Genome 559
Intro to Statistical and Computational Genomics
2009

Lecture 16b:
Biopython

Larry Ruzzo
(Thanks again to Mary Kuhner for many slides)

1 Minute Responses

- Bioinformatics
  - Liked Harrow summary & data integration
  - More on LLR?
- Python
  - Classes finally clicking (several)
  - Repetitive today (some)
  - Liked "dir()" & other details
  - Is "return" the diff between "mutable" and not?
  - Can Python process non-.txt files? .ppt, .doc, etc?

Biopython

What is Biopython?

How do I get it to run on my computer?

What can it do?

Biopython

Biopython is tool kit, not a program–a set of Python modules useful in bioinformatics

Features include:

- Parsing files in different database formats
- Interfaces to progs/DBs like Blast, Entrez, PubMed
- Sequence class (can transcribe, translate, invert, etc)
- Code for handling alignments of sequences
- Clustering algorithms

Useful tutorials at http://biopython.org

Making Biopython available on your computer

http://biopython.org/DIST/docs/install/installation.html

1.49 is latest; works with Python 2.5 (& newer?)

Runs on Windows, MaxOSX, and Linux

Sequence class

```
>>> from Bio.Seq import Seq # sequence class
>>> myseq = Seq("AGTACACTGGT")
>>> myseq.alphabet
Alphabet()
>>> print myseq.tostring()
AGTACACTGGT
```

More functionality than a plain string

```
>>> myseq
Seq('AGTACACTGGT', Alphabet())
>>> myseq.complement()
Seq('TCATGTGACCA', Alphabet())
>>> myseq.reverse.complement()
Seq('ACCAGTGTACT', Alphabet())
```

A sequence in a specified alphabet

```
>>> from Bio.Seq inport Seq
>>> from Bio.Alphabet import IUPAC
>>> myseq=Seq('AGTACACTGGT',IUPAC.unambiguous.dna)
>>> myseq
Seq('AGTACACTGGT', IUPACUnambiguousDNA())
```

Transcribe

```
>>> from Bio import Transcribe
>>> mydna = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC",
... IUPAC.unambiguous.dna)
>>> myrna = mydna.transcribe()
>>> print myrna
Seq('GAUCGAUGGGCCUAUAUAGGAUCGAAAAUCGC',IUPACUnambiguousRNA())
>>> myprot = myrna.translate()
Seq('DRWAYIGSKI', ExtendedIUPACProtein())
>>> s2 = Seq("AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG",
... IUPAC.unambiguous_rna)
>>>s2.translate()
Seq('MAIVMGR*KGAR*', HasStopCodon(IUPACProtein(), '*'))
```

Parsing a database format

FASTA database file named "ls_orchid.fasta":

```
>gi|2765658|emb|Z78533.1|CIZ78533 C.irapeanum 5.8S rRNA gene
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAACGATCGAGTG
AATCCGGAGGACCGGTGTACTCAGCTCACCGGGGGCATTGCTCCCGTGGTGACCCTGATTTGTTGTTGGG
....
from Bio import SeqIO
handle = open("ls_orchid.fasta")
for seq.record in SeqIO.parse(handle, "fasta") :
  print seq.record.id
  print seq.record.seq
  print len(seq.record.seq)
handle.close()

gi|2765658|emb|Z78533.1|CIZ78533
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTG ...',
   SingleLetterAlphabet())
740
```

Searching GenBank

```
from Bio import GenBank
gilist = GenBank.search_for("Opuntia AND rpl16")

# gilist will be a list of all of the GenBank
# identifiers that match our query:
print gilist
['6273291', '6273290', '6273289', '6273287',
'6273286', '6273285', '6273284']
```

Searching GenBank

```
ncbidict = GenBank.NCBIDictionary("nucleotide", "genbank")
gbrecord = ncbidict[gilist[0]]
print gbrecord

LOCUS      AF191665 902 bp DNA PLN 07-NOV-1999
DEFINITION Opuntia marenae rpl16 gene; chloroplast gene for
           chloroplast product, partial intron sequence.
ACCESSION  AF191665
VERSION    AF191665.1 GI:6273291
...
```

How would I use Biopython?

Browse the documentation;become familiar with its capabilities

When doing bioinformatics, keep Biopython in mind

Prefer it to writing your own code for:

- Defining and handling sequences and alignments
- Parsing database formats
- Interfacing with databases

Biopython is not a program itself; it's a collection of tools for Python bioinformatics programs

You don't have to use it all: pick out one or two elements to learn first

Code re-use

If someone has written solid code that does what you need, use it

Don't "re-invent the wheel" unless you're doing it as a learning project

Python excels as a "glue language" which can stick together other peoples' programs, functions, classes, etc.