

Genome 559

Intro to Statistical and Computational Genomics 2009

Lecture 19b:

Biopython

Larry Ruzzo

(Thanks again to Mary Kuhner for many slides)

I Minute Responses

biopython makes me appreciate python more; but I have new questions: if python was used to make biopython what was used to make python? In other words, how do we get from bit code (0's & 1's) to a package like python?

Biopython seems really useful. Giving us ways to see how to use file was great.

thank you so much for the explanation of the homework! I feel a lot better. I still have some trepidation regarding biopython, but I feel better with the example.

the exercise were helpful to get used to Biopython. Going over the homework approaches was also good.

Biopython seems like a very cool & useful set of tools! I'm looking forward to digging through it more and hopefully using it in my work.

Nice class. Going over the homework was helpful.

I really appreciate that you're going over biopython - very useful.

Thanks for showing us how to extract features -- this will be really helpful for using biopython.

Good class today, just need more time to play with everything. It all seems a mystery until I get into the homework, then the pieces begin to come together. No success yet w/ biopython & Vista, but I will continue to play

HW notes

```
def func(x):  
    handle = open(...)  
    ...  
    return something  
    handle.close()
```

Solution I

```
from Bio import SeqIO
handle = open("ls_orchid.fasta")
for seqrec in SeqIO.parse(handle, "fasta"):
    print seqrec.id
    s = seqrec.seq
    print s
    print len(s),
    na = s.count('A')
    nc = s.count('C')
    ng = s.count('G')
    nt = s.count('T')
    print "GC%=", (ng+nc)*100.0/(na+nc+ng+nt)
handle.close()
```

Q1: there's also a Biopython func to
calc gc%; can you find it?

Q2: Why did I *not* use (G+C)/len(s)?

GenBank Format, too

```
from Bio import SeqIO
handle = open("ls_orchid.gbk")
for seq_record in SeqIO.parse(handle, "genbank") :
    print seq_record.id
    print repr(seq_record.seq)
    print len(seq_record)
handle.close()
```

This should give:

```
Z78533.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGT
GG...CGC', IUPACAmbiguousDNA())
740
...
```

Exercise 2

Change above example to save the records in a list called `seqrecs`

Solution 2

```
from Bio import SeqIO
handle = open("ls_orchid.gbk")
seqrecs = []
for seq_record in SeqIO.parse(handle, "genbank") :
    seqrecs.append(seq_record)
    print seq_record.id
    print repr(seq_record.seq)
    print len(seq_record)
handle.close()
```

This should give:

```
Z78533.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGT
GG...CGC', IUPACAmbiguousDNA())
740
...
```

And...

Feature Tables


```
>>> seqrecs[0]
SeqRecord(seq=Seq('CGTA...CGC', IUPACAmbiguousDNA()),
id='Z78533.1', name='Z78533', description='C.irapeanum
5.8S rRNA gene and ITS1 and ITS2 DNA.', dbxrefs=[])
>>> print seqrecs[0]
ID: Z78533.1
Name: Z78533
Description: C.irapeanum 5.8S rRNA gene and ITS1 and ITS2 DNA.
Number of features: 5
/sequence_version=1
/source=Cypripedium irapeanum
/taxonomy=['Eukaryota', ..., 'Cypripedium']
/keywords=['5.8S ribosomal RNA', ... 'ITS2']
/references=[<Bio.SeqFeature.Reference instance...]
/accessions=['Z78533']
/data_file_division=PLN
/date=30-NOV-2006
/organism=Cypripedium irapeanum
/gi=2765658
Seq('CGTAACAAGGTTTCCGTAGGTGA...CGC', IUPACAmbiguousDNA())
```


Extracting Features

(Lists of objects with dicts of lists of lists of dicts of ...Oh my!)

```
>>> seqrecs[0].annotations
```

```
{'sequence_version': 1, 'source': 'Cypripedium  
irapeanum', 'taxonomy': ['Eukaryota', ... ..]}
```



```
# it's a dictionary!  What keys does it have?
```


```
>>> seqrecs[0].annotations.keys()
```

```
['sequence_version', 'source', 'taxonomy', 'keywords',  
'references', 'accessions', 'data_file_division', 'date',  
'organism', 'gi']
```

```
# grab one dict entry
```

```
>>> seqrecs[0].annotations['keywords']
```

```
['5.8S ribosomal RNA', '5.8S rRNA gene', 'internal  
transcribed spacer', 'ITS1', 'ITS2']
```



```
#It's a list!  We can index into it...
```

```
>>> seqrecs[0].annotations['keywords'][1]
```

```
'5.8S rRNA gene'
```

Searching GenBank

```
# This example & next require internet access

from Bio import GenBank
gilist = GenBank.search_for("Opuntia AND rpl16")
# (that's RPL-sixteen, not RP-one-one-six)

# 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
...
```

Exercise 3: What kind of a thing is “gbrecord”? Is there other stuff hidden with it like annotations or feature tables? How do I access it?

Solution 3

```
>>> type(gbrecord)
<type 'str'>
```

```
# Aha, it's just a plain string.
```

```
>>> gbrecord
'LOCUS          AY851612                892 bp    DNA
linear    PLN 10-APR-2007\nDEFINITION  Opuntia subulata
rpl16 gene, intron; chloroplast.\nACCESSION
AY851612\nVERSION      AY851612.1    GI:
57240072\nKEYWORDS      .\nSOURCE      chloroplast
Austrocylindropuntia subulata\n    ... .. '

```

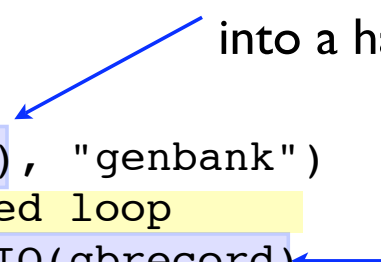
Can we get Biopython to parse it?

To parse a string

```
>>> SeqIO.parse(gbrecord, "genbank")
Traceback (most recent call last): blah blah blah...
```

```
# Oops, a string isn't a handle...
```

Turn a string
into a handle



```
>>> import cStringIO
>>> SeqIO.parse(cStringIO.StringIO(gbrecord), "genbank")
<generator object at 0x5254b8> ## Oops, need loop
>>> for rec in SeqIO.parse(cStringIO.StringIO(gbrecord),
... "genbank"):
...     print rec
...
ID: AY851612.1
Name: AY851612
Description: Opuntia subulata rpl16 gene, intron;
chloroplast.
Number of features: 3
/sequence_version=1
/source=chloroplast Austrocylindropuntia subulata ...
```

(Some) Other Capabilities

AlignO

- consensus

- PSSM (weight matrix)

BLAST

- both local and internet

Entrez EUtils

- including GenBank and PubMed

Other Databases

- SwissProt, Prosite, ExPASy, PDB

How would I use Biopython?

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

When doing bioinformatics, keep Biopython in mind

Browse the documentation; become familiar with its capabilities

Use `help()`, `type()`, `dir()` & other built-in features to explore

You might prefer it to writing your own code for:

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

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.

Exercises

Many!

As one suggestion, look at the “Cookbook” section of the tutorial. Figure out how to read my hem6.txt Phylip alignment & make a WMM from it.

Feel free to do something with one of the other pieces instead.