De novo transcriptome assembly workflow

Jared Mamrot, Roxane Legaie, Stacey J Ellery, Trevor Wilson, Torsten Seemann, David Gardner, David W Walker, Peter Temple-Smith, Anthony T Papenfuss, Hayley Dickinson

Abstract

This protocol describes the production of a reference-quality *de novo* transcriptome assembly for the spiny mouse (*Acomys cahirinus*). These methods can be applied to other RNA-Seq datasets to generate high-quality transcriptome assemblies in other species. Validation and description of assembly output is described in 'De novo transcriptome assembly for the spiny mouse (*Acomys cahirinus*)' *bioRxiv*, p.076067.

Citation: Jared Mamrot, Roxane Legaie, Stacey J Ellery, Trevor Wilson, Torsten Seemann, David Gardner, David W Walker, Peter Temple-Smith, Anthony T Papenfuss, Hayley Dickinson De novo transcriptome assembly workflow. **protocols.io**
dx.doi.org/10.17504/protocols.io.ghebt3e
Published: 06 Mar 2017

Protocol

Import and organise raw data

Step 1.

Download raw data from the NCBI to working directory and archive a copy (read-only). To efficiently transfer data the NCBI recommends using Aspera connect, a FASP® transfer program which facilitates high-speed data transfer.

Many commands in this protocol take hours/days to complete: to avoid processes being killed if connection to the server is lost, employ the 'nohup' command and/or run processes in the background ('&') and disown them from the terminal ('disown %1'). Where possible, follow good scientific practices eg. Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L. and Teal, T.K., 2016. Good Enough Practices in Scientific Computing. *arXiv preprint arXiv:1609.00037*.

Aspera connect:
Download - http://downloads.asperasoft.com/en/downloads/8?list (ver3.6.2)

Documentation - https://www.ncbi.nlm.nih.gov/books/NBK242625/
Requirements - NCBI SRA toolkit

NCBI SRA toolkit:
Download - https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software (ver2.8.1-3)

Documentation - https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc

DATASET
Illumina HiSeq 1500 RNA-Seq data NCBI BioProject Accession: PRJN
COMMAND (Unix - bash)
```
#Create working directory and directory for installed software
mkdir $HOME/projects $HOME/projects/spiny_mouse
export WORKDIR=$HOME/projects/spiny_mouse/
cd $WORKDIR && mkdir user_installed_software
export PROGRAMDIR=$WORKDIR/user_installed_software

#Download, unpack, and install aspera connect
cd $PROGRAMDIR
wget http://download.asperasoft.com/download/sw/connect/3.6.2/aspera-connect-3.6.2.117442-l
inux-64.tar.gz -O aspera.tar.gz
tar zxvf aspera.tar.gz && rm aspera.tar.gz
bash aspera-connect*
cd ~/.aspera/connect/bin
#add binaries to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Download reads from the NCBI
cd $WORKDIR
ascp -i ~/.aspera/connect/etc/asperaweb_id_dsa.openssh -T anonftp@ftp-
trace.ncbi.nlm.nih.gov:/sra/sra-instant/reads/ByRun/sra/SRR/SRR427/SRR4279903 anonftp@ftp-
trace.ncbi.nlm.nih.gov:/sra/sra-instant/reads/ByRun/sra/SRR/SRR427/SRR4279904 .

#Obtain reads in fastq format using the ncbi SRA Toolkit
find . -name "*.sra" -exec fastq-dump --split-spot --split-files --skip-technical -I -F -
Q 33 -W -T -R pass '{}' \;
cd SRR4279903/pass/1 && mv fastq Lane1_R1.fastq
cd ../2 && mv fastq Lane1_R2.fastq
cd ../../../SRR4279904/pass/1 && mv fastq Lane2_R1.fastq
cd ../2 && mv fastq Lane2_R2.fastq

#Move fastq files to WORKDIR
cd $WORKDIR
find . -name "Lane*" -exec mv '{}' $WORKDIR \; && rm -R SRR427990*

#Delete sra files
cd $WORKDIR/
find . -name "*.sra" -delete

#Archive a read-only copy of the raw data
cd $WORKDIR/
mkdir protected_data && cd protected_data
cp ../Lane* .
chmod 444 Lane*
```

Decompress and examine RNA-Seq read quality

Step 2.

Decompress gzipped files (*.gz), and use FastQC for preliminary read quality assessment. GNU zip (gzip) is a popular compression utility free from patented algorithms. FastQC is a quality control tool for high throughput sequence data which assesses multiple metrics and provides a QC report.

gzip:
Download - https://www.gnu.org/software/gzip/ (ver1.2.4)
Documentation - http://www.math.utah.edu/docs/info/gzip_toc.html

fastQC:
Download - http://www.bioinformatics.babraham.ac.uk/projects/download.html#fastqc (ver0.11.15)
Documentation - http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/
Requirements - A suitable Java Runtime Environment (https://www.java.com/en/) and the Picard BAM/SAM Libraries (included in download)

cmd COMMAND (Unix - bash)

```
#Download and install GZIP and FastQC
cd $PROGRAMDIR
wget http://www.gzip.org/gz124src.zip
wget http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.5.zip
#unpack
unzip *.zip
#install
cd gzip124src
./configure --prefix=`pwd`
make
make install
#add binaries to a directory contained in PATH, or add current directory to PATH
cd bin && echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Set file permissions for FastQC
cd $PROGRAMDIR/FastQC
chmod 755 fastqc
#add binaries to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#If required, decompress fastq files downloaded from the NCBI (may not be necessary dependi
ng on SRAtoolkit version)
cd $WORKDIR
gunzip *.fastq.gz

#Run FastQC on each sample to assess read quality
for f in Lane*; do fastqc $f; done
```

Trim adapters and re-examine read quality

Step 3.

Trim_galore is a tool that implements cutadapt to consistently apply quality and adapter trimming to FastQ files: it seeks out and removes adapter sequences from RNA-Seq data.

Trim_galore:
Download - https://github.com/FelixKrueger/TrimGalore/releases (ver0.4.2)
Documentation -

http://www.bioinformatics.babraham.ac.uk/projects/trim_galore/Trim_Galore_User_Guide_v0.4.2.pdf
Requirements: Cutadapt and FastQC

Cutadapt:
Download - https://pypi.python.org/pypi/cutadapt (ver1.9.1)
Documentation - https://media.readthedocs.org/pdf/cutadapt/stable/cutadapt.pdf
Reference - Martin, M., 2011. Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet. journal, 17(1), pp.pp-10.

*Minimum read length parameter ('--length 80') is dependant on length of original reads (in this case 150bp, paired-end)

cmd COMMAND (Unix - bash)

```
#Download and install cutadapt using pip
cd $PROGRAMDIR
pip install --user --upgrade cutadapt

#Download and install cutadapt using anaconda (https://www.continuum.io/downloads)
cd $PROGRAMDIR
conda install -c bioconda cutadapt && python setup.py install --user

#Download trim_galore
cd $PROGRAMDIR
wget https://github.com/FelixKrueger/TrimGalore/archive/0.4.2.tar.gz -O trim_galore.tar.gz
#unpack
tar zxvf trim_galore.tar.gz && cd TrimGalore-0.4.2
#add binary to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Run trim_galore on each lane
mkdir $WORKDIR/adapter_trimmed_reads && cd $WORKDIR
trim_galore --paired --retain_unpaired --length 80 --
output ./adapter_trimmed_reads Lane1_R1.fastq Lane1_R2.fastq 1>trim_galore_lane_1_reads.log
 2>trim_galore_lane_1_reads.err &

trim_galore --paired --retain_unpaired --length 80 --
output ./adapter_trimmed_reads Lane2_R1.fastq Lane2_R2.fastq 1>trim_galore_lane_2_reads.log
 2>trim_galore_lane_2_reads.err &

#Stdout ('1>') and stderr ('2>') are redirected to .log/.err files to aid in 'disowning' pr
ocesses from the terminal session ('disown %1')
```

Remove poor quality reads

Step 4.

Remove poor quality reads and trim poor quality nucleotides from read ends using trimmomatic. Trimmomatic is a flexible read trimming tool for Illumina NGS data. Re-assess 'cleaned' reads using FastQC to check read metrics have been improved.

Using a reletively high quality threshold improves assembler performance and reduces memory requirements, however lowly expressed transcripts may be lost as proportionately more reads are excluded. Conversely, a relatively low quality threshold may improve retention of lowly expressed transcripts, but negatively affect assembler performance (https://doi.org/10.3389/fgene.2014.00013). The optimal degree of read trimming and removal can differ for each dataset.

Several read quality thresholds were used in the publication associated with this protocol (Mamrot, J., Legaie, R., Ellery, S.J., Wilson, T., Gardner, D., Walker, D.W., Temple-Smith, P., Papenfuss, A.T. and Dickinson, H., 2016. De novo transcriptome assembly for the spiny mouse (Acomys cahirinus). *bioRxiv*, p.076067). Each dataset was assembled (Steps 5-9), validated and compared for accuracy and completeness. For this dataset a relatively low quality threshold (as outlined in the COMMAND section below) produced higher quality assemblies.

In addition to differing levels of trimming/filtering, read error correction was conducted using SEECER (http://sb.cs.cmu.edu/seecer/). Assembly metrics were slightly improved when error-corrected reads were assembled with Trinity (this 'error-corrected' assembly was included in downstream analyses), however there were no noticeable improvements assembling error-corrected reads with SOAPdenovo-Trans and Velvet/Oases with this dataset. Despite this unexpected outcome, error correction is still recommended for best-practice transcriptome assembly (http://oyster-river-protocol.readthedocs.io/en/master/; http://dx.doi.org/10.1101/035642; https://doi.org/10.7717/peerj.113)

Trimmomatic:
Download - http://www.usadellab.org/cms/?page=trimmomatic (ver0.36)
Documentation - http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf
Reference - Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. Bioinformatics, btu170.

SEECER:

Download - http://sb.cs.cmu.edu/seecer/install.html#Download (ver1.3)

Documentation - http://sb.cs.cmu.edu/seecer/downloads/manual.pdf

Reference - Hai-Son Le, Marcel H. Schulz, Brenna M. McCauley, Veronica F. Hinman and Ziv Bar-Joseph (2013). Probabilistic error correction for RNA sequencing. Nucleic Acids Research /

Requirements - GNU Scientific Library, SeqAn, Jellyfish, OPENMP API

cmd COMMAND (Unix - bash)

```
#There is substantial overlap in the capabilities of trim_galore and trimmomatic; each prog
ram was used here for different purposes: trim galore to seek out and remove adapters, trim
momatic for trimming poor quality bases/reads.

#Download and install trimmomatic using linuxbrew (http://linuxbrew.sh/)
cd $PROGRAMDIR
brew tap homebrew/science
brew install trimmomatic
#alternatively, download trimmomatic from source
cd $PROGRAMDIR
wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.36.zip
unzip Trimmomatic-0.36.zip && rm Trimmomatic-0.36.zip && cd Trimmomatic-0.36/
#add binary to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

cd $WORKDIR/adapter_trimmed_reads
trimmomatic PE -
phred33 Lane1_R1_val_1.fq Lane1_R2_val_2.fq lane1_R1_pairedout.fq lane1_R1_unpairedout.fq l
ane1_R2_pairedout.fq lane1_R2_unpairedout.fq LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 AVGQUA
L:20 MINLEN:36 1>trimmomatic_1.log 2>trimmomatic_1.err &
trimmomatic PE -
phred33 Lane2_R1_val_1.fq Lane2_R2_val_2.fq lane2_R1_pairedout.fq lane2_R1_unpairedout.fq l
ane2_R2_pairedout.fq lane2_R2_unpairedout.fq LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 AVGQUA
L:20 MINLEN:36 1>trimmomatic_2.log 2>trimmomatic_2.err &

mkdir $WORKDIR/cleaned_trimmed_reads
mv *_pairedout.fq $WORKDIR/cleaned_trimmed_reads/

#Assess read quality using FastQC, and compare with untrimmed/unfiltered reads
cd $WORKDIR/cleaned_trimmed_reads
for f in *pairedout.fq; do fastqc $f; done

#Concatenate lanes
cat lane1_R1_pairedout.fq lane2_R1_pairedout.fq > R1_pairedout.fastq
cat lane1_R2_pairedout.fq lane2_R2_pairedout.fq > R2_pairedout.fastq

#optionally, unpairedout.fq reads can be added to R2_pairedout.fq prior to assembly

#Download and install SEECER
cd $PROGRAMDIR
wget http://sb.cs.cmu.edu/seecer/downloads/SEECER-0.1.3.tar.gz
tar zxvf SEECER-0.1.3.tar.gz && rm SEECER-0.1.3.tar.gz
cd SEECER-0.1.3
./configure --prefix=`pwd`
make
make install
#add binary to a directory contained in PATH, or add current directory to PATH
cd bin && echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Conduct error correction on trimmed/filtered reads using SEECER
#run 'run_seecer.sh' on testdata to check install, then symlink real data into ./testdata d
irectory and run the 'run_seecer.sh' script
cd $PROGRAMDIR/SEECER-0.1.3/testdata.
ln -s $WORKDIR/cleaned_trimmed_reads/R1_pairedout.fastq R1_pairedout.fastq
ln -s $WORKDIR/cleaned_trimmed_reads/R2_pairedout.fastq R2_pairedout.fastq
cd $PROGRAMDIR/SEECER-0.1.3/
bash ./bin/run_seecer.sh -
t tmpDirectory ./testdata/R1_pairedout.fastq ./testdata/R2_pairedout.fastq 1>seecer.log 2>s
eecer.err &
```

ANNOTATIONS

Jared Mamrot 31 Jul 2017

A consequence of SEECER is that your fastq reads are converted into fasta format (i.e. base quality scores are lost). This limits your options further down the pipeline. Rcorrector is a great alternative provided you have a reasonable number of reads (eg >30 million): https://github.com/mourisl/Rcorrector / install via linuxbrew: "brew install rcorrector" / Song, L., Florea, L., Rcorrector: Efficient and accurate error correction for Illumina RNA-seq reads. GigaScience. 2015, 4:48.

Prepare for SOAPdenovo-Trans assembly

Step 5.

SOAPdenovo-Trans is a de Bruijn graph-based assembler for transcriptome data, derived from the SOAPdenovo2 genome assembler. It incorporates the innovative error-removal model from Trinity and combines this with the robust heuristic graph traversal method for solving isoform-related sub-graphs from Oases (both of Trinity and Oases are subsequently employed in this protocol). Further details on the algorithms and applications of SOAPdenovo-Trans are available at https://doi.org/10.1093/bioinformatics/btu077

SOAPdenovo-Trans requires user-specified parameters to be listed in a 'config' file.

SOAPdenovo-Trans:
Download - http://soap.genomics.org.cn/SOAPdenovo-Trans.html (ver1.03)
Documentation - http://soap.genomics.org.cn/SOAPdenovo-Trans.html / https://github.com/aquaskyline/SOAPdenovo-Trans
Reference - Xie, Y., Wu, G., Tang, J., Luo, R., Patterson, J., Liu, S., Huang, W., He, G., Gu, S., Li, S. and Zhou, X., 2014. SOAPdenovo-Trans: de novo transcriptome assembly with short RNA-Seq reads. Bioinformatics, 30(12), pp.1660-1666.

cmd COMMAND (Unix - bash)

```
#Download
cd $PROGRAMDIR
mkdir SOAPdenovo-Trans
wget https://downloads.sourceforge.net/project/soapdenovotrans/SOAPdenovo-Trans/bin/v1.03/S
OAPdenovo-Trans-bin-v1.03.tar.gz -O SOAPdenovo-Trans/SOAPdenovo-Trans-bin-v1.03.tar.gz
cd SOAPdenovo-Trans/
tar zxvf SOAPdenovo-Trans-bin-v1.03.tar.gz
#add binary to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Create config file- options are described/explained in the SOAPdenovo-Trans documentation
cd $WORKDIR/cleaned_trimmed_reads
cat > SOAP.config <<END_TEXT
max_rd_len=150
[LIB]
avg_ins=192
```

```
reverse_seq=0
asm_flags=3
q1=R1_pairedout.fastq
q2=R2_pairedout.fastq
END_TEXT
```

Assemble reads using SOAPdenovo-Trans

Step 6.

Assemble reads at multiple kmer sizes using SOAPdenovo-Trans.

Use SOAPdenovo-Trans31mer for kmer sizes 21 - 31.

Use SOAPdenovo-Trans127mer for kmer sizes 41 - 121.

After assembly is complete, statistics are contained in the *.scafStat file, and scaffolded contigs in the *.scafSeq file.

cmd COMMAND (Unix - bash)

```
cd $WORKDIR/cleaned_trimmed_reads
mkdir SOAPdenovo-Trans_output
for ((n=21; n<33; n=n+2)); do SOAPdenovo-Trans-31mer all -K $n -p 32 -s SOAP.config -
o SOAPdenovo-Trans_output/SOAP_$n 1>SOAP_k$n.log 2>SOAP_k$n.err; done
for ((n=41; n<131; n=n+10)); do SOAPdenovo-Trans-127mer all -K $n -p 32 -s SOAP.config -
o SOAPdenovo-Trans_output/SOAP_$n 1>SOAP_k$n.log 2>SOAP_k$n.err; done
for ((n=35; n<135; n=n+10)); do SOAPdenovo-Trans-127mer all -K $n -p 32 -s SOAP.config -
o SOAPdenovo-Trans_output/SOAP_$n 1>SOAP_k$n.log 2>SOAP_k$n.err; done

#SOAPdenovo-
Trans is a relatively fast assembler, however assembly may require a large amount of RAM (>
500GB)
```

Assemble reads with Trinity

Step 7.

Assemble reads using Trinity (k-mer size set at 25). Trinity is the most widely used de novo transcriptome assembler: it is a de Bruijn graph-based assembler that implements an error removal model when constructing transcripts. It retains and clusters transcript isoforms, and is often as a benchmark to evaluate novel assemblers.

In silico normalization can be employed in the Trinity pipeline to reduce read counts before assembly. The normalized reads can be obtained (from /trinity_out_dir/insilico_read_normalization/*.fq) and assembled using other assemblers (SOAPdenovo-Trans and Velvet/Oases; Repeating steps 5-9).

Assembly statistics can be generated using the bundled 'TrinityStats.pl' script.

Trinity:
Download - https://github.com/trinityrnaseq/trinityrnaseq/releases (ver2.3.2)
Documentation - https://github.com/trinityrnaseq/trinityrnaseq/wiki

Requirements - Trinity comes bundled with required programs, including fastool, jellyfish, parafly, samtools and trimmomatic.
Reference - Grabherr, M.G., Haas, B.J., Yassour, M., Levin, J.Z., Thompson, D.A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q. and Chen, Z., 2011. Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nature biotechnology, 29(7), pp.644-652.

cmd COMMAND (Unix - bash)

```
#Install Trinity
cd $PROGRAMDIR
wget https://github.com/trinityrnaseq/trinityrnaseq/archive/Trinity-v2.3.2.tar.gz -O Trinit
y-v2.3.2.tar.gz
tar zxvf Trinity-v2.3.2.tar.gz
cd trinityrnaseq
make
make plugins
#test installation
cd sample_data/test_Trinity_Assembly/ && ./runMe.sh
#if successful, add trinityrnaseq, trinity-
plugins, util, misc, and support_scripts directories to PATH
echo export PATH=$PATH$( find $PROGRAMDIR/trinityrnaseq/ -type d -
printf ":%p" ) >> ~/.bashrc && source ~/.bashrc

#Use Trinity to assemble error-corrected (fasta) and non-error-
corrected (fastq) reads. Some user-defined parameters for Trinity are version-
specific: check documentation before use

cd $WORKDIR/cleaned_trimmed_reads
Trinity --seqType fq --JM 40G --left R1_pairedout.fastq --right R2_pairedout.fastq --
CPU 12 --min_kmer_cov 2 --output Trinity-DN 1>trinity.log 2>trinity.err

#Move Trinity assembly to Trinity-DN directory
mv $WORKDIR/cleaned_trimmed_reads/Trinity-DN/trinity_out_dir/Trinity.fasta ..

#Basic assembly statistics can be generated using the TrinityStats.pl script included in th
e package
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/
TrinityStats.pl Trinity.fasta > Stats.txt
```

Align reads to Trinity.fasta

Step 8.

Align reads to assembled transcripts using Bowtie to provide a measure of transcript 'completeness', and to calculate the average insert size if unknown (required for Velvet / Oases). Bowtie is a fast, memory-efficient short read aligner and Picard tools is a collection of command line utilities for processing next-gen sequencing data.

Bowtie:
Download - https://sourceforge.net/projects/bowtie-bio/files/bowtie/1.2.0/ (ver1.2)
Documentation - http://bowtie-bio.sourceforge.net/index.shtml
Reference - Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10:R25.

Picard tools:
Download - https://broadinstitute.github.io/picard/ (ver2.1.1)
Documentation - https://broadinstitute.github.io/picard/command-line-overview.html
Reference - https://broadinstitute.github.io/picard/

cmd COMMAND (Unix - bash)

```
#Install Bowtie with anaconda
conda install bowtie
#install bowtie from tarball
cd $PROGRAMDIR
wget https://downloads.sourceforge.net/project/bowtie-bio/bowtie/1.2.0/bowtie-1.2-linux-x86
_64.zip -O bowtie-1.2.zip
#unpack
unzip bowtie-1.2.zip && cd bowtie-1.2
#add binaries to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Install Picard tools using linuxbrew
brew install picard-tools
#install from source
wget https://github.com/broadinstitute/picard/releases/download/2.1.1/picard-tools-2.1.1.zi
p -O picard-tools-2.1.1.zip
unzip picard-tools-2.1.1.zip
#add java binaries to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Build a bowtie alignment database
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/
bowtie-build Trinity.fasta Trinity_bowtie

#Run bowtie to align fastq reads to the assembly
bowtie -q --phred33-quals -n 2 -e 99999999 -l 25 -I 1 -X 1000 -p 20 -a -m 200 --
chunkmbs 128 -
S Trinity_bowtie -1 R1_pairedout.fastq -2 R2_pairedout.fastq trinity_backmap_paired.sam 1>b
owtie_trinity_backmapping.log 2>bowtie_trinity_backmapping.err

#Convert alignment file in SAM format (human readable) to BAM format (binary)
samtools view -
bS trinity_backmap_paired.sam > trinity_backmap_paired.bam && rm trinity_backmap_paired.sam

#Classify alignments as properly paired (i.e. on the same contig with a reasonable insert l
ength), aligned, or not aligned using the samtools flagstat tool
samtools flagstat trinity_backmap_paired.bam > trinity_backmap_paired_flagstat.txt

#Calculate the average insert size
java -
jar picard.jar CollectInsertSizeMetrics I=Trinity_backmap_paired.bam O=insert_size_output.t
```

```
xt H=insert_size_output.pdf M=0.5
```

Assembling reads with Velvet / Oases

Step 9.

Velvet is a de Bruijn graph based short-read assembler designed for de novo genome assembly. Oases takes contigs assembled using Velvet and solves sub-graphs caused by transcript isoforms to build transcripts.

Use Velvet to assemble reads at multiple kmer sizes (21, 23, 25, 27, 29, 31, 35, 41, 51, 61, 71, 81, 91, 101, 111, 121), then use Oases to assemble contigs into transcripts.

Assembly statistics are contained in the file 'stats.txt'.

Velvet:
Download - https://www.ebi.ac.uk/zerbino/velvet/ (ver1.2.10)
Documentation - http://www.ebi.ac.uk/zerbino/velvet/Manual.pdf
Reference - Velvet: algorithms for de novo short read assembly using de Bruijn graphs. D.R. Zerbino and E. Birney. Genome Research 18:821-829.

Oases:
Download - http://www.ebi.ac.uk/zerbino/oases/ (ver0.2.08)
Documentation - http://www.ebi.ac.uk/zerbino/oases/OasesManual.pdf
Reference -M.H. Schulz, D.R. Zerbino, M. Vingron and Ewan Birney. Oases: Robust de novo RNA-seq assembly across the dynamic range of expression levels. Bioinformatics, 2012. DOI: 10.1093/bioinformatics/bts094.

cmd COMMAND (Unix - bash)

```
#Download velvet
cd $PROGRAMDIR
wget http://www.ebi.ac.uk/~zerbino/velvet/velvet_1.2.10.tgz -O velvet_1.2.10.tar.gz
#unpack
tar zxvf velvet_1.2.10.tar.gz && cd velvet_1.2.10
#specify number of threads to use for assembly and set variables for velvet
export OMP_NUM_THREADS=31
export OMP_THREAD_LIMIT=32
make 'MAXKMERLENGTH=141' 'BIGASSEMBLY=1' 'OPENMP=1' 'LONGSEQUENCES=1'
#add binaries to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Download Oases
cd $PROGRAMDIR
git clone --recursive https://github.com/dzerbino/oases.git
cd oases
export OMP_NUM_THREADS=31
export OMP_THREAD_LIMIT=32
make 'MAXKMERLENGTH=141' 'BIGASSEMBLY=1' 'OPENMP=1' 'LONGSEQUENCES=1'
#add binary to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Run velveth
cd $WORKDIR/cleaned_trimmed_reads/
mkdir velvet-oases_output && cd velvet-oases_output/
~/velvet_1.2.10/velveth dir 21,37,2 -shortPaired -fastq -
separate ../R1_pairedout.fastq ../R2_pairedout.fastq 1>velveth_21-35.log 2>velveth_21-35.er
r
~/velvet_1.2.10/velveth dir 41,131,10 -fastq -shortPaired -
separate ../R1_pairedout.fastq ../R2_pairedout.fastq 1>velveth_41-121.log 2>velveth_41-121.
err

#Run velvetg
cd $WORKDIR/cleaned_trimmed_reads/velvet-oases_output/
for f in dir_*; do velvetg $f -read_trkg yes -ins_length 215 1>$f.log 2>$f.err; done

#Run Oases (include insert length calculated in step 8)
cd $WORKDIR/cleaned_trimmed_reads/velvet-oases_output/
for f in dir_*; do oases $f -min_trans_lgth 100 -
ins_length 215 1>oases_$f.log 2>oases_$f.err; done
```

Remove redundant transcripts

Step 10.

Reduce transcript redundancy using the CD-HIT algorithm. CD-HIT is used to cluster similar biological sequences together, reducing sequence redundancy and improving the performance and accuracy of specific downstream sequence analyses.

De novo assembly often produces highly similar sequences. These can be biological, for example transcript isoforms and transcripts from the same family, or artifacts of the assembly process, such as chimeric transcripts, unsupported insertions, incomplete/fragmented/misassembled or duplicate transcripts. The threshold for clustering depends on the specific analysis being performed: to cluster exact duplicates a threshold of '1.0' is used, to cluster transcripts that are share 99% similarity a threshold of '0.99' is used, and so on.

CD-HIT-EST:
Download - https://github.com/weizhongli/cdhit (ver4.6.4)
Documentation - http://weizhongli-lab.org/lab-wiki/doku.php?

Reference - 'Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences', Weizhong Li & Adam Godzik Bioinformatics, (2006) 22:1658-9. / Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu and Weizhong Li, CD-HIT: accelerated for clustering the next generation sequencing data. Bioinformatics, (2012), 28 (23): 3150-3152. doi: 10.1093/bioinformatics/bts565.

*All analyses and validation steps described in this protocol use 'un-clustered' assembly .fasta files.

cmd COMMAND (Unix - bash)

```
#Install CD-HIT-EST
cd $PROGRAMDIR
git clone --recursive https://github.com/weizhongli/cdhit.git
cd cdhit
make
#add binaries to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#for SOAPdenovo-Trans
cd $WORKDIR/cleaned_trimmed_reads/SOAPdenovo-Trans_output/ && mkdir SOAP_cdhit
for f in SOAP*/*.contig; do cd-hit-est -i $f -o SOAP_cdhit/clustered_95_$f -c 0.95 -n 8 -
p 1 -g 1 -M 200000 -T 12 -d 40 1>SOAP_cdhit/$f.log 2>SOAP_cdhit/$f.err; done

#for Trinity
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/ && mkdir trinity_cdhit
cd-hit-est -i Trinity.fasta -o trinity_cdhit/Trinity_clustered_0.95 -c 0.95 -n 8 -p 1 -
g 1 -M 200000 -T 12 -d 40 1>trinity_cdhit/Trinity_cd-hit-
est_0.95.log 2>trinity_cdhit/Trinity_cd-hit-est_0.95.err

#for Velvet/Oases
cd $WORKDIR/cleaned_trimmed_reads/velvet-oases_output/ && mkdir velvet-oases_cdhit
for f in dir_*; do cd-hit-est -i $f/transcripts.fa -o velvet-oases_cdhit/clustered_95_$f -
c 0.95 -n 8 -p 1 -g 1 -M 200000 -T 12 -d 40 1>velvet-oases_cdhit/$f.log 2>velvet-
oases_cdhit/$f.err; done
```

Remove SOAPdenovo-Trans 'gaps'

Step 11.

Run GapCloser on the SOAPdenovo-Trans assemblies. GapCloser is bundled with SOAPdenovo2, and removes strings of 'N' introduced in transcripts during scaffolding by SOAPdenovo-Trans. Removal of these gaps can improve accuracy and performance in specific downstream analysis. GapCloser is designed to remove assembly artifacts (N's) using the abundant pair relationships of short reads.

GapCloser:

Download - https://sourceforge.net/projects/soapdenovo2/files/GapCloser/ (ver1.12-r6)
Documentation - Read the GapCloser_Manual.pdf in GapCloser-bin-v1.12-r6.tgz
Reference - Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., Liu, Y. and Tang, J., 2012. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. Gigascience, 1(1), p.18.

*Use of 'non-GapClosed' or 'post-GapClosed' SOAPdenovo-Trans assemblies depends on the analyses applied. In many cases the inclusion of long strings of 'N' will cause errors or poor performance: if this occurs, use the GapClosed assemblies.

cmd COMMAND (Unix - bash)

```
#Download and install GapCloser
cd $PROGRAMDIR
wget https://sourceforge.net/projects/soapdenovo2/files/GapCloser/src/r6/GapCloser-src-v1.1
```

```
2-r6.tgz/download --no-check-certificate -O GapCloser.tar.gz
tar zxvf GapCloser.tar.gz
cd v1.12-r6 && make
#add binaries to a directory contained in PATH, or add current directory to PATH
cd bin && echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Copy assemblies to a single directory and rename to *.fasta
cd $WORKDIR/cleaned_trimmed_reads/SOAPdenovo-Trans_output/
mkdir SOAP_assembly_fasta_files
for f in SOAP*; do cp $f/*.scafSeq SOAP_assembly_fasta_files; done
cd SOAP_assembly_fasta_files
for f in *.scafSeq; do mv $f $(echo $f | sed 's/.scafSeq/.fasta/g'); done

#Create a config file (similar to Step 5)
cd $WORKDIR/cleaned_trimmed_reads/SOAPdenovo-Trans_output/SOAP_assembly_fasta_files
cat > GapCloser.config <<END_TEXT
max_rd_len=150
[LIB]
avg_ins=192
reverse_seq=0
asm_flags=3
q1=$WORKDIR/cleaned_trimmed_reads/R1_pairedout.fastq
q2=$WORKDIR/cleaned_trimmed_reads/R2_pairedout.fastq
END_TEXT

#Run GapCloser
cd $WORKDIR/cleaned_trimmed_reads/SOAPdenovo-Trans_output/SOAP_assembly_fasta_file
for f in *.fasta; do GapCloser -b GapCloser.config -a $f -o GapClosed_$f -l 150 -
t 24 1>GapCloser_{$f}.log 2>GapCloser_{$f}.err; done
```

Merge single k-mer assemblies

Step 12.

Move all transcriptome assemblies to a single directory. Use the EvidentialGene tr2aacds pipeline and/or Transfuse to identify high-quality, non-redundant transcripts from all single-kmer assemblies, and combine these to form a 'merged' or 'clustered' assembly. EvidentialGene tr2aacds.pl processes de novo assemblies with different kmer sizes (and from different assemblers), into the most biologically useful 'best' set of mRNA, classified into primary and alternate transcripts.
Similarly, Transfuse intelligently merges multiple de novo transcriptome assemblies, combining 'high-grade' transcripts into a single high quality transcriptome.

EvidentialGene tr2aacds

Download - http://arthropods.eugenes.org/genes2/about/EvidentialGene_trassembly_pipe.html

Documentation - http://arthropods.eugenes.org/EvidentialGene/evigene/

Reference - Gilbert, Donald (2013) Gene-omes built from mRNA seq not genome DNA. 7th annual arthropod genomics symposium. Notre Dame.
http://arthropods.eugenes.org/EvidentialGene/about/EvigeneRNA2013poster.pdf and
http://globalhealth.nd.edu/7th-annual-arthropod-genomics-symposium/ and
doi:10.7490/f1000research.1112594.1 / Gilbert D. (2016) Accurate & complete gene construction with EvidentialGene.
Talk at Galaxy Community Conference 2016, Bloomington IN. F1000Research, 5:1567 (slide set).

doi:10.7490/f1000research.1112467.1
http://eugenes.org/EvidentialGene/about/evigene_bothgalmod1606iu.pdf (Galaxy+GMOD full slide set)

Transfuse

Download - https://github.com/cboursnell/transfuse/releases/ (ver0.5.0)

Documentation - https://github.com/cboursnell/transfuse

Requirements - dependencies are bundled with the transfuse tarball

Reference - https://github.com/cboursnell/transfuse

cmd COMMAND (Unix - bash)

```
#Download and install EvidentialGene tr2aacds
wget ftp://arthropods.eugenes.org/evigene.tar
tar xvf evigene.tar
cd evigene/
#add executables from scripts directory and subdirectories to PATH
echo export PATH=$PATH$( find $PROGRAMDIR/evigene/scripts/ -type d -
printf ":%p" ) >> ~/.bashrc

#Concatenate all fasta assembly files
cd $WORKDIR/cleaned_trimmed_reads
mkdir all_assemblies
cd SOAPdenovo-Trans_output/SOAP_assembly_fasta_files
cp SOAP_* ../../all_assemblies; done
cd ../../velvet-oases_output
mkdir velvet-oases_fasta_files
for f in dir_*; do mv $f/transcripts.fa $f/velvet-oases_{$f}.fasta; done
for f in dir_*; do cp $f/velvet-oases* velvet-oases_fasta_files; done
cd velvet-oases_fasta_files && cp *.fasta ../../all_assemblies
cd ../Trinity-DN
cp Trinity.fasta ../all_assemblies
cd ../all_assemblies
cat *.fasta > All_assemblies.fasta
mkdir tr2aacds_merge && mv All_assemblies.fasta tr2aacds_merge/

#Rename fasta headers
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies/tr2aacds_merge/
perl -
ane 'if(/\>/){$a++;print ">Locus_$a\n"}else{print;}' All_assemblies.fasta > All_renamed.fas
ta

#Reformat fasta headers, then run the tr2aacds pipeline
trformat.pl -output All_assemblies.tr -input All_renamed.fasta
rm All_assemblies.fasta All_renamed.fasta
tr2aacds.pl -mrnaseq All_assemblies.tr -NCPU=40 1>tr2aacds.log 2>tr2aacds.err

#Download and install transfuse using Ruby
gem install transfuse
#or, alternatively, from source
```

```
wget https://github.com/cboursnell/transfuse/releases/download/v0.5.0/transfuse-0.5.0-linux
-x86_64.tar.gz -O transfuse.tar.gz
tar -zxvf transfuse.tar.gz
#add binary to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Run transfuse on the fasta assemblies to merge non-redundant and read-
supported contigs together
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies
mkdir transfuse
LIST=`ls -m *.fasta | tr -d " \t\n\r"`
transfuse -a $LIST -l R1_pairedout.fastq -r R2_pairedout.fastq -
o transfuse/transfused_assembly -t 40 1>tfuse.log 2>tfuse.err

#If transfuse fails due to insufficient MCP, symlink snap-
aligner from TransRate into the transfuse /bin/ directory (or /ruby*/gems/transrate/ direct
ory if install using Ruby) after editing the snap.rb file to increase MCP as described in S
tep 15.
```

ANNOTATIONS

Pablo García 05 Jan 2018

Hello, first of all, great work here thank you for that detailed protocol.

My question is, why we have did cdhit step and then we didn't take its ouput to do this step?

Thank you

kat coyk 24 May 2018

Hi,
Thank you so much for your detailed protocol. I am having issues with Evigenes that I hope you or someone else here can clarify. First of all, FYI, the ftp site doesn't work and the downloaded file is "evigenes18may07" not "evigene" (I mention this because this may be helpful to a newbie like me who gets easily derailed if the instructions aren't exact).
I have 4 assemblies, 2 done with Trinity and 2 with rnaSPAdes. I've concatenated them into one file.
I made it through the
#add executables from scripts directory and subdirectories to PATH
#Concatenate all files (didn't follow your protocol - just concatenated and put it in evigenes18may07 folder)
#Rename fasta headers (This file is in evigenes10may07 as well)

Now here is where I have a problem
#Reformat fasta headers
trformat.pl -output Lpe10_05ALLrenamed.tr -input Lpe10_05RenamedALL.fasta

I get an error:
"Can't use an undefined value as a symbol reference at /home/me/evigenes18may07/scripts/rnaseq/trformat.pl line 296"

Incidentally, line 296 of that script is:
print $outh $_ if($ok); # testformat got here .. bad

Unfortunately, the documentation for evigenes is challenging to follow for me. Any help would be appreciated.

Search for BUSCOs

Step 13.

Search all assemblies for the presence/absence of Benchmarking Universal Single-Copy Orthologs (BUSCOs). BUSCO provides a quantitative measure of transcriptome quality and completeness, based on evolutionarily-informed expectations of gene content from near-universal single-copy orthologs selected from OrthoDB v9.

Aligned BUSCOs are classified as 'complete', 'fragmented', or 'missing'. 'Complete' genes are further categorised as 'single' or 'duplicate', with genomes ideally containing a single copy of each gene (transcriptomes often have multiple copies of transcripts). This tool provides a genome-free/reference-free validation of transcriptome assembly quality, and allows comparison between multiple assemblies.

BUSCO:
Download - https://gitlab.com/ezlab/busco (ver1.22)
Documentation - http://busco.ezlab.org/

Requirements - species-specific genesets (from http://busco.ezlab.org/)
Reference - Simão, F.A., Waterhouse, R.M., Ioannidis, P., Kriventseva, E.V. and Zdobnov, E.M., 2015. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics, p.btv351.

cmd COMMAND (Unix - bash)

```
#For easy install use homebrew / linuxbrew (http://linuxbrew.sh/)
cd $PROGRAMDIR
brew install busco
#to download and install from source
cd $PROGRAMDIR
wget https://gitlab.com/ezlab/busco/repository/archive.tar.gz?ref=master -O BUSCO.tar.gz
tar -zxvf BUSCO.tar.gz && rm BUSCO.tar.gz && mv busco* BUSCO_v1.22
cd BUSCO_v1.22
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Download the required BUSCO catalogue (eg vertebrata and/or eukaryota) and unpack
cd $PROGRAMDIR/BUSCO_v1.22/
wget http://busco.ezlab.org/v1/files/eukaryota_buscos.tar.gz -O eukaryota.tar.gz
wget http://busco.ezlab.org/v1/files/vertebrata_buscos.tar.gz -O vertebrata.tar.gz
tar -zxvf *.tar.gz
```

```
#Run busco
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies
for f in *.fasta; do python BUSCO_v1.22.py -o busco_vert_$f -i $f -
l $PROGRAMDIR/BUSCO_v1.22/vertebrata -m tran -c 32 -
f 1>{$f}_busco_vert.log 2>{$f}_busco_vert.err; done

for f in *.fasta; do python BUSCO_v1.22.py -o busco_euk_$f -i $f -
l $PROGRAMDIR/BUSCO_v1.22/eukaryota -m tran -c 32 -
f 1>{$f}_busco_euk.log 2>{$f}_busco_euk.err; done
```

Search for CEGMAs

Step 14.

Run CEGMA on assemblies to identify the presence/absence of core eukaryotic genes. The most accurate and complete transcriptome assemblies are expected to contain a greater number of CEGMA core genes. CEGMA is considered deprecated software: it is no longer supported, and has effectively been superseded by BUSCO, however CEGMA scores may still be useful for comparing new transcriptome assemblies to previously established assemblies.

CEGMA is difficult to install; a guide can be found at
http://korflab.ucdavis.edu/datasets/cegma/ubuntu_instructions_1.txt

Additional information on compiling genewise is available at
http://seqanswers.com/forums/archive/index.php/t-24027.html

CEGMA:
Download - http://korflab.ucdavis.edu/datasets/cegma/#SCT3 (ver2.5)
Documentation - http://korflab.ucdavis.edu/datasets/cegma/README /
http://korflab.ucdavis.edu/Datasets/cegma/faq.html

Requirements - there are a number of dependancies, as detailed in the install guide.
Reference - G. Parra, K. Bradnam and I. Korf. 'CEGMA: a pipeline to accurately annotate core genes in eukaryotic genomes.'
Bioinformatics, 23: 1061-1067 (2007) / Genis Parra, Keith Bradnam, Zemin Ning, Thomas Keane, and Ian Korf. Assessing the gene space in draft genomes' Nucleic Acids Research, 37(1): 298-297 (2009)

cmd COMMAND (Unix - bash)

```
#Download and install CEGMA
cd $PROGRAMDIR
wget http://korflab.ucdavis.edu/datasets/cegma/CEGMA_v2.5.tar.gz
tar zxvf CEGMA_v2.5.tar.gz
rm CEGMA_v2.5.tar.gz && CEGMA_v2.5
make
#add executables to a directory contained in PATH, or add current directory to PATH
cd bin && echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

###Ensure all required dependencies are installed as detailed in the installation guide

#Run CEGMA on each assembly
```

```
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies
mkdir cegma_output
for f in *.fasta; do mkdir cegma_output/dir_$f && cp $f cegma_output/dir_$f; done
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies/cegma_output
export CEGMA=$PROGRAMDIR/cegma
for f in $WORKDIR/cleaned_trimmed_reads/all_assemblies/cegma_output/dir_*/*.fasta; do ~/ceg
ma/bin/cegma --genome $f -
o $WORKDIR/cleaned_trimmed_reads/all_assemblies/cegma_output/dir_*/cegma_$f -T 32; done
#If the 'for loop' fails, run CEGMA on assemblies individually
```

Run TransRate

Step 15.

TransRate is software for de-novo transcriptome quality analysis. It examines each assembly in detail and reports quality scores for contigs and assemblies, allowing you to identify the optimal assembly, filter out contigs from an assembly that are not supported by the reads, and help decide when to stop trying to improve the assembly.

TransRate:

Download - https://github.com/blahah/transrate/releases (ver1.2)

Documentation - http://hibberdlab.com/transrate/index.html

Requirements - snap-aligner (v1.0beta18), bam-read (v1.0), salmon (v0.6.0), vsearch (ver1.8.0): bundled with transrate tarball

Reference - TransRate: reference free quality assessment of de-novo transcriptome assemblies (2016). Richard D Smith-Unna, Chris Boursnell, Rob Patro, Julian M Hibberd, Steven Kelly. Genome Research doi: http://dx.doi.org/10.1101/gr.196469.115

cmd COMMAND (Unix - bash)

```
#To install using ruby
cd $PROGRAMDIR
gem install transrate
transrate --install-deps all
#to install from source
cd $PROGRAMDIR
wget https://bintray.com/artifact/download/blahah/generic/transrate-1.0.2-linux-x86_64.tar.
gz -O transrate.tar.gz
tar -zxvf transrate.tar.gz
cd transrate-1.0.2-linux-x86_64/bin
#add executables to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Prior to run, edit /transrate/lib/app/lib/transrate/snap.rb to increase MCP
#edit the function "build_paired_cmd", by finding the line "> cmd << " -
omax 10" # max alignments per pair/read" and inserting "> cmd << " -
mcp 10000000" # maximum candidate pool size" after it.

#Run TransRate
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies/
for f in *.fasta; do transrate --assembly $f --left R1_pairedout.fastq --
```

```
right R2_pairedout.fastq --output ./transrate_$f --
threads 32 1>trate_$f.log 2>trate_$f.err; done
```

Align reads to each assembly

Step 16.

Use Bowtie to align reads to each assembly. The proportion of properly aligned reads is stored in the *_flagstat.txt files. This provides a general assessment of assembly completeness: if many reads align properly (i.e. on the same contig), it suggests the assembly quality is high.

Bowtie:
Download - https://sourceforge.net/projects/bowtie-bio/files/bowtie/1.2.0/ (ver1.2)
Documentation - http://bowtie-bio.sourceforge.net/index.shtml
Reference - Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10:R25.

cmd COMMAND (Unix - bash)

```
#Bowtie download and install detailed at Step 8

#Build an alignment database for each assembly
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies
for f in *.fasta; do bowtie-build $f $f_bowtie; done

#Run bowtie and convert output from SAM format (human readable) to BAM format (binary)
for f in *.fasta; do bowtie -q --phred33-quals -n 2 -e 99999999 -l 25 -I 1 -X 1000 -p 20 -
a -m 200 --chunkmbs 128 -
S $f_bowtie -1 $WORKDIR/cleaned_trimmed_reads/R1_pairedout.fq -2 $WORKDIR/cleaned_trimmed_r
eads/R2_pairedout.fq $f.sam 1>bowtie_$f_backmapped.log 2>bowtie_$f_backmapping.err && samto
ols view -bS $f.sam > $f.bam && rm $f.sam; done

#Classify alignments as properly paired (i.e. on the same contig with a reasonable insert l
ength), aligned, or not aligned using the samtools flagstat tool
for f in *.bam; do samtools flagstat $f > $f_flagstat.txt; done
```

ANNOTATIONS

Pablo García 30 Jan 2018

Hello, I have a question related with the pipeline in terms of redundancy reduction.

I have performed these steps:

* Trinity > cdhit > output to merge in my "final redundant transcriptome"

* SOAP > gapcloser > cdhit > output to merge in my "final redundant transcriptome"

* Velvet-Oases > cdhit > output to merge in my "final redundant transcriptome"

* "final redundant transcriptome" > EvidentialGene > Transcriptome.okay.tr/cds/aa

Since I'm reducing the size of my transcriptome, and also deleting these transcripts which were considered "not good", it's normal to found worst mapping % at the end of the pipeline? In my specific case, I came from > 90% in my Trinity.fasta to 70-75% in my Transcriptome.okay.tr

What de you think about apply cd-hit after gapcloser? could be a problem?

Thank you!

Annotation

Step 17.

Idenitfy the most complete and accurate de novo transcriptome assemblies based on BUSCO/CEGMA scores, transrate scores, and properly paired read alignment, and annotate them using the Trinotate or Dammit pipelines. Annotation of transcripts aids inferrence of biological function, for instance identifying transcripts that have functional domains (Pfam domains), signaling proteins, and transmembrane proteins.

Trinotate is a suite of tools for functional annotation of de novo assembled transcriptomes. Trinotate employs sequence homology search to known transcripts (BLAST+/SwissProt), protein domain identification (HMMER/PFAM), protein signal peptide and transmembrane domain prediction (signalP/tmHMM), and leveraging various annotation databases (eggNOG/GO/Kegg databases).

Dammit performs essentially the same task: it is a simple, yet comprehensive, de novo transcriptome annotator born from the observations that annotation is time-consuming, mundane and often frustrating, with many existing solutions being overly complicated or relying on non-free software.

First, install perl module URI::Escape using cpan or cpanm (curl -L https://cpanmin.us | perl - App::cpanminus). Download the swissprot/uniprot database and prepare for blast.

Trinotate:

Download - https://github.com/Trinotate/Trinotate/releases (ver2.0.2)

Documentation - https://trinotate.github.io/

Requires - transdecoder, sqlite, ncbi-blast+, HMMER/PFAM, signalP, tmhmm, RNAmmer (further details available at https://trinotate.github.io/)

Reference - https://trinotate.github.io/

Dammit:

Download - https://github.com/camillescott/dammit

Documentation - http://dammit.readthedocs.io/en/latest/ ;
http://dammit.readthedocs.io/en/latest/installing.html

Requires - many, many dependencies, run within a virtual python environment. Further explained in the doumentation above

Reference - Camille Scott (2016) 'dammit: an open and accessible de novo transcriptome annotator', in preparation; www.camillescott.org/dammit

cmd COMMAND (Unix - bash)

```
#A guide for installing and using Trinotate is available at https://trinotate.github.io/
cd $PROGRAMDIR
cpan URI::Escape
wget https://github.com/Trinotate/Trinotate/archive/v3.0.1.tar.gz -O Trinotate_v3.0.1.tar.g
z
tar -zxvf Trinotate_v3.0.1.tar.gz && rm Trinotate_v3.0.1.tar.gz
cd Trinotate_v3.0.1/sample_data
./runMe.sh
#ensure 'runMe.sh' test completed successfully, then run:
./cleanMe.pl
cd $PROGRAMDIR/Trinotate_v3.0.1/admin/
Build_Trinotate_Boilerplate_SQLite_db.pl  Trinotate

#add executables from Trinotate directory and subdirectories to PATH
echo export PATH=$PATH$( find $PROGRAMDIR/Trinotate_v3.0.1/ -type d -
printf ":%p" ) >> ~/.bashrc

#Trinotate
#Identify ORFs using TransDecoder
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/
TransDecoder.LongOrfs -t Trinity.fasta

#Download, unpack, and build the Uniprot/Swissprot database
cd $PROGRAMDIR
mkdir databases && cd databases/
wget ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/uni
prot_sprot.fasta.gz && gunzip uniprot_sprot.fasta.gz
makeblastdb -in uniprot_sprot.fasta -dbtype prot -parse_seqids

#Blast transcripts against the database to identify potential orthologs
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/
blastx -query Trinity.fasta -db $PROGRAMDIR/databases/uniprot_sprot -evalue 1e-5 -
num_threads 40 -max_target_seqs 1 -outfmt 6 > blastx_uniprot_1e05.outfmt6
blastp -query longest_orfs.pep -db $PROGRAMDIR/databases/uniprot_sprot -evalue 1e-5 -
num_threads 40 -max_target_seqs 1 -outfmt 6 > blastp_uniprot_1e05.outfmt6

#Use an HMM search to identify PFAM domains in the longest ORFs
hmmscan --cpu 20 --domtblout TrinotatePFAM.out Pfam-A.hmm longest_orfs.pep
```

```
#Use an HMM search to identify PFAM domains in uniprot_sprot blastp hits
TransDecoder.Predict -t target_transcripts.fasta --retain_pfam_hits TrinotatePFAM.out --
retain_blastp_hits blastp_uniprot_1e05.outfmt6

#Identify signalling proteins in the longest ORFs using SignalP
signalp -f short -n signalp.out longest_orfs.pep &

#Identify rRNA sequences within the transcriptome using RNAmmer
RnammerTranscriptome.pl --transcriptome Trinity.fasta --
path_to_rnammer $PROGRAMDIR/rnammer-1.2/rnammer

#Identify transmembrane proteins using tmhmm
tmhmm --short < transdecoder.pep > tmhmm.out

#Build the boilerplate trinotate sqlite database
Trinotate Trinotate.sqlite init --gene_trans_map Trinity.fasta.gene_trans_map --
transcript_fasta Trinity.fasta --
transdecoder_pep Trinity.fasta.transdecoder_dir/longest_orfs.pep

#Load annotation results into the SQLite database
Trinotate Trinotate.sqlite LOAD_swissprot_blastp blastp_uniprot_1e05.outfmt6
Trinotate Trinotate.sqlite LOAD_swissprot_blastx blastx_uniprot_1e05.outfmt6
Trinotate Trinotate.sqlite LOAD_pfam TrinotatePFAM.out
Trinotate Trinotate.sqlite LOAD_signalp signalp.out
Trinotate Trinotate.sqlite LOAD_tmhmm tmhmm.out
Trinotate Trinotate.sqlite LOAD_rnammer Trinity.fasta.rnammer.gffTrinotate

#Generate annotation reports
Trinotate Trinotate.sqlite report > trinotate_annotation_report.xls
Trinotate Trinotate.sqlite report -E 0.00001 > trinotate_annotation_report_evalue00001.xls

#Extract GO terms from the report
extract_GO_assignments_from_Trinotate_xls.pl --
Trinotate_xls trinotate_annotation_report.xls -G --
include_ancestral_terms > go_annotations.txt

#Use trinotate to import transcript name information into the report
import_transcript_names.pl Trinotate.sqlite Trinotate_report.xls

#Identify and collate all columns containing a value for each loaded result
awk '$3 ~ !/./' trinotate_report_ex_GO.xls > trinotate_report_uniprot_hits.xls
awk '$8 ~ !/./' trinotate_report_ex_GO.xls > trinotate_report_uniprot_protein_hits.xls
awk '$10 ~ !/./' trinotate_report_ex_GO.xls > trinotate_report_pfam_hits.xls
awk '$11 ~ !/./' trinotate_report_ex_GO.xls > trinotate_report_signalling_protein_hits.xls

#Include feature names/IDs
Trinotate_get_feature_name_encoding_attributes.pl Trinotate_report.xls > Trinotate_report.x
ls.annotate_ids

#Extract GO term assignments for further analysis
extract_GO_assignments_from_Trinotate_xls.pl Trinotate_report.xls > Trinotate_GO_assignment
s.txt

#Dammit
#Download and install
cd $PROGRAMDIR
wget https://github.com/camillescott/dammit/archive/v0.3.2.tar.gz -O dammit.tar.gz
tar -zxvf dammit.tar.gz & rm dammit.tar.gz
cd dammit-0.3.2
make
```

```
sudo apt-get update
sudo apt-get install python-pip python-dev python-numpy git ruby hmmer unzip infernal ncbi-
blast+ liburi-escape-xs-perl emboss liburi-perl build-
essential libsm6 libxrender1 libfontconfig1 parallel
sudo gem install crb-blast
curl -
LO https://github.com/TransDecoder/TransDecoder/archive/2.0.1.tar.gz -O TransDecoder_2.0.1.
tar.gz
tar -xvzf TransDecoder_2.0.1.tar.gz && rm TransDecoder_2.0.1.tar.gz
cd TransDecoder-2.0.1
make
echo 'export PATH=$PATH:$PROGRAMDIR/TransDecoder-2.0.1' >> ~/.bashrc
curl -LO http://last.cbrc.jp/last-658.zip
unzip last-658.zip
cd last-658
make
echo 'export PATH=$PATH:$PROGRAMDIR/last-658/src' >> ~/.bashrc
curl -LO http://busco.ezlab.org/v1/files/BUSCO_v1.22.tar.gz
tar -xvzf BUSCO_v1.22.tar.gz
chmod +x BUSCO_v1.22/*.py
echo 'export PATH=$PROGRAMDIR/BUSCO_v1.22:$PATH' >> ~/.bashrc
sudo pip install -U setuptools
sudo pip install dammit
dammit databases --install --all --busco-group vertebrata

#to get the latest version of dammit: 'pip install git+https://github.com/camillescott/damm
it.git'

#load dependencies and virtual python environment per http://dammit.readthedocs.io/en/lates
t/installing.html then activate environment
cd $PROGRAMDIR
source activate dammit

#Run dammit
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/
dammit annotate Trinity.fasta --busco-group eukaryota --
n_threads 32 1>dammit_Trinity.log 2>dammit_Trinity.err &
```

Counting full-length transcripts

Step 18.

Take all blastx hits from annotated transcriptomes with evalue <1e-20, and use the 'analyze_blastPlus_topHit_coverage.pl' script bundled with Trinity to align assembled transcripts to corresponding entries in the Swissprot/Uniprot database. This metric provides a measure of assembly completeness: if many transcripts are near-full-length compared to the reference transcript/protein, it suggests the transcripts are well assembled. If few are near-full-length, it suggests transcripts are fragmented/poorly assembled. This metric is useful for comparing multiple assemblies produced using the same dataset.

Only the single best matching Trinity transcript is reported for each top matching database entry. If a target protein matches multiple Trinity transcripts as their best hits, that target protein is counted only once along with that Trinity transcript that provides the highest BLAST bit score and longest match length. Also compare distribution and size of contigs using bundled script 'trinity_component_distribution'.

cmd COMMAND (Unix - bash)

```
#Parse blastx output to retain transcripts with evalues <=1e-20
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/
awk ‘$11 <=1e-20’ blastx_uniprot_1e-05.outfmt6 > blastx_uniprot_1e-20.outfmt6

#Use the script bundled with Trinity to analyse blast hit coverage for each transcript
analyze_blastPlus_topHit_coverage.pl blastx.outfmt6_1e-20 Trinity.fasta $WORKDIR/databases/
uniprot_sprot

#Reformat SOAPdenovo-Trans assemblies for 'trinity_component_distribution.pl'
cd $WORKDIR/cleaned_trimmed_reads/SOAPdenovo-Trans_output/SOAP_assembly_fasta_files
for f in *.fasta; do perl -
ane 'if(/\>/){$a++;print ">c$a\n"}else{print;}' $f > tmp.fasta && sed 's/>.*/&_g1/' tmp.fas
ta > {$f}_trinity_format.fasta && rm tmp.fasta; done
for f in *_trinity_format.fasta; do trinity_component_distribution.pl $f -
o dist_$f 1>comp_distr.log 2>comp_distr.err; done

#Reformat Velvet-oases assemblies for 'trinity_component_distribution.pl'
cd $WORKDIR/cleaned_trimmed_reads/velvet-oases_output/velvet-oases_fasta_files
for f in *.fasta; do perl -
ane 'if(/\>/){$a++;print ">c$a\n"}else{print;}' $f > tmp.fasta && sed 's/>.*/&_g1/' tmp.fas
ta > {$f}_trinity_format.fasta && rm tmp.fasta; done
for f in *_trinity_format.fasta; do trinity_component_distribution.pl $f -
o dist_$f 1>comp_distr.log 2>comp_distr.err; done
```

Count full-length transcripts for all assemblies

Step 19.

Blastx is the recommended tool for aligning transcripts to the Uniprot/Swissprot database, however alignment takes a relatively long time for large transcriptome assemblies (>1 week with 40 threads). A considerably faster tool is DIAMOND ('DIAMOND blastx' is 20,000X faster than NCBI Blast+ 'blastx').

Download - https://github.com/bbuchfink/diamond/releases/ (ver0.8.36)

Documentation - https://github.com/bbuchfink/diamond

Reference - B. Buchfink, Xie C., D. Huson (2015), 'Fast and sensitive protein alignment using DIAMOND', Nature Methods 12, 59-60.

cmd COMMAND (Unix - bash)

```
#Download and install DIAMOND
cd $PROGRAMDIR
mkdir Diamond && cd Diamond
wget http://github.com/bbuchfink/diamond/releases/download/v0.8.36/diamond-linux64.tar.gz -
O diamond.tar.gz
tar -zxvf diamond.tar.gz && rm diamond.tar.gz
#add executables to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Prepare the Swissprot/Uniprot database
cd $PROGRAMDIR/databases/
```

```
diamond makedb --in uniprot_sprot.fasta -d uniprot_sprot

#run DIAMOND blastx on each assembly
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies/
for f in *.fasta; do diamond blastx -d $PROGRAMDIR/databases/uniprot_sprot -q $f -
o diamond_$f --sensitive -p 32 -k 1 -e 1e-05 -b 10 -c 1; done

#parse DIAMOND blastx output, retaining transcripts with evalues <=1e-20
mkdir blast_hits_1e-20
for f in diamond*; do awk '$11 <=1e-20' $f > blast_hits_1e-20/1e-20_$f; done

#analyse transcript length compared to the length of the reference transcript
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies
for f in *.fasta; do analyze_blastPlus_topHit_coverage.pl ./blast_hits_1e-20/1e-20_diamond_
$f $f $PROGRAMDIR/databases/uniprot_sprot; done

#count unique blastx hits (i.e. the number of reference transcripts that align to one or mo
re assembled transcripts)
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies/
mkdir number_of_unique_blastx_hits
for f in diamond_*; do cut -f2 $f | sort | uniq -c | wc -
l > ./number_of_unique_blastx_hits/count_$f; done

cd $WORKDIR/cleaned_trimmed_reads/all_assemblies/blast_hits_1e-20/
mkdir number_of_unique_blastx_hits_1e-20
for f in 1e-20_diamond_*; do cut -f2 $f | sort | uniq -c | wc -
l > ./number_of_unique_blastx_hits_1e-20/count_$f; done
```

ANNOTATIONS

Jared Mamrot 12 Mar 2017

Diamond blastx is less sensitive than NCBI BLAST+ blastx, however the significant gain in speed facilitates comparison with larger databases, such as the UniRef90 and nr.

Compare transcripts to a closely-related species

Step 20.

Use the NCBI blast+ utility to identify Mus musculus orthologs within spiny mouse transcriptome assemblies. Download the Mus RefSeq protein and RNA catalogues and convert them to blast databases. Run blastn (transcripts vs transcripts) and blastx (transcripts vs proteins), parse the output to retain hits <=1e-20, then identify which transcripts/proteins in the RefSeq catalogue correspond to one (or more) transcripts in the spiny mouse transcriptome assembly used.

Download - ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ (ver2.6.0)

Documentation - https://www.ncbi.nlm.nih.gov/books/NBK279690/

Reference - Altschul et al. (1990) Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. Journal of Molecular Biology. 1990;215(3):403–410. doi: 10.1016/S0022-2836(05)80360-2.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K. and Madden, T.L., 2009. BLAST+: architecture and applications. BMC bioinformatics, 10(1), p.421.
Madden (2002) Madden T. The BLAST sequence analysis tool. In: McEntyre J, Ostell J, editors. The NCBI

Handbook. Bethesda: National Center for Biotechnology Information; 2002. Chapter 16. https://www.ncbi.nlm.nih.gov/books/NBK153387/

cmd COMMAND (Unix - bash)

```
#Obtain, unpack, and build blast databases for Mus musculus RefSeq sequences
cd $WORKDIR/databases
ascp -i ~/.aspera/connect/etc/asperaweb_id_dsa.openssh -QT -l 100M anonftp@ftp-
private.ncbi.nlm.nih.gov:/refseq/M_musculus/mRNA_Prot/mouse.1.rna.fna.gz anonftp@ftp-
private.ncbi.nlm.nih.gov:/refseq/M_musculus/mRNA_Prot/mouse.1.protein.faa.gz .
gunzip *.gz
makeblastdb -in mouse.1.rna.fna -dbtype nucl -parse_seqids
makeblastdb -in mouse.1.protein.faa -dbtype prot -parse_seqids

#Blast transcripts against the RefSeq databases
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN
blastn -query Trinity.fasta -db $WORKDIR/databases/mouse.1.rna.fna -num_threads 32 -
max_target_seqs 1 -outfmt 6 > blastn_Trinity.fasta_vs_mus_musculus_refseq.outfmt6
blastx -query Trinity.fasta -db mouse.1.protein.faa -num_threads 32 -max_target_seqs 1 -
outfmt 6 > blastx_Trinity.fasta_vs_mus_musculus_refseq.outfmt6

#Retain hits at <=1e-20
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN
awk ‘$11 <=1e-20’ blastn_Trinity.fasta_vs_mus_musculus_refseq.outfmt6 > blastn_Trinity.fast
a_vs_mus_musculus_refseq_1e-20.outfmt6
awk ‘$11 <=1e-20’ blastx_Trinity.fasta_vs_mus_musculus_refseq.outfmt6 > blastx_Trinity.fast
a_vs_mus_musculus_refseq_1e-20.outfmt6

#Identify the number of unique RefSeq transcripts with >=1 blast hit in the transcriptome a
ssembly
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN
cut -f2 blastn_Trinity.fasta_vs_mus_musculus_refseq_1e-20.outfmt6 | sort | uniq -c | wc -
l > count_uniq_blastn_vs_refseq_1e-20_hits.txt
cut -f2 blastx_Trinity.fasta_vs_mus_musculus_refseq_1e-20.outfmt6 | sort | uniq -c | wc -
l > count_uniq_blastx_vs_refseq_1e-20_hits.txt
```

Identify non-coding RNAs

Step 21.

Use the Coding-Non-Coding Index (CNCI) to identify non-coding transcripts. CNCI profiles adjoining nucleotide triplets to distinguish protein-coding from non-coding sequences, independent of known annotations. Align transcripts to the NONCODE database for *Mus musculus* and identify unique single best blast hits.

CNCI:

Download - https://github.com/www-bioinfo-org/CNCI (version 2)

Documentation - https://github.com/www-bioinfo-org/CNCI

Reference - Liang Sun, Haitao Luo, Dechao Bu, Guoguang Zhao, Kuntao Yu, Changhai Zhang, Yuanning Liu, RunSheng Chen and Yi Zhao* Utilizing sequence intrinsic composition to classify protein-coding and long non-coding transcripts. Nucleic Acids Research (2013), doi: 10.1093/nar/gkt646

NONCODE:

Download - http://www.noncode.org/download.php (ver NONCODE2016_mouse.fa.gz)

Documentation - http://www.noncode.org/

Reference - Zhao Y, Li H, Fang S, Kang Y, Hao Y, Li Z, Bu D, Sun N, Zhang MQ, Chen R. NONCODE 2016: an informative and valuable data source of long non-coding RNAs. Nucleic acids research. (2015) Nov 19:gkv1252.

cmd COMMAND (Unix - bash)

```
#Download and install CNCI
cd $PROGRAMDIR
git clone https://github.com/www-bioinfo-org/CNCI.git
cd CNCI
unzip libsvm-3.0 && cd libsvm-3.0 && make && cd ..
chmod 755 *.py *.pl
#add scripts to a directory contained in PATH, or add current directory to PATH
echo export PATH=\$PATH:`pwd`\ >> ~/.bashrc && source ~/.bashrc

#Run the CNCI script
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/
python CNCI.py -f Trinity.fasta -o Trinity_ncRNA -p 32 -m ve

#Identify 'confident' non-coding transcripts (cutoff <= -0.05)
cd $WORKDIR/cleaned_trimmed_reads/Trinity-DN/Trinity_ncRNA/
awk '$3 <=-0.05' CNCI.index > CNCI_Trinity_ncRNA.txt

#Download the NONCODE database and prepare for blast
cd $WORKDIR/databases
wget http://www.noncode.org/datadownload/NONCODE2016_mouse.fa.gz
gunzip NONCODE2016_mouse.fa.gz
makeblastdb -in NONCODE2016_mouse.fa -dbtype nucl -parse_seqids

#Blast each assembly against the NONCODE database
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies
for f in *.fasta; do blastn -query $f -db $WORKDIR/databases/NONCODE2016_mouse.fa -
num_threads 40 -max_target_seqs 1 -evalue 1e-05 -outfmt 6 > blastn_vs_noncode_$f; done
mkdir noncode_output && mv blastn_vs_noncode* noncode_output

#Parse blast output
cd $WORKDIR/cleaned_trimmed_reads/all_assemblies/noncode_output
mkdir evalue_1e-20
for f in blastn*; do awk '$11 <=1e-20' $f > 1e-20_$f; done
mv 1e-20_* evalue_1e-20 && cd evalue_1e-20
for f in 1e-20*; do cut -f2 | sort | uniq -c | wc -l > count_best_blast_hits_$f; done
```