Comparative Metatranscriptomics WorkFlow (CoMW)

Authors: Muhammad Zohaib Anwar; *MZA* - mzanwar@envs.au.dk, Anders Lanzen; *AL*

Department of Environmental Sciences, Aarhus University, Frederiksborgvej 399, DK-4000 Roskilde, Denmark

Overall Dependencies

Third Party tools to be installed and add in working $PATH

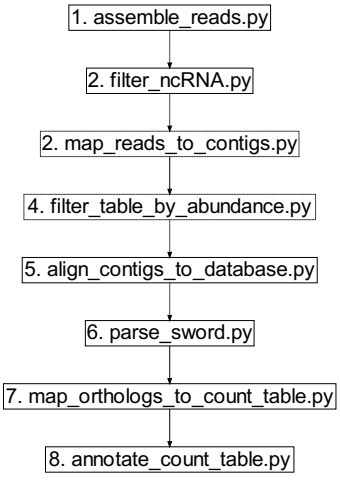
1. SWORD - https://github.com/rvaser/sword
2. Burrows-Wheeler Aligner (BWA) - https://github.com/lh3/bwa
3. EMBOSS - http://emboss.sourceforge.net/download/
4. Trinity - https://github.com/trinityrnaseq/trinityrnaseq
5. Infernal - https://github.com/EddyRivasLab/infernal

Python libraries to be installed and add in $PYTHONPATH

1. Pyfasta - https://github.com/brentp/pyfasta
2. BioPython - https://biopython.org/

Overview

The CoMW is written in python and is available with number independent and reproducible scripts that can be used based on metatranscriptomics datasets.Help (Description, input, output and parameters) are provided with each script below. CoMW is based on the results and findings from comparison of approaches, however it has two (2) optional steps such as abundance based and non-coding RNA filtering which can be different in data sets from a different environment.



1. assemble_reads.py

```
python assemble_reads.py -h
usage: assemble_reads.py [-h] [-i INPUTDIR] [-o OUTPUTDIR] [-c CPUS]
                         [-m MEMORY] [-l LIBTYPE] [-s STRANDLIBTYPE]

Author: MZA
License: GPL v3.0

Description:
Trinity is the state-of-the-art transcriptomic assembler. Trinity can be used in
independently however for a relatively straight-forward use we have wrapped it in
this script which assembles transcriptomic short reads from NGS (MiSeq, HiSeq or NextSeq)
reads to longer contigs. Short reads can be single or paired end as described in
examples below.

Dependencies:
1. Trinity https://github.com/trinityrnaseq/trinityrnaseq

Examples:
1. python assemble_reads.py -i $Fastq_dir -o $Output_dir -c 16 -m 100G -l paired -s RF
Given an input directory $Fastq_dir {$[*]R1.fastq.gz, $[*]R2.fastq.gz}, paired-end RF
orientation reads are assembled into contigs using Trinity in parallel with 16 cpus
and 100G of memory.

2. python assemble_reads.py -i $Fastq_dir -o $Output_dir -c 16 -m 100G -l single -s F
Given an input directory $Fastq_dir {$[*].fastq.gz}, sinle F orientation reads are
assembled into contigs using Trinity in parallel with 16 cpus and 100G of memory.

optional arguments:
  -h, --help            show this help message and exit
  -i INPUTDIR, --inputdir INPUTDIR
                        Fastq file directory
  -o OUTPUTDIR, --outputdir OUTPUTDIR
                        Output directory
  -c CPUS, --cpus CPUS  Number of Threads to be used
  -m MEMORY, --memory MEMORY
                        Max-memory to be used in Gb e.g 20G
  -l LIBTYPE, --libtype LIBTYPE
                        Single or Paired-end library
  -s STRANDLIBTYPE, --strandlibtype STRANDLIBTYPE
                        Strand-specific RNA-Seq read orientation if paired:
                        RF or FR, if single: F or R.
```

2. filter_ncRNA.py

```
python filter_ncRNA.py -h
usage: filter_ncRNA.py [-h] [-f FASTAFILE] [-e EVALUE] [-t THREADS]
                       [-o OUTPUTFILE] [-r REMOVE]

Authors: AL & MZA
License: GPL v3.0

Description:
This is a script that uses Infernal (a secondary-structure-aware aligner). cmsearch
module of the infernal predicts the secondary structure of RNA sequences and similarities
based on the consensus structure models of RFam. This script uses the $utils/parsecm.py
then to parse the oputput and filter the non-coding RNA contigs based on the confidence
threshold of alignment.

Dependencies:
1. $CoMW/utils/parsecm.py
2. Infernal in path http://eddylab.org/infernal/
3. Bio.Seq http://biopython.org/DIST/docs/api/Bio.Seq-module.html
from biopython http://biopython.org

Example:
1. python filter_ncRNA.py -f $contigs.fasta -e 3 -t 16 -o $contigs_ncrna_filtered.fasta -r n
Given an input fasta file with contigs the script uses infernal to align the RNA contigs
using 16 threads in parallel against RFam and filter the non-coding RNAs with a confidence
threshold of 1E-1 and write to $contigs_ncRNA_filtered.fasta

optional arguments:
  -h, --help            show this help message and exit
  -f FASTAFILE, --fastafile FASTAFILE
                        Input fasta file
  -e EVALUE, --evalue EVALUE
                        Evalue in integar
  -t THREADS, --threads THREADS
                        Number of Threads
  -o OUTPUTFILE, --outputfile OUTPUTFILE
                        Output fasta file
  -r REMOVE, --remove REMOVE
                        Delete temporary files created [y/n], default y
```

3. map_reads_to_contigs.py

```
python map_reads_to_contigs.py -h
usage: map_reads_to_contigs.py [-h] [-f FASTAFILE] [-i READSDIR]
                               [-o OUTPUTFILE] [-t THREADS] [-m MERGED]

Author: MZA
License: GPL v3.0

Description:
This script aligns quality filtered mRNA (merged or paired-end reads) against the assembled
contigs from RNA-Seq de novo transcriptome assemblers (e.g. Trinity). Given a directory with
FASTQ files merged or paired-end and a FASTA file consisting the assembled contigs, the script
aligns using BWA mapper and produces an abundance table. This script can be parallelized
using the threads option -t.

Dependencies:
1. BWA mapper http://bio-bwa.sourceforge.net/
2. $CoMW/utils/MapReads_to_contigs.sh

Example:
1. python map_reads_to_contigs.py -f $Contigs.fasta -i $Fastq_dir -o $Output_dir -t 12 -m paired
Given the paired-end fastq reads present in $Fastq_dir this script aligns against $Contigs.fasta
using 12 threads and producing the abundance table in $Output_dir

1. python map_reads_to_contigs.py -f $Contigs.fasta -i $Fastq_dir -o $Output_dir -t 12 -m single
Given the merged or single-end fastq reads present in $Fastq_dir this script aligns against
$Contigs.fasta using 12 threads and producing the abundance table in $Output_dir


optional arguments:
  -h, --help            show this help message and exit
  -f FASTAFILE, --fastafile FASTAFILE
                        Fasta file of contigs
  -i READSDIR, --readsdir READSDIR
                        Fastq file directory
  -o OUTPUTFILE, --outputfile OUTPUTFILE
                        Output file
  -t THREADS, --threads THREADS
                        Number of Threads
  -m MERGED, --merged MERGED
                        Single or Paired-end, default = paired]
```

4. filter_table_by_abundance.py

```
python filter_table_by_abundance.py -h
usage: filter_table_by_abundance.py [-h] [-i INPUTFILE] [-f FASTAFILE]
                                    [-e EXPRESSION] [-o OUTPUTPREFIX]
                                    [-r REMOVE]

Authors: AL & MZA
License: GPL v3.0

Description:
This is an optional script filters the contigs less than a given threshold of
relative expression. eg if e=1 only contigs with sum > 1/sum(Minimum Reads)
are selected. Filters out contigs from both count table
[output from map_reads_to_contigs.py]and fasta file of contigs assembled.

Example:
Given an input count table and FATSA file generates a new count table and FASTA file that
includes only contigs that have a relative expression of higher than the threshold
specified by the user.

Dependencies:
1. $CoMW/utils/Filteration.R
2. Bio.Seq http://biopython.org/DIST/docs/api/Bio.Seq-module.html
from biopython http://biopython.org

Example
python filter_table_by_abundance.py -i abundance_table.tsv -f contigs.fasta -e 1
                                    -o out_prefix -r y
filters abundance_table.tsv and contigs.fasta using expression 1% and producing
the new abundance table and contigs file with output prefix in same directory

optional arguments:
  -h, --help            show this help message and exit
  -i INPUTFILE, --inputfile INPUTFILE
                        Table file from BWA mapper output
  -f FASTAFILE, --fastafile FASTAFILE
                        Fasta file
  -e EXPRESSION, --expression EXPRESSION
                        Relative expression in integars
  -o OUTPUTPREFIX, --outputprefix OUTPUTPREFIX
                        Output prefix for filtered table and fasta file
  -r REMOVE, --remove REMOVE
                        Delete temporary files created [y/n], default y
```

5. align_contigs_to_database.py

```
python align_contigs_to_database.py -h
usage: align_contigs_to_database.py [-h] -f INPUTFASTAFILE -s SPLITSIZE -n
                                    ORFS -o OUTPUTFILE -t THREADS -d DATABASE
                                    [-r REMOVE]

Author: MZA
License: GPL v3.0

Description:
This script will use SWORD to align the assembled contigs from previous step against
database of choice from following options
1. Md5nr https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-13-141
and eggNOG annotation http://eggnogdb.embl.de/#/app/home
2. CAZy https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2686590/
3. NCyc https://www.ncbi.nlm.nih.gov/pubmed/30165481 to provide alignment results
in BM9 format using multiple threads.

Dependencies:
1. Databases in $CoMW/databases
2. SWORD aligner https://github.com/rvaser/sword
3. EMBOSS Transeq - http://emboss.sourceforge.net/download/
4. pyfasta - https://pypi.python.org/pypi/pyfasta/

Example:
1. python align_contigs_to_database.py -f $Contigs.fasta -s 12 -n 6 -o $SWORD_result.tsv
-t 12 -d 1 -r y
Given an input FASTA file $Contigs.fasta  is aligned against Md5nr using 12 threads and
6 possible ORFs generated an alignment file $[*].tsv. The input file is splitted into
12 parts after translation in order to save running memory

2. python align_contigs_to_database.py -f $Contigs.fasta -s 12 -n 1 -o $SWORD_result.tsv
-t 12 -d 2 -r y
Given an input FASTA file $Contigs.fasta  is aligned against CAZy using 12 threads and
1 possible ORFs generated an alignment file $[*].tsv. The input file is splitted into
12 parts after translation in order to save running memory

3. python align_contigs_to_database.py -f $Contigs.fasta -s 12 -n 3 -o $SWORD_result.tsv
-t 12 -d 3 -r y
Given an input FASTA file $Contigs.fasta  is aligned against NCyc using 12 threads and
3 possible ORFs generated an alignment file $[*].tsv. The input file is splitted into
12 parts after translation in order to save running memory

optional arguments:
  -h, --help            show this help message and exit
  -f INPUTFASTAFILE, --inputfastafile INPUTFASTAFILE
                        Fasta file of assembled contigs, output from Trinity
  -s SPLITSIZE, --splitsize SPLITSIZE
                        Number of parts Fasta file to be splitted in
  -n ORFS, --ORFs ORFS  Number of ORFs (1-6) to be calculated for alignment
  -o OUTPUTFILE, --outputfile OUTPUTFILE
                        Output file .tsv format
```

```
-t THREADS, --threads THREADS
                      Number of threads to be run
-d DATABASE, --database DATABASE
                      Alignment database of choice 1: Md5nr, 2: CAZy, 3: NCyc
-r REMOVE, --remove REMOVE
                      Remove temporary files [y/n]
```

6. parse_sword.py

```
parse_sword.py -h
usage: parse_sword.py [-h] [-i INPUTFILE] [-o OUTPUTFILE] [-e EVALUE]
                      [-d DATABASE]

Author: MZA
License: GPL v3.0

Description:
This script is used for parsing BM9 output file from SWORD alignement to using a specific
threshold e.g. 1E-5 against a database of choice from following
1. Md5nr https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-13-141
and eggNOG annotation http://eggnogdb.embl.de/#/app/home
2. CAZy https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2686590/
3. NCyc https://www.ncbi.nlm.nih.gov/pubmed/30165481


Dependencies:
1. Databases and annotations in $CoMW/databases

Example:
python parse_sword.py -i $[*].BM9 -e 3 -o $parsed_[*].tsv -d 2
Given an input SWord_output in BM9 $[*].BM9 this script parses BM9 file to produce a
human readable format parsed_[*].tsv and a map file against the CAZy database

optional arguments:
  -h, --help            show this help message and exit
  -i INPUTFILE, --inputfile INPUTFILE
                        SWORD output in bm9 format
  -o OUTPUTFILE, --outputfile OUTPUTFILE
                        Parsed Result file in .tsv format
  -e EVALUE, --Evalue EVALUE
                        Evalue for threshold eg: 5,6
  -d DATABASE, --database DATABASE
                        1: Md5nr, 2: CAZy, 3: NCyc
```

7. map_orthologs_to_count_table.py

```
python map_orthologs_to_count_table.py -h
usage: map_orthologs_to_count_table.py [-h] [-i INPUTFILE] [-m MAPFILE]
                                       [-o OUTPUTFILE]

Author: MZA
License: GPL v3.0

Description:
This script will map the aligned genes to the count table using the map
generated in parse_sword.py

Dependencies:
1. $CoMW/utils/AggregateTables.R

Example:
python map_orthologs_to_count_table.py -i abundance_table.tsv -m SWORD_result_eggNOG.map
                                       -o eggNOG_Counttable.tsv
Given an input abundance table abundance_table.tsv this script maps the identified genes
using the map generated in parse_sword.py

optional arguments:
  -h, --help            show this help message and exit
  -i INPUTFILE, --inputfile INPUTFILE
                        Table file from BWA mapper output
  -m MAPFILE, --mapfile MAPFILE
                        Map file from SWORD parsed output
  -o OUTPUTFILE, --outputfile OUTPUTFILE
                        Output file in tsv file
```

8. annotate_count_table.py

```
annotate_count_table.py -h
usage: annotate_count_table.py [-h] [-i INPUTFILE] [-o OUTPUTFILE] [-d DATABASE]

Author: MZA
License: GPL v3.0

Description:
This script will annotate a given countatble against the database of choice from the following
1. Md5nr https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-13-141
and eggNOG annotation http://eggnogdb.embl.de/#/app/home
2. CAZy https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2686590/
3. NCyc https://academic.oup.com/bioinformatics/10.1093/bioinformatics/bty741/5085377

Dependencies:
1. Databases and annotations in $CoMW/databases

Example:
python annotate_count_table.py -i counttable.tsv -o counttable_annotated.tsv -d 1
Given an input count table counttable.tsv is annotated using eggNOG hierarchial annotation

python annotate_count_table.py -i counttable.tsv -o counttable_annotated.tsv -d 2
Given an input count table counttable.tsv is annotated using CAZy hierarchial annotation

python annotate_count_table.py -i counttable.tsv -o counttable_annotated.tsv -d 3
Given an input count table counttable.tsv is annotated using NCyc hierarchial annotation

optional arguments:
  -h, --help            show this help message and exit
  -i INPUTFILE, --inputfile INPUTFILE
                        Table file from mapping output
  -o OUTPUTFILE, --outputfile OUTPUTFILE
                        Output file .tsv format
  -d DATABASE, --database DATABASE
                        1: Md5nr, 2: CAZy, 3: NCyc
```

This work was supported by a grant from the European Commission's Marie Sklowdowska Curie Actions program under project number 675546.