

ENCODE Long Read RNA-Seq Analysis Protocol for Nanopore Direct-RNA

Prepared by Jasmine Sakr

December 2021

Ali Mortazavi Lab, University of California, Irvine

Contact Information

Jasmine Sakr

2300 Biological Sciences III
University of California, Irvine
Irvine, CA 92697-2300
Telephone: (714) 906-1978
Email: jsakr@uci.edu

Ali Mortazavi

2218 Biological Sciences III
University of California, Irvine
Irvine, CA 92697-2300
Telephone: (949) 824-6762
Email: ali.mortazavi@uci.edu

INTRODUCTION

The long-read RNA-seq data on the ENCODE portal was processed with the ENCODE DCC deployment of the TALON pipeline (available with documentation here: <https://github.com/ENCODE-DCC/long-read-rna-pipeline>).

This document describes 1) the steps used to generate the fastqs submitted to the DCC, and 2) individual tasks that make up the TALON pipeline as we run them in our lab. The software versions used can be found in Table 1.

Prior to DCC submission, the raw data was processed by basecalling and trimming of the adapter. First, the raw data was recorded by the MinKNOW software. It processed the signal into reads and were written out as .fast5 files. Guppy, the basecalling software, uses a Recurrent Neural Network (RNN) algorithm to generate the base sequence of the read and is written out as a fastq file. Both MinKNOW and Guppy are software from Oxford Nanopore Technologies.

These reads in the fastq files are the entry point to the DCC pipeline implementation. From this point, the reads are mapped to the genome using Minimap2 to generate output in the sam/bam format. TranscriptClean is run on the mapped reads to correct remaining errors such as noncanonical splice junctions and microindels. Finally, TALON is run to annotate the transcripts and quantify their abundance.

Table 1: Referenced Software

Name	Version	Available from
DCC pipeline	-	https://github.com/ENCODE-DCC/long-read-rna-pipeline
minKNOW	4.3.20	https://community.nanoporetech.com/downloads
Guppy	5.0.11+2b6dbff	https://community.nanoporetech.com/downloads
Minimap2 ¹	2.17	https://github.com/lh3/minimap2
TranscriptClean ²	2.0.3	https://github.com/mortazavilab/TranscriptClean
TALON ³	5.0	https://github.com/mortazavilab/TALON
Samtools ⁴	1.14	https://sourceforge.net/projects/samtools/files/samtools/1.14/

COMPUTATIONAL ANALYSIS

MinKNOW writes out the raw data as 4000 reads into one fast5 file and each file get basecalled by Guppy and written into a fastq file.

Basecalling and adapter trimming with Guppy

The basecalling was run in High-accuracy basecalling (HAC) mode. This provides a higher (up to 97.8%) single-molecule accuracy. For direct-RNA sequencing, where the strands are sequenced in the 3' to 5' direction, Guppy trimmed all the bases up to the polyA tail.

```
guppy_basecaller \
  --recursive \
  --flowcell FLO-MIN106 \
  --kit SQK-RNA002 \
  --disable_pings
  --reverse_sequence on
  --u_substitution on
  -x "cuda:0 cuda:1"
```

All the fastq files were concatenated together and submitted to the DCC. This fastq file is the starting input for the long-read RNA-seq pipeline.

Download References

Reference file sets for human long-read RNA-seq is listed on ENCODE Reference Sequences web page. The files are listed here: <https://www.encodeproject.org/references/ENCSR925QOG/>

Reference genome & annotation file

We aligned the reads to the GRCh38 XY v29 human reference genome. Download the genome from the ENCODE reference file set link and remove extra information from fasta headers with `awk '{print $1}'` command. Run the command as follows:

```
awk '{print $1}' GRCh38_no_alt_analysis_set_GCA_000001405.15.fasta >
GRCh38.v29.fa
```

We made the index using samtools, run the command:

```
samtools faidx GRCh38.v29.fa
```

Download and unzip the gencode v29 annotation gtf file.

Extract reference splice junctions & common variants

TranscriptClean (described in reference-based error correction section) requires a file of reference splice junctions in order to correct noncanonical junctions. To get this file, we ran a TranscriptClean module on the GENCODE v29 comprehensive gene annotation (reference chromosomes only).

The command is as follows:

```
python ${TranscriptClean_Path}/accessory_scripts/get_SJs_from_gtf.py \
    --f GRCh38.v29_gencode.annotation.gtf \
    --g GRCh38.v29.fa \
    --o gencode_v29_sj.tsv
```

In addition, TranscriptClean can run in variant-aware mode to avoid correcting away known variants. First, download the common human variants dbSNP Build150 (April 2017 release) in the VCF format. Make sure that the chromosome naming convention of this file matches the hg38 reference genome by changing chromosome convention so that chromosomes start with 'chr'. Run the following commands:

```
zcat 00-common_all.vcf.gz | \
    awk '{if($0 !~ /^#/ && $0 !~ /^chr/) print "chr"$0; else print $0}' \
    > tmp_00-common_all.vcf
gzip tmp_00-common_all.vcf
mv tmp_00-common_all.vcf.gz common-variants.vcf.gz
```

Alignment to the reference genome using Minimap2

Some of the parameters used are based on minimap2's recommendations for nanopore direct-RNA libraries

(<https://github.com/lh3/minimap2#map-long-mrnacdna-reads>).

Minimap2 v2.17 can take annotated genes as input and prioritize on annotated splice junctions. This annotation is created by minimap's gff2bed module, which takes the gene annotation in the GTF or GFF3 format and outputs a 12-column BED format. This bed file was generated by the following command:

```
paftools.js gff2bed \  
    GRCh38.v29_gencode.annotation.gtf > \  
    GRCh38.v29_gencode.annotation.bed
```

Minimap2 was run with the following parameters:

```
minimap2 \  
    -t 16 \  
    -ax splice \  
    --MD \  
    -u f \  
    -k 14 \  
    --secondary=no \  
    --junc-bed GRCh38.v29_gencode.annotation.bed \  
    --junc-bonus=5 \  
    GRCh38.v29.fa \  
    ${sample}.fastq > \  
    ${sample}_mapped.sam
```

Reference-based error correction using TranscriptClean

Mismatches and microindels that occur at the boundary of an intron may create the mistaken appearance of a novel splice junction. TranscriptClean is a Python program we developed to compare the sequences of mapped isoforms to the reference genome and correct mismatches, microindels, and noncanonical splice junctions in long read sam files from the PacBio Iso-seq and Oxford Nanopore. A file of reference splice junctions (described in previous section) serves as a reference for correcting junctions.

First, we sorted the sam files using samtools:

```
samtools view -Sb --threads=16 ${sample}_mapped.sam > ${sample}_mapped.bam  
samtools sort --threads=16 ${sample}_mapped.bam > ${sample}_sorted.bam  
samtools view -h --threads=16 ${sample}_sorted.bam > ${sample}_sorted.sam
```

Then we ran TranscriptClean:

```
python ~/TranscriptClean/TranscriptClean.py \  
  --sam ${sample}_sorted.sam \  
  --genome GRCh38.v29.fa \  
  --spliceJns encode_v29_sj.tsv \  
  --variants common-variants.vcf.gz \  
  -t 32 \  
  --canonOnly \  
  --primaryOnly \  
  --deleteTmp \  
  --outprefix ${sample}
```

The inclusion of the `--canonOnly` parameter ensures that all of the reads in the output contain only canonical splice junctions or noncanonical splice junctions that are supported by the reference annotation.

Annotate and quantify reads using TALON

TALON is a technology-agnostic long read annotation tool. Our python package is designed to annotate full-length reads as known or novel transcripts as well as report abundance for these transcripts.

Flagging reads for internal priming

Generating direct-RNA nanopore libraries rely on poly-(A) selection and therefore are prone to internal priming artifacts. The `talon_label_reads` module records the fraction of A's in the 20-bp interval following the end of the alignment. Run the command as follows:

```
talon_label_reads \  
  --f ${sample}_clean.sam \  
  --g GRCh38.v29.fa \  
  --t 16 \  
  --ar 20 \  
  --deleteTmp \  
  --o ${sample}
```

Create TALON config file

Next we created a comma-separated configuration file that provides the sample name/dataset ID, sample description, platform, and input sam file. There should be one line for each dataset, and dataset names must be unique.

Here is an example:

```
sample1,Direct-RNA,Nanopore_SQK-RNA002,{path_to_file}/sample1_labeled.sam  
sample2,Direct-RNA,Nanopore_SQK-RNA002,{path_to_file}/sample2_labeled.sam
```

Create TALON database

Next, we generated a TALON database which is built around an SQLite database initialized to contain known genes, transcripts, and exon models from the GENCODE v29 GTF transcriptome annotation. This only needs to be done once for analysis and keep track of the build and annotation names you choose, as these will be used downstream when running TALON and its modules.

```
talon_initialize_database \  
  --f GRCh38.v29_gencode.annotation.gtf\  
  --g GRCh38.v29 \  
  --a gencode_v29.gtf \  
  --l 0 \  
  --idprefix ENCODEH \  
  --5p 500 \  
  --3p 300 \  
  --o talon
```

Running TALON

In a TALON run, each input SAM transcript is compared to the existing transcript models in the database on the basis of its splice junctions, start, and end points. The input database is modified in place to track and quantify transcripts in the provided dataset(s). This allows us to not only assign a novel gene or transcript identity where appropriate, but to track new transcript models and characterize how they differ from known ones.

```
talon \  
  --f config.csv \  
  --db talon.db \  
  --build GRCh38.v29 \  
  --t 64 \  
  --o ${experiment}
```

Generating abundance files

The `talon_abundance` module can be used to extract a raw or filtered transcript count matrix from your TALON database. Each row of this file represents a transcript detected by TALON in one or more of your datasets. **NOTE:** to run this utility, you must provide genome build (-b) and annotation (-a) names that match those provided for the `talon_initialize_database`, otherwise it will not run.

To generate an unfiltered, raw abundance file, run the command as follows:

```
talon_abundance \
  --db talon.db \
  -a gencode_v29.gtf \
  -b GRCh38.v29 \
  --o ${experiment_name}
```

To quantifying results on the isoform level, it is important to filter the novel transcript models because long-read platforms are prone to several forms of artifacts. The `talon_filter_transcripts` module generates a whitelist of transcripts that are either known or observed at least n times in each of k datasets (see TALON documentation). Use this whitelist with the `talon_abundance` module.

To generate a filtered abundance file, run the command as follows:

```
talon_filter_transcripts \
  --db talon.db \
  -a gencode_v29.gtf \
  --maxFracA=0.5 \
  --minCount=5 \
  --minDatasets=2 \
  --o ${experiment_name}

talon_abundance \
  --db talon.db \
  -a gencode_v29.gtf \
  -b GRCh38.v29 \
  --whitelist ${experiment_name}_talon-list.csv
  --o ${experiment_name}
```

References

- (1) Li, H. Minimap2: Pairwise Alignment for Nucleotide Sequences. *Bioinforma. Oxf. Engl.* **2018**, 34 (18), 3094–3100. <https://doi.org/10/gdqbqt>.
- (2) Wyman, D.; Mortazavi, A. TranscriptClean: Variant-Aware Correction of Indels, Mismatches and Splice Junctions in Long-Read Transcripts. *Bioinformatics* **2019**, 35 (2), 340–342. <https://doi.org/10/gnr6rr>.
- (3) Wyman, D.; Balderrama-Gutierrez, G.; Reese, F.; Jiang, S.; Rahmanian, S.; Forner, S.; Matheos, D.; Zeng, W.; Williams, B.; Trout, D.; England, W.; Chu, S.-H.; Spitale, R. C.; Tenner, A. J.; Wold, B. J.; Mortazavi, A. A Technology-Agnostic Long-Read Analysis Pipeline for Transcriptome Discovery and Quantification. *bioRxiv* **2020**, 672931. <https://doi.org/10/gg9pwb>.
- (4) Li, H.; Handsaker, B.; Wysoker, A.; Fennell, T.; Ruan, J.; Homer, N.; Marth, G.; Abecasis, G.; Durbin, R.; 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map Format and SAMtools. *Bioinforma. Oxf. Engl.* **2009**, 25 (16), 2078–2079. <https://doi.org/10/ff6426>.