Week 4: Calling open reading frames with Prodigal, using BLAST, annotating genes with Interproscan

Rika Anderson

Abstract

Now that we've learned how to assemble our sequences, we'll next learn how to identify genes (or open reading frames) on our new assemblies. We'll do that using a software program called Prodigal. After we've identified ORFs, we will compare those ORFs to other sequences using a variety of algorithms within the BLAST suite, as well as other sequence comparison tools.

*A note: it is **extremely important** that you keep careful track of where you are putting your files, or you may lose them. This is especially the case for your projects. I recommend keeping a careful list, either in a notebook or on a document on your computer, that keeps track of where all of your important files are and what directories they are in.*

Citation: Rika Anderson Week 4: Calling open reading frames with Prodigal, using BLAST, annotating genes with Interproscan. **protocols.io**

dx.doi.org/10.17504/protocols.io.gz3bx8n

Published: 04 May 2017

Protocol

Logging in to liverpool

Step 1.

Boot as a Mac on the lab computer.

Logging in to liverpool

Step 2.

Find and open the Terminal application (Applications/Utilities).

Logging in to liverpool

Step 3.

Log on to the server using your Carleton username and password.

cmd COMMAND

```
ssh [username]@liverpool.its.carleton.edu
```

Searching for open reading frames

Step 4.

First, we're going to learn how to identify open reading frames (ORFs) on your assembled toy dataset from last week. To do that, we'll use a program called Prodigal. There are lots of programs that can identify ORFs, and there are strong distinctions between ORF callers for eukaryotes versus bacteria/archaea/viruses. Prodigal is one of the best ORF callers for microbial genomes and it is free, so that's what we're using today. If you find yourself wanting to identify ORFs on some sequence in the future, most ORF callers should work in a similar manner to Prodigal.

Go to your toy dataset directory:

cmd COMMAND

```
cd ~/toy_dataset_directory
```

Searching for open reading frames

Step 5.

We're going to try out Prodigal on your assembled dataset. However, rather than running it on your full dataset, we're going to run it on a subset of it because otherwise some downstream processes would take forever. So, copy toy_dataset_assembled_subsample.fa into your toy_assembly directory:

cmd COMMAND

```
cp /usr/local/data/toy_dataset_assembly_subsample.fa toy_assembly
```

Searching for open reading frames

Step 6.

Now that you're in the toy dataset directory, make a new directory and change into it.

cmd COMMAND

```
mkdir ORF_finding
cd ORF_finding
```

Searching for open reading frames

Step 7.

Now run Prodigal on your toy assembly subsample:

1. The "-i" flag gives the input file, which is the assembly you just made.
2. The "-o" flag gives the output file in Genbank format
3. The '-a" flag gives the output file in fasta format
4. The "-p" flag states which procedure you're using: whether this is a single genome or a metagenome. This toy dataset is a single genome so we are using –p single, but for your project dataset, you will use –p meta.

cmd COMMAND

```
prodigal -i ../toy_assembly/toy_dataset_assembly_subsample.fa -o toy_assembly_ORFs.gbk -
a toy_assembly_ORFs.faa -p single
```

Searching for open reading frames

Step 8.

You should see two files, one called "toy_assembly_ORFs.gbk" and one called "toy_assembly_ORFs.faa." Use the program less to look at toy_assembly_ORFs.faa.

cmd COMMAND

```
less toy_assembly_ORFs.faa
```

Finding the function of our open reading frames/genes/proteins

Step 9.

Now we have all of these nice open reading frames, but we don’t know what their function is. So we have to compare them to gene databases. There are lots of them out there. First, we will use BLAST and compare against the National Center for Biotechnology Information (NCBI) non-redundant database of all proteins. This is a repository where biologists are required to submit their sequence data when they publish a paper. It is very comprehensive and well-curated.

Take a look at your Prodigal file again using the program less (see above). Copy the sequence of the first ORF in your file. You can include the first line that includes the > symbol.

Finding the function of our open reading frames/genes/proteins

Step 10.

Navigate your browser to the BLAST suite at: https://blast.ncbi.nlm.nih.gov/Blast.cgi. Select Protein BLAST (blastp).

LINK:
https://blast.ncbi.nlm.nih.gov/Blast.cgi

Finding the function of our open reading frames/genes/proteins

Step 11.

Copy your sequence in to the box at the top of the page, and then scroll to the bottom and click “BLAST.” Give it a few minutes to think about it.

Finding the function of our open reading frames/genes/proteins

Step 12.

While that's running, go back to the BLAST suite home page and try blasting your protein using tblastn instead of blastp.

What’s the difference?

blastp is a protein-protein blast. When you run it online like this, you are comparing your protein sequence against the National Centers for Biotechnology Information (NCBI) non-redundant protein database, which is a giant database of protein sequences that is “non-redundant”—that is, each protein should be represented only once. In contrast, **tblastn** is a translated nucleotide blast. You are blasting your protein sequence against a translated nucleotide database. When you run it online like this, you are comparing your protein sequence against the NCBI non-redundant nucleotide database, which is a giant database of nucleotide sequnces, which can include whole genomes.

Isn't this a BLAST?!?

Yeah, ok, that wasn't funny.

Finding the function of our open reading frames/genes/proteins

Step 13.

Answer the following questions in a Word document that you will upload to the Moodle:

1. For both your blastp and tblastn results, list:
	-The top hit for each of your database searches

	-The e-value and percent identity of the top hit
	-What kind of protein it matched
	-What kind of organism it comes from
2. Take a look at the list of hits below the top hit.

	-Do they have the same function?
	-Do think you can confidently state what type of protein this gene encodes? Explain.
	-Describe the difference in your tblastn and blastp results, and explain in what scenarios you might choose to use one over the other.

Finding the function of our open reading frames/genes/proteins

Step 14.

This works well for a few proteins, but it would be exhausting to do it for every single one of the thousands of proteins you will likely have in your dataset. We are going to use software called Interproscan to more efficiently and effectively annotate your proteins. It compares your open reading frames against several protein databases and looks for protein "signatures," or regions that are highly conserved among proteins, and uses that to annotate your open reading frames. It will do this for every single open reading frame in your dataset, if it can find a match.

Interproscan takes a long time to run. With a big dataset, it could run for hours. So let's open a screen session. (Note: for your toy dataset, this will take minutes, not hours, so screen isn't entirely necessary. But it's good to practice-- and for your project datasets, you will definitely want to be in screen.)

cmd COMMAND

```
screen
```

Finding the function of our open reading frames/genes/proteins

Step 15.

Interproscan doesn’t like all of the asterisks that Prodigal adds to the ends of its proteins. So get rid of them by typing this:

Explanation: “sed” is another Unix command. It is short for “stream editor” and it is handy for editing files on occasion, like when you’re trying to do a “find/replace” operation like you might do in Word. Here, we’re using sed to replace all of the asterisks with nothing.

cmd COMMAND

```
sed 's/\*//g' toy_assembly_ORFs.faa > toy_assembly_ORFs.noasterisks.faa
```

Finding the function of our open reading frames/genes/proteins

Step 16.

Now run interproscan.

1. The “-i” flag gives the input file, which is your file with ORF sequences identified by Prodigal, with the asterisks removed.
2. The “-f” flag tells Interproscan that the format you want is a tab-separated vars, or “tsv,” file.

cmd COMMAND

```
interproscan.sh -i toy_assembly_ORFs.noasterisks.faa -f tsv
```

Finding the function of our open reading frames/genes/proteins

Step 17.

When it's done, terminate your screen session.

(You could also try Ctrl + A + k)

cmd COMMAND

```
exit
```

Finding the function of our open reading frames/genes/proteins

Step 18.

Now let's take a look at the results. The output should be called “toy_assembly_ORFs.noasterisks.faa.tsv”. The output is a text file that's most easily visualized in a spreadsheet application like Excel. The easiest way to do that is to copy this file from the server to your own desktop and look at it with Excel. We're going to learn how to use **FileZilla** to do that.

Locate FileZilla in the Applications folder on your computer and open it. Enter the following:

1. For the Host: sftp://liverpool.its.carleton.edu
2. For the Username: your Carleton username
3. For the password: your Carleton password
4. For the Port: 22
5. Then click QuickConnect.

The left side shows files in the local computer you're using. The right side shows files on liverpool. On the left side, navigate to whatever location you want your files to go to on the local computer. On the right side, navigate to the directory where you put your Interproscan results. (/Accounts/yourusername/toy_dataset_directory/ORF_finding/toy_assembly_ORFs.noasterisks.faa.tsv) All you have to do is double-click on the file you want to transfer, and over it goes!

Finding the function of our open reading frames/genes/proteins

Step 19.

Find your file on your local computer, and open your .tsv file in Excel. The column headers are in columns as follows, left to right:

1. Protein Accession (e.g. P51587)
2. Sequence MD5 digest (e.g. 14086411a2cdf1c4cba63020e1622579)
3. Sequence Length (e.g. 3418)
4. Analysis (e.g. database name-- Pfam / PRINTS / Gene3D)
5. Signature Accession Number (e.g. PF09103 / G3DSA:2.40.50.140)
6. Signature Description (e.g. BRCA2 repeat profile)
7. Start location
8. Stop location
9. Score - is the e-value of the match reported by member database method (e.g. 3.1E-52)
10. Status - is the status of the match (T: true)
11. Date - is the date of the run

Finding the function of our open reading frames/genes/proteins

Step 20.

Answer the following question on the Word document you will submit to the Moodle:

4. Take a look at the annotation of the contig you BLASTed earlier. If there is more than one hit, that's because Interproscan is returning hits from multiple protein databases. What is the description of the hit with the best score, according to Interproscan? (Remember, Google can be a handy resource sometimes.) Does that match your BLAST results?

Searching for matches within your own database

Step 21.

We just learned how to compare **all** of the proteins within your dataset against publicly available databases. This will be handy later on, because it gives you a nice list of all of the proteins in your assembly. But you may also wish to choose a specific protein of interest and see if you can find a match within your dataset. Why? There are a few reasons why you might do this. For example, you may have a gene you're interested in that's not available in public databases, or is relatively unknown. It won't show up in the Interproscan results. Or maybe you want to know of any of the ORFs or contigs in your dataset have a match to a particular sequence, even if it isn't the top hit that Interproscan found.

We will learn how to use BLAST to search for a gene, using your own dataset as the databse to search against.

First, make sure you are in the correct directory.

cmd COMMAND

```
pwd
/Accounts/[your username]/toy_dataset_directory/ORF_finding
```

Searching for matches within your own database

Step 22.

Turn your ORF dataset into a BLAST database. Here is what the flags mean:

1. makeblastdb invokes the command to make a BLAST database from your data
2. -in defines the file that you wish to turn into a BLAST database
3. -dbtype: choices are "prot" for protein and "nucl" for nucleotide.

cmd COMMAND

```
makeblastdb -in toy_assembly_ORFs.faa -dbtype prot
```

Searching for matches within your own database

Step 23.

Now your proteins are ready to be BLASTed. Now we need a protein of interest to search for. Let's say you are interested in whether there are any close matches to CRISPR proteins in this dataset. The Pfam database is a handy place to find 'seed' sequences that represent your protein of interest. Navigate your browser to:

http://pfam.xfam.org/family/PF03787.

This takes you to a page with information about the RAMP family of proteins, the "Repair Associated Mysterious Proteins." While indeed mysterious, they turn out to be CRISPR-related proteins.

🔗 LINK:
http://pfam.xfam.org/family/PF03787

Searching for matches within your own database

Step 24.

Click on "1319 sequences" near the top.

Searching for matches within your own database

Step 25.

Under "format an alignment," select your format as "FASTA" from the drop-down menu, and select gaps as "No gaps (unaligned)" from the drop-down menu. Click "Generate."

Searching for matches within your own database

Step 26.

Use FileZilla to transfer this file to your ORF_finding directory on liverpool.

Searching for matches within your own database

Step 27.

Take a look at the Pfam file using less or using a text editing tool on the local computer. It is a FASTA file with a bunch of “seed” sequences that represent a specific protein family from different organisms. If you want to learn more about a specific sequence, you can take the first part of the fasta title (i.e. “Q2RY06_RHORT”) and go to this website: http://www.uniprot.org/uploadlists/ and paste it in the box. Then select from “UniProt KB AC/ID” to “UniProtKB” in the drop-down menu on the bottom. You will get a table with information about your protein and which organism it comes from (in this case, a type of bacterium called *Rhodospirillum*).

🔗 LINK:
http://www.uniprot.org/uploadlists/

Searching for matches within your own database

Step 28.

Now, BLAST it! There are many parameters you can set. The following command illustrates some common parameters.

1. blastp invokes the program within the blast suite that you want. (other choices are blastn, blastx, tblastn, tblastx.)
2. -query defines your blast query-- in this case, the Pfam seed sequences for the CRISPR RAMP proteins.
3. -db defines your database-- in this case, the toy assembly ORFs.
4. -evalue defines your maximum e-value, in this case 1x10^{-5}
5. -outfmt defines the output format of your blast results. option 6 is common; you can check out https://www.ncbi.nlm.nih.gov/books/NBK279675/ for other options.
6. -out defines the name of your output file. I like to title mine with the general format

'query_vs_db.blastp' or something similar.

cmd COMMAND
```
blastp -query PF03787_seed.txt -db toy_assembly_ORFs.faa -evalue 1e-05 -outfmt 6 -
out PF03787_vs_prodigal_ORFs_toy.blastp
```

Searching for matches within your own database

Step 29.

Now let's check out your blast results. Take a look at your output file using less. (For easier visualization, you can also either copy your results (if it’s a small file) and paste in Excel, or transfer the file using FileZilla and open it in Excel.)

Searching for matches within your own database

Step 30.

Each blast hit is listed in a separate line. The columns are tabulated as follows, from left to right:

1. query sequence name
2. database sequence name
3. percent identity
4. alignment length
5. number of mismatches
6. number of gaps
7. query start coordinates
8. query end coordinates
9. subject start coordinates
10. subject end coordinates
11. e-value
12. bitscore

Searching for matches within your own database

Step 31.

Take a look at your BLAST results and answer the following questions for the Word document you will submit on the Moodle.

5a. Which protein among your Pfam query sequences had the best hit?

5b. What was the percent identity?

5c. What organism does the matching Pfam protien query sequence come from?

5d. Which of your ORFs did it match?

5e. Does this ORF have hits to other sequences within your query file? What do you think this means?

Searching for matches within your own database

Step 32.

Let’s try this another way. Run your blast again, but this time use a lower e-value cutoff.

cmd COMMAND

```
blastp -query PF03787_seed.txt -db toy_assembly_ORFs.faa -evalue 1e-02 -outfmt 6 -
out PF03787_vs_prodigal_ORFs_toy_evalue1e02.blastp
```

Searching for matches within your own database

Step 33.

Answer the following question on the Word document you will submit to the Moodle:

6. How do these BLAST results differ from your previous BLAST? Explain why.

Searching for matches within your own database

Step 34.

Now let’s try a different gene. Pfam is a good place to find general protein sequences that are grouped by sequence structure, with representatives from many different species. Let’s say you’re interested in finding a specific sequence instead.

1. Go to https://www.ncbi.nlm.nih.gov/protein/
2. Let’s look for the DNA polymerase from *Thermus aquaticus*, the famous DNA polymerase that is used in PCR. Type “DNA polymerase Thermus aquaticus’ in the search bar at the top.
3. Click on the first hit you get. You’ll see lots of information related to that sequence.
4. Click on “FASTA” near the top. It will give you the FASTA sequence for that protein (protein rather than DNA).
5. Copy that sequence and paste it into a new file on liverpool. Here is one way to do that: use nano to create a new file (see command). Once in nano, paste your sequence into the file. Type Ctrl+O, Enter, and then Ctrl+X.

LINK:

http://www.ncbi.nlm.nih.gov/protein/

cmd COMMAND

```
nano DNA_pol_T_aquaticus.faa
```

Searching for matches within your own database

Step 35.

Now you have a sequence file called “DNA_pol_T_aquaticus.faa”. BLAST it against your toy dataset:

cmd COMMAND

```
blastp -query DNA_pol_T_aquaticus.faa -db toy_assembly_ORFs.faa -outfmt 6 -
out DNA_pol_T_aq_vs_prodigal_ORFs_toy.blastp
```

Searching for matches within your own database

Step 36.

Take a look at your blastp output using less. Notice the e-value column, second from left. Most of your e-values for this range between 2.4 (at best) and 7.7 (at worst). These e-values are TERRIBLE. This means you probably didn’t have very good hits to this protein in your dataset. (What constitutes a “good” e-value is debatable and depends on your application, but usually you want it to be at least 0.001 or lower, as a general rule of thumb.)

Searching for matches within your own database

Step 37.

Now choose your own protein and BLAST it against your dataset. **Answer the following questions on the Moodle Word document:**

7. Describe the protein you chose and how you found the sequence for that protein.

8. Show the command you executed for the blast.

9. Where there any matches? If so, which contig was the best match?

Applying these analyses to your project datasets

Step 38.

Now we’re going to apply these analyses to your own datasets for your projects.

Identify open reading frames on your project assembly using Prodigal, then determine their functions using Interproscan. Use this lab handout for reference. **Don't forget to use the 'fixed' version of your contigs** (the version that has been run throuh 'anvi-script-reformat-fasta.') Your TAs and I are here to help if you have questions or get stuck—usually the problem is a typo or you’re in the wrong directory, so check that first! **Your Interproscan runs will take several hours. Be sure to run them on screen so they can run overnight.**

****Running Inteproscan on your large project datasets will be extremely computationally intensive. Please stagger your runs so that we are not all running it at the same time: for Thursday lab, 5 people should start in lab today, and 4 people should log on tomorrow and start their runs once the others have finished.****

When you are finished, copy your assembly, your ORFs, and your Interproscan results to your group's shared folder. For example, if your Interproscan results are called 'ERR598966_assembly_ORFs.noasterisks.faa.tsv' and you are in the Arabian Sea group, copy them over like this:

cmd COMMAND

```
cp ERR598966_assembly _ORFs.noasterisks.faa.tsv /usr/local/data/Tara_datasets/Arabian_Sea
```

Applying these analyses to your project datasets

Step 39.

Compile all of the Moodle questions above into a single document. There are nine questions listed above.

Finally, develop a question about your project dataset that you can answer using BLAST or using your Interproscan results. Remember that you can search for specific genes in your own dataset, or you can use the NCBI BLAST website to compare your own contigs to the NCBI non-redundant database, or you can search through your Interproscan results. For example, your question could be something like: 'how many *different* open reading frames in my dataset have a match to photosystem II?' or 'what genes in the NCBI non-redundant database have the best match to my longest contig?' Describe your question, explain how you went about answering it, and describe your results.

Compile all of this together into a Word (or similar) document and submit via Moodle by lab next week.