ILLUMINA READ DATA PARSING

ANDOLFATTO LAB

1. Background

1.1. **What is *read parsing*?** Read parsing is the process of extracting the sequencing reads belonging to a particular project, or even a particular sample, from the complete read set from a sequencing run. After you send your sample out for sequencing, the sequencing facility will provide you with at least two FASTQ files, corresponding to all of the reads (and corresponding *index reads*, see subsection 1.4) from a single run of the sequencer.

1.2. **Why do we parse?** For various reasons, including cost, it is beneficial to sequence multiple individuals, and even multiple projects, in the same sequencing run (a process known as *multiplexing*). However, we cannot just throw the DNA from all these individuals straight into the same tube and expect to be able to tell which reads come from which individuals. In order to distinguish the DNA from different individuals or projects, we must add *index adapters* and/or *barcode adapters* to each sample prior to *pooling*. The process of *demultiplexing* (aka *read parsing*) involves using the sequence information from these adapters to separate reads by sample. Thus, we can sequence many samples in the same run, and afterwards be sure which reads came from which sample.

1.3. **How are sequencing libraries made?** Illumina sequencing libraries are generally prepared using one of two methods: TruSeq and Nextera. These two methods follow the same general procedure of:

(1) Fragmenting the sample DNA (whether genomic DNA or cDNA)
(2) Ligating sequencing adapters (no PCR)
(3) Adding index adapters by PCR
(4) Quality control checking the fragment size of the library and concentration

The differences between these two methods are primarily in steps 1 and 2. In the Andolfatto lab, we often use customized protocols derived from either of these two methods. Specifically, you will often hear the term *Tn5 library* used. This refers to a customized version of the Nextera protocol, which uses the Tn5 transposome in order to fragment the sample DNA (step 1). Because of the way the Tn5 transposome works, no *barcode adapters* are used, and instead **two** sets of *index adapters* are used. If, instead, *MseI* is used for fragmentation, barcode adapters are used with a single index adapter (the barcode adapters identify individuals in a single 96-well plate, and the index adapter identifies the plate and/or project).
Thus, generally a DNA fragment from a completed library will look like this:

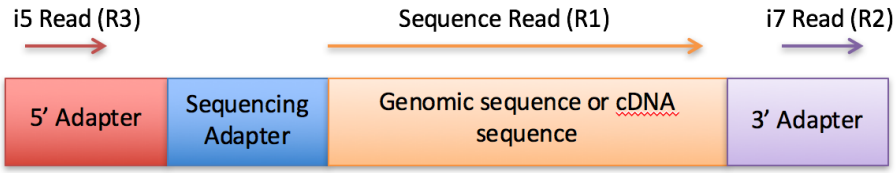

FIGURE 1. Typical library fragment, where the 3' adapter is always an index adapter, but the 5' adapter may or may not be an index adapter.

1.4. **Types of reads.** Due to the way the Illumina HiSeq performs indexed sequencing, a single run can produce between 2 and 4 FASTQ files, corresponding to 1-2 *sequence reads* (1 for single-end, 2 for paired-end), and 1-2 *index reads*. All libraries prepared for use with MSG are single-end libraries, so you will get 2-3 FASTQ files from the sequencing facility. R1 (aka Read 1) is **always** the *sequence read*, and R2 and R3 (aka Reads 2 and 3, or Index Reads 1 and 2) are **always** the *index reads*. Traditionally, the *index adapters* are referred to as the i7 and i5 adapters (i7 is the 3' adapter, and i5 is the *index adapter* version of the 5' adapter). According to the Illumina Dual-indexed sequencing User Guide[1], Index Read 1 (aka R2) thus contains the i7 index sequence, and Index Read 2 (if it exists) contains the i5 index sequence.

2. Pre-parsing information to gather

Prior to submitting your samples for sequencing, you should record several pieces of information critical to proper demultiplexing:

(1) Knowledge of which library preparation protocol you used
(2) A file of some kind specifying which combinations of indices and barcodes correspond to which samples
(3) A rough idea of how many reads you expect per sample

The library prep protocol used determines the number of index reads and the specific index and barcode sequences used for demultiplexing. Often, the index adapters we use correspond to standard Illumina adapters, so you can always look up the index sequences using the Illumina Adapter Sequences Document[2].
This information helps tremendously in diagnosing parsing and sequencing problems. For example, you can diagnose a problem with normalization during library pooling if there is a mismatch between your expected number of reads per sample and the realized distribution.

3. Nomenclature

Here is a quick overview of what the different parts of commands in later sections imply:

`\t`
is the TAB character

`\n`
is the NEWLINE character, produced by pressing the ENTER key

`[file name]`
implies that you should substitute the appropriate file name in this position, e.g.
`cp [file to copy] [place to copy it]`
might become
```
cp /tigress/ANDOLFATTO/Sequencing_raw_data/MyReadSet_R1_001.fastq.gz \
/tigress/ANDOLFATTO/myname/myproject/
```

`This type of text`
typically implies commands to be run or the contents of a file

4. Parsing

4.1. **Index Parsing.** Index parsing consists of two simple steps that are repeated for every pair of sequence read and index read:

(1) Scan the index/barcode file to match (allowing for n possible mismatches) the current index read
(2) Output the sequence read to the appropriate sample file (and also output the index read to its appropriate sample file)

The sequence read and index read files provided by the sequencing facility are always line-matched. This implies that the first sequence read corresponds with the first index read, the second sequence read corresponds with the second index read, and so on. Thus, we can provide the `barcode_splitter.py` script with the sequence read and index read files, and a couple of other parameters including n, and it will iterate through these pairs, splitting the sequence and index reads into individual files per index sequence. That is, if index sequence `ATACGT`, found as the first index read, corresponds to sample 1, and index sequence `CTTAGA`, found as the second index read, corresponds to sample 2, then sequence read 1 will be placed in the sample 1 sequence read file, and sequence read 2 will be placed in the sample 2 sequence read file.

Table 1. Example of how index matching works for the i5 parsing command in subsection 4.2

Sequence read	Index read	Matched Index	Index ID	Output file for sequence read
ACGTATCCGA	ATACGT	ATACGA	Plate1	Plate1_read_1.fastq.gz
TTAGCGCGTA	CTTAGA	CTTACA	Plate2	Plate2_read_1.fastq.gz
TTCACTCAGG	ATACGG	ATACGA	Plate1	Plate1_read_1.fastq.gz
AGGGATTCTC	ATTACA	CTTACA	Plate2	Plate2_read_1.fastq.gz
CGGATTATCG	CTTACA	CTTACA	Plate2	Plate2_read_1.fastq.gz

4.2. **Tn5 Libraries.** Tn5 libraries generally have two sets of index reads, R2 and R3, corresponding to the i7 and i5 indices, respectively. Index parsing requires an index file conforming to the following format:

```
[index ID]\t[index sequence]\n
```

Thus, a typical i5 index file will look like:

```
i5_1\tGGAGGTTT\n
i5_2\tAACGCCAA\n
i5_3\tGCGCTGAT\n
i5_4\tCCTATTTA\n
i5_5\tCCGAGATC\n
i5_6\tATTATTCG\n
i5_7\tCATGCTGC\n
i5_8\tTAGATCAA\n
i5_9\tAATATGAC\n
i5_10\tACGTAAAC\n
i5_11\tAGGGTAAA\n
i5_12\tGATTACTT\n
```

As discussed above, `barcode_splitter.py` allows for index matching within a certain number of mismatches we called n. This same n is used in all calls to `barcode_splitter.py` below as the argument to the "`--mismatches`" parameter. In order to parse Tn5 libraries into plates, we must parse all three read files using the R3 (i.e. i5) indices using the following command (path is specific to Malinche; see the subsection on Della/Tigress paths for Della):

```
/home/guest/bin/barcode_splitter.py --mismatches=[n] --idxread=3 --gzip \
--suffix=".fastq.gz" --bcfile=[i5 index file] [R1 file] [R2 file] [R3 file]
```

This will produce a set of 3 files for each index in the i5 index file, corresponding to the R1, R2, and R3 reads associated with that i5 index. Of course, that means the R3 files are not useful to the end-user after i5 parsing.

After parsing by i5 index, we must parse each pair of i5-parsed R1 and R2 files by their i7 indices using the following command (again, path is specific to Malinche):

```
/home/guest/bin/barcode_splitter.py --mismatches=[n] --idxread=2 --gzip \
--suffix=".gz" --bcfile=[i7 index file] [i5-parsed R1 file] [i5-parsed R2 file]
```

This will produce a set of 2 files for each index in the i7 index file, corresponding to the R1 and R2 reads associated with that i7 index.

Note that if the i5 index file has p lines, and the i7 index file has q lines, this will produce $p \times q$ R1 files, ignoring the "unmatched" files.

Also note that the final R1 files often need to be renamed and processed (e.g. adapter and/or quality trimming) for further analysis pipelines, like MSG.

4.3. **Barcode Parsing.** The distinction between a *barcode* and an *index* is often ignored, but is very important during demultiplexing. Barcode adapters are prepended to the genomic sequence by ligation and contain a pre-defined sequence called the *linker*, whereas index adapters are generally PCRed.

Why does this matter? Well, for the sequencer, it means the barcode sequence is not as easily distinguished from the genomic read (since it is not sequenced separately), and so we need to computationally remove the barcode sequence and the linker region simultaneously to demultiplex individuals from a plate.

What do barcoded libraries look like? Take a look:

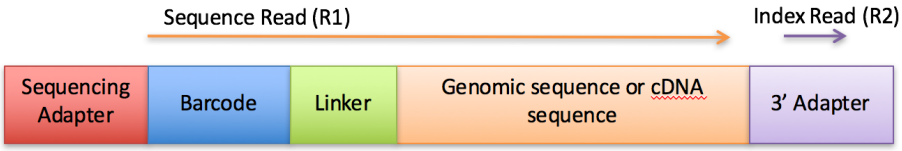

FIGURE 2. A typical fragment from a barcoded library. Note the position of the sequencing adapter 5' of the barcode, linker, and genomic sequence such that the sequence read (R1) captures them all.

4.4. **MseI Libraries.** MseI libraries are prepared by performing a restriction digest of genomic DNA using the MseI enzyme, which recognizes the sequence:

```
5'-T^TAA-3'
3'-AAT^T-5'
```

In order to incorporate these sticky-end fragments into a library, we use adapters with complementary sticky ends, namely the barcode adapters, and FC2 adapters. The FC2 adapters are specially-modified i7 adapters, so it is important to make sure that the index sequence of the FC2 adapter you use is **NOT** the same as **ANY** i7 index sequence in **ANY** other library being sequenced in the same lane. This is because MseI libraries do not use an i5 adapter, so R3 will be gibberish for MseI library reads.

Clearly there are only a few FC2 adapters (around 12), so these are used to identify 96-well plates. To identify the well within a 96-well plate, we use barcode adapters. Each individual is uniquely identified by a combination of i7 index sequence and barcode sequence. So when parsing MseI libraries from a run, the first task is i7 parsing, to isolate

the MseI library from all other libraries in the lane.

Once the MseI plate is parsed out, we use a script specially written for the barcode adapters used here in the Andolfatto lab (and more specifically, designed for MSG library preparations) called `parse_BCdata2BWA.pl` in order to demultiplex by barcode, and strip the sequence reads (R1) down to only the genomic read after the MseI cut site. That is, the script scans each read for the linker region sequence, and upon recognition, searches for the barcode sequence. When the barcode sequence is matched, the script deletes the barcode, linker, and restriction site from the read, leaving only the genomic read, and places this in the appropriate file for that individual.

Incidentally, as this script is intended for use with MSG, it produces files formatted for MSG (e.g. `indiv15_AGTTCT`), and an extra set of files used to compile parsing statistics (e.g. `junk, bad_barcodes`, etc.).

5. Recommendations

5.1. **Quality Control.** Quality control should be performed after parsing, but may additionally be performed prior to parsing. This involves checking for biases in the sequencing data or quality scores, typically using FastQC[3]. FastQC is available on both Malinche and Della/Tigress (see section 7).

5.2. **Pre-parsing.** One of the simplest diagnostics possible prior to parsing is to generate a sorted histogram of various indices and compare these read counts to your expected read counts. For instance, generating a sorted histogram of i5 indices and comparing the results to expected read counts per plate can quickly show that a plate was not pooled properly, or did not sequence successfully. The same can be done for i7 indices after i5 parsing to see if individuals from a plate had the aforementioned problems, or before i5 parsing to see if other sequencing or library preparation problems occurred. A quick script for generating histograms for $n = 0$ is:

```
zcat [gzipped FASTQ file] | awk 'NR%4==2' | sort | uniq -c | sort -k1nr \
> [histogram TSV file]
```

5.3. **During parsing.** One minor but potentially important strategy to maximize the yield of demultiplexing is to perform a second round of parsing on the "unmatched" files. Oftentimes, it is easiest to do a first pass demultiplexing with "`--mismatches=0`", and then perform a second round of parsing on the unmatched reads while increasing n. It may also be useful to perform i7 parsing on the i5 unmatched, then i5 parsing on the results of the i7 parsing, since the i7 parsing filters out most of the i5 noise.

5.4. **Post-parsing.** Most downstream analysis pipelines require that the raw reads produced by demultiplexing be adapter and/or quality trimmed. In the Andolfatto lab, we use `TQSfastq_gz.py` for quality trimming, using a threshold of quality score Q20 over 30 consecutive bases (i.e. the output files will have an additional suffix "`_T20C30.trim.fastq.gz`").

The command for quality trimming is thus (on Malinche):

```
/home/guest/bin/TQSfastq_gz.py -f [FASTQ file] -q -t 20 -c 30 -z
```

A consensus has yet to be reached regarding adapter trimming software, but `cutadapt` and its wrapper Trim Galore! are suggested.

6. Troubleshooting

6.1. `barcode_splitter.py` **is taking a long time.** If `barcode_splitter.py` is taking longer than a day to parse your Illumina data, you may have run into the pathological case for that program, where none or very few of the index reads match your barcode list. `barcode_splitter.py` is very slow when it has to add the majority of your reads into the `unmatched_read_[123].fastq.gz` files, and this also indicates something is wrong with your barcode file. The most common error here is putting the reverse complement of the index into your barcode file, which generally only happens for the i7 parsing step. Recalling that the i7 adapter sequence's 5‘ end attaches to the 3‘ end of the genomic DNA insert, the i7 index is sometimes reported in the 5‘ to 3‘ direction. However, when Index Read 1 (aka Read 2) occurs, the sequencing primer binds on the opposite strand, so your i7 index read will actually be the reverse complement of the reported i7 index sequence.

Thus, the simple solution is to reverse complement all of the index sequences in your i7 barcode file, and re-run the i7 parsing step.

You should be able to tell if reverse complementing is a problem even before i7 parsing by generating a histogram of the i7 index reads, and comparing this to your i7 barcode file. Thus, it is **ALWAYS** recommended that you generate at least one i5 histogram and one i7 histogram before performing the associated parsing, and cross-validate your barcode files against these histograms.

7. Program paths

7.1. **Malinche.**

- **FastQC**: `/home/guest/bin/FastQC/fastqc`
- **barcode_splitter.py**: `/home/guest/bin/barcode_splitter.py`
- **TQSfastq_gz.py**: `/home/guest/bin/TQSfastq_gz.py`

7.2. **Della/Tigress.**

- **FastQC**: `/tigress/ANDOLFATTO/bin/FastQC/fastqc`
- **barcode_splitter.py**: `/tigress/ANDOLFATTO/patrick/barcode_splitter.py`
- **TQSfastq_gz.py**: `/tigress/ANDOLFATTO/bin/TQSfastq_gz.py`
- **parse_BCdata2BWA.pl**: `/tigress/ANDOLFATTO/MSG.custom/msg/parse_BCdata2BWA.pl`
- **cutadapt (version 1.7.1)**: `/tigress/ANDOLFATTO/bin/cutadapt-1.7.1/bin/cutadapt`
- **Trim Galore! (version 0.4.0)**: `/tigress/ANDOLFATTO/bin/trim_galore_zip/trim_galore`

8. Glossary

- **Adapter**: A piece of DNA ligated or PCRed onto the end of a DNA fragment, containing specialized sequence used by the sequencer (e.g. to attach the fragment to the flowcell, or bind the sequencing primer)
- **Barcode**: A short DNA sequence near the linker region of the sequencing adapter that is used (often in combination with an index) to identify the sample of origin
- **Barcode adapter**: A special type of sequencing adapter that contains a sequencing primer binding site, barcode sequence (usually 6 bp), and linker
- **Index**: A short DNA sequence within the index adapter that is used to identify the sample or project of origin; multiple indices (indexes) can be used combinatorically to multiplex a large number of samples
- **Index adapter**: A special type of adapter that contains a binding site for the index read sequencing primer followed by the index sequence (6-8 bp), and some extra sequence for binding to the sequencing adapter
- **Index read**: A read produced by the sequencer that contains an index sequence; the FASTA/Q header of an index read matches the FASTA/Q header of its corresponding sequence read (except for a one-character difference)
- **Linker**: A segment of DNA used to connect a complementary sticky end (for ligation to restriction enzyme-cut genomic DNA or cDNA) with the sequencing adapter; the final pattern is: sequencing adapter-¿barcode-¿linker-¿sticky end-¿genomic DNA or cDNA
- **Multiplexing**: Combining sequencing libraries from multiple samples, identifying the samples with a combination of index adapters (and sometimes barcode adapters)
- **Read parsing**: (aka demultiplexing) Using index (and sometimes barcode) sequences to separate out the sequence reads corresponding to different samples from the main sequence read file from the sequencer
- **Pooling**: The physical process of combining multiple sequencing libraries into a single tube or well
- **Sequence read**: A read produced by the sequencer that contains the sequence of genomic or cDNA of interest; may also contain the barcode sequence and a linker sequence, if a barcode adapter was used

References

[1] Illumina. Sequencing Dual-Indexed Libraries on the HiSeq System - User Guide;. Available from: `https://support.illumina.com/content/dam/illumina-support/documents/myillumina/7f899767-773e-48d7-b03e-d126c80943dc/dualindexedlibraries_onhiseq_15032071_b.pdf`.

[2] Illumina. Illumina Adapter Sequences;. Available from: `https://support.illumina.com/downloads/illumina-customer-sequence-letter.html`.

[3] Andrews S. FastQC: A quality control tool for high throughput sequence data.;. Available from: `http://www.bioinformatics.babraham.ac.uk/projects/fastqc/`.