Demultiplexing Illumina sequencing data containing unique molecular identifiers (UMIs)

Introduction

This document outlines an example workflow for demultiplexing Illumina sequencing data containing UMIs, starting from Illumina basecall (BCL) files through generating FASTQ files. The high level workflow is:

A. Determine the sample barcode for every read in an Illumina sequencer lane

B. Demultiplex each lane of Illumina BCL files and include the UMI data in the per-sample BAM files

C. Convert BAM files to FASTQ files

D. Map FASTQ reads and merge alignment data from aligned BAM files with molecular index data from unaligned BAM files

Note: If you require molecular index data for downstream, post-mapping analysis, extract the molecular index data from the unaligned BAM files, then merge the data with the aligned BAM files after mapping the FASTQ reads.

Referenced software package

The following software package is used in the examples within this document:

Package	License	URL
Picard	MIT	https://github.com/broadinstitute/picard

A. Determine the sample barcode for every read in an Illumina sequencer lane

Before demultiplexing Illumina basecall data, you must determine the barcode for each read in an Illumina sequencer lane. Use Picard's *ExtractIlluminaBarcodes* to determine the barcode for each read in a lane by parsing the files in the basecalls directory. The tool determines the numbers of reads containing barcode-matching sequences and provides statistics on the quality of these barcode matches.

The Read Structure refers to a String that describes the order, the number of cycles, and the type of those cycles (T = template, B = sample barcode, M = molecular barcode, and S = skip). Illumina sequences can contain at least two types of barcodes: sample and molecular. Sample barcodes (B) are used to demultiplex pooled samples, while molecular barcodes (M) are used to differentiate multiple reads of a template when performing paired-end sequencing. Note that *ExtractIlluminaBarcodes* only extracts sample barcodes (B) and not molecular barcodes (M). An example invocation follows:

```
java -Xmx4g -jar picard.jar ExtractIlluminaBarcodes \
    BASECALLS_DIR=inputBCLdir \
    BARCODE_FILE=barcodesFile \
    READ_STRUCTURE=100T8B9M8B100T \
    LANE=1 \
    OUTPUT_DIR=barcodes_outputdir \
    METRICS_FILE=barcode_metrics.txt
```

INPUT:

1. Basecalls directory (required):
   The following input files from the basecalls directory are needed:
   - *.bcl files
   - *.stats files
   - *.filter files
   - *.control files
   - *.locs files

2. Barcodes (required):
   Barcodes are provided in the form of a list (BARCODE_FILE) or a string representing the barcode (BARCODE). The BARCODE_FILE contains multiple fields including "barcode_sequence_1", "barcode_sequence_2" (optional), "barcode_name", and "library_name". In contrast, the BARCODE argument is used for runs with reads containing a single barcode (nonmultiplexed) and can be added directly as a string of text (e.g., BARCODE = CAATAGCG). An example invocation follows:

barcode_name	library_name	barcode_sequence_1	barcode_sequence_2
20160808-AP1SG-S1	Mix1_Rep1	CTGATCGTNNNNNNNNN	GCGCATAT
20160808-AP1SG-S2	Mix1_Rep2	ACTCTCGANNNNNNNNN	CTGTACCA
20160808-AP1SG-S3	Mix1_Rep3	TGAGCTAGNNNNNNNNN	GAACGGTT

Read structure (required):

A read structure is a description of the logical structure of clusters in an Illumina sequencing run. It should consist of integer/character pairs describing the number and type of cycles (T = template, B = sample barcode, M = molecular barcode, and S = skip). For example, if the input data consists of 225-base clusters and we provide a read structure of "100T8B9M8B100T", then the sequence may be split into four reads:

a. Read 1, with 100 cycles (bases) of template
b. Read 2, with 8 cycles (bases) of sample barcode followed by 9 cycles (bases) of molecular barcode (e.g., unique molecular barcode)
c. Read 3, with 8 cycles (bases) of sample barcode
d. Read 4, with 100 cycles (bases) of template

3. Lane (required):
   ExtractIlluminaBarcodes determines the barcode for each read in an Illumina sequencer lane. Therefore, a lane number must be provided as an input.

OUTPUT:

1. Extracted barcode files:
   For each tile on the sequencing flowcell, a barcode file is written to the basecalls directory with the following file naming convention: *s_<lane>_<tile>_barcode.txt*. Barcode files contain the following tab-separated columns:
   a. Read subsequence at barcode position
   b. Y or N indicating the presence of a barcode match
   c. Matched barcode sequence (empty if the read did not match one of the barcodes)
      If there is no match, but it is close to the threshold of calling it a match, the barcode would be output as if a match was called, but indicated with lowercase letters. Threshold values can be adjusted to accommodate barcode sequence mismatches from the reads.
   d. Number of mismatches
   e. Number of mismatches in the second-best barcode

   Example output barcode file (partial file shown):

```
ACGAATTC    Y    ACGAATTC    0    4
ACGTCCTG    Y    ACGTCCTG    0    4
TGCTTCTC    N    ggaatctc    3    5
TAAGATTA    Y    TAAGATTA    0    5
TTATGAAT    Y    TTCTGAAT    1    5
TCAGATTA    Y    TAAGATTA    1    4
CTCGCCTA    N    cttcgcct    5    5
```

2. Extracted barcode metrics file:
   The metrics file produced by the *ExtractIlluminaBarcodes* program indicates the number of matches (and mismatches) between the barcode reads and the actual barcodes. These metrics are provided both per barcode and per lane. For poorly matching barcodes, the order of specification of barcodes can cause arbitrary output differences.

Metrics files contain the following fields:

Field	Description*
BARCODE	The barcode (from the set of expected barcodes) for which the following metrics apply. Note that the "symbolic" barcode of NNNNNN is used to report metrics for all reads that do not match a barcode.
BARCODE_NAME	Barcode name provided in the BARCODE_FILE.
LIBRARY_NAME	Library name provided in the BARCODE_FILE.
READS	The total number of reads matching the barcode.
PF_READS	The number of passing-filter (PF) reads matching this barcode (always less than, or equal to, READS).
PERFECT_MATCHES	The number of all reads matching this barcode that matched with 0 errors or no-calls.
PF_PERFECT_MATCHES	The number of PF reads matching this barcode that matched with 0 errors or no-calls.
ONE_MISMATCH_MATCHES	The number of all reads matching this barcode that matched with 1 error or no-call.
PF_ONE_MISMATCH_MATCHES	The number of PF reads matching this barcode that matched with 1 error or no-call.
PCT_MATCHES	The fraction of all reads in the lane that matched this barcode.
RATIO_THIS_BARCODE_TO_BEST_BARCODE_PCT	The rate of all reads matching this barcode to all reads matching the most prevalent barcode. For the most prevalent barcode, this will be 1; for all others it will be less than 1 (except for the possible exception of when there are more orphan reads than for any other barcode, in which case the value may be arbitrarily large). One divided by the lowest number in this column, is the fold-difference in representation between barcodes.
PF_PCT_MATCHES	The fraction of PF reads in the lane that matched this barcode.
PF_RATIO_THIS_BARCODE_TO_BEST_BARCODE_PCT	The rate of PF reads matching this barcode to PF reads matching the most prevalent barcode. For the most prevalent barcode this will be 1; for all others it will be less than 1 (except when there are more orphan reads than reads matching other barcodes, in which case the value may be arbitrarily large). One divided by the lowest number in this column is the fold-difference in representation of PF reads between barcodes.
PF_NORMALIZED_MATCHES	The "normalized" matches to each barcode. This is calculated as the number of PF reads matching this barcode divided by the average of all PF reads matching any barcode (excluding orphans). If all barcodes are represented equally, this will be 1.

* Table is based on Picard documentation. For the most up-to-date information, please refer to the Picard website at http://broadinstitute.github.io/picard/picard-metric-definitions.html#ExtractIlluminaBarcodes.BarcodeMetric.

B. Demultiplex each lane of Illumina BCL files and include the UMI data in the per-sample BAM files

Picard's *IlluminaBasecallsToSam* program collects, demultiplexes, and sorts reads across all of the tiles of a lane using barcodes generated by Picard's *ExtractIlluminaBarcodes* program, producing an unmapped BAM (uBAM) file. By default, *IlluminaBasecallsToSam* will check the presence of sequencing adapter sequences in the reads and mark them with the *XT* tag in the output BAM file. An example invocation follows:

```
java -Xmx8g -jar picard.jar IlluminaBasecallsToSam \
    BASECALLS_DIR=inputBCLdir \
    BARCODES_DIR=extracted_Barcodes_Files \
    LANE=1 \
    READ_STRUCTURE=100T8B9M8B100T \
    RUN_BARCODE=160808_M01253_AP1SG \
    LIBRARY_PARAMS=library_params.AP1SG.1.txt \
    TMP_DIR=java_io_tmpDir \
    MOLECULAR_INDEX_TAG=RX \
    ADAPTERS_TO_CHECK=INDEXED \
    TAG_PER_MOLECULAR_INDEX=ZA \
    TAG_PER_MOLECULAR_INDEX=ZB
```

INPUT:

1. BASECALLS_DIR (required):
   The following input files from the basecalls directory are needed:
   - *.bcl files
   - *.stats files
   - *.filter files
   - *.control files
   - *.locs files

2. BARCODES_DIR:
   The directory contains the extracted barcodes files (*_barcode.txt* files) generated by *ExtractIlluminaBarcodes*.

3. READ_STRUCTURE (required):
   A read structure is a description of the logical structure of clusters in an Illumina sequencing run. It should consist of integer/character pairs describing the number and type of cycles (T = template, M = molecular barcode, B = sample barcode, and S = skip). For example, if the input data consists of 225-base clusters and we provide a read structure of "100T8B9M8B100T", then the sequence may be split into four reads:
   a. **Read 1, with 100 cycles (bases) of template**
   b. **Read 2, with 8 cycles (bases) of sample barcode followed by 9 cycles (bases) of molecular barcode** (e.g., unique molecular barcode)

c. Read 3, with 8 cycles (bases) of sample barcode

d. Read 4, with 100 cycles (bases) of template

4. LANE (required):
   IlluminaBaseCallsToSam processes data one lane at a time. Therefore, a lane number must be provided as an input.

5. RUN_BARCODE (required):
   The barcode of the run is prefixed to read names.

6. LIBRARY_PARAMS (required):
   All barcode, sample, and library data are provided in the library parameters file as input for running *IlluminaBaseCallsToSam*. The following is an example of a correctly formatted LIBRARY_PARAMS file:

```
OUTPUT                              SAMPLE_ALIAS        LIBRARY_NAME  BARCODE_1           BARCODE_2
/mnt/bam/L001/AP1SG-S1_unmapped.bam 20160808-AP1SG-S1   Mix1_Rep1     CTGATCGTNNNNNNNNN   GCGCATAT
/mnt/bam/L001/AP1SG-S2_unmapped.bam 20160808-AP1SG-S2   Mix1_Rep2     ACTCTCGANNNNNNNNN   CTGTACCA
/mnt/bam/L001/AP1SG-S3_unmapped.bam 20160808-AP1SG-S3   Mix1_Rep3     TGAGCTAGNNNNNNNNN   GAACGGTT
/mnt/bam/L001/Unmatched.bam         Unmatched           Unmatched     N
```

7. MOLECULAR_INDEX_TAG and TAG_PER_MOLECULAR_INDEX:
   If at least one molecular index is present, you can specify the *Molecular_Index_Tag* (default: *RX*). If more than one molecular index is present, you can specify a tag for each molecular index by setting the *TAG_PER_MOLECULAR_INDEX* tag. For example, if two molecular indexes are present, you can use *ZA* and *ZB* for each molecular index tag.

8. ADAPTERS_TO_CHECK:
   This setting allows you to choose which adapters to look for in the read. The default value is [INDEXED, DUAL_INDEXED, NEXTERA_V2, FLUIDIGM]. This option can be set to "null" to clear the default value. Possible values: {PAIRED_END, INDEXED, SINGLE_END, NEXTERA_V1, NEXTERA_V2, DUAL_INDEXED, FLUIDIGM, TRUSEQ_SMALLRNA, ALTERNATIVE_SINGLE_END}. This option may be specified 0 or more times. The *XT* tag stores the position at which the SAM record contains sequencing adapter sequence.

OUTPUT:

The output BAM files are in the location specified in the LIBARAY_PARAMS file. One unaligned BAM file is generated per lane per sample. These BAM files contain molecular index tags with molecular index sequences and *XT* tags for sequencing adapter sequences present in the reads.

C. Convert BAM files to FASTQ files

Picard's *SamToFastq* tool extracts read sequences and base quality scores from the input SAM/BAM file and writes them into the output file in FASTQ format. Sequencing adapter sequences present in the reads in the BAM file can be trimmed before writing to the FASTQ files. An example invocation follows:

```
java -Xmx4g -jar picard.jar SamToFastq \
    INPUT=input.bam \
    FASTQ=S1_R1.fastq \
    SECOND_END_FASTQ=S1_R2.fastq \
    INCLUDE_NON_PF_READS=false \
    CLIPPING_ATTRIBUTE=XT \
    CLIPPING_ACTION=X \
    CLIPPING_MIN_LENGTH=minFragSize
```

INPUT:

1. INPUT (required):
   Input the unaligned BAM file generated by *IlluminaBasecallsToSam*.

2. FASTQ (required):
   The output FASTQ file is single-end FASTQ or, if paired, the first end of the pair FASTQ.

3. SECOND_END_FASTQ:
   The output FASTQ file is, if paired, the second end of the pair FASTQ. The default value is null.

4. INCLUDE_NON_PF_READS:
   Non-PF reads are reads whose "not passing quality controls" flag is set. Non-PF reads from the SAM file are included in the output FASTQ files. The default value is false. Possible values: {true,false}.

5. CLIPPING_ATTRIBUTE:
   The clipping attribute stores the position at which the SAM record should be clipped. The default value is null.

6. CLIPPING_ACTION:
   The clipping action indicates what action should be taken with clipped reads:
   - "X" means the reads and qualities should be trimmed at the clipped position.
   - "N" means the bases should be changed to Ns in the clipped region.
   - An integer means that the base qualities should be set to that value in the clipped region.
   - The default value is null.

7. CLIPPING_MIN_LENGTH:
   CLIPPING_MIN_LENGTH is the minimum read length after clipping with the CLIPPING_ATTRIBUTE and CLIPPING_ACTION parameters. If the original read is shorter than minimum read length, then the original read length will be maintained. The default value is 0. This option can be set to "null" to clear the default value.

OUTPUT:

A single-end FASTQ file or two paired-end FASTQ files will be generated from each unaligned BAM file for each sample per lane. These FASTQ files do not contain any molecular index information.

D. Map FASTQ reads and merge alignment data from aligned BAM files with molecular index data from unaligned BAM files

After aligning the FASTQ files using a short-read aligner, such as Burrow-Wheeler Aligner (bwa) (https://github.com/lh3/bwa), you will need to run Picard's *MergeBamAlignment* tool to merge the alignment data in the aligned BAM file with the Molecular Index Tag data in the unaligned BAM file (generated by Picard's *IlluminaBasecallsToSam* in **step B** above).

Picard's *MergeBamAlignment* produces a new BAM file that includes all aligned and unaligned reads, and also contains additional read attributes from the uBAM that may otherwise be lost during the alignment process. Note that *MergeBamAlignment* expects to find a sequence dictionary in the same directory as REFERENCE_SEQUENCE and expects it to have the same base name as the reference FASTA, but with the extension ".dict". The sequence dictionary can be easily generated using Picard's *CreateSequenceDictionary*. An example invocation follows:

```
java -Xmx4g -jar picard.jar MergeBamAlignment \
    ALIGNED=aligned.bam \
    UNMAPPED=unaligned.bam \
    OUTPUT=merged.bam \
    REFERENCE_SEQUENCE=hg38.fa \
    EXPECTED_ORIENTATIONS=FR \
    MAX_GAPS=-1 \
    SORT_ORDER=coordinate \
    CREATE_INDEX=true \
    ALIGNER_PROPER_PAIR_FLAGS=false
```

INPUT:

1. ALIGNED (required):
   These SAM or BAM file(s) contain alignment data.
2. UNMAPPED (required):
   The original SAM or BAM file contains unmapped reads, which must be in queryname order.

3. OUTPUT (required):
   This is the merged SAM or BAM file that will contain the final data from this alignment process.
4. REFERENCE_SEQUENCE (required):
   You must provide the path to the FASTA file for the reference sequence.
5. EXPECTED_ORIENTATION:
   This is the expected orientation of proper read pairs. Possible values: {FR, RF, TANDEM}. This option may be specified 0 or more times.
6. MAX_GAPS:
   This is the maximum number of allowable insertions or deletions for an alignment to be included. Alignments with more than this many insertions or deletions will be ignored. You may set the MAX_GAPS to –1 to allow any number of insertions or deletions. The default value is 1.
7. SORT_ORDER:
   This specifies the order in which the merged reads will be output. The default value is coordinate. Possible values: {unsorted, queryname, coordinate, duplicate}.
8. CREATE_INDEX:
   This specifies whether or not to create a BAM index when writing a coordinate-sorted BAM file. The default value is false. Possible values: {true, false}.
9. ALIGNER_PROPER_PAIR_FLAGS:
   You may use the aligner software definition of a proper pair rather than compute proper pairs in this program. The default value is false. Possible values: {true, false}.

OUTPUT:

You will receive a new BAM file that includes all aligned and unaligned reads and also contains additional read attributes from the uBAM that may otherwise be lost during the alignment process.

Revision history

Version	Date released	Description of changes
2	June 2018	Clarified instructions for constructing a read structure that matches the library.
1	October 2017	Original analysis guideline.