PoreCamp2016 : De novo and hybrid assembly

0 Overview

By the end of this tutorial, you will be able to generate a de novo assembly of your nanopore reads, improve the assembly with nanopolish, and evaluate the quality of your assembly. You will also do a hybrid nanopore/Illumina assembly.

WARNING: Copying and pasting from this document into a terminal window seems to introduce strange characters, so if something doesn't work, please type in the commands from scratch.

1 Background

Sequence assembly is the process by which we infer the original sequence of the DNA (or RNA) molecule. Ideally, we would be able to recover the linear or circular sequence of all chromosomes present in the sample would be recovered in their entirety, with 100% accuracy, using a de novo method that does not use any information other than what is in your set of reads derived from your sample. Long reads are useful for resolving the order of nonunique or repetitive stretches of DNA, but high per read error rates make it harder to call each nucleotide accurately.

Additional information:

- Assembly and error correction for nanopore reads : `http://porecamp.github.io/2015/pdf/151215_jts_porecamp.pdf` (Jared Simpson's slides from PoreCamp2015)
- De novo genome assembly: what every biologist should know : `http://www.nature.com/nmeth/journal/v9/n4/full/nmeth.1935.html`
- Assembly Algorithms for NextGeneration Sequencing Data : `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2874646/`
- CANU assembler (a fork of the Celera assembler for long reads) : `http://canu.readthedocs.io/en/stable/#`
- Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences : `http://bioinformatics.oxfordjournals.org/content/early/2016/03/18/bioinformatics.btw152`
- SPAdes: A New Genome Assembly Algorithm and Its Applications to SingleCell Sequencing : `http://online.liebertpub.com/doi/abs/10.1089/cmb.2012.0021`

2 Miniasm de novo assembly on GroupA data

- Log into CLIMB
- Create a directory for your files and 'cd' (change directory into it)
```
cd
mkdir -p Tutes/Assembly
cd Tutes/Assembly
```
- Create a subsample of NUMBER (e.g., 10000 or 0.8 of the records in the fastq file) fastq records
```
/home/ubuntu/.linuxbrew/bin/seqtk seq -f 0.8 \
/data2/minion/R9/GroupA/Ecoli/Ecoli.pass.1400.fastq \
> Ecoli.pass.fastq
```

- Generate overlaps
```
minimap -Sw5 -L100 -m0 -t8 Ecoli.pass.fastq Ecoli.pass.fastq | \
gzip -1 > Ecoli.pass.reads.paf.gz
```
- Generate layout
```
miniasm -f Ecoli.pass.fastq Ecoli.pass.reads.paf.gz \
> Ecoli.pass.contigs.gfa
```
- Convert to FASTA
```
awk '/^S/{print ">"$2"\n"$3}' Ecoli.pass.contigs.gfa | fold \
> Ecoli.pass.contigs.fa
```
- Run QUAST to evaluate the quality of the assembly (open quast_results/report.html)
```
quast.py -R /data2/refdata/NC_000913.fna Ecoli.pass.contigs.fa
```

3 Use nanopolish to improve the assembly

- Create a bwa index for the Miniasm assembly so you can use it as a “reference”
```
bwa index Ecoli.pass.contigs.fa
```
- Use bwa mem to align the reads to the assembly, then sort by coordinate and save as BAM
```
bwa mem -x ont2d -t 8  Ecoli.pass.contigs.fa Ecoli.pass.fastq \
| samtools view -Sb - \
| samtools sort -o Ecoli.pass.sorted.bam -
```
- Create a BAM index file
```
samtools index Ecoli.pass.sorted.bam
```
- Check your alignment looks sensible (using tview here, but IGV or Savant would look nicer)
```
samtools tview Ecoli.pass.sorted.bam Ecoli.pass.contigs.fa
```
- Copy the nanopolish model files into the working directory to here
```
cp -p /home/ubuntu/src/nanopolish /etc/r9-models/* .
```
- Create a symbolic link to the “downloads” directory mentioned in the FASTQ header of the FASTQ reads file
```
cat Ecoli.pass.fastq \
| sed "s,downloads,\/data2\/minion\/R9\/GroupA\/Ecoli\/downloads,g" \
> Ecoli-abspath.pass.fastq
```
- Convert FASTQ to FASTA
```
seqtk seq -q Ecoli-abspath.pass.fastq > Ecoli-abspath.pass.fasta
```
- Align the reads to the assembly in event space: the reads are the FASTQ nanopore reads the bam file is the nanopore reads mapped to the de novo assembly contigs
```
/home/ubuntu/src/nanopolish-master/nanopolish eventalign \
-t 8 \
--sam \
--reads Ecoli- abspath.pass.fasta \
--bam Ecoli.pass.sorted.bam \
--genome Ecoli.pass.contigs.fa \
--models nanopolish_models.fofn \
| samtools view -Sb - \
| samtools sort -o Ecoli.pass.eventalign.sorted.bam -
```
- Set an environment variable to tell Python where to look for BioPython package export
```
PYTHONPATH=/usr/local/lib/python2.7/dist packages
```
- Merge the individual segments together into the final assembly
```
python \
/nanodata/home/ubuntu/software/nanopolish/scripts/nanopolish_merge.py
\
polished.*.fa > Ecoli.pass.polished_genome.fa
```

- Run QUAST to evaluate the quality of the polished assembly

```
quast.py -R /data2/refdata/NC_000913.fna Ecoli.pass.polished_genome.fa
```

4 Repeat the process with the CANU assembler

CANU gives better de novo assemblies, but it might take 1.52 hours to run for our example data. So later, when you have time, perhaps try to replicate the above analysis with CANU instead of Miniasm.

- Create a new directory for our files

```
mkdir canuassembly
cd canuassembly
```

- Invoke CANU without any arguments to see the usage message. Then, generate an assembly

```
nohup nice \
canu -p Ecoli -d Ecoli_pass_canu -genomeSize=4.8m -nanopore- raw \
../Ecoli-abspath.pass.fastq &> canu.nohup.out &
```

- As before, run QUAST on the output assembly file ASSEMBLY.fasta, then run nanopolish, and run QUAST again on the polished_genome.fa 5 Hybrid assembly with SPAdes You could try doing a hybrid assembly, map the reads back to the assembly, then view the aligned reads to see how they compare.

```
nohup nice \
spades.py --only-assembler -k 21,51,71 \
/data/ref/EcoliK12MG1655_1.fastq.gz  \
/data/ref/EcoliK12Mg1655_2.fastq.gz  --nanopore Ecoli-abspath.fastq \
-o SPADES &> spades.nohup.out &
```