Assessing Transcriptome Assembly

Matt Johnson

July 9, 2015

1 Introduction

Now that you have assembled a transcriptome, you are probably wondering about the sequence content. Are the sequences from the target organism, or are they contaminants? Which genes do the assembled contigs represent?

To answer these questions, you have to annotate the transcriptome, by assigning identities to each of the sequences in your data. Given that the original reason for using *de novo* sequencing was the lack of a reference genome, this seems difficult. However, in this case the sheer amount of publicly available genomic data can help you with your annotation.

In a 2013 paper, O'Neil and Emrich proposed several metrics for assessing the quality of a transcriptome that relies on aligning your genes to a reference proteome. A protein alignment may be possible even between very distantly related organisms: in the paper they align bee transcriptomes to the *Drosophila* genome, a divergence of over 300 million years. Protein sequences do not evolve as quickly as DNA sequences, and also have a larger alphabet.

In this tutorial you will learn several bioinformatics tools that will help you assess the completeness of your transcriptome through annotation.

2 NCBI BLAST

The National Center for Biotechnology Information (NCBI) hosts a number of highly useful services for bioinformatics. They host a very large database of DNA and peptide sequences known as GenBank, a collection of publicly available data from researchers all over the world. They also host the Basic Local Alignment Search Tool (BLAST), which can be used to search GenBank with your own sequence query.

To access the web portal for BLAST click the link below: `http://blast.ncbi.nlm.nih.gov/Blast.cgi`

Under **Basic BLAST**, there are several options.

blastn	Nucleotide blast: both your query and the database are DNA sequences.
blastp	Protein blast: both your query and the database are peptide sequences.
blastx	Translated search: your query (DNA) is translated and searched against peptide sequences.
tblastn	Reverse translated search: your query (proteins) is searched against a database of translated nucleotides.
tblastx	Both the query and the search database are DNA sequences, but they are translated into proteins before searching.

Open your transcriptome assembly file in a text editor and copy the first contig sequence. Choose the *nucleotide blast* from the NCBI page and paste the contig sequence into the large text box. Under the *database* setting, make sure it has "Nucleotide collection" and click the **BLAST** button at the bottom.

Your search may take several seconds to perhaps a minute. At the end you will (hopefully) be presented with several sequences of high similarity to your query (if not, choose another contig and try again). If you scroll down you will see how your query sequence aligned with sequence in the database. BLAST hits score highest if they have a high percentage identity along a long, continuous region.

The e-value refers to the probability that this hit arose due to chance. It depends on not just the quality of the hit but also the size of the database. The bit score also measures the amount of similarity between the query and the BLAST hit, and higher is better.

Exercise Try the BLAST search again, but this time with the blastx tool. How do the results differ?

3 Local BLAST

A large transcriptome assembly may contain tens of thousands of contigs. Conducting individual BLAST searches is not feasible. Although you may submit an entire fasta file containing many sequences, this search will take a very long time. It may also cause your bandwidth to NCBI to be reduced for submitting too many requests.

Conducting BLAST searches locally is also an option, using freely available command line tools. This is especially useful if you want to restrict your search to BLAST hits against a custom-made database. In our case we will be building a database of *P. patens* proteins to search.

We will be using the BLAST executables installed on the remote computer for the workshop. If you would like them for your own computer, download the BLAST executables appropriate for your system from here:
`ftp://ftp.ncbi.nlm.nih.gov/blast/executables/LATEST/`

3.1 Obtaining plant genomes from Phytozome

Phytozome is a database for plant genomes maintained by the Joint Genomes Institute at the US Department of Energy. Genomic sequence data is available for dozens of plant genomes in three categories:

- Flagship genomes: these plants are "most important to the DOE mission" and receive the most resources for functional genomics. Currently these include Poplar, Sorghum, *Brachypodium*, the algae *Chlamydomonas*, Soybean, Foxtail, and the moss *Physcomitrella*.
- Released genomes: These genomes have been fully sequenced, annotated, and published. There is no restriction on the use of the data in other studies.
- Early Release Genomes: These genomes may not be 100% complete, but some sequence and annotation is available. You may see reference to the *Fort Lauderdale Agreement*, made among genomics researchers to ensure that while data is available, the publication rights are retained by the Principal Investigators. It's okay to use the data for exploratory analysis.

To download genomic and proteomic data, go to the main website: `http://phytozome.jgi.doe.gov/pz/portal.html`

You will have to create a (free) account on their website to download data. Once that's completed, on the main menu bar select **Download** and then **Phytozome v.10**. This will bring you to the genome portal, where you will see all the Flagship and Released Genomes.

For these exercises you will want to download data from a genome that is as closely related to your transcriptome as possible. For example, if your transcriptome is a moss, you will want to download data from `Ppatens` (*Physcomitrella patens*). For *Oenothera*, `Athaliana` (*Arabidopsis thaliana*) is probably a good choice, as it is the most complete Rosid genome.

Click on the arrow next to your reference genome of choice. In addition to some text documents about data usage and assembly, there will be two sub-directories: `annotation` and `assembly`. The assembly directory will contain the full genomic sequence as a gzipped FASTA file.

Since we are working with transcriptomes in this tutorial, we only want to consider the coding part of the genome. This is in the annotation directory. There are two FASTA format files to download:

`Organism.cds.fa.gz` This file contains the *Coding Domain Sequences* (CDS) of the organism. These are DNA sequences that represent the coding portions of the genome, the part that gets turned into proteins. These sequences are *in frame*, meaning that the first position of each sequence also represents the first codon.

`Organism.protein.fa.gz` This file will contain the translated peptide sequences of all of the sequences in the CDS file. It should be roughly one-third the size of the CDS file.

Download both of these files to Treubia and uncompress them with `gunzip`.

3.2 Building a local BLAST database

A BLAST database is an index of sequences that the BLAST algorithm uses to align query sequences. Converting a FASTA file to a BLAST is relatively straightforward using the `makeblastdb` tool. The files generated by `makeblastdb` are *binary* files (not text files). They cannot be read by a standard text editor, but are used by BLAST to speed up the search.

You can then build a blast database with this command:

```
makeblastdb -in Athaliana_167_TAIR10.protein.fa -out arabidopsis_prot -dbtype prot
```

This generates three database files all with the prefix `arabidopsis_prot`.

3.3 Running a local blastx search

The local BLAST search tools are extremely thorough and sensitive. However, this means they are also very slow, even with a small local database. Enter the command `blastx -help` to see all the options for a local search. A few of the required and useful options:

query (Required) Your input file, in this case a fasta file of assembled contigs.

db	(Required) The database to search, usually one that you have created with `makeblastdb`. You do no have to enter any of the suffixes (`.pin`), just the first part of the name.
max_target_seqs	If set to 1, will return only the best hit for each search sequence. Otherwise, the output will have hundreds of hits for each sequence.
evalue	The default is 10, which is essentially any hit. A more reasonable value such as 1e-10 will only return significant hits.
num_threads	If multiple CPUs are available, will speed the search by roughly that amount.
outfmt	The default is the same output generated by the webtool. Will generate an XML file if set to 4, and a tabular output if set to 6.

These commands will search the `phyco_prot` database with the first few dozen sequences in your output file (to save time).

Terminal 1: A simple blastx

```
blastx -query mini_climacium.fasta -db /scratch/refseq/physco/physco_prot\
 -outfmt 6 -max_target_seqs 1 -evalue 1e-20 >mini_climacium_blastx.results
```

The output file might look something like this:

```
comp57785_c0_seq1 Pp1s267_89V6.1|PACid:18072090 88.03 969 79 2 2 2884 1156 2095 0.0 1462
comp57803_c0_seq1 Pp1s217_32V6.1|PACid:18045021 65.84 281 96 0 710 1552 16 296 1e-110  351
comp57844_c0_seq1 Pp1s83_168V6.1|PACid:18073480 64.40 854 279 7 3095 546 441 1273 0.0 1047
comp57860_c0_seq1 Pp1s475_12V6.1|PACid:18064056 83.83 699 111 2 2776 683 1 698 0.0 1186
comp58127_c0_seq1 Pp1s212_44V6.1|PACid:18042238 86.63 1631 208 5 5611 737 1 1627 0.0 2862
```

Twelve columns are listed for each contig with a successful hit in the database. With `outfmt 6` those columns are:

1. contig name (query)
2. *P. patens* protein name (target)
3. Percent identiy
4. Query length (length of the query sequence that matched a target sequence)
5. Mismatches between query and target
6. Gaps inserted
7. Start position of match in the query sequence
8. End position in query sequence
9. Start position in the target sequence
10. End position in the target sequence
11. E-value
12. Bit Score

Exercise: Conduct a tblastn search to find genome reference protein sequences matching transcriptome assembly nucleotide sequences. *Hints*: You will have to build a database from your assembly, and query sequences must be in fasta format.

3.4 Annotation Statistics

In the previous section, we noted that evaluating your assembly based solely on contig length statistics (N50) may not be the best approach for transcriptomes. Indeed, a recent paper by O'Neill and Emrich ("Assessing de novo transcriptome assembly metrics for consistency and utility" *BMC Genomics* 2013) raises further questions about N50. Instead, they suggest that annotation metrics may be more appropriate.

For this approach, you will need a reasonably-well annotated genome somewhat closely to your organism. "Closely related" is relative: O'Neill and Emrich used *Drosophila* and *Bombyx*, which may have shared a common ancestor over 300 million years ago. The next step is to perform BLASTX and TBlASTN searches between your assembled transcriptome and the target protein database. The metrics all quantify the level of orthology between the two sequence databases:

- The number of unique proteins found in the blastx search.
- The number of unique contigs hit by proteins in the tblastn search.
- The number of Reciprocal Best Hits (a contig and a peptide were each other's best hit in the two searches)
- Average Collapse Factor. The CF for each contig is the number of peptides with a best match to it. This is the only metric expected to decrease with better assemblies.
- Average Ortholog Hit Ratio. The OHR is the proportion of a target protein matched by a query contig.

You can view all of these statistics by using the Python script `annotation_stats.py`

```
/home/mjohnson/scripts/annotation_stats.py -b blastx.results -t tblastn.results -p reference_proteins.fasta
```

The BLAST results must be in tabular (`outfmt 6`) format.

Terminal 2: annotation_stats.py

```
[mjohnson@treubia pooled]$ /home/mjohnson/scripts/annotation_stats.py \
        -b pooled.blastx -t pooled.tblastn \
        -p ../reference_proteins.fa
```

```
Annotation statistics
---------------------
45773 contigs had a best hit in the protein database!
There were 13662 unique peptides found.
The average Ortholog Hit Ratio was 0.37

Reverse Search Statistics
---------------------------
29559 peptides had a best hit in the assembly!
There were 18376 unique contigs with hits.
There were 8875 reciprocal best hits!
The average collapse factor was 1.61
```

The statistics Ortholog Hit Ratio, Reciprocal Best Hits, and Collapse Factor are as they are defined in the O'Neil and Emrich paper:

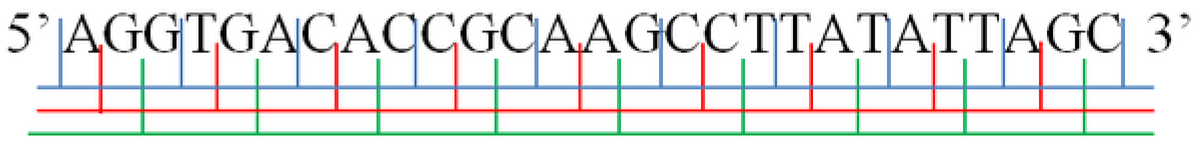
OHR	The number of bases in the matched region divided by the length of the reference sequence.
RBH	A count of transcripts that hit a protein via BLASTx, if that protein's best tBLASTn hit is also that transcript.
CF	How many transcripts matched the same protein, on average. Could be high due to incomplete assembly or whole genome duplication.

4 Transdecoder

Another helpful tool from the same folks at the Broad Institute who made Trinity helps us translate our transcriptomes into protein sequences in a smart way. In order to turn nucleotides into amino acids, we have to know the proper *reading frame*.

Consider the nucleotide sequence: `5'AGGTGACACCGCAAGCCTTATATTAGC 3'`

The codons produced by this sequence are different depending on which base we use to start the reading frame:



These three reading frames would produce different amino acid sequences (using the standard genetic code and single letters to represent amino acids):

Frame 1 `R*HRKPYIS`

Frame 2 `GDTASLIL`

Frame 3 `VTPQALY*`

And this is only half of all the choices, because the DNA could be read from right to left instead:

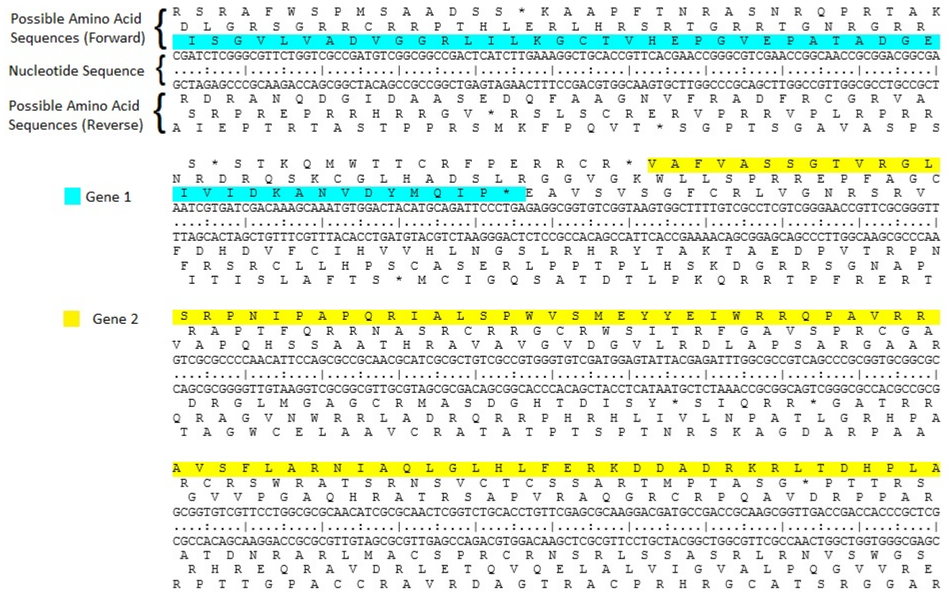
Frame 4 `ANIRLAVSP`

Frame 5 `LI*GLRCH`

Frame 6 `*YKACGVT`

The asterisks indicate the *stop codon*. Most of the time, each transcript has just one reading frame that represents a real protein. Which one should we choose?

Most programs that do translation look for the longest reading frame that does not contain a stop codon.

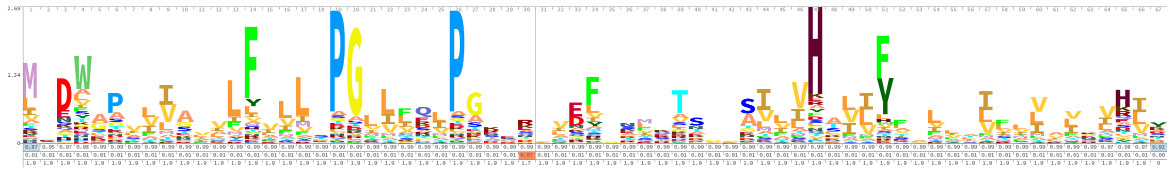

Possible Amino Acid Sequences (Forward)
Nucleotide Sequence
Possible Amino Acid Sequences (Reverse)
Gene 1
Gene 2

(images from `https://en.wikipedia.org/wiki/Reading_frame`)

Transdecoder does this six-frame translation, and then goes a step further to look for reading frames that are likely to contain a real protein sequence. To do this, it compares the initial translation *pfam*, the protein family database. Pfam (`http://pfam.xfam.org/`) contains hundreds of thousands of protein sequences from all life. The program that Transdecoder uses to search the Pfam database is called `hmmsearch` (`http://hmmer.janelia.`

org/)

Rather than compare proteins based on sequence identity (as in BLAST), hmmsearch compares the protein sequence to a Hidden Markov Model of proteins from many different organisms. One way to visualize a protein HMM is shown below.



The graphic shows the amino acids as single letters, and the size of the letter represents how commonly that amino acid is found in this protein across many different types of organisms.

When you run Transdecoder, valid proteins will have long open reading frames and will also have matches to known proteins in the pfam database. This generates much more reliable protein translations than six-frame translation alone.

Terminal 3: Running Transdecoder on Treubia

```
/opt/apps/Trinity/trinity-plugins/transdecoder/Transdecoder \
        -t Trinity.fasta
        -search_pfam /opt/apps/Trinity/trinity-plutins-transdecoder/pfam/Pfam-AB.hmm.bin\
        --CPU 12
```

The file we are using is for the `Pfam-AB` dataset. This includes HMMs built from lots of different organisms, including some that are not completely annotated functionally. This is different than the `Pfam-A` dataset that we will use for the Trinotate pipeline below.

Transdecoder will take some time to complete the hmmsearch, and at the end you will be left with several files:

mRNA	Full transcripts, including any untranslated regions.
cds	Coding domain sequences, in frame.
pep	Translated peptide sequences.
bed	Format for viewing the locations of open reading frames in a genome viewer.
gff	A format that annotates the locations of open reading frames.
pfam.dat	The results of the HMM scan.

In the peptide and CDS files, the sequences will now have IDs in the format: `c1_g1_i1|m.1`. There may be multiple proteins identified from each transcript. After a space, other information about the transcript will be listed, including whether the translation is:

complete	A full transcript from Methionine to a stop codon.
5' partial	A long reading frame that begins with Met but no stop codon.
3' partial	A long reading frame that ends with a stop codon but no obvious start.
internal	A long reading frame with no obvious start or stop codon.

5 Functional Annotation

For a complete annotation of your transcripts and the translated proteins, I recommend the Trinotate pipeline: `http://trinotate.github.io/`

The tutorial on the main website is very useful, and you should now have all of the skills necessary to generate the Trinotate report for your transcriptome.

Some notes on Trinotate:

- The input is your Trinity.fasta file and the Trinity.fasta.pep file generated by Transdecoder.
- The search against UniRef90 takes a very long time, and may not add a lot to the results, versus UniProt. I normally skip the search against UniRef90.
- Although you may have done a search with Pfam during the Transdecoder step, Trinotate requires a slightly different Pfam database (`PfamA`). You will have to rerun hmmsearch with this database.
- You need to use a clean pre-generated SQLite database for each run of Trinotate.
- Although the report will have an extension `.xls`, it is not actually a Microsoft Excel file. It is a normal tab-delimited text file.

Some notes specifically for Treubia:

- The latest version of trinotate is here: `/data/mjohnson/trinotate/Trinotate_r20140708`
- The databsases (Pfam, uniprot, and uniref90) are here: `/data/mjohnson/trinotate/dbs`
- The other software (`signalp`, `tmhmm`, `rnammer`) are here: `/usr/local/bin`