Homology and Pairwise Alignment

Stephen A. Smith[1], James B. Pease[2]

[1]*Department of Ecology and Evolutionary Biology, University of Michigan;*
[2]*Department of Biology, Wake Forest University*

EMBO Computational Molecular Evolution 2018

1. How to use this tutorial

1.1. In the tutorial below, individual command lines are present in boxes using `monospaced font`.

1.2. Comments and commands are numbered to help with asking questions.

1.3. Lines in the command boxes ending in "\" indicate a single command syntax is continued the next line of text.

1.4. While you *could* copy-paste the command lines, I **strongly** encourage you to practice typing them in manually, as this will help you remember and practice terminal command technique.

2. Getting Started

2.1. Download the gzip-compressed TAR archive http://peaselab.github.io/crete2018/exercise_1.tar.gz.

2.2. Decompress the file using this command:

2.2.1.
```
tar -xzf exercise_1.tar.gz
```

2.2.2. This should create a directory called `exercise_1`.

2.3. Change directory to this folder and be sure to run all commands below within this folder.

2.3.1.
```
cd exercise_1
```

3. Working with pairwise alignments

3.1.
```
python test.py
```

3.1.1. You should see a Needleman-Wunsch pairwise alignment.

3.2. Open test.py in a plain text editor. This is a Python file that can run Needleman-Wunsch and Smith-Waterman in a verbose (but slow) way so you can experiment. The `sc_mat` is the scoring matrix. I have included EDNAFULL that is a simple DNA scoring matrix (used by NCBI) and I have included BLOSUM30,BLOSUM50,BLOSUM62,PAM50,PAM120, and PAM500 for amino acids (also used by NCBI).

3.3. Change the `g` (gap penalty) to `-2`.

3.3.1.
```
python test.py
```

3.3.2. Note not only how the alignment changes but how the matrix changes

3.4. Change the `g` (gap penalty) to `-4`

3.4.1. `python test.py`

3.4.2. Any changes?

3.5. Comment the current line (line 6) and uncomment the next line (line 7)

3.5.1. `python test.py`

3.5.2. How high does the gap penalty have to be to make it so there is only one gap?

3.6. Leave that line uncommented and uncomment the next line (8)

3.6.1. `python test.py`

3.6.2. Note the changes.

3.7. The second one is a Smith-Waterman pairwise alignment of the same set of sequences.

3.8. Make the `g = -10`

3.8.1. `python test.py`

3.8.2. What changes?

3.9. Change the gap penalty to 0

3.9.1. `python test.py`

3.9.2. Are the alignments exactly the same? Similar?

3.10. Comment those two lines and uncomment the next two (lines 9 and 10).

3.10.1. `python test.py`

3.10.2. Are these alignments similar or exactly the same?

3.10.3. What penalty is required to make these different?

3.11. Let's look at a protein alignment. Comment lines 9 and 10, and uncomment lines 12, 13 and 14.

3.11.1. `python test.py`

Browse to ftp://ftp.ncbi.nih.gov/blast/matrices/ and see where you can download more PAM and BLOSUM matrix files. There are some already provided and you can change these to see if anything makes a difference.

3.12. Now we will look at run.py. (We will examine what species these are later.)

3.12.1. `python run.py test1.fasta test2.fasta`

3.12.2. `python run.py test1.fasta test3.fasta`

3.12.3. How does this one look?

3.13. Increase the gap penalty and try again.

3.13.1. What could be the reasons that this does not appear as well aligned?

3.14. The script **revcomp.py** computes the reverse complement of a simple DNA sequence.

3.14.1. `python revcomp.py test3.fasta test3.revcomp.fasta`

3.14.2. `python run.py test1.fasta test3.revcomp.fasta`

3.15. Try this with a pairwise BLAST.

3.15.1. `blastn -query test1.fasta -subject test3.fasta`

3.15.2. `blastn -query test1.fasta -subject test3.revcomp.fasta`

3.15.3. How is blast letting you know that the first is reverse?

4. Working with SWIPE and BLAST

4.1. SWIPE is a Smith-Waterman tool for conducting pairwise Smith-Waterman alignments on a database of sequences. The first thing we have to do for SWIPE or local BLAST is construct a special database based on a set of sequences that we want to compare. We can do that with the `makeblastdb` command.

4.1.1. `makeblastdb -in rall.fasta -dbtype prot`

4.2. Now we will run SWIPE on this database with the file

4.2.1. `swipe -d rall.fasta -i r.fasta -v 1 -b 1`

If we change the b to 0 it will just list the scores and the e values. This can be helpful for scanning results. Increase the v to get more results

4.2.2. `swipe -d rall.fasta -i r.fasta -v 1 -b 0`

4.2.3. `swipe -d rall.fasta -i r.fasta -v 10 -b 0`

4.2.4. What does this sequence appear to be?

4.2.5. `swipe -d rall.fasta -i r2.fasta -v 10 -b 0`

4.2.6. What does it seem like this sequence is?

4.3. Now lets look at the actual alignments. We will set the `-b` flag to `8` to show 8 alignments.

4.3.1. `swipe -d rall.fasta -i r.fasta -v 10 -b 8`

4.3.2. Is there anything interesting at this last sequences?

4.4. Try another query with a different file.

4.4.1. `swipe -d rall.fasta -i r3.fasta -v 10 -b 8`

4.4.2. Does this look like it is represented in the database?

4.5. We will now look at BLAST with the same set of files.

4.5.1. `blastp -db rall.fasta -query r.fasta -num_alignments 0 -num_descriptions 1`

This one might look like the cool one to begin with. To compare you (or I) can rerun the SWIPE with:

4.5.2. `swipe -d rall.fasta -i r.fasta -v 1 -b 1`

4.5.3. Why might the scores be different?

4.6. Try this query again with BLAST.

4.6.1. `blastp -db rall.fasta -query r3.fasta -num_alignments 0 -num_descriptions 1`

4.6.2. When we did it with SWIPE, why was it different?

4.7. Prepare another database from a different FASTA file, and run BLASTp.

4.7.1. `makeblastdb -in r_p_3.fasta -dbtype prot`

4.7.2. `blastp -db r_p_3.fasta -query r3.fasta -num_alignments 0 -num_descriptions 1`

4.7.3. What does it seem like this sequence is?

4.8. Look at the FASTA file `r4.fasta`.

4.8.1. The other files we have looked at were amino acid sequence, what is this one?

4.9. To use this, we need to use which BLAST program?

4.9.1. `blastx -db r_p_3.fasta -query r4.fasta -num_alignments 0 -num_descriptions 1`

4.10. We can look at nucleotide blasts as well.

4.10.1. `makeblastdb -in all.unall -dbtype nucl`

4.10.2. `blastn -db all.unall -query r4.fasta -num_alignments 0 -num_descriptions 1`

4.11. We can also blast our protein sequence against the nucleotide database with tBLASTn.

4.11.1. `tblastn -db all.unall -query r3.fasta -num_alignments 0 -num_descriptions 1`

5. Working with BLAST through the web portal.

5.1. Often you will want to conduct BLAST analyses on the web before conducting or constructing pipelines that will use either BLAST or SWIPE

5.2. Go to http://blast.ncbi.nlm.nih.gov/

5.3. Open `nerd.fasta` in a plain text editor and copy its contents to the clipboard.

5.4. Click on "Nucleotide BLAST" and paste the sequence into the query sequence space. Then under the program selection, click on "somewhat similar", then click the BLAST button.

5.4.1. This is one of the more common exercises for BLAST, but it is fun . This is the sequence used by Crichton in The Lost World. What do you notice about the sequence? What is it closely related to (top hit).

5.5. Go back and run a blastx on the same sequence. Scroll to the first few alignments. What do you notice in the gaps? (If you are wondering Mark Boguski was a scientist at NCBI.)

5.6. Conduct a blastn on the r4 sequence. Does the label match the sequence?

5.6.1. Click on the accession for the first hit and you can see all the info for the sequence. Including some analyses.

5.6.2. If you go back you can see some interesting tools.

5.6.3. At the descriptions, click select all and then graphics. This gives a nice overview of the comparison of the sequences.

5.6.4. Go back and click on "Taxonomy Reports" at the top of the page. This will give you a rundown of the hits by taxon.

5.6.5. Go to the first BLAST page and go to the bottom and click on algorithm parameters. This will expand the options for the blast searches.

5.6.6. Change the word size to `15` and match/mismatch to `1,-4`

5.6.7. If you click open results in new window you can compare things more easily. Compare this with the one that you did before. (You can rerun with default parameters if you didn't leave them open).

5.7. Go to "blastp" and in the box type:

5.7.1.
```
>test
NIRVANA
```

5.7.2. Click "Show Results" in new window and then click the BLAST button.

5.8. Now go back and click "algorithm", then check "change parameters for short input sequences".

5.8.1. Click the BLAST button again.

5.9. Now go back and click algorithm, make sure "change parameters for short input sequences" still unchecked and increase "Expect threshold;" to 100000 then click the BLAST button again. The "Expect threshold" is the number of hits you expect by random with larger being less stringent. Lower values are important for significance.

5.10. Conduct a blastp on `r3.fasta` but change the organism to *Amborella*.

5.10.1. Is the first hit significant?

5.10.2. This is another way to shrink the results but not the database.

5.11. Compare the search summary of the first to a search summary without the organism limit. The statistics are the same so the size of the database didn't change.

6. Extended Exercise 1: Fun with online BLAST

6.1. Go to blastp and try some different names that happen to overlap with the amino acid alphabet (ACDEFGHIKLMNPQRSTVWY). Note the E-values as you increase the letters.

6.1.1. You can try: DARWIN, WALLACE, KRETE, ELLENIKA, BLASTISEASY, FAKEPEPTIDE

6.1.2. or try your name (or as close as you can with the letters available).

6.1.3. Report your best matches

7. Extended Exercise 2: Exploring Significance in Nucleotide vs. Protein

7.1. We have a set of sequences that we can make a database of and then compare a nucleotide to that database and a protein to that database.

7.1.1. This is making a protein database (you can check the file in your text editor).

7.1.2.
```
makeblastdb -in acorus\_protein\_gb.fasta -dbtype prot
```

7.2. Then we are blasting a file (`r4p.fasta`) against the newly made database. This is a protein file.

7.2.1.
```
blastp -db acorus_protein_gb.fasta -query r4p.fasta \
-num_alignments 1 -num_descriptions 1
```

7.3. Then we are BLAST-ing a file (`r4.fasta`) against the newly made database. This is a nucleotide file.

7.3.1. Do you notice any differences?

7.3.2. Which do you think might be more conservative (i.e., less likely to say they are similar)?

7.3.3.
```
blastx -db acorus_protein_gb.fasta -query r4.fasta \
-num_alignments 1 -num_descriptions 1
```

This is making a nucleotide database (you can check the file in your text editor) of the same sequences as above. We are then doing the blasts on this database (protein file against nucleotide database and nucleotide against nucleotide database).

7.3.4.
```
makeblastdb -in acorus_nuc_gb.fasta -dbtype nucl
```

7.3.5.
```
tblastn -db acorus_nuc_gb.fasta -query r4p.fasta \
-num_alignments 1 -num_descriptions 1
```

7.3.6.
```
blastn -db acorus_nuc_gb.fasta -query r4.fasta \
-num_alignments 1 -num_descriptions 1
```

8. Extended Exercise 3: Database sizes

8.1. Here we have two files, a small and large.

8.1.1. View them in your text editor.

8.2. Make databases of the two and blast our files against these databases to check whether they change the significance.

8.2.1.
```
makeblastdb -in ITS.sm -dbtype nucl
```

8.2.2.
```
makeblastdb -in ITS.all -dbtype nucl
```

8.2.3.
```
blastn -db ITS.sm -query ITS.test -num_alignments 0 -num_descriptions 10
```

8.2.4.
```
blastn -db ITS.all -query ITS.test -num_alignments 0 -num_descriptions 10
```

8.2.5. The results are all significant, but you should be able to see that the significance values change.