iSAAC User Guide

Overview

iSAAC Aligner

iSAAC (short for iSAAC Sequence Aligner And Counter) is an end-to-end aligner that starts from data produced by an Illumina sequencing instrument and produces sorted, deduped, indel-realigned, indexed Bam files and reports. iSAAC has the following characteristics:

- Is simple to use: a single command with few mandatory options and reasonable default settings.
- Efficiently uses all available resources (CPU, memory, caching, IOs) and contains optimized algorithms and implementation.
- Is very robust: single node, positive control of resource usage.
- Sensitivity and accuracy have not been compromised.
- Integrated indel realignment.
- High flexibility for downstream variant calling. For example, iSAAC is seamlessly integrated with the HiSeq Analysis Software variant caller and works well with GATK (has BWA-style features including mapping qualities, random mapping, soft clipping, etc.).
- Provides comprehensive alignment stats and metrics

Use Cases

iSAAC suppots the follwoing use cases:

- Whole genome sequencing (small and large genomes).
- Targeted resequencing.
- Works with tags (Chip-Seq data)

Note that RNA mode is not supported.

iSAAC Command Descriptions

Aligning

The basic workflow to align the data is **isaac-align**. This workflow starts from BCL files and performs the entire alignment workflow: BAM generation and indexing, generation of the reports. More granular workflows are possible, using the "--start-from" and "--stop-at" command line options.

For instructions, see Aligning with isaac-align.

Reference Commands

The other iSAAC commands are for the creation of the sorted reference:

- **isaac-unpack-reference** creates the sorted reference from a compact version available on iGenome (or anywhere else). If your reference is not on iGenome, contact Illumina and ask to have it included.
- **isaac-pack-reference** creates a compact archive of the sorted reference meta data that is easy to transfer.
- **isaac-sort-reference** creates the sorted reference from the original fasta, if you need a reference not available on iGenome.
- **isaac-reorder-reference** is for users who need their reference in a different order (e.g. MT first)

iSAAC Limitations

Keep in mind the following when running iSAAC:

- iSAAC can only align reads of 32 bases or longer. Runs with very short reads (31 bases or less) should use another aligner.
- Processing of mRNA samples may be much slower than genomic DNA. When data is heavily mismatching, iSAAC spends a lot of time trying to realign reads to all possible locations within the template length. As iSAAC so far has been targeting standard DNA templates, the assumption is that the shadow rescuing would not have to deal with length over one thousand bases. Obviously with RNA this is not the case. For the moment, you can try to reduce the negative effects of rescuing by using --shadow-scan-range (see --help for details). However, the usefulness of iSAAC for RNA will be limited until there is a specific support for it.

Installing iSAAC

Downloading package

The iSAAC can be downloaded from MyIllumina (https://my.illumina.com) as rpm, deb or source tarball.

RPM Installation

Before starting this installation procedure, you need to have root access.The command line for installing the RPM file is as follows:

```
sudo rpm -i path-to-RPM-file/iSAAC-xx.xx.xx.xx-Linux-x86_64.rpm
```

Alternatively yum can be used to ensure all dependencies are installed automatically:

```
sudo yum install path-to-RPM-file/iSAAC-xx.xx.xx.xx-Linux-x86_64.rpm
```

Debian installation

Before starting this installation procedure, you need to have root access.The command line for installing the deb file is as follows:

```
sudo dpkg -i path-to-deb-file/iSAAC-xx.xx.xx.xx-Linux-x86_64.deb
```

Installation from source tarball

Please follow the steps described in the package src/INSTALL file.

Memory Requirements

As a ballpark figure, if there is Y GBs of BCL data, then iSAAC roughly does the following:

- Reads 2xY GBs of BCL files
- Reads 50 GB of sorted reference (for human)
- Writes 4xY GBs of Temporary data
- Reads 4xY GBs of Temporary Data
- Writes Y GBs of BAM Data

As a rule of thumb, given a reasonably high end modern CPU and enough memory for a lane of BCL files plus the reference, then the scratch storage should be able to do over 200 MB/s to avoid IO dominating the processing time and preferable over 500 MB/s.

Obtaining Sorted Reference

iSAAC requires a pre-processed reference to do the alignment. The pre-processing extracts all possible 32-mers from the reference genome and stores them in the format that is easily accessible to isaac-align along with the metadata. The metadata file also keeps the absolute path to the original .fa file and isaac-align uses this file. It is important to ensure that this file is available in its original location at the time isaac-align is being run.

As the metadata uses absolute paths to reference files, manually copying or moving the sorted refernce is not recommended. Instead, using the isaac-pack-reference/isaac-unpack-reference tool pair is advised.

There are two ways to obtain a sorted reference:

- Download pre-sorted packed reference from iGenomes and unpack it using isaac-unpack-reference
- Sort the reference genome of your choice using isaac-sort-reference.

Aligning with isaac-align

iSAAC uses the command isaac-align for aligning tasks. This chapter explains the use of isaac-align.

Command Line examples

Analyze all data from a bcl run

```
isaac-align
-r <Genome_Folder>/Sequence/iSAACIndex.xxxxxxxx/sorted-reference.xml
-b <Run_Folder>/Data/Intensities/BaseCalls -m 40
```

Analyze subset of lanes from a bcl run

```
isaac-align
-r <Genome_Folder>/Sequence/iSAACIndex.xxxxxxxx/sorted-reference.xml
-b <Run_Folder>/Data/Intensities/BaseCalls -m 38 --tiles s_[1234]_
```

Analyze paired read data from fastq.gz files

Notice that the fastq files need to be named or sym-linked in a special way so that iSAAC can recognize them.

```
$ ls Fastq/
lane1_read1.fastq.gz  lane1_read2.fastq.gz  lane2_read1.fastq.gz
lane2_read2.fastq.gz


isaac-align
-r <Genome_Folder>/Sequence/iSAACIndex.xxxxxxxx/sorted-reference.xml
-b Fastq -m 40 --base-calls-format fastq-gz
```

Analyze single-ended data from fastq.gz files

Notice that the fastq files need to be named or sym-linked in a special way so that iSAAC can recognize them.

```
$ ls Fastq/
lane1_read1.fastq.gz  lane2_read1.fastq.gz


isaac-align
-r <Genome_Folder>/Sequence/iSAACIndex.xxxxxxxx/sorted-reference.xml
-b Fastq -m 40 --base-calls-format fastq-gz
```

What to Do Next

After alignment is done, a common analysis step is variant detection. A tool that works well with iSAAC is the Genome Analysis Toolkit (GATK). A best practice wiki for variant detection with GATK is maintaned here: http://www.broadinstitute.o

rg/gatk/guide/topic?name=best-practices.

You can also use GATK for other analysis tasks; see documentation for supported GATK workflows here: http://www.broadinstitute.org/gatk/gatkdocs/.

Output folder structure

The standard output folder when running isaac_align looks like this:

```
Aligned
|-- Projects (output data files)
|   |-- <project name>
|   |   |-- <sample name>
|   |   |   |-- Casava (subset of CASAVA variant calling results data)
|   |   |   |-- sorted.bam (bam file for the sample. Contains data for the
project/sample from all flowcells)
|   |   |   `-- sorted.bam.bai
|   |   |-- ...
|   `-- ...
|-- Reports (navigable statistics pages)
|   |-- gif
|   |   |-- <flowcell id>
|   |   |   `-- all
|   |   |       `-- all
|   |   |           `-- all
|   |   |               |-- <per-tile statistic plot images>
|   |   `-- ...
|   `-- html
|       `-- index.html (root html for the analysis reports)
`-- Stats
    |-- BuildStats.xml (chromosome-level duplicate and coverage statistics)
    |-- DemultiplexingStats.xml (information about the barcode hits)
    `-- MatchSelectorStats.xml (tile-level yield, pair and alignment quality
statistics)
```

Tips and tweaks

Analyzing human genomes on low RAM System

As the human reference requires about 46 gigabytes to stay in RAM, the 48 gigabyte (or smaller) systems are not able to keep it in the cache. In this situation it pays to minimize the number of reference scanning passess the MatchFinder need to perform.

The number of tiles processed at a time is usually capped by the **--temp-parallel-save** parameter, which is set to 16 by default. This limit prevents the match files to be excessively fragmented on the systems where temporary data is stored on a local hard drive. If the temporary data is stored on a distributed network storage such as Isilon, the fragmentation is usually not an issue. In this case **--temp-parallel-save 64** will ensure that entire lane (64 is the currently known maximum number of tiles per lane produced by an instrument) is loaded into memory for MatchFinder.

On the systems with more than 64 gigabytes of physical RAM, it makes sense to keep the defaults as in this case the reference has enough room to stay in the IO cache.

Reducing Linux swappiness

iSAAC is designed to take the full advantage of the hardware resources available on the processing node. On systems with default Linux configuration, this causes the operating system to swap pages out to make room for the IO cache, when the

iSAAC gets to a memory/IO intenstive stages. Since iSAAC operates on data volumes that normally exceed the amount of RAM of an average system, there is little or no benefit from IO-caching the data iSAAC reads and writes. As result, the data processing takes longer than it would if the system did not prioritize IO cache over the code and data that is already in RAM.

Reducing the value /proc/sys/vm/swappiness to 10 or lower from default 60 solves the problem.

Toolkit Commands for Reference

isaac-align

The parameters -r, -b, and -m (if ulimit is not set) are required. All other parameters are optional.

Option	Description
-h [--help]	produce help message and exit
-v [--version]	print program version information
-b [--base-calls-directory] arg	full path to the base calls directory. Multiple entries allowed.
--base-calls-format arg	Multiple entries allowed. Each entry is applied to the corresponding base-calls-directory. Last entry is applied to all --base-calls-directory that don't have --base-calls-format specified.
	- bcl(default) : common bcl files, no compression.
	- bcl-gz : bcl files are individually compressed and named s_X_YYYY.bcl.gz
	- fastq : One fastq per lane/read named lane<X>_read<Y>.fastq and located directly in the specified base-calls-directory
	- fastq-gz : One compressed fastq per lane/read named lane<X>_read<Y>.fastq.gz and located directly in the specified base-calls-directory
--default-adapters arg	Multiple entries allowed. Each entry is associated with the corresponding base-calls-directory. Flowcells that don't have default-adapters provided, don't get adapters clipped in the data.
	Each entry is a comma-separated list of adapter sequences written in the direction of the reference. Wildcard (* character) is allowed only on one side of the sequence. Entries with * apply only to the alignments on the matching strand. Entries without * apply to all strand alignments and are matched in the order of appearance in the list.
	Examples:
	ACGT*,*TGCA : Will clip ACGT and all subsequent bases in the forward-strand alignments and mirror the behavior for the reverse-strand alignments.

	ACGT,TGCA : Will find the following sequences in the reads: ACGT, TGCA, ACGTTGCA (but not TGCAACGT!) regardless of the alignment strand. Then will attempt to clip off the side of the read that is shorter. If both sides are roughly equal length, will clip off the side that has less matches.
	Nextera : Nextera standard, Same as CTGTCTCTTATACACATCT*,*AGATGTGTATAAGAGACAG
	NexteraMp : Nextera mate-pair. Same as CTGTCTCTTATACACATCT,AGATGTGTATAAGAGACAG
-s [--sample-sheet] arg	Multiple entries allowed. Each entry is applied to the corresponding base-calls-directory.
	- none : process flowcell as if there is no sample sheet
	- default : use <base-calls-directory>/SampleSheet.csv if it exists. This is the default behavior.
	- <file path> : use <file path> as sample sheet for the flowcell.
--barcode-mismatches arg (=0)	Multiple entries allowed. Each entry is applied to the corresponding base-calls-directory. Last entry applies to all the bases-calls-directory that do not have barcode-mismatches specified. Last component mismatch value applies to all subsequent barcode components should there be more than one. Examples:
	- 1:0 : allow one mismatch for the first barcode component and no mismatches for the subsequent components.
	- 1 : allow one mismatch for every barcode component.
	- 0 : no mismatches allowed in any barcode component. This is the default.
--realign-gaps arg (=yes)	For reads overlapping the gaps occurring on other reads, check if applying those gaps reduces mismatch count. Significantly reduces number of false SNPs reported around short indels.
	- no : no gap realignment
	- yes : discard all existing gaps in the read and apply the combination of gaps found in the region that yields maximum reduction of mismatches
--bam-gzip-level arg (=1)	Gzip level to use for BAM

--expected-bgzf-ratio arg (=1)	compressed = ratio * uncompressed. To avoid memory overallocation during the bam generation, iSAAC has to assume certain compression ratio. If iSAAC estimates less memory than is actually required, it will fail at runtime. You can check how far you are from the dangerous zone by looking at the resident/swap memory numbers for your process during the bam generation. If you see too much showing as 'swap', it is safe to reduce the --expected-bgzf-ratio.
--tiles arg	Comma-separated list of regular expressions to select only a subset of the tiles available in the flow-cell.
	- to select all the tiles ending with '5' in all lanes: --tiles [0-9][0-9][0-9]5
	- to select tile 2 in lane 1 and all the tiles in the other lanes: --tiles s_1_0002,s_[2-8]
	Multiple entries allowed, each applies to the corresponding base-calls-directory.
--use-bases-mask arg	Conversion mask characters:
	- Y or y : use
	- N or n : discard
	- I or i : use for indexing
	If not given, the mask will be guessed from the B<config.xml> file in the base-calls directory.
	For instance, in a 2x76 indexed paired end run, the mask I<Y76,I6n,y75n> means:
	use all 76 bases from the first end, discard the last base of the indexing read, and use only the first 75 bases of the second end.
--seeds arg (=16:0:32:64)	Seed descriptors for each read, given as a comma-separated list-of-seeds for each read. A list-of-seeds is a colon-separated list of offsets from the beginning of the read.
	Examples:
	- 0:32,0:32:64 : two seeds on the first read (at offsets 0 and 32) and three seeds on the second read (at offsets 0, 32, and 64) and on subsequent reads.
	- 0:32:64 : three seeds on all the reads (at offsets 0, 32 and 64)

	Note that the last list-of-seeds is repeated to all subsequent reads if there are more reads than there are colon-separated lists-of-seeds.
-f [--first-pass-seeds] arg (=1)	the number of seeds to use in the first pass of the match finder
-r [--reference-genome] arg	Full path to the reference genome XML descriptor. Multiple entries allowed.Each entry applies to the corresponding --reference-name. The last --reference-genome entry may not have a corresponding --reference-name. In this case the default name 'default' is assumed.
-n [--reference-name] arg	Unique symbolic name of the reference. Multiple entries allowed. Each entry is associated with the corresponding --reference-genome and will be matched against the 'reference' column in the sample sheet.
	Special names:
	- unknown : default reference to use with data that did not match any barcode.
	- default : reference to use for the data with no matching value in sample sheet 'reference' column.
-t [--temp-directory] arg (="./Temp")	Directory where the temporary files will be stored (matches, unsorted alignments, etc.)
-o [--output-directory] arg (="./Aligned")	Directory where the final alignment data be stored
-j [--jobs] arg (=32)	Maximum number of compute threads to run in parallel
--input-parallel-load arg (=64)	Maximum number of parallel file read operations for --base-calls-directory
--temp-parallel-load arg (=8)	Maximum number of parallel file read operations for --temp-directory
--temp-parallel-save arg (=64)	Maximum number of parallel file write operations for --temp-directory
--output-parallel-save arg (=8)	Maximum number of parallel file write operations for --output-directory
--repeat-threshold arg (=10)	Threshold used to decide if matches must be discarded as too abundant (when the number of repeats is greater or equal to the threshold)
--shadow-scan-range arg (=-1)	-1 - scan for possible mate alignments between template min and max
	>=0 - scan for possible mate alignments in range of template median += shadow-scan-range

--neighborhood-size-threshold arg (=0)	Threshold used to decide if the number of reference 32-mers sharing the same prefix (16 bases) is small enough to justify the neighborhood search. Use large enough value e.g. 10000 to enable alignment to positions where seeds don't match exactly.
--verbosity arg (=2)	Verbosity: FATAL(0), ERRORS(1), WARNINGS(2), INFO(3), DEBUG(4) (not supported yet)
--start-from arg (=Start)	Start processing at the specified stage:
	- Start : don't resume, start from beginning
	- MatchFinder : same as Start
	- MatchSelector : skip match identification, continue with template selection
	- AlignmentReports : regenerate alignment reports and bam
	- Bam : resume at bam generation
	- CasavaReset : resume at CASAVA configuration (will reset CASAVA analyses)
	- CasavaResume : resume at CASAVA execution (will resume interrupted CASAVA execution)
	- Finish : Same as CasavaResume.
	- Last : resume from the last successful step
	Note that although iSAAC attempts to perform some basic validation, the only safe option is 'Start' The primary purpose of the feature is to reduce the time required to diagnose the issues rather than be used on a regular basis.
--stop-at arg (=Finish)	Stop processing after the specified stage is complete:
	- Start : perform the first stage only
	- MatchFinder : same as Start
	- MatchSelector : don't generate alignment reports and bam
	- AlignmentReports : don't perform bam generation
	- Bam : finish when bam is done
	- CasavaReset : finish after CASAVA configuration (will reset CASAVA analyses)
	- CasavaResume : finish after CASAVA execution completes
	- Finish : stop at the end.

	- Last : perform up to the last successful step only
	Note that although iSAAC attempts to perform some basic validation, the only safe option is 'Finish' The primary purpose of the feature is to reduce the time required to diagnose the issues rather than be used on a regular basis.
--ignore-neighbors arg (=0)	When not set, MatchFinder will ignore perfect seed matches during single-seed pass, if the reference k-mer is known to have neighbors.
--ignore-repeats arg (=0)	Normally exact repeat matches prevent inexact seed matching. If this flag is set, inexact matches will be considered even for the seeds that match to repeats.
--mapq-threshold arg (=0)	Threshold used to filter the templates based on their mapping quality: the BAM file will only contain the templates with a mapping quality greater than or equal to the threshold. Templates (or fragments) with a mapping quality of 4 or more are guaranteed to be uniquely aligned. Those with a mapping quality of 3 or less are either mapping to repeat regions or have a large number of errors.
--pf-only arg (=1)	When set, only the fragments passing filter (PF) are generated in the BAM file
--scatter-repeats arg (=0)	When set, extra care will be taken to scatter pairs aligning to repeats across the repeat locations
--base-quality-cutoff arg (=20)	3' end quality trimming cutoff. 0 turns off the trimming.
--variable-fastq-read-length arg (=0)	Unless set, iSAAC will fail if the length of the sequence changes between the records of the fastq file.
--ignore-missing-bcls arg (=0)	When set, missing bcl files are treated as all clusters having N bases for the corresponding tile cycle. Otherwise, encountering a missing bcl file causes the analysis to fail.
--ignore-missing-filters arg (=0)	When set, missing filter files are treated as if all clusters pass filter for the corresponding tile. Otherwise, encountering a missing filter file causes the analysis to fail.
--keep-unaligned arg (=discard)	Available options:
	- discard : discard clusters where both reads are not aligned
	- front : keep unaligned clusters in the front of the BAM file
	- back : keep unaligned clusters in the back of the BAM file

--clip-semialigned arg (=1)	When set, semialigned reads have their mismatching part soft-clipped
--gapped-mismatches arg (=5)	Maximum number of mismatches allowed to accept a gapped alignment.
--gap-scoring arg (=eland)	Gapped alignment algorithm parameters:
	- eland : equivalent of 2:-1:-15:-3
	- bwa : equivalent of 0:-3:-11:-4
	- m:mm:go:ge : colon-delimited string of values where:
	m : match score
	mm : mismatch score
	go : gap open score
	ge : gap extend score
--dodgy-alignment-score arg (=Zero)	Controls the behavior for templates where alignment score is impossible to assign:
	- Unaligned : marks template fragments as unaligned
	- Zero : sets the fragment and pair alignment scores to 0
	- Unknown : assigns value 255 for bam MAPQ. Ensures SM and AS are not specified in the bam
--keep-duplicates arg (=0)	Keep duplicate pairs in the bam file (with 0x400 flag set in all but the best one)
--bin-regex arg	Comma-separated list of regular expressions. If not empty, bam files will be filtered to contain only the bins that match.
--memory-control arg (=off)	Define the behavior in case unexpected memory allocations are detected:
	- warning : Log WARNING about the allocation.
	- off : Don't monitor dynamic memory usage.
	- strict : Fail memory allocation. Intended for development use.
-m [--memory-limit] arg (=0)	Limits major memory consumption operations to a set number of gigabytes. 0 means no limit, however 0 is not allowed as in such case iSAAC will most likely consume all the memory on the system and cause it to crash. Default value is taken from ulimit -v.
-c [--cluster] arg	Restrict the alignment to the specified cluster Id (multiple entries allowed)

--tls arg	Template-length statistics in the format 'min:median:max:lowStdDev:highStdDev:M0:M1', where M0 and M1 are the numeric value of the models (0=FFp, 1=FRp, 2=RFp, 3=RRp, 4=FFm, 5=FRm, 6=RFm, 7=RRm)
--casava*	Any option beginning with --casava will have the --casava prefix removed and passed along with its arguments to the configureBuild.pl CASAVA script at the CasavaReset stage. For example --casava--variantsConsensusVCF will add --variantsConsensusVCF to the configureBuild.pl command line
--stats-image-format arg (=gif)	Format to use for images during stats generation
	- gif : produce .gif type plots
	- none : no stat generation
--qscore-bin arg (=0)	Toggle QScore binning, this will be applied to the data after it is loaded and before processing
--qscore-bin-values arg	Overwrite the default QScore binning values. Default bins are 0:0,1:1,2-9:6,10-19:15,20-24:22,25-29:27,30-34:33,35-39:37,40-63:40. Identity bins 1:1,2:2,3:3,4:4,5:5,6:6,7:7,8:8,9:9,10:10,11:11,12:12,13:13,14:14,15:15,16:16,17:17,18:18,19:19,20:20,21:21,22:22,23:23,24:24,25:25,26:26,27:27,28:28,29:29,30:30,31:

isaac-reorder-reference

Produces new .fa file with given order of contigs and an updated iSAAC sorted reference metadata. This is useful if the variant caller (such as GATK) requires a particular order of reference contigs that does not match the one in the .fa file originally supplied to isaac-sort-reference.

Usage

```
isaac-reorder-reference [options]
```

Options

Option	Description
-h [--help]	Produce help message and exit
-v [--version]	Print program version information
-r [--reference-genome] arg	Full path to the reference genome XML descriptor
-f [--output-fasta] arg	Path for the reordered fasta file

--order arg	Comma-separated list of contig names in the order in which they will appear in the new .fa file
-b [--bases-per-line] arg (=70)	Number of bases per line to print into .fa file
-x [--output-xml] arg	Path for the new xml file

isaac-sort-reference

Analyzes the reference genome and converts it into a set of files required for isaac-align

> Note that available RAM could be a concern when sorting big genomes. Human genome reference sorting will require ~150 gigabytes of RAM.

Usage

```
isaac-sort-reference [options]
```

Time highly repends on the size of the genome. For *E. coli* it takes about 1 minute. For human genomes it takes about 11 hours.

Options

Option	Description
-h [--help]	Print this message
-n [--dry-run]	Don't actually run any commands; just print them
-v [--version]	Only print version information
-j [--jobs] arg (=1)	Maximum number of parallel operations
-w [--mask-width] arg (=6)	Number of high order bits to use for splitting the data for parallelization
-g [--genome-file] arg	Path to fasta file containing the reference contigs
-o [--output-directory] arg (./iSAACIndex.20120704)	Location where the results are stored

Example

```
nohup /illumina/development/iSAAC/latest/bin/isaac-sort-reference -g
$(pwd)/Homo_sapiens_assembly19.fasta -j 24&
```

isaac-unpack-reference

Expand sorted reference into the current folder. The reverse command is isaac-pack-reference.

Usage

```
isaac-unpack-reference [options]
```

Options

Options	Description
-h [--help]	Print this message
-n [--dry-run]	Don't actually run any commands; just print them
-v [--version]	Only print version information
-j [--jobs] arg (=1)	Maximum number of parallel operations
-i [--input-file] arg	Archive path

Example

```
 nohup /illumina/development/iSAAC/testing/bin/isaac-unpack-reference -i
../packed-reference.tar.gz -j 24 &
```

isaac-pack-reference

Package sorted reference meta data into a compact archive which is easy to transfer. The reverse command is isaac-unpack-reference.

Usage

```
isaac-pack-reference [options]
```

Options

Option	Description
-h [--help]	Print this message
-n [--dry-run]	Don't actually run any commands; just print them
-v [--version]	Only print version information

-j [--jobs] arg (=1)	Maximum number of parallel operations
-r [--reference-genome] arg	Path to sorted-reference.xml
-o [--output-file] arg (./packed-reference.tar.gz)	Archive path

Example

```
/illumina/development/iSAAC/testing/bin/isaac-pack-reference -r
HumanUCSC.hg19.complete..nohap.20120918/sorted-reference.xml -j 2
```

Appendix

iSAAC Alignment Quality Scoring Model

This section describes the iSAAC alignment quality scoring model.

> ⚠ The iSAAC alignment quality scoring model is very similar to the model used in CASAVA

Probability of a Correct Read

This probability is the product of the probability for each base to be correct, as determined by the quality score of the base and the alignment of the base against the reference. If `Q[i]` is the Phred quality score of a base at position `i`, we have:

- `pBaseError[i] = 10^(-Q[i]/10)`
- `pBaseCorrect[i] = 1 - pBaseError[i]`
- `pBaseMatch[i] = pBaseCorrect[i]`
- `pBaseMismatch[i] = pBaseError[i]/3`
- `pReadCorrect = product(i=0..readLength-1, pBase[i])`, where `pBase[i]`is
  - `pBaseMatch[i]` if the base at position `i` matches the reference, or is part of an indel
  - `pBaseMismatch[i]` otherwise

Note: using `pBaseMatch[i]` for indels is an arbitrary choice (doing otherwise would require a model for indels)

Alignment Quality of a Single Read

The alignment quality depends on the intrinsic quality of the alignment (inferred from `pReadCorrect` above), but also on the specificity of the alignment (i.e. the probability that the read aligns somewhere else). This is inferred from two quantities:

- `pNeighbourhood = sum of pReadCorrect for all alignments in the neighbourhood` (all other identified alignment positions)
- `rogCorrection = 2*GenomeLength/(4^ReadLength)`

The `rogCorrection` is the "rest-of-genome" correction that gives an indication of the probability of having a random read aligning to the reference. This value tends to (and should) be very small and allows differentiating between the quality of reads with unique alignments (in which case `pNeighbourhood == 0`).

iSAAC Accumulates Actual Probablilities in pNeighborhood

Note that in CASAVA (unlike iSAAC), `pNeighborhood` is normalized to take into account all the estimated values of `pReadCorrect` for all the seeds that matched the reference with up-to two mismatches. These estimated values are pessimistic in

the sense that they assume that extending the alignment does not introduce any additional mismatches. The assumption is that a seed with one mismatch will lead to an alignment descriptor with exactly one match on the base with the worst quality (using the definition of `pReadCorrect` given above). Similarly, a seed with two mismatches will lead to an alignment descriptor with exactly two mismatches (on the bases with the two worst qualities).

```
pNormalized = rogCorrection + pNeighbourhood
```

finally, the alignment quality is:

```
alignmentQuality == -10 * log10(pNormalized/(pNormalized + pReadCorrect))
```

Alignment Quality of a Pair

This is simply the sum of the alignment quality of each fragment when there is exactly one resolved fragment. Otherwise, the alignment score is corrected by the total alignment score of all the resolved fragments:

```
alignmentQuality = -10 * log(pBestTemplateCorrect / pTotalTemplateCorrect)
```

where:

- `pTemplateCorrect = product(pReadCorrect for all reads)`
- `totalRogCorrection = rogCorrection for the total length of all reads`
- `pTotalTemplateCorrect = totalRogCorrection + sum(pTemplateCorrect for all resolved templates)`
- `pBestTemplateCorrect = max(pTemplateCorrect for all resolved templates)`

iSAAC bam files

iSAAC produces separate bam file per project/sample.

Unaligned pairs

Pairs where both reads are unaligned are stored depending on the argument of --keep-unaligned command line option.

--keep-unaligned	
discard	Ensures unaligned pairs are not present in the bam file
front	Places unaligned pairs in the beginning of the bam file before the first aligned pair of the first chromosome. The iSAAC-generated bam index file is specially crafted to skip those. This approach makes it easier to locate the unaligned clusters compared to the standard implementations which require reading past the last aligned pair of the last chromosome in the genome to locate the first unaligned pair. The drawback is that the standard `samtools index` command is unable to process such bam files. Be sure to keep the bam index files produced by iSAAC.

back	Makes unaligned pairs appear at the end of the bam file. Although this makes it somewhat difficult to extract unaligned data, this is the option that produced bam file that is compatible with `samtools index` command

Singleton/Shadow pairs

Singleton/shadow pairs refer to pairs in which aligner was unable to decide on the alignment of one of the ends (shadow). In this case, the shadows are assigned the position of the end that does align (singleton). In order to keep compatibility with CASAVA, the shadows are stored in the bam file immediately after their singleton.

Read names

Although the format of iSAAC bam read names was kept compatible with CASAVA, the content is significantly different:

```
read-name     = flowcell-id "_" flowcell-idx ":" lane-number ":"
tile-number ":" cluster-id ":0"

flowcell-id   = ;flowcell identifier from BaseCalls/config.xml.
"unknown-flowcell" if the identifier cannot be determined from config.xml
file
flowcell-idx  = ;unique 0-based index of the flowcell within the analysis
lane-number   = ;Lane number 1-8
tile-number   = ;Unpadded tile number
cluster-id    = ;Unpadded 0-based cluster id in the order in which the
clusters appear in the bcl tile.
```

Bam flags usage

Bit	Description	iSAAC notes
0x1	template having multiple segments in sequencing	
0x2	each segment properly aligned according to the aligner	Pair matches dominant template orientation. Single-ended templates don't have this flag set.
0x4	segment unmapped	
0x8	next segment in the template unmapped	
0x10	SEQ being reverse complemented	
0x20	SEQ of the next segment in the template being reversed	
0x40	the first segment in the template	Read 1

0x80	the last segment in the template	Read 2
0x100	secondary alignment	iSAAC does not produce secondary alignments
0x200	not passing quality controls	PF flag from RTA
0x400	PCR or optical duplicate	If --keep-duplicates is turned off, duplicates are excluded from the bam file. If --mark-duplicates is turned off, duplicates are not marked in the bam file.

Extended tags

iSAAC generates following tags in the output bam files. The list of tags stored can be controlled by --bam-exclude-tags command-line argument.

Tag	iSAAC meaning
AS	Pair alignment score
BC	Barcode string.
NM	Edit distance (mismatches and gaps) including the soft-clipped parts of the read
OC	Original CIGAR for the realigned reads. See --realign-gaps.
RG	iSAAC read groups correspond to flowcell/lane/barcode
SM	Single read alignment score
ZX	Cluster X pixel coordinate on the tile times 100 (disabled by default)
ZY	Cluster Y pixel coordinate on the tile times 100 (disabled by default)

MAPQ

Bam MAPQ for pairs that match dominant template orientation is `min(max(SM, AS), 60)`. For reads that are not members of a pair machting the dominant template orientation, the MAPQ is `min(SM, 60)`. The MAPQ might be downgraded to 0 or set to be unknown (255) for alignments that don't have enough evidence to be correctly scored. This behavior depends on the `--dodgy-alignment-score` argument.