Stranded Transcript Count Table Generation from Long Reads

Version 2

Forked from Transcript Coverage Analysis from Long Reads

David Eccles[1]

[1]Malaghan Institute of Medical Research (NZ)

dx.doi.org/10.17504/protocols.io.vyne7ve

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

David Eccles
Malaghan Institute of Medical ...

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

PROTOCOL STATUS

In development
We are still developing and optimizing this protocol

BEFORE STARTING

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)

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

2 Prepare barcode adapter index

barcode_base.fa

```
lastdb -uNEAR -R01 barcode_base.fa barcode_base.fa
```

This is an open access protocol distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

3 Prepare cDNA adapter index

adapter_seqs.fa

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

Read Correction

4 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 (v1.8+). 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.

```
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

```
~/scripts/fastx-length.pl <(pv canu_corrected/canu_corrected.correctedReads.fasta.gz) | \
 awk '{print $2}' | gzip > corrected_readNames.txt.gz
```

7 Extract uncorrected reads using fastx-fetch.pl

```
pv called_fail.fastq.gz called_pass.fastq.gz | ~/scripts/fastx-fetch.pl -v -i corrected_readNames.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.

```
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 *uncorrected reads* to barcode sequences to generate CSV file of assignments. Canu v1.8 is very good at trimming off repetitive adapter sequences (including barcodes).

```
lastal -Q 1 -P 10 barcode_base.fa <(pv called_*.fastq.gz | \
   zcat) | ~/scripts/maf_bcsplit.pl | \
 gzip > barcode_assignments_all.csv.gz
```

This is an open access protocol distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

10 Map *uncorrected reads* to adapter sequences to generate CSV file of assignments.

```
lastal -Q 1 -P 10 adapter_seqs.fa <(pv called_*.fastq.gz | \
   zcat) | ~/scripts/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/env 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','SSPrev','SSPfwd','VNPrev','VNPfwd');

## read is considerd 'valid' (for now) if at least one primer matches
bc.valid.tbl <- filter(bc.unique.tbl, (SSPrev | VNPfwd | VNPrev | SSPfwd));

## ideal reads have forward and reverse cDNA adapters in opposing orientations bc.ideal.tbl <- filter(bc.unique.tbl,
 ((SSPrev & !SSPfwd & VNPfwd & !VNPrev) |
  (!SSPrev & SSPfwd & !VNPfwd & VNPrev)));
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

This is an open access protocol distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

```
zcat barcode-adapter_assignments_ideal.csv.gz | awk -F',' '{print $2}' | \
 sort | uniq -c | awk '{if($1 > 100){print $2}}' > used_barcodes.txt
```

13 Demultiplex valid reads (combined corrected and uncorrected) by barcodes using fastx-fetch.pl

```
cat used_barcodes.txt | while read bc
 do echo "** ${bc} **"
 mkdir -p demultiplexed/${bc};
 pv uncorrected_corrected_all.fasta.gz |  \
  ~/scripts/fastx-fetch.pl -i <(zgrep ${bc} barcode-adapter_assignments_ideal.csv.gz | awk -F',' '{print $1}')  | \
 gzip > demultiplexed/${bc}/${bc}_reads_all.fasta.gz;
done
```

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.

```
cat used_barcodes.txt | while read bc
 do echo "** ${bc}/fwd **";
 pv demultiplexed/${bc}/${bc}_reads_all.fasta.gz |  \
  ~/scripts/fastx-fetch.pl -i <(zgrep 'FALSE,TRUE,TRUE,FALSE$' barcode-adapter_assignments_ideal.csv.gz | awk -F','
'{print $1}')  | \
 gzip > demultiplexed/${bc}/${bc}_reads_fwd.fasta.gz;
 echo "** ${bc}/rev **";
 pv demultiplexed/${bc}/${bc}_reads_all.fasta.gz |  \
  ~/scripts/fastx-fetch.pl -i <(zgrep 'TRUE,FALSE,FALSE,TRUE$' barcode-adapter_assignments_ideal.csv.gz | awk -F','
'{print $1}')  | \
 fastx-rc.pl | gzip > demultiplexed/${bc}/${bc}_reads_rev.fasta.gz;
 pv demultiplexed/${bc}/${bc}_reads_fwd.fasta.gz demultiplexed/${bc}/${bc}_reads_rev.fasta.gz | zcat | \
  gzip > demultiplexed/${bc}/${bc}_reads_dirAdjusted.fasta.gz
done
```

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.

```
mkdir -p mapped
cat used_barcodes.txt | while read bc
 do echo "** ${bc} **";
 lastal -P 10 Mus_musculus.GRCm38.cds.all.fa <(pv demultiplexed/${bc}/${bc}_reads_dirAdjusted.fasta.gz | zcat) | \
  last-map-probs | ~/scripts/maf_bcsplit.pl | tail -n +2 | \
  awk -F',' '{print $1,$2,$3}' | sort | uniq | \
  awk -v "bc=${bc}" '{print bc,$2,$3}' | sort | uniq -c | gzip > mapped/trnCounts_LAST_${bc}_vs_Mmus_transcriptome.txt.gz;
done
```

This is an open access protocol distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

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/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 <- 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);
```

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

This is an open access protocol distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited