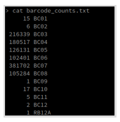

```
> cat barcode_counts.txt
     15 BC01
      6 BC02
 216339 BC03
 180517 BC04
 126131 BC05
 102401 BC06
 381702 BC07
 105284 BC08
      1 BC09
     17 BC10
      5 BC11
      2 BC12
      1 RB12A
```

Demultiplexing Nanopore reads with LAST

David Eccles[1]

[1]Malaghan Institute of Medical Research (NZ)

dx.doi.org/10.17504/protocols.io.xj3fkqn

MinION user group for high molecular weight DNA extraction from all kingdoms

David Eccles
Malaghan Institute of Medical ...

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

PROTOCOL STATUS

In development
We are still developing and optimizing 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).

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-*

4 *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 tab | grep -v '^#' | cut -f 2,7 | uniq | \
 gzip > barcode_assignments.txt.gz
```

For a more 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 tab | grep -v '^#' | 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.

Optional [but recommended]: filtering chimeric reads

5 Identify reads with multiple barcodes (i.e. potentially chimeric reads). The *sort* command sorts by the second field (read ID), then *uniq* identifies duplicated lines when ignoring the first field (barcode).

```
pv barcode_assignments.txt.gz | zcat | sort -k 2,2 | \
 uniq -f 1 -D | gzip > duplicate_assignments.txt.gz
```

6 Exclude duplicate read IDs from the fastq file. This uses one of my own scripts, fastx-fetch.pl, to do this directly from a FASTQ file.

```
~/scripts/fastx-fetch.pl -v -i <(zcat duplicate_assignments.txt.gz | \
  awk '{print $2}') <(pv reads_all.fastq) | gzip > reads_noChimeric.fastq.gz
```

Splitting Read File Per Barcode

7 Create a file containing barcode read counts

```
pv barcode_assignments.txt.gz | zcat | awk '{print $1}' | \
 sort | uniq -c > barcode_counts.txt
```

8 For each discovered barcode, 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:

```
for bc in $(awk '{print $2}' barcode_counts.txt);
 do echo "** ${bc} **";
 fastx-fetch.pl -i <(zgrep ${bc} barcode_assignments.txt.gz | awk '{print $2}') \
  <(pv reads_all.fastq.gz) | \
  gzip > reads_${bc}.fastq.gz;
done
```

Or, if working from the non-chimeric reads:

```
for bc in $(awk '{print $2}' barcode_counts.txt);
 do echo "** ${bc} **";
 fastx-fetch.pl -i <(zgrep ${bc} barcode_assignments.txt.gz | awk '{print $2}') \
  <(pv reads_noChimeric.fastq.gz) | \
  gzip > reads_${bc}.fastq.gz;
done
```

Note: this step processes through the read file once per barcode, which could take a while depending on how many barcodes are detected.