

Aug 08, 2019 Version 7

Stranded Transcript Count Table Generation from Long Reads V.7

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

DOI

dx.doi.org/10.17504/protocols.io.59zg976

David A Eccles¹

¹Malaghan Institute of Medical Research (NZ)



David A Eccles

Malaghan Institute of Medical Research (NZ)

OPEN  ACCESS



DOI: dx.doi.org/10.17504/protocols.io.59zg976

Protocol Citation: David A Eccles 2019. Stranded Transcript Count Table Generation from Long Reads. **protocols.io**
<https://dx.doi.org/10.17504/protocols.io.59zg976>

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: August 08, 2019

Last Modified: August 08, 2019

Protocol Integer ID: 26649



Abstract

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

Input(s): demultiplexed and oriented fastq files (see protocol [Preparing Reads for Stranded Mapping](#)), transcript reference fasta file, annotation file

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

Before start

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

Demultiplex Reads

1

Demultiplex and orient reads as per the protocol [Preparing Reads for Stranded Mapping](#). It is expected that these demultiplexed reads will be split up in the current directory, and coupled with a '*barcode_counts.txt*' file. If that's the case, the following should work:

```
for bc in $(awk '{print $2}' barcode_counts.txt);  
do ls oriented/${bc}_reads_dirAdjusted.fq.gz;  
done
```

Example expected output:

```
oriented/BC03_reads_dirAdjusted.fastq.gz  
oriented/BC04_reads_dirAdjusted.fastq.gz  
oriented/BC05_reads_dirAdjusted.fastq.gz  
oriented/BC06_reads_dirAdjusted.fastq.gz  
oriented/BC07_reads_dirAdjusted.fastq.gz  
oriented/BC08_reads_dirAdjusted.fastq.gz
```

If the '*barcode_counts.txt*' file is not present, this error will appear:

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

If one or more of the oriented read files is missing, it will look something like this:

```
oriented/BC03_reads_dirAdjusted.fastq.gz  
oriented/BC04_reads_dirAdjusted.fastq.gz  
ls: cannot access 'oriented/BC05_reads_dirAdjusted.fastq.gz':  
No such file or directory  
ls: cannot access 'oriented/BC06_reads_dirAdjusted.fastq.gz':  
No such file or directory  
oriented/BC07_reads_dirAdjusted.fastq.gz  
oriented/BC08_reads_dirAdjusted.fastq.gz
```

Index Preparation

2 Prepare transcript index (see Guidelines for data sources)

```
lastdb Mus_musculus.GRCm38.cds.all.fa <(zcat
Mus_musculus.GRCm38.cds.all.fa.gz)
```

Transcriptome Mapping

3 Reads are mapped to the transcriptome with LAST.

The results of that mapping can be piped through *last-map-probs* to exclude unlikely hits, then through '*maf-convert -n tab*' to convert to a one-line-per-mapping CSV format. This CSV format is further processed to make sure that there is only one mapping per transcript-read pair, and then aggregated to sum up counts per transcript.

```
mkdir -p mapped
for bc in $(awk '{print $2}' barcode_counts.txt);
do echo "*** ${bc} ***";
lastal -Q 1 -P 10 Mus_musculus.GRCm38.cds.all.fa <(pv
oriented/${bc}_reads_dirAdjusted.fq.gz | zcat) | \
last-map-probs | maf-convert -n tab | cut -f 2,7,10 | sort |
uniq | \
awk -F'\t' -v "bc=${bc}" '{print bc,$1,$3}' | sort | uniq -c |
\
gzip >
mapped/trnCounts_LAST_${bc}_vs_Mmus_transcriptome.txt.gz;
done
```

Annotation and Result generation

4 count_analysis.r

Transcript counts are merged with ensembl gene annotation, then converted into wide format (one line per transcript) using an R script.

The transcript annotation in this case is from ensembl BioMart (see Guidelines for more details).

```
#!/usr/bin/env Rscript
library(dplyr);
library(tidyr);

## load ensemble transcript metadata (including gene name)
ensembl.df <-
as.tbl(read.delim('ensembl_mm10_geneFeatureLocations.txt.gz',
  col.names=c('transcript','Description','Start','End',
    'Strand','Gene','Chr'),
  stringsAsFactors=FALSE));
ensembl.df$Description <- sub(' \\
[.*$', '', ensembl.df$Description);
ensembl.df$Description <- sub('^(.
{50}).+$', '\\1...', ensembl.df$Description);
ensembl.df[,1:7] <- ensembl.df[,c(1,7,5,3,4,2,6)];
colnames(ensembl.df)[1:7] <- colnames(ensembl.df)
[c(1,7,5,3,4,2,6)];
options(scipen=15); ## don't show scientific notation for large
positions
## load used barcode identifiers
bcNames <- read.table("barcode_counts.txt",
stringsAsFactors=FALSE)[,2];
## load count data into 'narrow' array (one line per count)
trn.counts <- tibble(); for(bc in bcNames){
  trn.counts <-
    bind_rows(trn.counts,
      as.tbl(read.table(
sprintf('mapped/trnCounts_LAST_%s_vs_Mmus_transcriptome.txt.gz',
bc),
      col.names=c('count','barcode','transcript','dir'),
      stringsAsFactors=FALSE)));
}

## remove revision number from transcript names (if present)
trn.counts$transcript <- sub('\\.[0-
9]+$', '', trn.counts$transcript);
## convert to wide format (one line per transcript)
trn.counts.wide <- spread(trn.counts, barcode, count) %>%
  mutate(dir = c('+='fwd', '-='rev')[dir]);
for(bd in colnames(trn.counts.wide[, -1])){
  trn.counts.wide[[bd]] <- replace_na(trn.counts.wide[[bd]], 0);
}
## merge ensembl metadata with transcript counts
```

```
gene.counts.wide <- inner_join(ensembl.df, trn.counts.wide,  
by='transcript');  
gene.counts.wide <- gene.counts.wide[order(-  
rowSums(gene.counts.wide[, -(1:8)])),];  
## write result out to a file  
write.csv(gene.counts.wide,  
file='wide_transcript_counts_LAST.csv',  
row.names=FALSE);
```

Downstream Workflows

- 5 A downstream workflow is still [in development] (<https://support.bioconductor.org/p/120764/>). I have had reasonable succes doing transcript-level differential expression analysis using [DESeq2] (<https://bioconductor.org/packages/release/bioc/html/DESeq2.html>).

I would like to emphasise that batch effects should be considered for nanopore sequencing, given how frequently the technology changes. Make sure that at least the sequencing *library* (i.e. samples prepared in tandem on the same day from the same kit) is added into the statistical model, and try to make sure that sequencing libraries are fairly heterogeneous - replicates from a sample with skewed transcript distributions could influence the outcome of statistical tests.