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.

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 non-unique 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 Next-Generation 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 Single-Cell Sequencing : `http://online.liebertpub.com/doi/abs/10.1089/cmb.2012.0021`

2 Data, software and directories

MinION R9 data (pass only)
`/data2/minion/R9/Ecoli/EcoliLambda/downloads/pass/*.fast5`

Reference sequence for *E. coli* K-12 strain MG1655 (NC_000913 from NCBI RefSeq):
`/data2/minion/R9/Ecoli/EcoliLambda/references/NC_000913.fna`

The programs we'll be using are in these directories, but if you just type the program name without the absolute path, it should work (because these directories are already in your PATH environment variable).

The following assumes you are part of analysis group A, and pair-group 1 (so you should change this as appropriate).

```
# Create environment variables to reduce typing when changing directories
export TUTE=/data2/analysis/GroupX/Testing

# create a working dir
mkdir -p $TUTE
cd $TUTE
```

3 De novo assembly with Miniasm and nanopolish

```
# Make sub-dir for this part of the tutorial
mkdir 01-miniasm
cd 01-miniasm

# Create symbolic links to the input and reference files
ln -s \
/data2/minion/R9/Ecoli/R9_Ecoli_K12_MG1655_lambda_MinKNOW_0.51.1.62/R9_Ecoli_K12_MG1655_lambda_MinKNOW_0.51.1.62-2D-pass-20k.fastq \
Ecoli_2d_pass.fastq
ln -s
\/data2/minion/R9/Ecoli/R9_Ecoli_K12_MG1655_lambda_MinKNOW_0.51.1.62/R9_Ecoli_K12_MG1655_lambda_MinKNOW_0.51.1.62-2D-pass-20k.fasta \
Ecoli_2d_pass.fasta
ln -s \
/data2/minion/R9/Ecoli/R9_Ecoli_K12_MG1655_lambda_MinKNOW_0.51.1.62/references/NC_000913.fna \
.

# Count number of records in the FASTQ file
# N.B. We do this the slightly difficult way because the base quality
# line of a FASTQ file could legitimately start with the '@' character.
grep -B 2 '^+$' Ecoli_2d_pass.fastq | grep '^@' | wc -l

# What does the read length distribution look like?
# Extract the lengths of the reads
cat Ecoli_2d_pass.fastq \
| awk '{if(NR%4==2) print length($1)}' \
> Ecoli_2d_pass.readlen
# Then you could use R to plot a histogram
R
data <- read.table('Ecoli_2d_pass.readlen', header=TRUE)
```

```
hist(data$readlen, breaks=100)
q()
n
```

```
# Overlap
minimap -Sw5 -L100 -m0 -t8 \
Ecoli_2d_pass.fastq Ecoli_2d_pass.fastq \
| gzip -1 \
> Ecoli_2d_pass.reads.paf.gz
```

Question: How long did it take to run this command in wall-clock seconds (real time) and CPU seconds?

```
# Layout
miniasm -f Ecoli_2d_pass.fastq Ecoli_2d_pass.reads.paf.gz \
> Ecoli_2d_pass.contigs.gfa
```

We need to convert the unitigs file into a FASTA file:

```
awk '/^S/{print ">"$2"\n"$3}' Ecoli_2d_pass.contigs.gfa | fold >
Ecoli_2d_pass.contigs.fa
```

Now, let's use QUAST to evaluate the assembly.

```
quast -R NC_000913.fna Ecoli_2d_pass.contigs.fa
```

And transfer it to your laptop to look at the results in `quast_results/report.html`

```
# On the CLIMB server
tar -zcvf quast_results.tgz quast_results/
# On your laptop
scp ubuntu@147.188.173.NN:/path/to/01-miniasm/quast_results.tgz .
tar -zxvf quast_result.tgz
```

Question: How good is your assembly?

Now, polish the assembly with nanopolish, generate a consensus assembly, and comment on how it compares with the first assembly

```
# Index the reference genome, which in this case, is the assembly you just made with Miniasm
bwa index Ecoli_2d_pass.contigs.fa
```

```
# Align the nanopore reads to the assembly you just made with Miniasm
bwa mem -x ont2d -t 8  Ecoli_2d_pass.contigs.fa Ecoli_2d_pass.fastq \
| samtools view -Sb - \
```

```
| samtools sort -o Ecoli_2d_pass.sorted.bam -
samtools index Ecoli_2d_pass.sorted.bam

What does your alignment look like? For a basic look at your file, try samtools (you can try
IGV, which is nicer and graphical, later):
samtools tview Ecoli_2d_pass.sorted.bam Ecoli_2d_pass.contigs.fa

# Copy the nanopolish model files into the working directory
cp -p /home/ubuntu/src/nanopolish-master/etc/r9-models/* .

# You might have to fix theEcoli_2d_pass.fasta file to have the absolute
# path name in the FASTA header, and save this file as Ecoli_2d_pass_abspath.fasta

# Align the reads in event space. Here, the reads are the FASTQ nanopore reads
# the  bam file is the nanopore reads mapped to the de novo assembly contigs
~/src/nanopolish-master/nanopolish eventalign -t 8 --readsEcoli_2d_pass_abspath.fasta
--bam Ecoli_2d_pass.sorted.bam --genome Ecoli_2d_pass.contigs.fa --models
nanopolish_models.fofn | samtools view -Sb - | samtools sort -o
Ecoli_2d_pass.eventalign.sorted.bam -

# Generate a consensus sequence
python /PATH/TO/nanopolish_makerange.py Ecoli_2d_pass.contigs.fa \
| parallel --results nanopolish.results -P 8 \
nanopolish variants --consensus polished.{1}.fa -w {1} -rEcoli_2d_pass_abspath.fasta
-b reads.sorted.bam -g draft.fa -e reads.eventalign.sorted.bam -t 4
--min-candidate-frequency 0.1 --models nanopolish_models.fofn

# After all polishing jobs are complete, you can merge the individual segments
# together into the final assembly:

python /PATH/TO/nanopolish_merge.py polished.*.fa > polished_genome.fa

# Now you can run QUAST and compare the Miniasm assembly to the Miniasm+nanopolish
assembly
```

4 Repeat the process with the CANU assembler

CANU probably gives better de novo assemblies, but it might take 1.5-2 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

```
cd ../
mkdir 02-canu
cd 02-canu
ln -s /data2/minion/R9/Ecoli/EcoliLambda.2d.pass.fastq \
EcoliLambda_2d_pass.fastq
```

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

```
canu -p EcoliLambda -d EcoliLambda_canu \
genomeSize=4.8m \
```

```
-nanopore-raw EcoliLambda.2d.pass.fastq
```

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.

```
spades.py --only-assembler -k 21,51,71 -1 ILLUMINAREADS_1.fastq.gz -2
ILLUMINAREADS_2.fastq.gz --nanopore PREFIX.fasta -o SPADES &
```