Oyster River Protocol Documentation

Release latest

May 03, 2021

Contents

The Oyster River Protocol for (eukaryotic) transcriptome assembly is an actively developed, evidenced based method for optimizing transcriptome assembly. The manuscript corresponding to this protocol is here: https://peerj.com/articles/5428/ In brief, the protocol assembles the transcriptome using a multi-kmer multi-assembler approach, then merges those assemblies into 1 final assembly.

CHAPTER 1

Contact Information

- Gitter (preferred)
- Email (good): Matthew.MacManes@unh.edu
- Twitter (good): @MacManes
- Phone (discouraged): 603-862-4052
- Office (I'm hiding under my desk): 434 Gregg Hall

Some method you'd like me to benchmark? File an issue

1.1 1. Installing the software

Docker is the preferred installation method!!!!

CHAPTER 2

docker_install

If you cannot use docker, then look at these instructions for installing manually, on Linux operating systems.

CHAPTER 3

aws_setup

3.1 2. List of dependencies

Sorry there are so many. Assembly is complex.. The makefile should take care of this.

- Rcorrector, Trimmomatic, Trinity, SPAdes, TransABySS, MCL, Metis, OrthoFuser, BLAST, seqtk, BUSCO (make sure to install databases), TransRate (the ORP version packaged here).
- Python modules numpy, scipy, biopython, cvxopt.

3.2 3. *oyster.mk* Usage

After activating the *orp* conda environment. this command will run the entire ORP in one shot! You can add the `--dry-run` flag to the end to see the individual commands that it will run, if you are curious. The *STRAND=RF* allows for strand specific assembly in version 2.1.0 of the ORP.

You must use the full PATH to the oyster.mk script for it to work

```
source activate orp

/path/to/Oyster_River_Protocol/oyster.mk \
TPM_FILT=1 \
STRAND=RF \
MEM=150 \
CPU=24 \
READ1=SRR2016923_1.fastq \
READ2=SRR2016923_2.fastq \
RUNOUT=SRR2016923
```

3.3 4. *strandeval.mk* Usage

After activating the *orp_v2* conda environment. this command will run the evaluate the strandedeness of your assembly in ORP version 2.1.0. It should help you understand if you have assembled the reads using the proper flags. You can add the `--dry-run` flag to the end to see the individual commands that it will run, if you are curious. The evaluation script was modified from a similar script in the Trinity distribution (https://github.com/trinityrnaseq/trinityrnaseq/wiki/Examine-Strand-Specificity).

See strandexamine for some help in interpreting the results.

You must use the full PATH to the strandeval.mk script for it to work

```
source activate orp

/path/to/Oyster_River_Protocol/strandeval.mk main \
ASSEMBLY=assembly.fasta \
CPU=24 \
READ1=SRR2016923_1.fastq \
READ2=SRR2016923_2.fastq \
RUNOUT=SRR2016923
```

3.4 5. *report.mk* Usage

After activating the *orp* conda environment. this command will generate a transcriptome assembly report, in ORP version 2.1.0. You can add the `--dry-run` flag to the end to see the individual commands that it will run, if you are curious. It can be run on an assembly generated by any method.

** The *LINEAGE=* flag must be specified, and the database you specify must be in */path/to/Oyster_River_Protocol/busco_dbs*. The Eukaryotic database is there by default.

```
source activate orp

/path/to/Oyster_River_Protocol/report.mk main \
ASSEMBLY=assembly.fasta \
CPU=24 \
LINEAGE=eukaryota_odb9
READ1=SRR2016923_1.fastq \
READ2=SRR2016923_2.fastq \
RUNOUT=SRR2016923
```

3.5 6. Changelog

Version 2.2.2

- FIXED a critical bug whereby the incorrect transcript was picked from a given orthogroup. This fix will potentially improve BUSCO scores dramatically.
- ADDED the ability to install using Docker!!!
- ADDED a flag to filter lowly expressed transcripts out of the dataset. Implement via *TPM_FILT=<float>*. The unfiltered assembly is available in the *assemblies/working/* folder. We also implement some methods to try and make sure that we don't eliminate any "real" transcripts in this process of TPM filtering.
- ADDED a check to make sure that your read files exist at the specified location.

- FIXED a bug the prevented proper BUSCO checkpointing (thanks @AdamStuckert).
- UPDATED Salmon to 0.13.1 and added the *–validateMappings* flag to the Salmon commands.
- UPDATED code such that you no longer need to specify *main* when running *oyster.mk*.

Version 2.1.1

- Updated conda environment name to *orp* rather than *orp_v2*
- Users may now specify kmer length to be used for Trinity using flag *TRINITY_KMER=INT* and for SPAdes using flags *SPADES1_KMER=INT* for the 1st SPAdes run and *SPADES2_KMER=INT* for the 2nd SPAdes run. Note the max kmer for trinity is 32, and for SPAdes it is 96. For all assemblies, the kmer length must be read_length-1 at a maximum.
- Add a check to make sure reads are of sufficient length given your selected assembly kmer length

Version 2.1.0

- Strand specific libraries are now assembled properly, this is enabled by adding the *STRAND=* flag. Both *RF* and *FR* are options, tho *RF* is the most common option.
- There is a new tool, *strandeval.mk*, which helps you evaluate the strandedness of your assembly.
- There is a new tool, *report.mk*, which generates an assembly report for you.
- There is a new tool, *quant.mk*, which facilities the quantitation procedure.
- Typing *oyster.mk help*, *report.mk help*, *strandeval.mk help* will print a help message.

Version 2.0

- The final assembly is now called *$RUNOUT.ORP.fasta*.
- Shannon has been removed, and TransABySS has been added in it's place. MANY users (and myself) have struggled with the RAM use and runtime of Shannon. TransABySS is much faster, and uses much less RAM.
- Diamond is leveraged for transcript recovery. It had been noted by some users that a few "real" transcripts were getting lost during the OrthoFuser steps.. Diamond, which is run after, recovers those.
- The use of LinuxBrew has been removed, in favor of conda. Dependencies are now managed by conda. You will need to launch the *orp_v2* conda environment before assembling.
- cd-hit-est is now run as default.