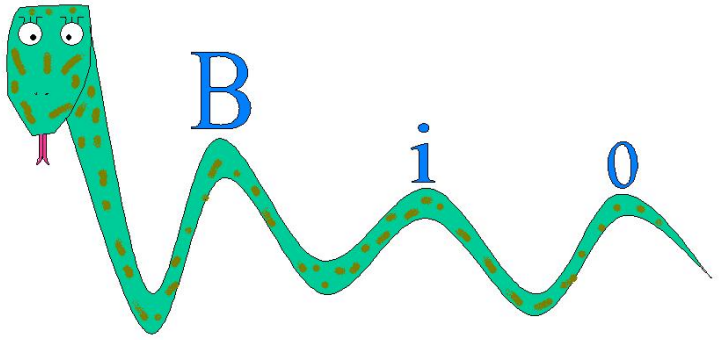
Biopython and the Lab Scientist

Brad Chapman, University of Georgia

1 August 2002

The Basics of Biopython

- International association of developers of freely available python tools for bioinformatics



- Established in 1999
- Part of Open Bioinformatics Foundation
- Active mail lists, CVS servers, and so on
- `http://www.biopython.org`

Messy page full of what Biopython can do

Parse Blast results (standalone and web); run biology related programs (blastall, clustalw, EMBOSS); deal with FASTA formatted files; parse GenBank files; parse PubMed, Medline and work with on-line resource; parse Expasy, SCOP, Rebase, UniGene, SwissProt; deal with Sequences; data classification (k Nearest Neighbors, Bayes, SVMs); Aligning sequences; CORBA interaction with Bioperl and BioJava; SQL database storage through BioSQL; Neural Networks; Genetic Algorithms; Hidden Markov Models; creating pretty PDF files for posters; format flatfiles with random access to entries; structural biology – PDB, FSSP; create specialized substitution matrices; okay, my head hurts. . .

The Basics of Python

- Interpreted, interactive, object-oriented programming language
- Designed to have a clear syntax and be easy to learn
- Scripting language: good for quick projects and scales to larger undertakings
- Has all of the bells and whistles – database access, XML parsing, graphical user interfaces . . .
- `http://www.python.org`

Example of some python code

```
def bottle(n):
    try:
        return { 0: "no more bottles",
                 1: "1 bottle"} [n] + " of beer"
    except KeyError:
        return "%d bottles of beer" % n

for i in range(99, 0, -1):
    new_bottle = bottle(i)
    bottles_left = bottle(i-1)
    print "%s on the wall, %s," \
      % (new_bottle, new_bottle)
    print "take one down, pass it around,"
    print "%s on the wall." % bottles_left
```

Our goal – Solving Problems

- Talk will focus on using Biopython to solve biological problems

- Who am I?
  - Biologist
  - Work in a large size academic lab on programming and wet-lab problems
  - Not much formal training in programming

- The random theme of the talk, and why you should feel very sorry for me.

Goals of this talk

- Show examples of how Biopython can be used in an everyday lab environment.
- Illustrate the use of Biopython to give an idea of how it looks to write python code with Biopython
- Emphasize some major concepts in Biopython code.
- Demonstrate that learning Biopython is worth your time

First problem – FASTA files and fixing a flat

- Common problem is dealing with FASTA formatted files
  - Reading FASTA files – getting sequences
  - Writing FASTA files – input into other programs

Reading and Rewriting a FASTA file

- A common task when getting files ready for input into a program.

- Good example might be extracting a subsequence in a number of proteins to align with Clustalw.

- Start with reading the file

```
from Bio import Fasta
iterator = Fasta.Iterator(open(filename), Fasta.RecordParser())
while 1:
    rec = iterator.next()
    if not(rec): break
    print rec.title
    print rec.sequence
    print rec
```

Reading a FASTA file in detail

- Set up an iterator which gives you one FASTA record at a time.

```
from Bio import Fasta
iterator = Fasta.Iterator(open(filename), Fasta.RecordParser())
```

- Use the iterator – each record returned is a python class. This is a standard Biopython theme.

```
while 1:
    rec = iterator.next()
    if not(rec): break
    print rec.title
    print rec.sequence
    print rec
```

Writing out a FASTA file

- Create a Biopython FASTA Record class.

```
rec = Fasta.Record()
rec.title = "My Sequence"
rec.sequence = "GATCGATC"
```

- The string representation of the class is FASTA.

```
>My Sequence
GATCGATC
```

```
out = open(filename, "w")
out.write(str(rec))
out.close()
```

Second Problem – BLASTs and Broken Fuel Pumps

- BLAST fuels much biology research; fast way to compare your sequence to every known sequence.

- Automating BLAST searches allows expanding your research beyond what you can do by hand. Or, it just keeps you from going crazy.

Running many local BLASTs

- BLASTing a sequenced EST against the nr database.

```
def blast_a_record(fasta_rec):
    from Bio.Blast import NCBIStandalone
    open("to_blast.fasta", "w").write(str(fasta_rec))
    out, error = NCBIStandalone.blastall(\
       "blastall", "blastn",
       "nr", "to_blast.fasta")
    parser = NCBIStandalone.BlastParser()
    rec = parser.parse(out)
    return rec.descriptions[0].title
```

Running many local BLASTs continued

- Use this function to BLAST all ESTs in a file and write out the name of the EST and the description of the first BLAST hit

```
iterator = Fasta.Iterator(open(est_file), Fasta.RecordParser())
while 1:
    rec = iterator.next()
    if not(rec): break
    first_blast_hit = blast_a_record(rec)
    print "-------"
    print rec.title
    print first_blast_hit
```

Whew. So what have we learned?

- Many tasks make use of similar functionality (reading FASTA files), so learning one module can be readily applicable to your next program.

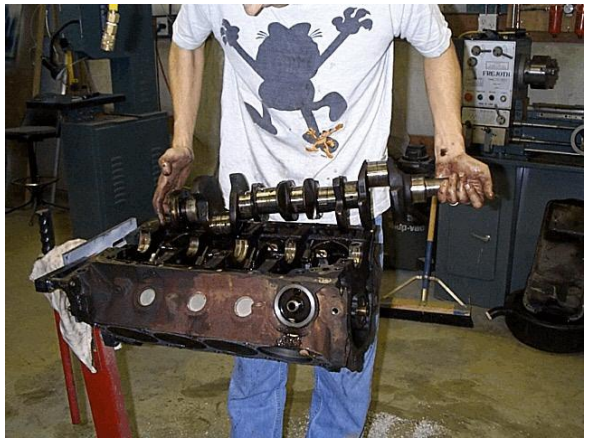
- Biopython code is modular (Fasta, Blast, . . .) so using it is a matter of stringing together modules you are involved in.

- Learning enough to write a simple program like the BLAST program can save tons of repetitive work.

- Many Biopython concepts (parsers, iterators, records) are applicable from module to module.

Final Problem – Designing primers and crankshaft rebuilding

- "To rebuild a crankshaft, you need special tools."



- Many things that come up in lab require writing specialized programs because needs are so specific.

- One task which requires tieing together parts of several programs is batch design of primers.

Description of our primer design goals

- Start with a FASTA file of sequences we want to design primers for.
- We want to design primers to span a central region in each of the FASTA records.
- Write out primer pairs to a file which can be readily loaded into Excel.

This code is simplified from some real work I did for one of my friends who was looking at designing hundreds of primers by hand. It took a little over an hour to program.

Running primer3 to design primers for a FASTA record

```
def get_primers(fasta_record, start, end):
    from Bio.Emboss.Applications import Primer3Commandline
    from Bio.Emboss.Primer import Primer3Parser
    from Bio.Application import generic_run

    open("in.pr3", "w").write(str(fasta_record) + "\n")

    primer_cl = Primer3Commandline()
    primer_cl.set_parameter("-sequence", "in.pr3")
    primer_cl.set_parameter("-outfile", "out.pr3")
    primer_cl.set_parameter("-productsizerange", "350,10000")
    primer_cl.set_parameter("-target", "%s,%s" % (start, end))
    result, r, e = generic_run(primer_cl)

    parser = Primer3Parser()
    return parser.parse(open("out.pr3"))
```

The main program to design the primers

```
def main(fasta_file, output_file):
    output_handle = open(output_file, "w")
    output_handle.write("name,forward_primer,reverse_primer\n")

    parser = Fasta.RecordParser()
    iterator = Fasta.Iterator(open(fasta_file), parser)
    while 1:
        cur_record = iterator.next()
        if not(cur_record): break
        primer_record = get_primers(cur_record, 100, 250)
        if len(primer_record.primers) > 0:
            primer = primer_record.primers[0]
            output_handle.write("%s,%s,%s\n" % (
              cur_record.title, primer.forward_seq,
              primer.reverse_seq))
        else:
            print "No primers found for %s" % cur_record.title
```

Finally – restoring your own car

- This talk showed a few of the tasks which can be more easily accomplished using the Biopython libraries.

- Biopython is also appropriate for much larger tasks and can fit well into large analysis pipelines.

- Hopefully this illustrates some of the possibilities for answering your own questions using Python and Biopython.

Acknowledgements

Thanks for Biopython go out to all of the great volunteers that have contributed to it: Jeff Chang, Andrew Dalke, Iddo Friedberg, Thomas Sicheritz-Ponten, Kate Linder, Johann Visagie, Yair Benita, Gavin E Crooks and many others