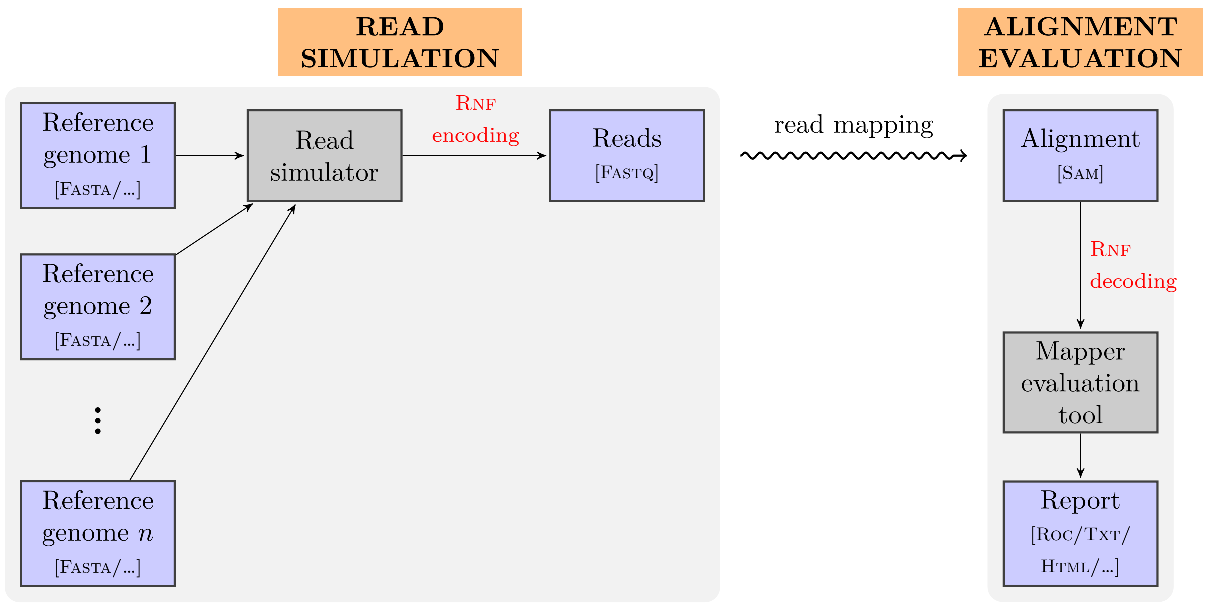

Figure 1: Evaluation of mappers of NGS reads using an RNF-compatible read simulator and an RNF-compatible alignment evaluation tool. Reads are simulated from one or more genomes (some of them possibly random), mixed together. After aligning them back to the reference sequence, alignments are assessed by an alignment evaluation tool, which subsequently creates overall reports.

1 Terminologies and concepts

Read tuple A tuple of sequences (possibly overlapping) obtained from a sequencing machine from a single fragment of DNA.

Reads Members of a *read tuple*. For example, every "paired-end read" is a *read tuple* and both of its "ends" are individual *reads* in our notation.

Segments Substrings of a *read* which are spatially distinct in the reference. They correspond to individual lines in a SAM file. Thus, each *read* has an associated chain of *segments* and we associate a *read tuple* with *segments* of all its *reads*.

Examples:

- A "single-end read" consists of a single *read* with a single *segment* unless it comes from a region with genomic rearrangment
- A "paired-end read" or a "mate-pair read" consists of two *reads*, each with one *segment* (under the same condition).
- A "strobe read" consists of several *reads*.

```
Coor        12345678901234-5678901234567890123456789

Source 1 - reference genome
chr 1       ATGTTAGATAA-GATAGCTGTGCTAGTAGGCAGTCAGCCC
chr 2       ttcttctggaa-gaccttctcctcctgcaaataaa

Source 2 - generator of random sequences

READS:
r001        ATG-TAGATA ->
r002/1         TTAGATAACGA ->
r002/2                                  <- TCAG-CGGG
r003/1                               tgcaaataa ->
r003/2              gaa-gacc-t ->
r004                       ATAGCT............TCAG ->
r005                                 GTAGG ->
             <- agacctt
                        <- TCGACACG
r006       ATATCACATCATTAGACACTA
```

(a) Simulated reads

r. tuple	LRN	SRN
r001	`sim__1__(1,1,F,01,10)__[single_end]`	#1
r002	`sim__2__(1,1,F,04,14),(1,1,R,31,39)__[paired_end]`	#2
r003	`sim__3__(1,2,F,09,17),(1,2,F,25,33)__[mate_pair]`	#3
r004	`sim__4__(1,1,F,15,36)__[spliced],C:[6=12N4=]`	#4
r005	`sim__5__(1,1,R,15,22),(1,1,F,25,29),(1,2,R,05,11)__[chimeric]`	#5
r006	`rnd__6__(2,0,N,00,00)__[random]`	#6

(b) Long and short read names

Figure 2: Example of simulated *read tuples* and their corresponding Long Read Names and Short Read Names which are used in the final FASTQ files.

- A chimeric *read* (i.e., read corresponding to a genomic fusion, a long deletion, or a translocation) has at least two *segments*.

Simulator of NGS reads A program which creates artificial simulated reads from one or more (possibly random) reference genomes.

Evaluation tool of NGS mappers A program which evaluates alignments of simulated NGS reads with known original genomic positions. It assesses if each individual read is aligned correctly. Finally it usually creates overall statistics.

1-based coordinate system A coordinate system where the first position has number 1 and intervals are closed (the same system is used by the SAM format).

2 Read tuple names

To every *read tuple*, we assign two names: Short read name (SRN) and Long read name (LRN).

SRN contains a hexademical unique *read tuple* ID prefixed by '#'.

LRN consists of four parts delimited by double-underscore:

i) a prefix (possibly containing expressive information for a user or a particular string for sorting or randomization of order of tuples),

ii) the *read tuple* ID,

iii) information about origins of all *segments* that constitute *reads* of the *tuple*,

iv) a suffix containing arbitrary comments or extensions (for holding additional information).

Preferred final read names are LRNs. If an LRN exceeds 255 (maximum allowed read length in SAM), SRNs are used instead and a SRN–LRN correspondence file must be created.

2.1 Read tuple ID

It is a positive integer, which is unique within a single file with genomic data. These IDs are assigned continuously from 1. Zero is reserved for “not available” (such value should be used only temporarily).

2.2 SRN – Short Read Name

Matching regular expression: `\#([0-9a-f]+)`.

SRN consists of *read tuple* ID prefixed by ‘`#`’. It is displayed as zero padded hexadecimals in lowercase such that all SRNs share the same string length within a single file.

2.3 LRN – Long Read Name

Matching regular expression: ``^([!-?A-^`-~]*)__([0-9a-f]+)__([!-?A-^`-~]+)__([!-?A-^`-~]*)$``.

LRN consists of four double-underscore-delimited parts: i) prefix part, ii) *read tuple* ID, iii) segmental part, iv) suffix part.

Prefix part

Matching regular expression: ``^[!-?A-^`-~]*$``

It can be an empty string, a string containing expressive “visual” information for the user (e.g., for easy distinguishing random reads from the others), or a string used for randomization of *read tuples* (randomly taken prefix and *read tuples* sorted in lexicographical order).

Length of all prefix parts within a single file must be equal.

Read tuple ID part

Matching regular expression: `^[0-9a-f]+$`

It displays *read tuple* ID as hexadecimals in lowercase. All *read tuple* ID parts are zero padded such that they all share the same string length within a single file.

Segmental part

Matching regular expression: `^(?:(\([0-9FRN,]*\))(?:,(?!$)|$))+$`

Segmental part consists of one or more comma-delimited segments.

Segment

Matching regular expression: `^\(([0-9]+),([0-9]+),([FRN]),([0-9]+),([0-9]+)\)$`

Every segment is parenthesized and consists of five comma-delimited values: i) genome ID, ii) chromosome ID, iii) direction, iv) leftmost coordinate, and v) rightmost coordinate.

Genome ID	ID (positive integer) of the source genome (a randomly generated genome, a genome saved in a Fasta file, etc.) or zero for “not available”. All numbers of genomes are displayed as decimals and they are zero padded such that they all share the same string length within a single file.
Chromosome ID	ID (positive integer) of the source chromosome or zero for “not available”. All numbers of chromosomes are displayed as decimals and they are zero padded such that they all share the same string length within a single file. IDs are assigned continuously from 1, order chromosomes is the same as in the file, where the genome is saved. In case of a random genome, zero should be used.
Direction	Direction in the reference genome. ‘F’ = forward direction ‘R’ = reverse direction ‘N’ = not available For random reads, ‘N’ should be used.
Leftmost coordinate	The leftmost coordinate of the segment in the reference in 1-based coordinate system or zero for “not available”.
Rightmost coordinate	The rightmost coordinate of the segment in the reference in 1-based coordinate system or zero for “not available”.

Suffix part

Matching regular expression: ``^(?:((?:[a-zA-Z0-9]+:){0,1})\[([!-?A-Z\\^`-~]*)\](?:,(?!$)|$))+$``

It contains arbitrary number of comma-delimited comments and extensions in any order.

Comment

Matching regular expression: ``^[([!-?A-Z\\^`-~]*)]$``

Comments are displayed as square-bracketed strings. They can contain, e.g., information about the simulated technology or the program used for simulation.

Extension

Matching regular expression: ``^([A-Za-z0-9]+):\[([!-?A-Z\\^`-~]*)\]$``

An extension consist of an extension’s code, a colon, and a square-bracketed extension’s content. Extensions can supplement the basic set of information provided in segmental part. Some of them are part of this standard, see Section 4.

3 SRN–LRN correspondence file

To encode information about correspondence between SRN and LRN, a special file is created. Its file name is formed of prefix of the Fastq file(s) and `.sl` suffix.

Examples:

Read files	SRN-LRN correspondence file
`reads_se.fq`	`reads_se.sl`
`reads_se.fastq`	`reads_se.sl`
`reads_pe.1.fq`, `reads_pe.2.fq`	`reads_pe.sl`

It is a tab delimited file with two columns (containing SRN and the corresponding LRN). File is sorted by *read tuple* ID.

4 Extensions

Extensions can supplement the basic set of information provided in the segmental part (Section 2.3).

C – CIGAR strings

Extension's code

C

Extension's content

Matching regular expression: `^(?:([0-9]+[=XIDNSHPM]+)(?:,(?!$)|$))+$`

Specification

The extension can be used to encode edit operations using CIGAR (Compact Idiosyncratic Gapped Alignment Report) strings.

Supported operations:

letter	operation	comment
`=`	match	
`X`	mismatch	
`I`	insertion	
`D`	deletion	
`N`	skipping bases	skipping intron regions in spliced mapping
`S`	soft clipping	for cutting unaligned prefixes and suffixes
`H`	hard clipping	for cutting unaligned prefixes and suffixes
`P`	padding	unused padding in padded reference
`M`	match or mismatch	DEPRECATED, reserved for situations when distinguishing `X` vs. `=` is impossible

CIGAR strings should be provided in the same order as their corresponding segments in the segmental part (Section 2.3). Adjacent edit operations should be different.

Example

```
demonstration__004__(1,1,F,16,40)__C:[6=14N5=],[spliced_read]
```

References

[1] Li, H. *et al.* (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**(16): 2078–2079.