# Abstract

This SOP describes the workflow used for assembling transcript sequences from short Illumina reads. Steps covered include adapter trimming, assembly and subsequent quality control. Expression of individual transcripts may be quantified from the assembly.

# Summary

Transcripts should be assembled using the Trinity software. Expression within and across samples can be measured using utility scripts included with Trinity. Deliverables for the data

management are:

* Assembly of transcripts (FASTA format, required)
* Annotation of transcripts (GFF3 format, per assembly, optional)
* Expression count tables (TAB format, per sample and assembly, optional)

# Introduction

De-novo transcriptome assembly is commonly used to study gene expression in non-model systems for which no (complete) genome sequence is available. Assembled transcripts can be functionally annotated to reveal the coding content of a genome. In addition, transcript assemblies can serve to quantify the expression of individual genes (and isoforms).

# Requirements

## Input reads

The recommended input is one or several sets of paired-end Illumina reads. It is highly recommended to have these generated with a strand-specific library protocol to increase the quality of the assembly (e.g. dUTP).

Reads should be adapter but not quality trimmed as low quality reads may nevertheless help during the assembly process. Please see the SOP on read processing for a recommended procedure.

## Software tools

De-novo transcriptome assemblies should be performed using the Trinity software (3.x): <https://github.com/trinityrnaseq/trinityrnaseq/wiki/Installing%20Trinity>

Functional annotation can be performed using the Trinotate package from the same developer: <http://trinotate.github.io/>

BUSCO to gauge completeness of the assembly: <http://busco.ezlab.org/>

# Procedure

## Adapter removal

This procedure is described in another SOP. As with other assembly applications, it is recommended to focus on removing sequencing adapters rather than low quality bases.

## Assembly

Assembly of short reads using Trinity requires the following basic syntax:

Trinity --SS\_lib\_type RF --output MY\_SAMPLE --seqType fq --left MY\_SAMPLE\_R1.fastq.gz --right MY\_SAMPLE\_R2.fastq.gz --CPU 16 --max\_memory 50G --full\_cleanup

The output file will be called Trinity.fasta, located in trinity.MY\_SAMPLE and contain the assembled transcript sequences in FASTA format, where each sequence is named using the following convention:

TRINITY\_DN1000|c115\_g5\_i1

This translates to: cluster “TRINITY\_DN1000|c115”, gene 5, isoform 1.

Assembly quality can be further improved if a draft genome is available (assuming that this draft has been checked for structural issues). For this, reads are first aligned against the reference. The resulting BAM file – including both aligned AND unaligned reads – can then be passed to Trinity for assembly.

bowtie2 --sensitive --rf –x <bowtie\_index> -1 MY\_SAMPLE\_R1.fastq.gz -2 MY\_SAMPLE\_R2.fastq.gz –S MY\_SAMPLE.sam –p 16

samtools view –b –o MY\_SAMPLE.sam MY\_SAMPLE.bam

samtools sort MY\_SAMPLE.bam MY\_SAMPLE.sorted

Trinity --output MY\_SAMPLE --genome\_guided\_bam MY\_SAMPLE.sorted.bam --max\_memory 50G --genome\_guided\_max\_intron 10000 --CPU 16 --full\_cleanup

A full list of Trinity arguments is available at: https://github.com/trinityrnaseq/trinityrnaseq/wiki/Running%20Trinity

The assembly should be checked into iRODS pointing to the study and samples if was generated from.

## Quality control

There are several metrics that can be used to gauge the overall quality of a de-novo assembled transcriptome. A full list is provided on the Trinity website: https://github.com/trinityrnaseq/trinityrnaseq/wiki/Transcriptome-Assembly-Quality-Assessment

Gene space coverage using BUSCO

BUSCO can compare a transcriptome assembly against a set of expected sequences for specific taxonomic groups to gauge overall completeness:

python BUSCO\_v1.1b.py -o BUSCO\_MY\_SAMPLE -in Trinity.fasta -l vertebrates -m trans

The “Busco” score for “completeness” should be attached to the assembly when you check it into the CRC iRODS.

## Functional annotation

Running Trinotate involves multiple steps – finding similarities between the assembled transcripts and external databases, identifying putative functional motifs, and finally combining the outputs from these stages to create a combined best-guess functional inference. The full workflow is outlined on the Trinotate website: <http://trinotate.github.io/>

The resulting annotation in GFF3 format should be check into the CRC iRODS as part of this analysis collection and referencing the assembly file it belongs to.

## Expression

The general steps for measuring expression from individual samples involve mapping of reads against the assembly and quantification using the RSEM package.

$TRINITY\_HOME/util/align\_and\_estimate\_abundance.pl --transcripts Trinity.fasta --seqType fq --left MY\_SAMPLE\_R1.fastq.gz --right MY\_SAMPLE\_R2.fastq.gz –output\_dir expression.MY\_SAMPLE --aln\_method bowtie2 --SS\_lib\_type RF --thread\_count 16 --trinity\_mode --prep\_reference

The result will be a tabular file with expression estimates (FPKM) for each assembled transcript.

This file should be checked into iRODS pointing to the sample and assembly it was generated from.

## Differential expression

# Implementation

Not yet available.

# Related Documents

# Revision history