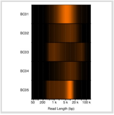


Oct 03, 2019

Demultiplexing Nanopore reads with LAST V.6

David Eccles[1]

[1]Malaghan Institute of Medical Research (NZ)

In Development dx.doi.org/10.17504/protocols.io.7vmhn46

David A. Eccles
Malaghan Institute of Medical Research (NZ)

ABSTRACT

This protocol is for a semi-manual method for read demultiplexing, as used after my presentation *Sequencing DNA with Linux Cores and Nanopores* to work out the number of reads captured by different barcodes.

Input: reads as a FASTQ file, barcode sequences as a FASTA file
Output: reads split into single FASTQ files per target [barcode]

Note: barcode / adapter sequences are not trimmed by this protocol

EXTERNAL LINK

https://doi.org/10.5281/zenodo.2535894

Generating Barcode Index

1 Prepare a FASTA file containing barcode sequences (see attached FASTA file). To reduce the chance of mismatched adapters, this should *only* contain the barcode sequences. That restriction means this approach will not work for short reads, where the barcode sequences are very likely to occur within sequences.

barcode_base.fa

2 Prepare the LAST index for the barcode file. Following Martin Frith's recommendation, the '-uNEAR' seeding scheme is used to slightly increase sensitivity. This will generate seven additional files of the form <index name>.XXX:

```
lastdb -uNEAR barcode_base.fa barcode_base.fa
```

3 Prepare a substitution matrix for barcode mapping. The default substitution matrix is swayed too much by INDELs in the barcode sequences, so here's one that I've developed using a combination of trial & error and last-train:

```
#last -Q 0
#last -a 10
#last -A 10
#last -b 5
#last -B 5
#last -S 1
# score matrix (query letters = columns, reference letters = rows):
       A      C      G      T
A      4    -24     -9    -24
C    -24      5    -24    -14
G     -9    -24      7    -24
T    -24    -14    -24      8
```

bc.mat

This matrix has a moderate penalty for opening gaps (i.e. insertions and deletions), and a lower penalty for inserting them. Insertions and deletions are considered to be equally likely in the barcode region. It also has a moderate penalty for A/G transition variants, and a higher penalty for C/T transition variants (but still lower than other substitution penalties).

Mapping Reads to Barcodes

4 Combine all input reads into a single file

```
pv ../called_all/*.fastq | gzip > reads_all.fastq.gz
```

Note: I'm using the pipe viewer command *pv* to produce a progress indicator while the command is running. If this command is not available, it can be replaced with *cat* with no change in function (apart from not showing progess).

5 Use LAST, ignoring FASTQ quality scores for substitution (-Q 0), using the pre-defined substitution matrix to map the reads. In this example, it is distributed over 10 processing threads (-P 10). Here *maf-convert* is used to convert to a single line per match, *cut* retains only the barcode and read IDs, and *uniq* is used to make sure that multiple same barcodes per read (e.g. for reverse / complement barcodes at each end) will not produce duplicates:

```
lastal -Q 0 -P10 -p bc.mat barcode_base.fa <(pv reads_all.fastq.gz) | \
  maf-convert -n tab | cut -f 2,7 | sort | uniq | \
  gzip > barcode_assignments.txt.gz
```

For a more stringent search, lastal can do an overlap match (-T 1) rather than the default local match, which will make sure that only the ends of the barcode are matched (hopefully the entire barcode) when demultiplexing. The downside is that this is more likely to drop reads due to slight mismatches in the barcode portion of the read:

```
lastal -T 1 -Q 0 -P10 -p bc.mat barcode_base.fa <(pv reads_all.fastq.gz) | \
  maf-convert -n tab | cut -f 2,7 | sort | uniq | \
  gzip > barcode_assignments.txt.gz
```

Stringency can also be altered by adjusting the query letters per random alignment setting (-D <value>, 1e6 by default). Lowering this number will produce more matches, at the expense of more false positive matches:

```
lastal -D 1e5 -Q 0 -P10 -p bc.mat barcode_base.fa <(pv reads_all.fastq.gz) | \
  maf-convert -n tab | cut -f 2,7 | sort | uniq | \
  gzip > barcode_assignments.txt.gz
```

The output of this command will be a gzipped tab-separated 2-column file with barcode names in the first column, and read IDs in the second column.

Splitting Read File Per Barcode

6 For each discovered barcode, using the appropriate read category assignment file, find the corresponding read IDs, then extract those IDs out of the read FASTQ file. This uses one of my scripts, fastx-fetch.pl, to do this directly from a FASTQ file. The '-*lengths*' command-line parameter also outputs sequence lengths for each read (see next step):

□fastx-fetch.pl

```
mkdir -p demultiplexed
fastx-fetch.pl -lengths -demultiplex barcode_assignments.txt.gz \
  -prefix 'demultiplexed/reads' <(pv reads_all.fastq.gz) > barcode_counts.txt
```

Note: this demultiplexing code will only by default put reads into a barcode bin if they have a single unique barcode sequence detected. Otherwise, they will be put into a 'BCchim' bin if multiple adapters are detected (i.e. a chimeric read), or a 'BCmiss' bin if no adapters are detected. If these reads should be duplicated and put in one bin per barcode, then the *-chimeric* option can be added to the command arguments:

```
mkdir -p demultiplexed
fastx-fetch.pl -lengths -demultiplex barcode_assignments.txt.gz -chimeric \
  -prefix 'demultiplexed/reads' <(pv reads_all.fastq.gz) > barcode_counts.txt
```

[optional] Displaying Read Length Statistics

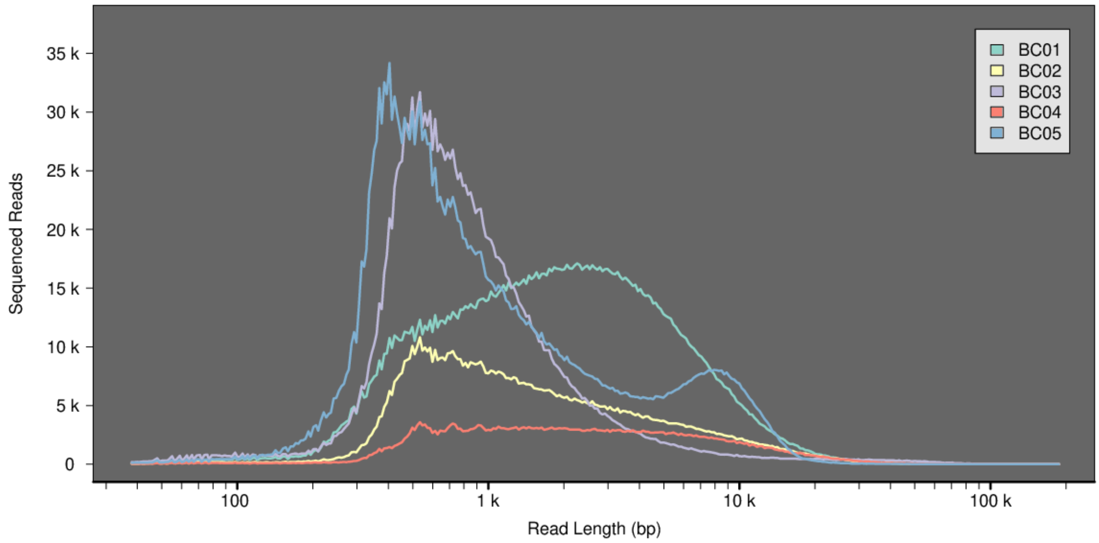
7 The *lengths* output from the demultiplexing step can be fed into another one of my scripts, length-plot.r, in order to display length-based QC plots:

□length_plot.r

```
fastx-length demultiplexed/lengths_*.txt.gz
```

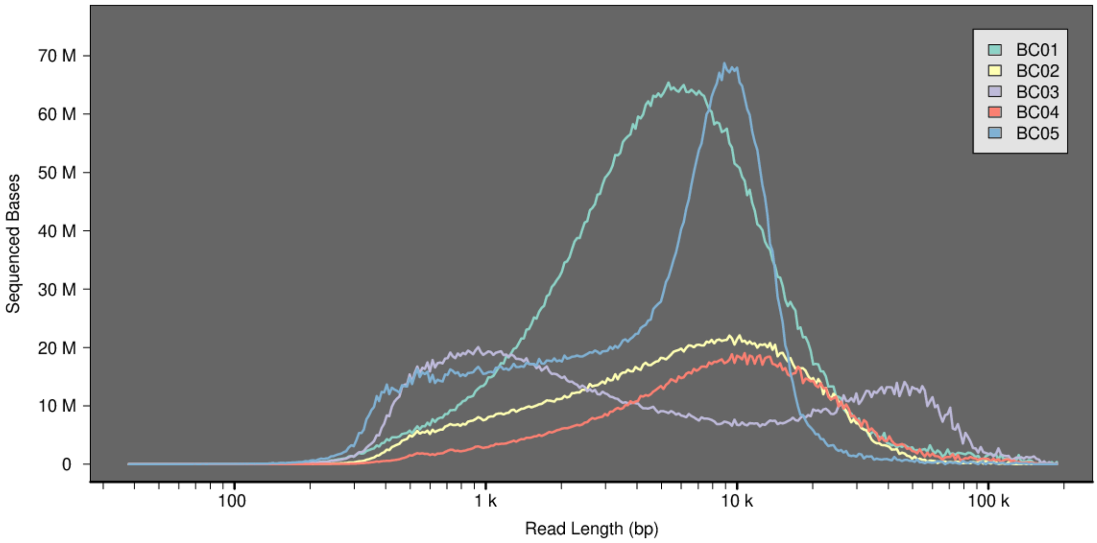
As output, this produces a multi-page PDF file, *Sequence_curves.pdf*. Here are some examples of the plots that are produced:

1. Read Count Frequency Curve



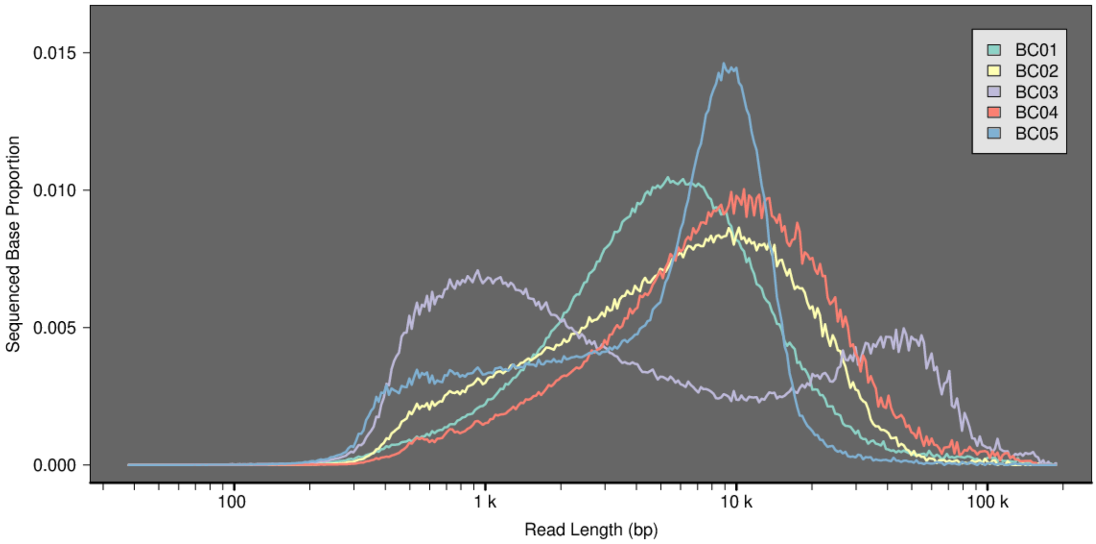
Read count frequency curve for five samples, showing a variety of different read length distributions

2. Called Bases Frequency Curve



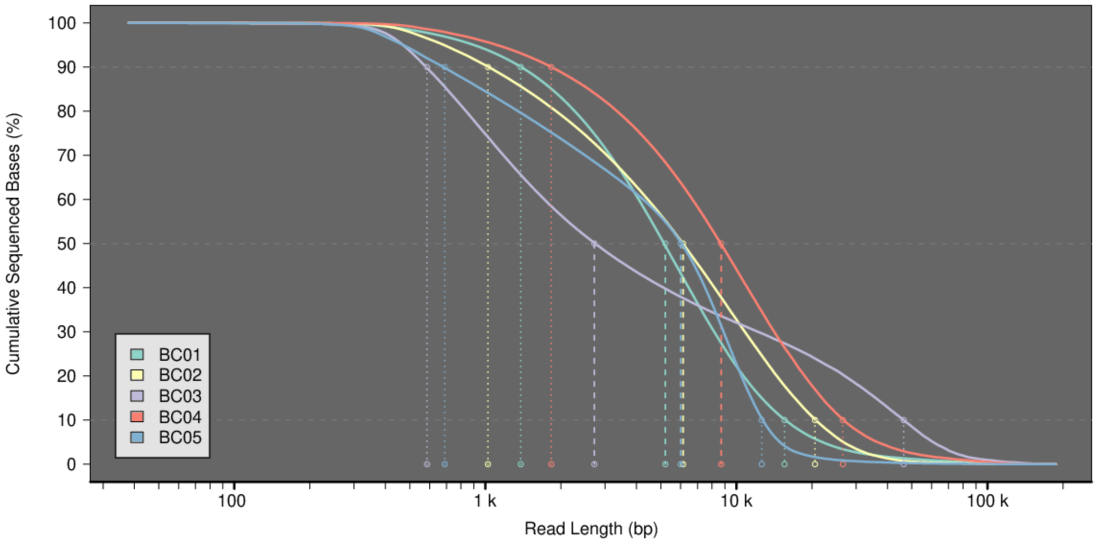
Called bases frequency curve for five samples, showing a variety of different read length distributions

3. Called Bases Density Curve



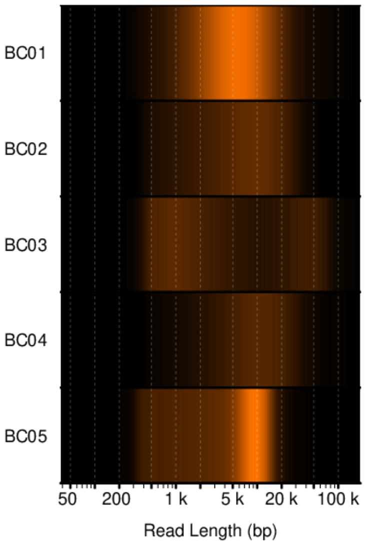
Sample-normalised called bases density curve for five samples, showing a variety of different read length distributions

4. Cumulative Sequenced Bases Curve



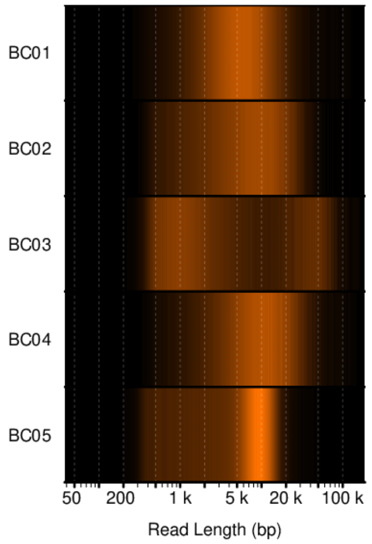
Sample-normalised cumulative sequenced base curve for five samples, showing a variety of different read length distributions

5. Digital Electrophoresis Plot (relative frequency)



Relative digital electrophoresis plot for five samples, showing a variety of different read length distributions

5. Digital Electrophoresis Plot (sample-normalised)



Sample-normalised digital electrophoresis plot for five samples, showing a variety of different read length distributions

Downstream Workflows

8 Following on from here, cDNA reads can be oriented in preparation for stranded mapping:

- Preparing Reads for Stranded Mapping