

Aug 16, 2018

Version 1

# Stranded Transcript Count Table Generation from Long Reads V.1

 Forked from [Transcript Coverage Analysis from Long Reads](#)

DOI

[dx.doi.org/10.17504/protocols.io.smuec6w](https://dx.doi.org/10.17504/protocols.io.smuec6w)

David A Eccles<sup>1</sup>

<sup>1</sup>Malaghan Institute of Medical Research (NZ)

High molecular weight DNA extraction from all kingdoms

Tech. support email: [See@each.protocol](mailto:See@each.protocol)



David A Eccles

GrinGene Bioinformatics

## Create & collaborate more with a free account

Edit and publish protocols, collaborate in communities, share insights through comments, and track progress with run records.

Create free account

OPEN  ACCESS



DOI: <https://dx.doi.org/10.17504/protocols.io.smuec6w>

**Protocol Citation:** David A Eccles 2018. Stranded Transcript Count Table Generation from Long Reads. **protocols.io**

<https://dx.doi.org/10.17504/protocols.io.smuec6w>



**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 15, 2018

**Last Modified:** August 16, 2018

**Protocol Integer ID:** 14740

**Keywords:** stranded transcript count table generation from long read, stranded transcript count table generation, transcript reference fasta file, stranded mapping from long read, transcript table, stranded fastq file, different samples at the transcript level, transcript, using long read, transcript level, fastq file, long read, stranded mapping, gene, gene name, annotation file, file

## Abstract

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

**Input(s):** stranded fastq files (see steps 1-8 of **Stranded Mapping from Long Reads**), transcript reference fasta file, annotation file

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

## Troubleshooting



## 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** is the most current at the time this protocol was created.
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

-



## Index Preparation

### 1 Prepare transcript index (see Guidelines for data sources)

#### Software

**LAST**

NAME

Debian GNU/Linux

OS

Martin Frith

DEVELOPER

<http://last.cbrc.jp/last/>

REPOSITORY

<http://last.cbrc.jp/>

SOURCE LINK

#### Command

Create the transcriptome index from the transcriptome fasta file using lastdb. An anonymous pipe is used

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

### 2 Prepare barcode adapter index



barcode\_base.fa

**Command****Index the barcode adapter file**

```
lastdb -uNEAR -R01 PCR_barcode_base.fa PCR_barcode_base.fa
```

**3 Prepare cDNA adapter index**

adapter\_seqs.fa

**Command****Index the VNP and strand switch adapter file**

```
lastdb -uNEAR -R01 adapter_seqs.fa adapter_seqs.fa
```

**Read Correction****4 Collate basecalled reads****Command****Collate basecalled reads into separate files for pass and fail (but all barcodes thrown together)**

```
pv workspace/fail/*/*.fastq | gzip > called_fail.fastq.gz  
pv workspace/pass/*/*.fastq | gzip > called_pass.fastq.gz
```

- 5 Correct collated reads with canu. To make sure that all reads are considered, the genomeSize parameter should be set to about 1/20 of the total number of uncorrected bases.

**Command**

**Correct Reads with canu (both passed and failed sequences), using minimap as the mapper**

```
canu -correct overlapper=minimap genomeSize=400M \ minReadLength=100  
minOverlapLength=30 -p canu_corrected -d canu_corrected -nanopore-raw  
./called_pass.fastq.gz \ ./called_fail.fastq.gz
```

- 6 Identify corrected reads using **fastx-length.pl**

**Command**

**create list of corrected sequence lengths**

```
pv canu_corrected/canu_corrected.correctedReads.fasta.gz | \nfastx-  
length.pl | awk '{print $2}' | gzip > names_corrected_all.txt.gz
```

- 7 Extract uncorrected reads using **fastx-fetch.pl**

#### Command

##### filter/extract uncorrected reads

```
pv called_pass.fastq.gz called_fail.fastq.gz | \n fastx-fetch.pl -v -  
i names_corrected_all.txt.gz | gzip > uncorrected_all.fastq.gz
```

- 8 Join corrected and uncorrected reads. The uncorrected reads are converted to fasta format with **fastq2fasta.pl** to make the joined file formats consistent.

#### Command

##### Concatenate corrected reads to uncorrected reads

```
pv uncorrected_all.fastq.gz | zcat | fastq2fasta.pl | gzip >  
uncorrected_all.fasta.gz  
pv uncorrected_all.fasta.gz  
canu_corrected/canu_corrected.correctedReads.fasta.gz | zcat | \ gzip  
> uncorrected_corrected_all.fasta.gz
```

## Demultiplexing

- 9 Map to barcode sequences to generate CSV file of assignments. The corrected and uncorrected reads are mapped separately to give the uncorrected reads the best chance of mapping with '-Q 1'; the corrected reads are in FASTA format, so the corrected mapping does not use quality scores.

**Command****Map to barcode sequences (excluding adapters)**

```
(lastal -P 10 barcode_base.fa <(pv  
canu_corrected/canu_corrected.correctedReads.fasta.gz | zcat);  
lastal -P 10 -Q 1 barcode_base.fa <(pv uncorrected_all.fastq.gz |  
zcat)) | \ maf_bcsplit.pl | gzip > barcode_assignments_all.csv.gz
```

- 10 Map to inner cDNA sequences to generate CSV file of assignments. The corrected and uncorrected reads are mapped separately because the corrected reads are in FASTA format.

**Command****Map to cDNA adapter sequences**

```
(lastal -P 10 adapter_seqs.fa <(pv  
canu_corrected/canu_corrected.correctedReads.fasta.gz | zcat);  
lastal -P 10 -Q 1 adapter_seqs.fa <(pv uncorrected_all.fastq.gz |  
zcat)) | \ maf_bcsplit.pl | gzip > adapter_assignments_all.csv.gz
```

- 11 Create 'wide' table indicating barcode/adapter assignments. This R script creates files 'barcode-adapter\_assignments\_ideal.csv.gz' and 'barcode-adapter\_assignments\_valid.csv.gz'.



```
#!/usr/bin/Rscript bc.df <-
read.csv('barcode_assignments_all.csv.gz'); ad.df <-
read.csv('adapter_assignments_all.csv.gz'); library(dplyr);
library(tidyr); ## Create table of adapter additions ad.tbl <-
group_by(ad.df, query, target, dir) %>% summarise() %>%
unite(tdir, target, dir, sep='.') %>% mutate(present=TRUE) %>%
spread(tdir, present); ## collapse multiple query/target pairs
into one bc.tbl <- group_by(bc.df, query, target) %>%
summarise(dir=paste(unique(dir), collapse='/')); bc.wide <-
spread(bc.tbl, target, dir); ## identify reads with a unique
barcode bc.unique.tbl <- group_by(bc.tbl, query) %>% summarise(n =
n()) %>% filter(n == 1) %>% select(-n) %>% left_join(bc.tbl,
by='query') %>% left_join(ad.tbl, by='query', copy=TRUE);
bc.unique.tbl$`ONT_SSP.-`[is.na(bc.unique.tbl$`ONT_SSP.-`)] <-
FALSE; bc.unique.tbl$`ONT_SSP.+`[is.na(bc.unique.tbl$`ONT_SSP.+`)]
<- FALSE; bc.unique.tbl$`ONT_VNP.-`[is.na(bc.unique.tbl$`ONT_VNP.-
`)] <- FALSE;
bc.unique.tbl$`ONT_VNP.+`[is.na(bc.unique.tbl$`ONT_VNP.+`)] <-
FALSE; colnames(bc.unique.tbl) <-
c('query', 'target', 'bcDir', 'SSPprev', 'SSP fwd', 'VNPprev', 'VNP fwd');
## read is considered 'valid' (for now) if at least one primer
matches bc.valid.tbl <- filter(bc.unique.tbl, (SSPprev | VNP fwd |
VNPprev | SSP fwd)); ## ideal reads have forward and reverse cDNA
adapters in opposing orientations bc.ideal.tbl <-
filter(bc.unique.tbl, ((SSPprev & !SSP fwd & VNP fwd & !VNPprev) |
(!SSPprev & SSP fwd & !VNP fwd & VNPprev))); write.csv(bc.ideal.tbl,
row.names=FALSE, file=gzfile('barcode-
adapter_assignments_ideal.csv.gz'), quote=FALSE);
write.csv(bc.valid.tbl, row.names=FALSE, file=gzfile('barcode-
adapter_assignments_valid.csv.gz'), quote=FALSE);
```

## 12 Create a list of used barcodes

**Command****Create a list of used barcodes**

```
zcat barcode-adapter_assignments_ideal.csv.gz | tail -n +2 | awk -F',' '{print $2}' | sort | uniq > used_barcodes.txt
```

**13 Demultiplex valid reads by barcodes using [fastx-fetch.pl](#)****Command**

```
cat used_barcodes.txt | while read bc do echo
```

**14 Demultiplex barcode-demultiplexed reads by SSP direction.**

Note that the last four values in the 'wide' table refer to the reverse and forward mappings of the SSP and VNP primers respectively). The reverse reads are reverse-complemented with [fastx-rc.pl](#), followed by a final concatenation to simplify the subsequent alignment steps.

**Command**

```
cat used_barcodes.txt | while read bc do echo
```

**Transcriptome Mapping**



## 15 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\\_bcsplit.pl](#) 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.

### Command

```
mkdir -p mapped
cat used_barcodes.txt | while read bc do echo
```

## Annotation and Result generation

### 16 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/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 <- readLines('used_barcodes.txt'); ## 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);
```