ANALYSIS PAPER

ABSTRACT

A vast number of small DNA sequences are pieced back together to create the original chromosomes from which the DNA originated. This process is known as genome assembly in the field of bioinformatics. Once next generation sequencing, PacBio SMRT sequencing, or Nanopore sequencing has been completed, one of the fundamental phases is sequence assembly. When genome assembly is unnecessary, expensive, or challenging, a transcriptome created using short-read RNA sequencing (RNA-seq) offers a freely accessible proxy catalog of protein-coding genes. The transcriptome must be put together from scratch utilizing only the data included in the RNA-seq reads in the absence of a sequenced genome to serve as a guide. To identify sequence-intrinsic and evolutionary traits in the sequences, they must then be annotated (for example, protein-coding regions). De novo transcriptome assembly and annotation, however initially seeming simple, might quickly turn out to be difficult tasks. Those interested must also navigate an overwhelming amount of options in addition to being familiar with the conceptual and technical complexity of the tasks at hand and the multiple pre- and post-processing stages required. The absence of defined workflows, the rapid development of new tools and approaches, and the scarcity of reliable literature have all contributed to the task's increased difficulty. We give a thorough overview of de novo transcriptome assembly and annotation in this article. We go over the processes, including the pre- and post-processing steps, and provide a list of the appropriate tools.

INTRODUCTION

Organism used: Escherichia coli

E. coli has long been used by biochemists and geneticists to examine the fundamental chemical processes that underlie life and to gather some of the earliest indications of how gene function is controlled. Scientists anticipated that knowing the entire genome sequence of the bacterium E. coli will help them understand this organism even better.

E. coli's genome sequence (from the lab strain K-12) was made public in 1997. One of the earliest genome sequences to be finished, it plays a significant role in genetics and biotechnology. The genomes of countless additional E. coli strains have now been made public.

By examining E. coli's genome sequence, we can learn a lot about how it functions. . We are aware that the majority of E. coli strains have roughly 4400 genes, for example. The specific DNA sequence of these genes is also known. With the aid of that knowledge, scientists can forecast how the proteins encoded by the genes of E. coli will behave.

The genomes of a large number of E. coli strains have been sequenced, allowing us to compare the DNA sequences of the genes in various E. coli strains. The function of genes, their relative importance, and the changes they have undergone over time can all be inferred from comparing gene sequences.

Genomic and biological data are accessible through NCBI.

In Sequence Read Archive, there are numerous data records. Using NCBI, readings for E. coli's DNA and RNA are discovered.

SRA: SRR22575573

- Source: DNA
- Library Layout: paired

- Platform: Illumina
- Strategy: Genome
- File Type: fastq

SRA : SRR21776475

- Source: RNA
- Library Layout: paired
- Platform: Illumina
- Strategy: RNAseq
- File Type: fastq

The main scope of this project is to understand the analysis of DNA and RNA seq reads performing Genome Assembly, Short Read Alignment, Transcriptome Assembly, Blast and Transdecoder, and Protein Classification.

Example codes for assemblies can be found in GitHub and articles.

RESULTS

Genome Assembly:

Genome assembly is the process of assembling a representation of the original chromosomes from which the DNA came from a large number of short DNA sequences [1]. De novo genome assemblies do not rely on prior information on the original DNA sequence's size, organization, or makeup. The DNA of the intended organism is divided into millions of tiny bits and read on a sequencing machine as part of a genome sequencing study. Depending on the sequencing technique employed, these “reads” length ranges from 20 to 1000 nucleotide base pairs (bp). Typically, reads between 36 and 150 bp are generated for short-read sequencing of the Illumina type. These reads can either be "paired-end" or "single-ended," as previously mentioned.

PROCEDURE

Load the required modules:

srun is used to **launch the processes**. If your program is a parallel MPI program, srun takes care of creating all the MPI processes. If not, srun will run your program as many times as specified by the --ntasks option.

- Get the NGS Data of E.coli filtering the given options and split the fastq sequences using

```
fasterq-dump --split-3 “SRRID”
```

- Quality Trimming the split data fastq reads.
- Trimmomatic is a Java-based quality trimmer that uses a sliding window to determine where quality scores have dropped below a specified threshold. In addition to trimming based on quality scores, Trimmomatic also removes any adapter sequences from the reads. Sometimes during library preparation, extra copies of adapters get attached to the beginning or end of the cDNA fragments. These adapters are what sequencers use to immobilize DNA fragments on the flow cell, but you don't want them treated as actual sequence data. If included in your sequence data, they confuse assemblers and read aligners.

```
trimmomatic PE \
-threads 1 -phred33 \

# Input details \

HEADCROP:0 \

ILLUMINACLIP:$PATH_TO_TRIMMOMATIC/adapters/TruSeq3-PE.fa:2:30:10 \

LEADING:20 TRAILING:20 SLIDINGWINDOW:4:30 MINLEN:36
```

Output :

Trimmomatic completes successfully and trims the sequences into paired and unpaired.

- The Third step is where genome assembly starts
- runSpades.sh

  run spades.py to get the help section and go through all available options. SPAdes Genome Assembler is an open-source tool for de novo genome sequencing.

Other parameters:

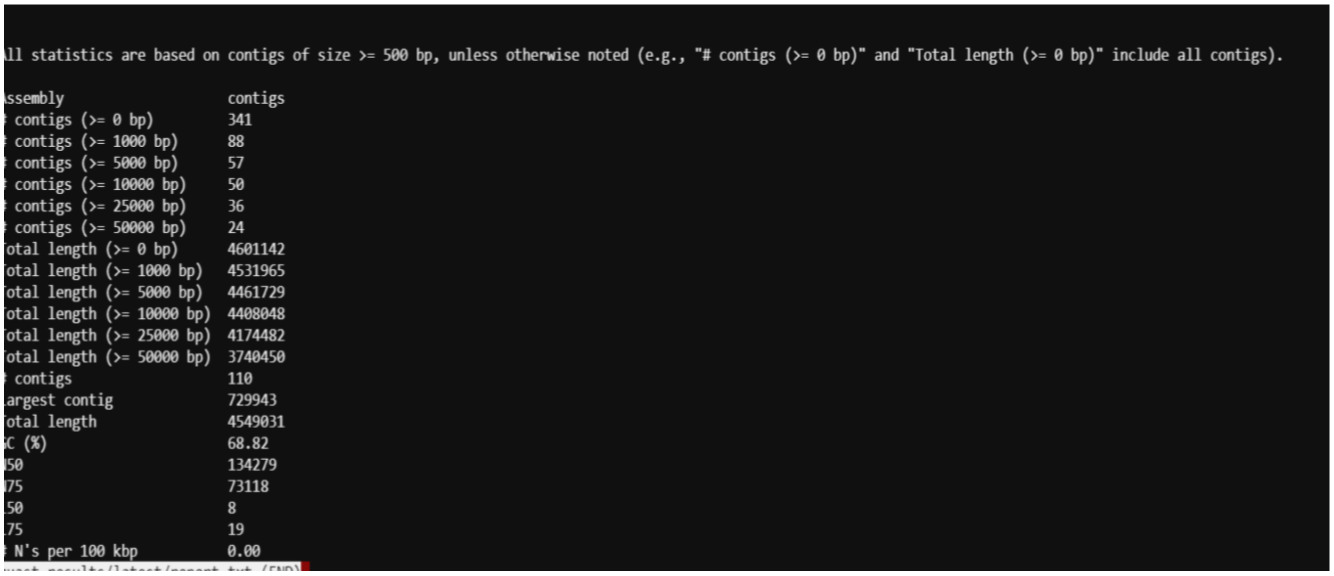
- -t number of threads (CPUs) to use for calculations
- --memory maximum memory usage in Gb
- -k k-mers to use (this gives room for experimenting!)
- -o name of the output folder

  Ouput files:

- error-corrected reads
- contigs for each individual k-mer assembly
- final contigs.fasta and scaffolds.fasta

  runQuast.sh

outputs a report file that has all of the parameters given below.



```
ll statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).

ssembly                   contigs
 contigs (>= 0 bp)        341
 contigs (>= 1000 bp)     88
 contigs (>= 5000 bp)     57
 contigs (>= 10000 bp)    50
 contigs (>= 25000 bp)    36
 contigs (>= 50000 bp)    24
otal length (>= 0 bp)     4601142
otal length (>= 1000 bp)  4531965
otal length (>= 5000 bp)  4461729
otal length (>= 10000 bp) 4408048
otal length (>= 25000 bp) 4174482
otal length (>= 50000 bp) 3740450
 contigs                  110
argest contig             729943
otal length               4549031
C (%)                     68.82
50                        134279
75                        73118
50                        8
75                        19
 N's per 100 kbp          0.00
uast_results/latest/report.txt (END)
```

Summary report

# contigs (≥ x bp) is total number of contigs of length ≥ x bp. Not affected by the --min-contig parameter

Total length (≥ x bp) is the total number of bases in contigs of length ≥ x bp. Not affected by the --min-contig parameter

# contigs is the total number of contigs in the assembly.

Largest contig is the length of the longest contig in the assembly.

Total length is the total number of bases in the assembly.

Reference length is the total number of bases in the reference genome.

GC (%) is the total number of G and C nucleotides in the assembly, divided by the total length of the assembly.

GC is the percentage of G and C nucleotides in the reference genome.

N50 the length for which the collection of all contigs of that length or longer covers at least half an assembly.

Genome assembly is a hierarchical process; it is performed in steps beginning from the assembly of the sequence reads into contigs, assembly of the contigs into scaffolds, and assembly of the scaffolds into chromosomes. The N50 contig value can be determined by first sorting all contigs in decreasing order of size, then adding the contigs until the total added size reaches at least half of the total size of all assembled contigs. The size of the smallest contig used in this addition process represents the N50. The scaffold N50 is calculated in the same fashion using the scaffold size. The larger the N50 value, the better is the assembly

SHORT READ ALIGNMENT:

We need additional programs to load beyond the existing environment for the tools gmap and gsnap to work

```
module avail gmap-gsnap


module load gmap-gsnap/[version]
```

GMAP is a tools for rapidly and accurately mapping and aligning cDNA sequences to genomic sequences.

GSNAP is designed to align short reads from NGS data and allow detection of short and long range splicing de novo or with a database of know juctions

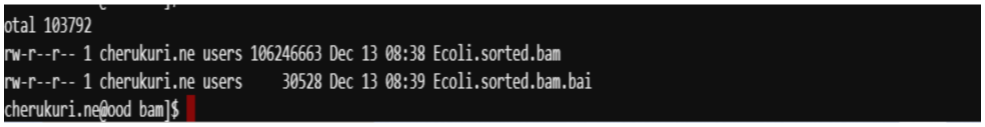
- Quality trimming the seq reads is the first most important step

- Building GMAP Database

- gmap_build -D data \ indicates to use data directory
- -d EcoliGmapDb \ creates the gmap database in data directory
- /home/username/module-directory/ShortReadAlignment/ecoli/contigs.fasta \ path to the f asta file. To get contigs.fasta file , execute genome assembly for RNA sequences
- head results/logs/AipBuild.log / confirm the results of log file
- gsnap \ add before the input and output paths
- Aligning gsnap result sam using samtools sort
- Quality trimming the seq reads using trimmomatic
- Aligning the trimmed reads
- Sorting the trimmed reads
- Indexing the trimmed reads

Example outputs:

For bam after sorting and indexing



```
otal 103792
rw-r--r-- 1 cherukuri.ne users 106246663 Dec 13 08:38 Ecoli.sorted.bam
rw-r--r-- 1 cherukuri.ne users     30528 Dec 13 08:39 Ecoli.sorted.bam.bai
cherukuri.ne@ood bam]$
```

Short read sequencing helps to filter the files sooner with gmap and gsnap

TRANSCRIPTOME ASSEMBLY:

In a wide variety of biological studies at the molecular level, transcriptome assembly utilizing next-generation sequencing data is a crucial step. The accuracy of computationally generated transcriptomes influences several downstream analyses, including isoform identification, gene expression analysis, and gene structure prediction. Assembled transcriptomes' real correctness, however, is frequently uncertain. The method utilized, the parameters (for example, k-mers) used with the approach, and the transcript that needs to be assembled are other variables that affect how well an assembly is done. Users frequently choose an assembly technique solely based on availability, disregarding the distinctions between methods and the parameters they chose. Insufficient benchmarking datasets are one factor in this. In this chapter, we offer a review of the computational methods utilized for transcriptome analysis assembly (genome-guided, de novo, and ensemble), factors influencing assembly success, especially those crucial for plant transcriptomes, and methods for evaluating the effectiveness of transcriptome assembly.

There are different approaches of DeNovo to transcriptome assembly:

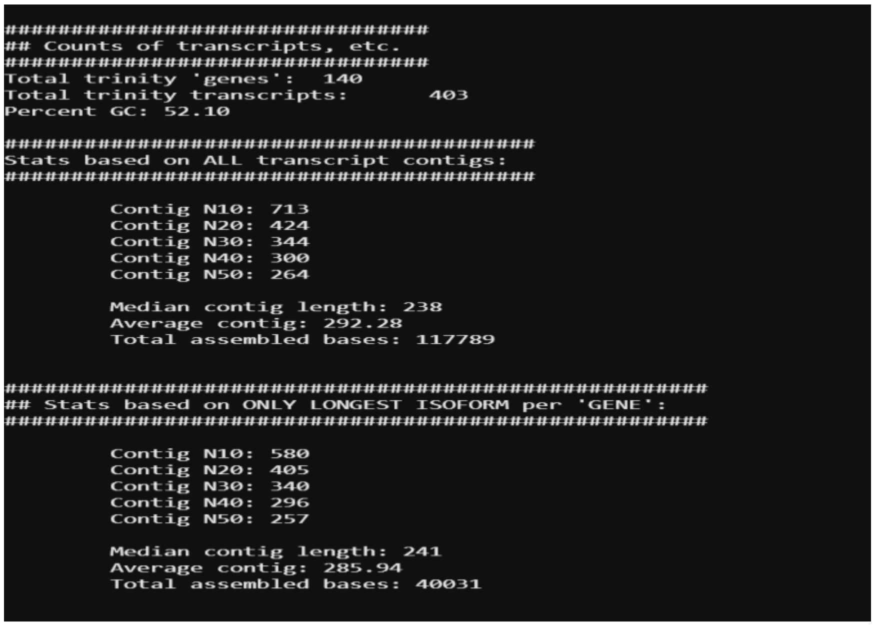
The majority of de novo transcriptome assembly methods, such as Trinity, IDBA-Tran, SOAPdenovo-Trans, and rnaSPAdes, use the de Bruijn graph based on k-mers. A k-mer of a sequence is a subsequence of length k, or k consecutive nucleotides. Inchworm, Chrysalis, and Butterfly are three of the modules that make up Trinity. Assembling reads into contigs is done by Inchworm using a greedy-extension based overlap approach after removing incorrect k-mers from the read sequences. Contigs are clustered by Chrysalis, and a de Bruijn graph is built for each cluster. Butterfly then navigates the graphs to create transcripts. The SOAPdenovo2 genome assembler has been extended to include SOAPdenovo-Trans. It eliminates edges that reflect incorrect k-mers using Trinity's mistake reduction techniques. The de Bruijn graphs' recovered contigs are mapped to reads to create linkages between them, and the contigs are then clustered into subgraphs based on the linkage data. To create the transcripts, each subparagraph is finally explored.

We need to load Trinity after loading modules to run assemblies

```
srun --partition=short --pty --export=ALL --nodes=1 --ntasks=1 --mem=10Gb --time=02:00:00 /b
in/bash
```

```
module load anaconda3/2021.11

source activate BINF-12-2021

Trinity#

--seqType <string> :type of reads: ('fa' or 'fq')

## --max_memory <string> :suggested max memory to use by Trinity where limiting can be enab
led. (jellyfish, sorting, etc)# provided in Gb of RAM, ie. '--max_memory 10G'
```

- Create symbolic links to copy the data from Short Read Alignment which has the files paired, unpaired, sam, and bam.
- Execute trinitydenovo.sh which has trimmed reads and outputs to trinity_de_novo
- Check the trinity_de_novo output directory and analyze it using Trinity.fasta
- Write a script to analyze runTrinity.sh script which outputs to trinity_guided_stats.txt
- Text file shows you the N50 assembly
- N50 is a metric widely used to assess the contiguity of an assembly, which is defined by the length of the shortest contig for which longer and equal-length contigs cover at least 50 % of the assembly. NG50 resembles N50 except the metric relates to the genome size rather than the assembly size.
- Output sample for text file.



```
################################
## Counts of transcripts, etc.
################################
Total trinity 'genes':  140
Total trinity transcripts:      403
Percent GC: 52.10

########################################
Stats based on ALL transcript contigs:
########################################

        Contig N10: 713
        Contig N20: 424
        Contig N30: 344
        Contig N40: 300
        Contig N50: 264

        Median contig length: 238
        Average contig: 292.28
        Total assembled bases: 117789


#####################################################
## Stats based on ONLY LONGEST ISOFORM per 'GENE':
#####################################################

        Contig N10: 580
        Contig N20: 405
        Contig N30: 340
        Contig N40: 296
        Contig N50: 257

        Median contig length: 241
        Average contig: 285.94
        Total assembled bases: 40031
```

BLAST AND TRANSDECODER

TransDecoder identifies candidate coding regions within transcript sequences. In the process of functional annotation, we attempt to attach names and actions to the transcripts we have assembled. Practically every method now in use uses similarity to achieve this. The function of this newly annotated transcript is probably the same if a translation product bears a striking resemblance to a protein that has already been given a function. Tools for de novo assembled transcriptomes, particularly those that are automatically functionally annotated, from model or non-model species. It employs BLAST+/SwissProt homology search to known sequence data, HMMER/PFAM protein domain identification, and signalP/tmHMM protein signal peptide and transmembrane domain prediction.

- Copy the trinity_out_dir out put from Trinity assembly

  Create longorf script

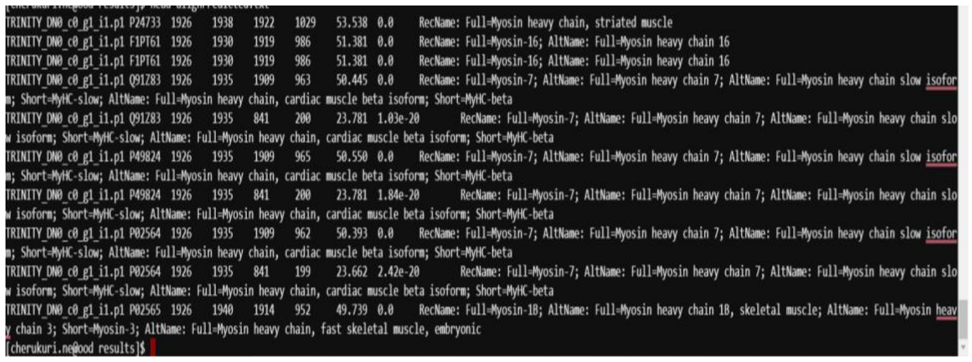
Finding the open reading frame is the second phase in the annotation process; after that, the transcript will be annotated. TransDecoder is the tool we utilize to complete this task. This tool has demonstrated its value, especially when used with the Trinotate pipeline.

The following steps are used by TransDecoder to determine potential coding sequences:

All transcripts that have open reading frames (ORFs) that are longer than a set minimum (by default, 100 amino acids) are recognized. No start, stop, or both are acceptable for incomplete ORFs.A reading frame-specific 5th-order Markov model is trained using these coding sequences using the top 500 longest ORFs.

- BLASTPEP scripts which outputs to outfm6
- PFAMSCAN scripts scan hmm files and outputs to pfam.domtblout
- Create a script predict protein with --retain_pfam_hits $4 \ --retain_blastp_hits $5 , also add the input and output
- Create alignpredicted script with Trinity.fasta.transdecoder.pep and analyze the sequences in tabular format by using:

"6 qseqid sacc qlen slen length nident pident evalue stitle"



```
TRINITY_DN0_c0_g1_i1.p1 P24733  1926    1938    1922    1029    53.538  0.0     RecName: Full=Myosin heavy chain, striated muscle
TRINITY_DN0_c0_g1_i1.p1 F1PT61  1926    1930    1919    986     51.381  0.0     RecName: Full=Myosin-16; AltName: Full=Myosin heavy chain 16
TRINITY_DN0_c0_g1_i1.p1 F1PT61  1926    1930    1919    986     51.381  0.0     RecName: Full=Myosin-16; AltName: Full=Myosin heavy chain 16
TRINITY_DN0_c0_g1_i1.p1 Q91Z83  1926    1935    1909    963     50.445  0.0     RecName: Full=Myosin-7; AltName: Full=Myosin heavy chain 7; AltName: Full=Myosin heavy chain slow isofor
m; Short=MyHC-slow; AltName: Full=Myosin heavy chain, cardiac muscle beta isoform; Short=MyHC-beta
TRINITY_DN0_c0_g1_i1.p1 Q91Z83  1926    1935    841     200     23.781  1.03e-20        RecName: Full=Myosin-7; AltName: Full=Myosin heavy chain 7; AltName: Full=Myosin heavy chain slo
w isoform; Short=MyHC-slow; AltName: Full=Myosin heavy chain, cardiac muscle beta isoform; Short=MyHC-beta
TRINITY_DN0_c0_g1_i1.p1 P49824  1926    1935    1909    965     50.550  0.0     RecName: Full=Myosin-7; AltName: Full=Myosin heavy chain 7; AltName: Full=Myosin heavy chain slow isofor
m; Short=MyHC-slow; AltName: Full=Myosin heavy chain, cardiac muscle beta isoform; Short=MyHC-beta
TRINITY_DN0_c0_g1_i1.p1 P49824  1926    1935    841     200     23.781  1.84e-20        RecName: Full=Myosin-7; AltName: Full=Myosin heavy chain 7; AltName: Full=Myosin heavy chain slo
w isoform; Short=MyHC-slow; AltName: Full=Myosin heavy chain, cardiac muscle beta isoform; Short=MyHC-beta
TRINITY_DN0_c0_g1_i1.p1 P02564  1926    1935    1909    962     50.393  0.0     RecName: Full=Myosin-7; AltName: Full=Myosin heavy chain 7; AltName: Full=Myosin heavy chain slow isofor
m; Short=MyHC-slow; AltName: Full=Myosin heavy chain, cardiac muscle beta isoform; Short=MyHC-beta
TRINITY_DN0_c0_g1_i1.p1 P02564  1926    1935    841     199     23.662  2.42e-20        RecName: Full=Myosin-7; AltName: Full=Myosin heavy chain 7; AltName: Full=Myosin heavy chain slo
w isoform; Short=MyHC-slow; AltName: Full=Myosin heavy chain, cardiac muscle beta isoform; Short=MyHC-beta
TRINITY_DN0_c0_g1_i1.p1 P02565  1926    1940    1914    952     49.739  0.0     RecName: Full=Myosin-1B; AltName: Full=Myosin heavy chain 1B, skeletal muscle; AltName: Full=Myosin heav
y chain 3; Short=Myosin-3; AltName: Full=Myosin heavy chain, fast skeletal muscle, embryonic
[cherukuri.ne@ood results]$
```

InterproScan Annotation

The functionality of InterPro annotations in Blast2GO allows to retrieved domain/motif information in a sequence-wise manner. Corresponding GO terms are then transferred to the sequences and merged with already existing GO terms. InterPro Scan (IPS) analysis showing the number of ESTs without IPS, with IPS, and GO domains in forward and reverse libraries.

Using IPS domain

- Create a script called prepare for IPS which outputs to a fasta file for number of lines you provide
- Create a script called runIPS which outputs a tsv file. Parameters that used are

  -goterms -pa -cpu 2 -f TSV, -dp. You can find about all this by interproscan.sh

- Create a script a file called summarizetsv and analyze the tsv file
- Analyze the text files

- go_ids.txt
- domains.txt
- path_ids.txt

Example output for domains.txt

```
681
              5 IPR001313     Pumilio RNA-binding repeat
              5 IPR000308     14-3-3 protein
              4 IPR008197     WAP-type 'four-disulfide core' domain
              2 IPR036645     Elafin-like superfamily
              2 IPR003593     AAA+ ATPase domain
```

Example output for go_ids.txt

```
1380
              9 -
              5 GO:0003723
              4 GO:0030414
              4 GO:0005576
```

Example output for path_ids.txt

```
1380
              8 Reactome: R-MMU-6798695
              8 Reactome: R-HSA-6798695
              8 Reactome: R-CEL-6807878
              8 Reactome: R-CEL-6798695
              7 Reactome: R-RNO-8854518
              7 Reactome: R-RNO-5620912
              7 Reactome: R-RNO-380320
              7 Reactome: R-RNO-380284
              7 Reactome: R-RNO-380270
              7 Reactome: R-RNO-380259
              7 Reactome: R-RNO-3371511
              7 Reactome: R-RNO-2565942
              7 Reactome: R-MMU-8854518
              7 Reactome: R-MMU-5620912
              7 Reactome: R-MMU-380320
              7 Reactome: R-MMU-380284
```

CONCLUSION

RNA-seq has established itself as a standard instrument in biology and has steadily spread to related disciplines of study like ecology. Long-read RNA-seq has opened up a number of promising new possibilities, including the direct sequencing of RNA molecules without the need for cDNA synthesis and the sequencing of RNA from individual cells. Bulk RNA-seq through short-read sequencing is still a popular technique despite these difficulties. One explanation is the persistent (and rising) acceptance of de novo transcriptome assembly and annotation for non-model organism research. In this situation, properly cataloging transcripts and discovering intriguing gene products can be accomplished at a low cost using well-annotated de novo constructed transcriptomes.

This ongoing and wide-ranging interest has meant that there is an unrelenting flood of constantly improved tools, databases, and workflows to aid assembly, annotation, and related analysis. While these tools have significantly reduced the effort involved in scientific discovery, the sheer number of resources accessible has made it difficult to select the best strategy for a given research subject. This may be especially true for novice practitioners who now have access to equipment that allows them to conduct RNA-seq tests totally in-house. De novo transcriptomics has considerable promise for these people because it now allows them to investigate almost any organism(s) of their choosing.

In order to achieve this, provided a thorough and user-friendly explanation of the main procedures and equipment needed for de novo transcriptome assembly and annotation of bulk short-read RNA-seq data. In their effort to acquire high-quality transcriptomes, new and seasoned researchers both should find this information useful.

REFERENCES

- Buccitelli, C., & Selbach, M. (2020). mRNAs, proteins and the emerging principles of gene expression control. *Nature Reviews Genetics*, *21*(10), 630-644.
- Wang, Z., Gerstein, M., & Snyder, M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, *10*(1), 57-63.
- Margulies, M., Egholm, M., Altman, W. E., Attiya, S., Bader, J. S., Bemben, L. A., ... & Rothberg, J. M. (2005). Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, *437*(7057), 376-380.
- Maher, C. A., Kumar-Sinha, C., Cao, X., Kalyana-Sundaram, S., Han, B., Jing, X., ... & Chinnaiyan, A. M. (2009). Transcriptome sequencing to detect gene fusions in cancer. *Nature*, *458*(7234), 97-101.
- Altschul, S. F., T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. 1997. "Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs." *Nucleic Acids Res* 25 (17): 3389–3402.
- Finn, Robert D., Jody Clements, William Arndt, Benjamin L. Miller, Travis J. Wheeler, Fabian Schreiber, Alex Bateman, and Sean R. Eddy. 2015. "HMMER Web Server: 2015 Update." *Nucleic Acids Res* 43 (W1): W30–W38.
- Haas, Brian J., Alexie Papanicolaou, Moran Yassour, Manfred Grabherr, Philip D. Blood, Joshua Bowden, Matthew Brian Couger, et al. 2013. "De Novo Transcript Sequence Reconstruction from RNA-Seq Using the Trinity Platform for Reference Generation and Analysis." *Nat Protoc* 8 (8): 1494–1512.