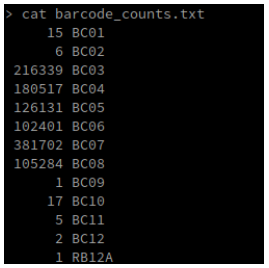
Aug 02, 2019 Version 4

Demultiplexing Nanopore reads with LAST V.4



DOI
dx.doi.org/10.17504/protocols.io.533g8qn

David A Eccles[1]

[1]Malaghan Institute of Medical Research (NZ)

David A Eccles
Malaghan Institute of Medical Research (NZ)

OPEN ACCESS

DOI: dx.doi.org/10.17504/protocols.io.533g8qn

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

Protocol Citation: David A Eccles 2019. Demultiplexing Nanopore reads with LAST. **protocols.io** https://dx.doi.org/10.17504/protocols.io.533g8qn

Manuscript citation:

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

Created: August 02, 2019

Last Modified: August 02, 2019

Protocol Integer ID: 26459

Keywords: demultiplexing, nanopore, high-throughput sequencing

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

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. This will generate seven additional files of the form <index name>.XXX:

```
lastdb barcode_base.fa barcode_base.fa
```

Mapping Reads to Barcodes

3 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).

4 Use LAST in FASTQ alignment mode (-Q 1) 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 1 -P10 barcode_base.fa <(pv reads_all.fastq.gz) | \
  maf-convert -n tab | cut -f 2,7 | uniq | \
  gzip > barcode_assignments.txt.gz
```

For an extremely stringent search, the output of lastal can be piped through last-map-probs, which will reduce the likelihood of a partial barcode match to other DNA

sequences. The downside is that this is more likely to drop reads due to slight mismatches in the barcode portion of the read:

```
lastal -Q 1 -P10 barcode_base.fa <(pv reads_all.fastq.gz) | last-
map-probs | \
  maf-convert -n tab | cut -f 2,7 | 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

5 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 own scripts, **fastx-fetch.pl**, to do this directly from a FASTQ file:

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

Note: this step discards chimeric reads that have multiple adapter sequences. If these reads are desired, then the *-chimeric* option can be added to the command arguments:

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