

Nov 11, 2019

Metatranscriptomic screening for genes of interest

DOI

dx.doi.org/10.17504/protocols.io.7vyhn7w

Helena Pound¹, Steven W Wilhelm²

¹University of Tennessee, Knoxville; ²The University of Tennessee, Knoxville

The Aquatic Microbial E...

Great Lakes Center for F...



Helena Pound

University of Tennessee, Knoxville

Create & collaborate more with a free account

Edit and publish protocols, collaborate in communities, share insights through comments, and track progress with run records.

Create free account

OPEN  ACCESS



DOI: <https://dx.doi.org/10.17504/protocols.io.7vyhn7w>

External link: <http://wilhelmlab.utk.edu/>

Protocol Citation: Helena Pound, Steven W Wilhelm 2019. Metatranscriptomic screening for genes of interest. **protocols.io** <https://dx.doi.org/10.17504/protocols.io.7vyhn7w>

License: This is an open access protocol distributed under the terms of the **[Creative Commons Attribution License](#)**, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited



Protocol status: Working

We use this protocol and it's working

Created: October 02, 2019

Last Modified: November 11, 2019

Protocol Integer ID: 28312

Keywords: metatranscriptome, blastx, CLC workbench, metatranscriptomic screening for gene, metatranscriptomic screening, metatranscriptomic library, assembled metatranscriptomic library, using blast, genes of interest, gene

Abstract

Find genes of interest in assembled metatranscriptomic library using blast.

Guidelines

You will need:

1. Metatranscriptomic assembly (fasta file) - recommended that you pool samples from individual experiments and assemble them together

(Called "assembly.fasta" in this protocol)

2. List of genes of interest (fasta file) - Protein sequences of genes; commonly genes of similar function with representatives from many species

(Called "reference.fasta" in this protocol)

Materials

- CLC Workbench
- command line blast
- command line diamond
- python

Troubleshooting

Establishing Candidates

- 1 Make a database from the list of genes of interest (referred to as the reference.fasta) in command line window.

Note

Protein sequences are required for phylogenetic placement in later steps.

1.1

Command

```
>makeblastdb -dbtype prot -in reference.fasta -parse_seqids  
>makeblastdb -dbtype prot -in reference.fasta -parse_seqids  
>makeblastdb -dbtype prot -in reference.fasta -parse_seqids
```

- 2 Blast assembly against reference database you just created
 - e-value can be changed to your preference
 - final.txt will produce list of contigs from assembly that had blast hits, to be referred to as "candidates"

2.1

Command

```
>blastx -query assembly.fasta -db reference.fasta -out final.txt -outfmt  
"6 qseqid sseqid pident length mismatch gapopen qstart qend sstart  
send eval evalue bitscore qframe" -e-value 1E-10
```

```
>blastx -query assembly.fasta -db reference.fasta -out final.txt -outfmt  
"6 qseqid sseqid pident length mismatch gapopen qstart qend sstart  
send eval evalue bitscore qframe" -e-value 1E-10
```

```
>blastx -query assembly.fasta -db reference.fasta -out final.txt -  
outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend  
sstart send eval evalue bitscore qframe" -e-value 1E-10
```

3 Extract only the aligned portion of candidate contigs.

Note

- Sometimes, contigs from assembly may be longer than a single gene. To accurately estimate transcript expression, we want to trim candidates to the exact length of the reference gene, by trimming it according to the start and finish locations of the blast alignment. Custom python script performs this trimming. Script relies on column order, so double-check that your .txt file contains all columns listed in 2.1. If a candidate contig hits to more than one reference sequence, the lowest start point and the highest end point from all hits will be used to capture the entire candidate sequence.

3.1 Custom python script found at https://github.com/Wilhelmlab/general-scripts/blob/master/Pound2019_Extract_aligned.py

4 **Convert** .txt file generated from step 3.1 to a .csv file

-Open .txt file in Microsoft Excel and save as .csv



5 **Import** .csv file into CLC Workbench as "sequences in .csv format"

6 Sort sequences by length and delete sequences smaller than 150 nucleic acids or personal preference

Note

This length cutoff can be changed based on personal preference.

7 **Export** sequence list as .fasta file and .csv file (named trimmed.fasta/csv or something similar).

8 Perform Diamond blastx against NCBI non-redundant database.

Note

- This step is a quality control mechanism to identify sequences that may not be of viral origin.
- Diamond is used to expedite the process (faster than traditional blast)

8.1 **Download** non-redundant database from NCBI in .fasta format

-rename nonredundant.fasta

8.2 **Make diamond database**

>diamond makedb --in non-redundant.fasta -d non-redundant

8.3 **Blast**

Command

```
>diamond blastx -d nonredundant -q trimmed.fasta -o candidates.txt -f
6 qseqid sseqid pident length mismatch gapopen qstart qend sstart
send eval evalue bitscore qframe stitle -k 1
```

```
>diamond blastx -d nonredundant -q trimmed.fasta -o candidates.txt -f
6 qseqid sseqid pident length mismatch gapopen qstart qend sstart
send eval evalue bitscore qframe stitle -k 1
```

```
>diamond blastx -d nonredundant -q trimmed.fasta -o candidates.txt -f
6 qseqid sseqid pident length mismatch gapopen qstart qend sstart
send eval evalue bitscore qframe stitle -k 1
```

- 9 **Open** candidates.txt file and trimmed.csv file in Excel. Copy sequences from trimmed.csv to a new column in candidates.txt file in excel (make sure sequence names match up).
- 10 **Delete** sequences that are not viral in origin (refer to diamond blast hit in the candidates.txt file).

Note

Be sure to delete the entire row.

- 11 **Create** a new .csv file with viral sequences and query frames from candidates.txt file
 - 3 columns: sequence name, sequence, query frame
- 12 **Sort** by query frame. Create a new .csv file for each query frame containing respective sequences

**Note**

Ex. CandidatesQF+1.csv

13 **Import** each query frame.csv file into CLC workbench as separate sequence lists

13.1 **Create** new sequence list and combine all query frame sequence lists

- rename "FinalCandidatesNucleicAcid"

14 Translate individual query frame sequence list to protein based on the query frame.

14.1 **Create** new sequence list and combine all protein sequences

- rename "FinalCandidatesProtein"

15 **Export** FinalCandidatesProtein sequence list as a .fasta file