

# Transcript Coverage Analysis from Long Reads

David Eccles

## 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

**Citation:** David Eccles Transcript Coverage Analysis from Long Reads. **protocols.io**  
dx.doi.org/10.17504/protocols.io.retd3en

**Published:** 02 Jul 2018

## 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:
  1. Transcript stable ID
  2. Gene description
  3. Gene start (bp)
  4. Gene end (bp)
  5. Strand
  6. Gene name
  7. Chromosome/scaffold name

## Protocol

### Transcriptome Mapping

#### Step 1.

## LAST

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

### cmd COMMAND

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

Create the transcriptome index from the transcriptome fasta file using lastdb. An anonymous pipe is used "<()" to avoid the need to decompress the file for index generation.

## Transcriptome Mapping

### Step 2.

Map the reads to the Mmus transcripts with LAST.

The results of that mapping can be piped through *last-map-probs* to exclude unlikely hits, then through my *maf\_bcsplit.pl* script 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.

### cmd COMMAND

```
for id in "fwd_4T1_BC06" "rev_4T1_BC06" "fwd_4T1_BC07" "rev_4T1_BC07"
do lastal Mus_musculus.GRCm38.cds.all.fa <(pv ${id}.correctedReads.uniqueOnly.fasta.gz |
zcat) | \
    last-map-probs | ~/scripts/maf_bcsplit.pl | awk -F',' '{print $1,$2,$3}' | sort | \
    uniq | awk '{print $2,$3}' | sort | uniq -
c > trnCounts_LAST_${id}_vs_Mmus_transcriptome.txt
done
```

LAST mapping; probable hit filtering, read counting, and conversion to count file

## Data Cleaning (R script)

### Step 3.

Transcript counts are converted into a narrow table (one line per transcript/count/barcode tuple) that includes strand direction and a barcode tag:

```
trnCounts <- lapply(c("fwd_4T1_BC06", "rev_4T1_BC06",
                     "fwd_4T1_BC07", "rev_4T1_BC07"),
                  function(x){
    res <-
read.table(sprintf(("trnCounts_%s_vs_Mmus_transcriptome.txt"),
                    x), stringsAsFactors=FALSE,
            col.names=c("Count", "Transcript",
                        "Direction"));
```

```

        if(grepl("rev",x)){
            res$Direction = c("-" = "+", "+" = "-
") [res$Direction];
        }
        res$DB <- paste0(sub("..._4T1_", "", x), res$Direction);
        return(res);
    });
trnCounts <- rbind(trnCounts[[1]], trnCounts[[2]], trnCounts[[3]], trnCounts[[4]]);

```

#### Data Cleaning (R script)

##### Step 4.

The transcript revision number (if any) is removed from the transcript ID:

```
trnCounts$Transcript <- sub("\\.[0-9]+$", "", trnCounts$Transcript);
```

#### Data Cleaning (R script)

##### Step 5.

The annotation file is loaded into memory:

```
ensembl.df <- read.delim("ensembl_mm10_geneFeatureLocations.txt.gz",
                        col.names=c("Transcript", "Description", "Start",
                                    "End", "Strand", "Gene", "Chr"),
                        stringsAsFactors=FALSE);
```

#### Data Cleaning (R script)

##### Step 6.

The *dplyr* and *tidyr* packages are used to convert to a wide format, and pull the associated annotation from the ensembl annotation file:

```
library(dplyr);
library(tidyr);

trnCounts.wide <- group_by(trnCounts, DB, Transcript) %>%
  summarise(Count = sum(Count)) %>% spread(DB, Count) %>%
  inner_join(ensembl.df, by="Transcript");
```

#### Data Cleaning (R script)

##### Step 7.

Missing data is set to a count of zero to simplify subsequent computation:

```
trnCounts.wide$WTfwd <- coalesce(trnCounts.wide$`BC06+`, 0L);
trnCounts.wide$WTrev <- coalesce(trnCounts.wide$`BC06-`, 0L);
trnCounts.wide$p0fwd <- coalesce(trnCounts.wide$`BC07+`, 0L);
```

```
trnCounts.wide$p0rev <- coalesce(trnCounts.wide$`BC07-`,0L);
```

#### Differential coverage (R script)

##### **Step 8.**

This was a pilot experiment with no replicate data (hence why I refer to this as 'differential coverage', rather than differential expression. I would usually use software with proper statistical modeling (such as DESeq2), but in this case I'm generating a quick rough-cut comparison using basic R.

I introduce a fudge factor to account for missing data, and use this factor in the calculation of differential coverage (as log2 fold change). The strand-specificity of the sequencing and transcript annotation allows this differential coverage to be determined for only those reads mapping in the correct direction. Results are rounded to 1 d.p. as an additional warning that these results have a low precision:

```
fcFudge <- 5;
trnCounts.wide$DCov <- round(ifelse(trnCounts.wide$Strand == 1,
                                   log2(trnCounts.wide$WTfwd+fcFudge) -
log2(trnCounts.wide$p0fwd+fcFudge),
                                   log2(trnCounts.wide$WTrev+fcFudge) -
log2(trnCounts.wide$p0rev+fcFudge)),1);
```

#### Differential coverage (R script)

##### **Step 9.**

Finally, the resultant table is writtent out to a CSV file, ordered by the absolute value of the differential coverage statistic:

```
trnCounts.wide <- arrange(trnCounts.wide, -abs(DCov));

write.csv(trnCounts.wide[,c("Transcript", "WTfwd", "WTrev", "p0fwd", "p0rev",
"DCov", "Gene", "Description", "Chr", "Start", "End")],
file="wide_transcript_Counts_WTvsp0.csv", row.names=FALSE);
```