BLAST with BioPython

You can use Python to make BLAST searches over the web using the NBCBI server, or a use a BLAST program installed on your own computer (or the CHIBI HPC server). Local BLAST allows you to search a custom database, or use a very large query. Many of the steps to set up BLAST require some Unix command line typing, but BioPython is very useful to parse large results files.

1) BioPython has a nice tool (NCBIWWW) to make BLAST queries over the web on the NCBI BLAST service. Of course, you can only search against NCBI databases.

```
from Bio import SeqIO
from Bio.Blast import NCBIWWW
my_query = SeqIO.read("test.fasta", format="fasta")
result_handle = NCBIWWW.qblast("blastn", "nt", my_query.seq)
blast_result = open("my_blast.xml", "w")
blast_result.write(result_handle.read())
blast_result.close()
result_handle.close()
```

This creates a file on your computer (in the current directory used by Python) called "my_blast.xml". XML is the most computer friendly format for BLAST output, especially if you will be using BioPython to parse the result. Give this code a try – you can use any single sequence on your computer (in FASTA format) as the "test.fasta" query.

2) Or you can run a local copy of BLAST on your own computer. This allows you to create custom local databases and run unlimited queries (limited only by your compute power)

   Install BLAST+ from NCBI on your computer
   http://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download

3) For this exercise, we will use as query, a set of transcripts from de novo assembly of RNA-seq data from an insecticide resistant housefly (*Musa domestica*):
   http://www.ncbi.nlm.nih.gov/bioproject/PRJNA170716
   http://www.ncbi.nlm.nih.gov/sites/nuccore?term=170716[BioProject]

4) You also need a set of protein sequences (in FASTA format) that will be your database. For this exercise, we will use the known UniProt proteins from *Drosophila melanogaster* (the most well studied insect genome). Data file: Dmel-UniP.fasta

5) Build a BLAST database with **`makeblastdb`**
   If you just type this command, it gives you a minimal set of "usage" information that should enable you to craft an appropriate command.

6) Create a blastx query (to compare translated RNA to protein database). Set the e-value cutoff at 1e-20 and choose the XML output format. It is often useful to limit the output to the best BLAST match for each query sequence, so you can include the following parameters: **-num_descriptions 1 -num_alignments 1**

blastx –query Musa_tx.fasta –db drospep -evalue 1e-20 –outfmt 5

7) This same query can be run using a Python function NcbiblastxCommandline in the module Bio.Blast.Applications. Tthis does not save you any effort vs the Blast command line unless you want to create a large number of queries.

```
from Bio.Blast.Applications import NcbiblastxCommandline
        blastx_cline = NcbiblastxCommandline(query="Musa_tx.fasta", db="drospep",    \
        evalue=1e-20, outfmt=5, out="my_blast.xml")
 stdout, stderr = blastx_cline()
```

8) From any BLAST search you get a BLAST output – remember to save it in XML format. Then use the Python Bio.BLAST. NCBIXML module to parse the result. The output contains one "record" for each query sequence. A BLAST record is a complex object which contains many alignments (matches), each of which contains multiple regions (hsps). The hsps have the e-value, query name, and subject (match target) names.

   For the test search (Musa_tx.fasta vs. Drosophila proteins), for matches with e-value better than 1e-20 (for any hsp segment), print the name of the query, the name of the matching protein, and the e-value. You can play with the output format to get whatever elements of the BLAST result you want to use to summarize the search.

```
from Bio.Blast import NCBIXML
E_VALUE_THRESH = 1e-20

for record in NCBIXML.parse(open("my_blast.xml")):
  if record.alignments :        #skip queries with no matches
      print "QUERY: %s" % record.query[:60]
      for align in record.alignments:
         for hsp in align.hsps:
            if hsp.expect < E_VALUE_THRESH:
               print "MATCH: %s " % align.title[:60]
               print hsp.expect
```

To hand in for this assignment, for the first **10 query sequences** with matches above the e-value threshold, send the ID and name of the query sequence, the ID and name of the matching database sequence, and the e-value of the first HSP in the match. Send to: assignments@FenyoLab.org