10X GENOMICS SUPERNOVA PREP MANUAL

ANDOLFATTO LAB

1. Background

1.1. Why 10x Genomics?

1.2. ?

1.3. ?

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[**?**], 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[**?**].
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. What does HudsonAlpha typically send us?

4.1.1. *QC data.* HudsonAlpha sends several metadata and QC data files prior to sending the sequencing data. These consist of:

(1) The sample submission sheets, as printed out when you sent your HMW DNA samples
(2) An initial QC spreadsheet including a gel image checking the length and quality of your input DNA
(3) A Chromium spreadsheet, which gives data on various library characteristics produced by the Chromium machine
(4) Bioanalyzer electropherograms for your libraries
(5) A Kapa qPCR quantification spreadsheet, giving a final piece of QC information for pooling of your libraries before putting them on the clusterbot.

4.1.2. *Sequencing data.* Depending on what is run on the lane at HudsonAlpha, and which sequencer your 10x Chromium library is run on, HudsonAlpha may provide us access to the file in a number of ways:

(1) Libraries parsed by full i7 index sequence using bcl2fastq2 (if there are non-10x libraries on the lane)
(2) Libraries parsed by quartets of i7 index sequences using 10x Genomics LongRanger (the

`longranger basic`

command).

I mention 'quartets' of i7 indices because every 10x Chromium lane ends up with (4) i7 indices. These can be identified in the 'Indexes' sheet of the "Chromium worksheet" Excel spreadsheet provided by HudsonAlpha. However, as long as the reads are either parsed into separate files per i7 index, or have the i7 index as part of their FASTQ header line (which is how LongRanger does it), Supernova will be able to distinguish between reads from the (4) i7 indices.

5. Adapter Trimming (optional)

You may choose to adapter-trim your read data prior to running Supernova. Some have reported assembly improvements after adapter-trimming, while others have seen negligible or negative impacts on assembly contiguity. If you're going to adapter-trim your data, make sure of two things:

(1) Read pairs remain synchronized between the two read files (this is VERY important if using cutadapt, but is easy to ensure with Trim Galore!, and is possible with other adapter-trimming software)
(2) Read pairs do not desynchronize from their indices (if your read files already have the i7 indices in their FASTQ header lines, you don't need to worry about this)

And one final thing to note:
Supernova generates an initial de Bruijn Graph using kmers with k=48, and there is a 16 bp barcode at the beginning of all R1 reads, so every read after adapter-trimming MUST be of length 64 bp or greater. Otherwise, Supernova will get almost all the way through, giving you hope, and then segfault during the final steps, thereby rending your hopes asunder. So, if you use Trim Galore!, be sure to set

`--length 64`

or higher.

6. Running Supernova

6.1. **supernova run.** Supernova is a fairly simple program to run. In fact, the command to run the assembly has a grand total of 4 arguments. For Supernova 1.1.x, the command is:

```
supernova run --sample [Sample ID] --fastqs [Path to FASTQs] \
--description [Description of assembly] --id [Assembly directory name]
```

For Supernova 1.0.x, the command is:

```
supernova run --fastqprefix [Sample ID] --fastqs [Path to FASTQs] \
--description [Description of assembly] --id [Assembly directory name]
```

You can also optionally include the

`--maxreads [# reads to use]`

argument, which limits the number of reads Supernova uses for the assembly. 10x Genomics officially says that 40-56x is optimal for Supernova, although some testing with *Drosophila melanogaster* yw ISO8 data indicates that the higher, the better, even beyond 56x.

This is the single longest step in the entire process, and also the most memory-consuming. Most assemblies we have run peaked at 250 GB RAM, and can take up to 5 days on 20 cores.

A preliminary report on various assembly statistics can be found in the in:

`[Path to directory specified by --id]/outs/report.txt`

This will provide an idea of the weighted mean molecule length, median number of reads per barcode, rough assembly size, edge N50, phaseblock N50, scaffold N50, and a rough number of scaffolds $\geq$ 10kb. You can use these contiguity statistics to compare different pre-processing and read subsetting effects on Supernova assemblies.

6.2. **supernova mkoutput.** But you're not quite done yet. You still need to make a FASTA of your assembly. This is where the

`supernova mkoutput`

(in Supernova 1.0.x,

`supernova mkfasta`

) command comes in. There are a variety of different options at play here:

- `--asmdir [Path to --id directory]/outs/assembly`
- `--outprefix [Prefix to output FASTA file]`
- `--style [Style of FASTA to output]`
- `--minsize [Minimum size of scaffold to output, default 1000]`

You can fiddle with the

`--minsize`

parameter if your assembly seems larger or smaller than expected, although doing so is at your own risk. The

`--style`

parameter will provide you assemblies at different scales:

(1) `raw`
    gives you the edges in the Supernova graph, which are roughly equivalent to DBG unitigs
(2) `megabubbles`
    gives you edge paths corresponding to larger haplotype bubbles ('megabubbles'), separate from intervening roughly-homozygous sequence and from each other
(3) `pseudohap`
    arbitrarily phases megabubbles relative to surrounding roughly-homozygous sequence as well as relative to each other, producing a pseudo-haploid assembly
(4) `pseudohap2`

also arbitrarily phases the megabubbles, but produces a pseudo-diploid assembly instead (i.e. every scaffold has a dual scaffold corresponding to the alternate mixed haplotype)

In general, you will want to use the

`pseudohap`

option for this.

Also note that Supernova 1.1.x produces a GZIPped FASTA file, rather than a plaintext FASTA, so be sure to run

`gunzip`

on the file produced by

`supernova mkoutput`

before proceeding any further.

7. Recommendations

7.1. **Post-assembly improvements.** It is strongly suggested that you correct SNP and indel misassemblies using Pilon, and if possible, make a dotplot of your assembled genome against a closely related reference genome using MUMmer.

Another good assembly quality check is mapping proteins from a well-annotated related reference genome to your assembly using Exonerate 2.4.0, and then checking for the fraction of mapped proteins that contain a frameshift. A good assembly will tend to have $\leq 10\%$ of proteins containing a frameshift.

You may also consider attempting to fill the gaps in your assembly using the Illumina paired-end data you produced. This can be accomplished with a variety of different software, including GapFiller (from BaseClear) and GapCloser (found in SOAPdenovo2, although also available as a separate package).

If you really want your assembly in chromosome form, and there is a closely related reference genome assembled into chromosome form, you should try aligning the assemblies with progressiveCactus, and then superscaffolding your assembly using Ragout (which uses the alignment produced by progressiveCactus). Your mileage may vary here, depending on the degree of divergence between the chromosome-scale reference and your assembly.

8. Troubleshooting

Generally, Supernova error messages aren't terribly informative. Most of the time, your errors are going to be due to poorly documented limitations on argument values. For instance, you might encounter an error due to using periods in any IDs passed in. Other errors may include incorrect FASTQ file names, desynchronized FASTQ files

9. Program paths

9.1. **gen-comp1.**

- **Supernova 1.0.0**: `/Genomics/grid/users/preilly/bin/supernova-1.0.0/`
- **Supernova 1.1.3**: `/Genomics/grid/users/preilly/bin/supernova-1.1.3/`

10. 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