VERSION 3

FEB 07, 2019

OPEN ACCESS

DOI:
dx.doi.org/10.17504/protocols.io.xu4fnyw

Protocol Citation: David A Eccles 2019. Stranded Transcript Count Table Generation from Long Reads. **protocols.io**
<https://dx.doi.org/10.17504/protocols.io.xu4fnyw>

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: Feb 07, 2019

Last Modified: Feb 07, 2019

PROTOCOL integer ID:
 20092

Stranded Transcript Count Table Generation from Long Reads V.3

Version 1 is forked from [Transcript Coverage Analysis from Long Reads](#)

David A Eccles¹

¹Malaghan Institute of Medical Research (NZ)

David A Eccles

Malaghan Institute of Medical Research (NZ)

ABSTRACT

This protocol is for comparing different samples at the transcript level, using long reads that are mapped to transcripts.

Input(s): demultiplexed fastq files (see protocol [Demultiplexing Nanopore reads with LAST](#)), transcript reference fasta file, annotation file

Output(s): transcript table, sorted by differential coverage, annotated with gene name / description / location

BEFORE START INSTRUCTIONS

Obtain a transcript fasta file, and an annotation file. For the mouse genome, I use the following files:

1. Transcript [CDS] sequences from [Ensembl](#); [this file](#) was the most current when I last checked.
2. Annotation file obtained from [Ensembl BioMart](#) (Ensembl Genes -> Mouse Genes) as a compressed TSV file with the following attribute columns:

- Transcript stable ID
- Gene description
- Gene start (bp)
- Gene end (bp)
- Strand
- Gene name
- Chromosome/scaffold name

Barcode Demultiplexing

1 Demultiplex reads as per protocol [Demultiplexing Nanopore reads with LAST](#).

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

```
for bc in $(awk '{print $2}' barcode_counts.txt); do ls  
reads_${bc}.fastq.gz; done
```

Example output:

```
reads_BC03.fastq.gz  
reads_BC04.fastq.gz  
reads_BC05.fastq.gz  
reads_BC06.fastq.gz  
reads_BC07.fastq.gz  
reads_BC08.fastq.gz
```

If the *barcode_counts.txt* file is missing, the output will look like this:

```
awk: fatal: cannot open file `barcode_counts.txt' for reading (No  
such file or directory)
```

If one or more of the barcode-demultiplexed files are missing, the output will look something like this:

```
reads_BC03.fastq.gz  
reads_BC04.fastq.gz  
reads_BC05.fastq.gz  
ls: cannot access 'reads_BC06.fastq.gz': No such file or directory  
ls: cannot access 'reads_BC07.fastq.gz': No such file or directory  
reads_BC08.fastq.gz
```