VERSION 1
DEC 15, 2020

OPEN ACCESS

DOI:
dx.doi.org/10.17504/protocols.io.bqq6mvze

Protocol Citation: David A Eccles 2020. Plasmid Sequence Assembly from Long Reads. **protocols.io** <https://dx.doi.org/10.17504/protocols.io.bqq6mvze>

License: This is an open access protocol distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Protocol status: In development
We are still developing and optimizing this protocol

Created: Dec 15, 2020

Last Modified: Dec 15, 2020

PROTOCOL integer ID:
45566

Plasmid Sequence Assembly from Long Reads V.1

David A Eccles¹

¹Malaghan Institute of Medical Research (NZ)

David A Eccles
Malaghan Institute of Medical Research (NZ)

ABSTRACT

This protocol demonstrates how to assemble reads from plasmid DNA, and generate a circularised and non-repetitive consensus sequence

At the moment, this protocol uses Canu to de-novo assemble high-quality single-cut reads.

Input(s):

- basecalled fastq files in *barcodeXX* directories, as produced from a rapid barcoding sequencing protocol (e.g. RBK-004 + MinKNOW)

Output(s):

- Consensus sequence per barcode as a fasta file

Read file preparation

- 1 This protocol assumes a base directory that includes the *fastq_pass* and *fastq_fail* folders, with barcode-split reads inside them.

If this has been done, then the following command should produce output without errors:

```
ls -d fastq_pass/barcode??
```

Example output:

```
fastq_pass/barcode01 fastq_pass/barcode03 fastq_pass/barcode05
fastq_pass/barcode07 fastq_pass/barcode11 fastq_pass/barcode02
fastq_pass/barcode04 fastq_pass/barcode06 fastq_pass/barcode09
fastq_pass/barcode12
```

- 2 Create a directory to store results files

```
mkdir results
```

- 3 Determine the N50/L50 read length for each barcode. This will be used as the initial guess at the assembly size.

```
(for bc in $(basename -a $(dirname $(ls fastq_pass/*/))) | uniq);
do echo -n ${bc};
fastx-length.pl fastq_pass/${bc}/*.fastq 2>&1 > /dev/null | \
grep L50 | awk '{print "\t"$5}';
done) > results/read_L50.txt
```

This file can be viewed to confirm the assembly lengths:

```
cat results/read_L50.txt
```

Example output:

barcode03	8.875
barcode04	8.863
barcode05	10.218
barcode07	11.06
barcode09	11.069
unclassified	10.177

Note: this file will be used for subsequent downstream processing. If any of these barcodes shouldn't be processed further, feel free to remove the corresponding lines from this file

Read filtering

- 4 Filter out any reads that are less than half of the target read length, and determine the average quality of the remainder, keeping information on (at most) 200 of the highest-quality reads:

```
cat results/read_L50.txt | while read bc len;
do echo ${bc} ${len};
~/scripts/fastx-compstats.pl fastq_pass/${bc}/*.fastq | sort -t
',' -k 16rn,16 | \
    awk -F ',' -v "len=${len}" '{if($18 > (len * 500)){print}}' | \
    head -n 200 | grep -v '^name' >
results/bestLong_200X_${bc}.csv;
done
```

Check how successful the sequencer was at getting 200X coverage by counting lines:

```
wc -l results/bestLong_200X_*.csv
```

Example output:

```
146 results/bestLong_200X_barcode03.csv
160 results/bestLong_200X_barcode04.csv
124 results/bestLong_200X_barcode05.csv
200 results/bestLong_200X_barcode07.csv
200 results/bestLong_200X_barcode09.csv
200 results/bestLong_200X_unclassified.csv
1030 total
```

- 5 Subset the original read sets to only include the high-quality long reads:

```
cat results/read_L50.txt | while read bc len;
do echo ${bc} ${len};
~/scripts/fastx-fetch.pl -i results/bestLong_200X_${bc}.csv \
fastq_pass/${bc}/*.fastq > results/bestLong_200X_${bc}.fastq;
done
```

Canu Assembly

- 6 Run a Canu assembly on each read set, using default options:

```
cat results/read_L50.txt | while read bc len;
do canu -nanopore results/bestLong_200X_${bc}.fastq \
-p ${bc} -d results/canu_${bc} genomeSize=${len}k;
done
```

- 7 Trim the assembled contigs based on Canu's trim recommendations:

```
cat results/read_L50.txt | while read bc len;
do echo ${bc} ${len};
samtools faidx results/canu_${bc}/${bc}.contigs.fasta \
$(grep '^>' results/canu_${bc}/${bc}.contigs.fasta | \
perl -pe 's/^>(.*?) .*trim=(.*)/$1:$2/' >
results/circTrimmed_${bc}.fasta;
done
```