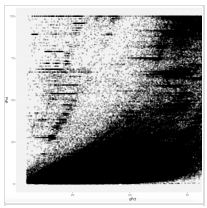

16

May 24, 2022

Stranded Transcript Count Table Generation from Long Reads V.16

Transcript Coverage Analysis from Long Reads

David A Eccles[1]

[1]Malaghan Institute of Medical Research (NZ)

1

dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

David Eccles
Malaghan Institute of Medical Research (NZ)

This protocol is for generating count tables for 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 summed coverage across all samples, annotated with gene name / description / location

DOI

dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

David A Eccles 2022. Stranded Transcript Count Table Generation from Long Reads. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16
David Eccles

Transcript Coverage Analysis from Long Reads, David Eccles

RNASeq, cDNASeq, transcript, nanopore, long reads

protocol ,

May 24, 2022

May 24, 2022

63128

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

1. Comprehensive transcript sequences from GENCODE; this should be the union of cDNA, CDS, and ncRNA sequences, usually the first file in the 'FASTA Files' section, named 'Transcript sequences'.
2. Annotation file obtained from Ensembl BioMart (Ensembl Genes -> Mouse Genes) as a compressed TSV file with the following attribute columns from the 'Features' section:

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

A recent version of these files can be obtained from This Zenodo Repository

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
```


Citation: David A Eccles Stranded Transcript Count Table Generation from Long Reads
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

```
oriented/BC07_reads_dirAdjusted.fastq.gz
oriented/BC08_reads_dirAdjusted.fastq.gz
```

Transcript Index Preparation

2 Prepare transcript index (see Guidelines for data sources).

The GENCODE transcript file is first modified so that the mapped reference sequence name is the first value (i.e. the Transcript id):

```
perl -pe 's/>(.*?)\|/>$1 /' gencode.vM28.transcripts.fa >
trName_gencode.vM28.transcripts.fa
```

In order to reduce the risk of polyA and polyT sequences matching to the polyA tail of reads (e.g. as found in Comt-204 / ENSMUST00000165430.8), any A or T homopolymers are masked out with Ns:

```
fasta_formatter -i trName_gencode.vM28.transcripts.fa | perl -pe
's/(A{9}A+|T{9}T+)/"N" x length($1)/eg' >
ATmasked_trName_gencode.vM28.transcripts.fa
```

A temporary shell variable is created to reference the transcript file location:

```
indexLoc=$(readlink -e ATmasked_trName_gencode.vM28.transcripts.fa)
```

2.1 If using LAST

Following Martin Frith's recommendation, the '-uNEAR' seeding scheme is used to slightly increase sensitivity; low-complexity regions are also converted to lower case. This will generate seven additional files of the form <index name>.XXX:

```
lastdb -uNEAR -R01 ${indexLoc} ${indexLoc}
```

Prepare a substitution matrix for transcript mapping. The default substitution matrix is swayed too much by INDELs in the barcode sequences, so here's one that I've developed using last-train with a recent nanopore cDNA sequencing run:

Citation: David A Eccles Stranded Transcript Count Table Generation from Long Reads
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

```
#last -Q 1
#last -t4.27969
#last -a 15
#last -A 17
#last -b 3
#last -B 4
#last -S 1
# score matrix (query letters = columns, reference
letters = rows):
       A      C      G      T
A      5    -37    -14    -33
C    -34      6    -36    -19
G    -15    -37      6    -35
T    -35    -19    -36      6
```

cDNA.mat

The variant matrix here was generated using last-train on one of the read files from a recent cDNA run, called using super-accuracy basecalling with guppy v5.1.15:

```
last-train -Q 1 ${indexLoc} <(zcat
oriented/BCXX_reads_dirAdjusted.fq.gz | head -n 100000)
```

*[note: this is a **different** matrix from that used for demultiplexing and read orientation]*

2.2 If using minimap2

A reference index is generated with the *-d* command-line argument:

```
minimap2 -d ${indexLoc}.idx ${indexLoc}
```

Note that minimap2 doesn't allow adjustment of alignment scores to have different penalties for insertions and deletions, or different base changes, and this could affect the mappability of reads. Nanopore basecalling has a higher likelihood of mis-calling transitions over transversions (presumably due to a more similar electrochemical structure), and slightly favours deletions (i.e. signal loss) over insertions (i.e. signal misinterpretations).

Transcriptome Mapping

3 Reads are mapped to the transcriptome to create an output associating each read with a transcript. The outcome of this process is a tab-separated file containing

Citation: David A Eccles Stranded Transcript Count Table Generation from Long Reads
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

barcode/transcript/read/direction quads (in that order). Reads may be mapped to multiple transcripts in this process, which is fine as long as the total sum of mapped transcript/read pairs doesn't play a role in subsequent downstream normalisation.

3.1 If using LAST

Reads are mapped to the transcriptome using LAST. The results of that mapping can be piped through *last-split* and *last-postmask* to exclude unlikely hits.

Following this, one of my scripts, maf2csv.pl, is used to convert to a one-line-per-mapping CSV format. Based on an eyeballing of mapped results (see here), hits are additionally filtered to exclude 100% matches (these tend to be short, low-complexity subsequences), as well as any mappings shorter than 50% of the read where the proportion of the query that is mapped is less than the proportion of the target.

LAST reports multiple hits per read (including multiple hits for each read/transcript pair), so this is further ordered by the gap-compressed identity column of the CSV output using *sort*, then filtered again by *sort* to only retain the first highest accuracy hit per read. This intermediate CSV file is preserved for additional read-level mapping and accuracy QC, if desired.

The CSV file is finally converted by *awk* to a space-separated format to retain only the transcript, read, and direction.

```
mkdir -p mapped
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo "** ${bc} **";
  lastal -P 10 -p cDNA.mat ${indexLoc} <(pv
oriented/${bc}_reads_dirAdjusted.fq.gz | zcat) | \
    last-split -n -m0.99 | last-postmask | maf2csv.pl |
\
    awk -F',' '{if(($14 == "gci") || (($14 < 100) &&
(($8>$13) || ($8 > 50)))){print}}' | \
    sort -t ',' -k 1r,1 -k 14rn,14 | \
    sort -t ',' -k 1r,1 -u | \
    gzip >
mapped/trnMapping_LAST_${bc}_vs_Mmus_transcriptome.csv.g
z;
  zcat
mapped/trnMapping_LAST_${bc}_vs_Mmus_transcriptome.csv.g
z | \
    awk -F ',' -v 'OFS=\t' '{print $2, $1, $3}' | gzip >
```


Citation: David A Eccles Stranded Transcript Count Table Generation from Long Reads
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

```
mapped/trnMapping_LAST_${bc}_vs_Mmus_transcriptome.txt.g
z
done
```

3.2 If using minimap2

Reads are mapped to the transcriptome using minimap2. The results of that mapping (as PAF format) are further processed to make sure that there is only one mapping per transcript-read pair, finally converting to a read/transcript/direction space-separated output:

```
mkdir -p mapped
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo "** ${bc} **";
  minimap2 -t 10 ${indexLoc}.idx <(pv
oriented/${bc}_reads_dirAdjusted.fq.gz | zcat) | \
    awk -v 'OFS=\t' '{print $1,$6,$5}' | \
    sort | uniq | gzip >
mapped/trnMapping_mm2_${bc}_vs_Mmus_transcriptome.txt.gz
;
done
```

Note: minimap2 may perform better via a spliced alignment to the genome to produce mapped BAM files (see this protocol), followed by gene/transcript counting using a genome annotation file. One approach for doing this can be found in the vignette for the long-read transcript mapping tool bambu.

4 The result is then aggregated to sum up counts per transcript:

```
mapper="LAST" # use if LAST is used for mapping
mapper="mm2" # use if minimap2 is used for mapping
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo "** ${bc} **";
  zcat
mapped/trnMapping_${mapper}_${bc}_vs_Mmus_transcriptome.txt.gz | \
    awk -F'\t' -v "bc=${bc}" '{print bc,$1,$3}' | sort | uniq -c |
\
    gzip >
mapped/trnCounts_${mapper}_${bc}_vs_Mmus_transcriptome.txt.gz;
done
```

Note: I've split this up into two steps so that an intermediate count of the total number of mapped transcripts per barcode can be done:

Citation: David A Eccles Stranded Transcript Count Table Generation from Long Reads
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

```
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo -n "${bc} ";
  zcat
mapped/trnMapping_${mapper}_${bc}_vs_Mmus_transcriptome.txt.gz | \
    awk '{print $2}' | sort | uniq | wc -l;
done
```

Annotation and Result generation

5 **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(tidyverse);

## load used barcode identifiers
bcNames <- read.table("barcode_counts.txt", stringsAsFactors=FALSE)
[,2];

## set gene annotation file location
annotationFile <- "ensembl_GRCm39_geneFeatureLocations.txt.gz";

## load ensemble transcript metadata (including gene name)
ensembl.df <- read_delim(annotationFile, delim="\t");

colnames(ensembl.df) <-
    c("Transcript stable ID" = "transcript",
      "Gene description" = "Description",
      "Gene name" = "Gene",
      "Gene start (bp)" = "Start",
      "Gene end (bp)" = "End",
      "Strand" = "Strand",
      "Chromosome/scaffold name" = "Chr")[colnames(ensembl.df)];

ensembl.df$Description <- sub(" \\[.*$","",ensembl.df$Description);
ensembl.df$Description <- sub("^(.
{50}).+$","\\1...",ensembl.df$Description);

options(scipen=15); ## don't show scientific notation for large
```


Citation: David A Eccles Stranded Transcript Count Table Generation from Long Reads
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

```
positions

for(mapper in c("LAST", "mm2")){
   checkName <-
sprintf("mapped/trnCounts_%s_%s_vs_Mmus_transcriptome.txt.gz",
                     mapper, bcNames[1]);
   if(!file.exists(checkName)){
      cat(sprintf("Warning: %s does not exist. If you are using
mapper '%s', consider this an error\n", checkName, mapper));
      next;
   }
   ## load count data into "narrow" array (one line per count)
   trn.counts <- tibble();
   for(bc in bcNames){
       trn.counts <-
           bind_rows(trn.counts,

sprintf("mapped/trnCounts_%s_%s_vs_Mmus_transcriptome.txt.gz",
                     mapper, bc) %>%
                 read_table2(col_names=c("count","barcode",
                                         "transcript","dir")));
   }

   ## 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=sprintf("wide_transcript_counts_%s.csv", mapper),
                          row.names=FALSE);
}
```


Citation: David A Eccles Stranded Transcript Count Table Generation from Long Reads
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16

Downstream Workflows

6 Here is a downstream workflow that carries out transcript-level differential expression analysis using DESeq2:

- Creating Differential Transcript Expression Results with DESeq2

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.

Citation: David A Eccles Stranded Transcript Count Table Generation from Long Reads
https://dx.doi.org/10.17504/protocols.io.5qpvonn2bl4o/v16