**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 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()

from Bio.Blast import NCBIXML

eval\_cut = 0.05

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 < eval\_cut:

print " MATCH: %s..." % align.title[:60]

... print hsp.expect

1. 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), or use BLAST that has been installed on a local computer cluster (such as the CHIBI HPC).

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

1. For this exercise, we will use as query, some of the transcripts that were assembled at CHIBI from *de novo* sequencing of the coffee borer beetle (cbb). This is an important economic pest of coffee with very little available genomic information. Data file = cbbtx.fa
2. You also need a set of protein sequences (in FASTA format) that will be your database. For this exercise, we will use the known RefSeq proteins from *Drosophila melanogaster* (the most well studied insect genome). The result will help to identify each gene from cbb by its closest match to a *Drosophila* protein.
3. 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.**

makeblastdb -in Dmel-UniP.fasta –dbtype prot –out Dmpep

1. Create a blastx query (to compare translated RNA to protein database).   
   First, make a human readable BLAST report, then repeat the search and choose the XML output format (-outfmt 5) . 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**

**blastp –query cbtx.fasta –db dmpep -num\_descriptions 1 -num\_alignments 1 -out cbtx.blast  
blastp –query cbtx.fasta –db dmpep -num\_descriptions 1 -num\_alignments 1 –outfmt 5 –out cbtx.blast.xml**

1. This same query can be run using a Python function NcbiblastxCommandline in the module Bio.Blast.Applications:

from Bio.Blast.Applications import NcbiblastpCommandline  
blastp\_cline = NcbiblastpCommandline(query="cbtx.fasta", db="Dmpep", evalue=0.01,

... outfmt=5, out="cbtx.blast.xml")

stdout, stderr = blastp\_cline()

1. From any BLAST search (browser, Python NCBIWWW, command line on your computer, or CHIBI server) you get a BLAST output file– remember to save it in XML format for best results. 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 (cbtx.fasta vs. Drosophila proteins), for matches with e-value better than e-10 (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.

For details, look at: <http://biopython.org/DIST/docs/api/Bio.Blast.Record.HSP-class.html>

from Bio.Blast import NCBIXML

E\_VALUE\_THRESH = 0.01

for record in NCBIXML.parse(open("cbtx.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

print hsp.sbjct