Biopython Tutorial and Cookbook

Je Chang, Brad Chapman, Iddo Friedberg, Thomas Hamelryck

Last Update–2 July 2006

–

Chapter 2

Quick Start – What can you do with Biopython?

The approach taken in the Biopython sequence class is to utilize a class that holds more complex information, yet can be manipulated as if it were a simple string. This is accomplished by utilizing operator

```
>>> new_seq = my_seq[0:5]
>>> print new_seq
Seq('GATCG', IUPACUnambiguousDNA())
>>> my_seq + new_seq
Seq('GATCGATGGGCCTATATAGGATCGAAAATCGCGATCG', IUPACUnambiguousDNA())
>>> my_seq[5]
'A'
>>> my_seq == new_seq
0
```

In all of the operations, the alphabet property is maintained. This is very useful in case you accidentally end up trying to do something weird like add a protein sequence and a DNA sequence:

```
>>> protein_seq = Seq('EVRNAK', IUPAC.protein)
>>> dna_seq = Seq('ACGT', IUPAC.unambiguous_dna)
```

```
>>> from Bio import Transcribe
>>> transcriber = Transcribe.unambiguous_transcriber
>>> my_rna_seq = transcriber.transcribe(my_seq)
>>> print my_rna_seq
Seq('GAUCGAUGGGCCUAUAUAGGAUCGAAAAUCGC', IUPACUnambiguousRNA())
```

The alphabet of the new RNA Seq object is created for free, so again, dealing with a Seq object is no more difficult then dealing with a simple string.

You can also reverse transcribe RNA sequences:

```
>>> transcriber.back_transcribe(my_rna_seq)
Seq('GATCGATGGGCCTATATAGGATCGAAAATCGC', IUPACUnambiguousDNA())
```

2.3 A usage example

Before we jump right into parsers and everything else to do with Biopython, let's set up an example to

2.4.3 Making it easier

2.5 Connecting with biological databases

One of the very common things that you need to do in bioinformatics is extract information from biological databases. It can be quite tedious to access these databases manually, especially if you have a lot of repetitive work to do. Biopython attempts to save you time and energy by making some on-line databases available from python scripts. Currently, Biopython has code to extract information from the following databases:

- ExPASy – http://www.expasy.org/ See section

Chapter 3

Cookbook – Cool things to do with it

3.1 BLAST

cgi?Jform=0

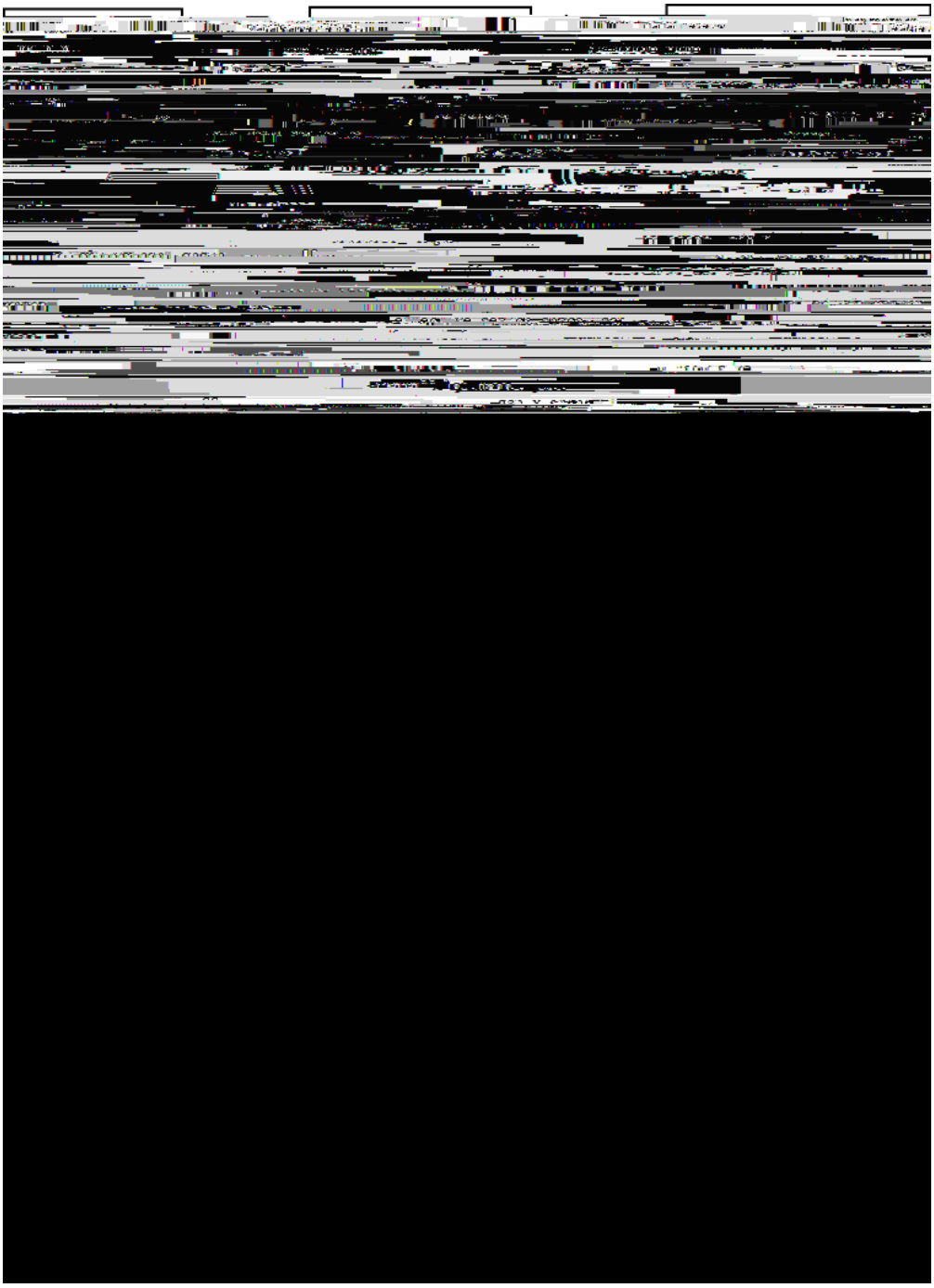

Figure 3.1: Class diagram for the Blast Record class representing all of the info in a BLAST report

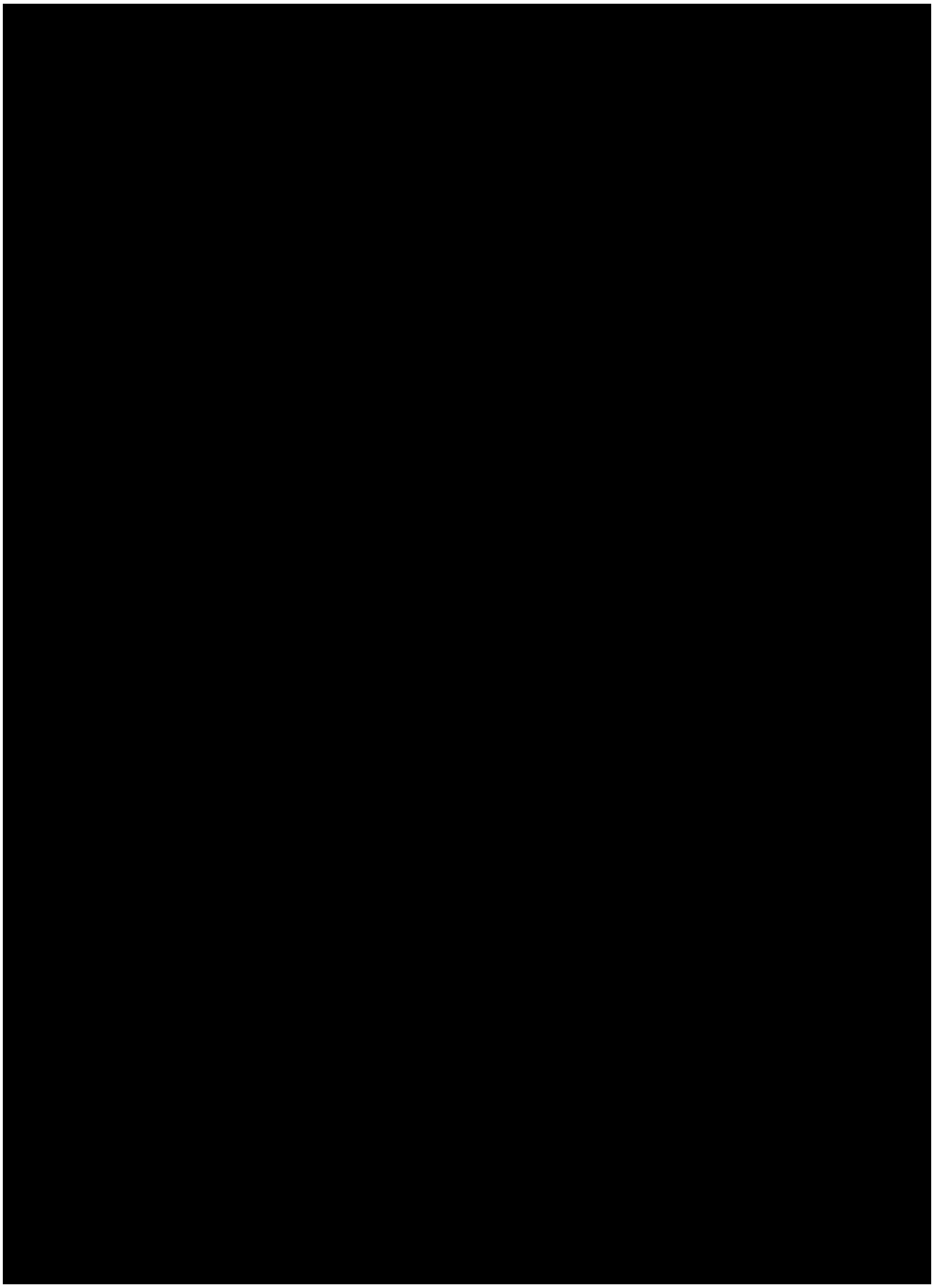

Figure 3.2: Class diagram for the PSIBlast Record class.

3.1.5 Parsing BLAST output from local BLAST

```
import os

b_file = os.path.join(os.getcwd(), 'blast_out', 'big_blast.out')
error_file = os.path.join(os.getcwd(), 'blast_out', 'big_blast.problems')
```

Now we want to get a `BlastErrorParser`:

```
from Bio.Blast import NCBIStandalone

error_handle = open(error_file, 'w')

b_error_parser = NCBIStandalone.BlastErrorParser(error_handle)
```

Notice that the parser take an optional argument of a handle. If a handle is passed, then the parser will write any blast records which generate a SyntaxError to this handle. Otherwise, these records will not be recorded.

Now we can use the `BlastErrorParser`

3.2 SWISS-PROT

3.2.1 Retrieving a SWISS-PROT record

SwissProt (http://www.expasy.org/sprot/sprot-top.html) is a hand-curated database of protein se-

3.4.3 Iterating over GenBank records

```
>>> len(gb_dict)
6
>>> gb_dict.keys()
['L31939', 'AJ237582', 'X62281', 'AF297471', 'M81224', 'X55053']
```

```
from Bio import Clustalw

alignment = Clustalw.do_alignment(cline)
```

3.6.1 Using common substitution matrices

3.6.2 Creating your own substitution matrix from an alignment

A very cool thing that you can do easily with the substitution matrix classes is to create your own substitution

- `factor` - The factor to multiply each matrix entry by. This defaults to 10, which normally makes the matrix numbers easy to work with.
- `round_digit` - The digit to round to in the matrix. This defaults to 0 (i. e. no digits).

ExactPosition – As its name suggests, this class represents a position which is specified as exact along the sequence. This is represented as just a a number, and you can get the position by looking at the `position` attribute of the object.

BeforePosition

```
>>> my_location.nofuzzy_start
5
>>> my_location.nofuzzy_end
8
```

3.8.2.2 Registering and Grouping databases

The basic registry objects are nice in that they provide basic functionality, but if you have a more advanced

```
['embl-dbfetch-cgi', 'embl-fast', 'embl', 'prosite-expasy-cgi',
'swissprot-expasy-cgi', 'nucleotide-genbank-cgi', 'pdb-ebi-cgi',
'genbank', 'nucleotide-genbank-local', 'interpro-ebi-cgi',
'embl-ebi-cgi', 'embl-xembl-cgi', 'protein-genbank-cgi', 'pdb',
'prodoc-expasy-cgi', 'nucleotide-dbfetch-cgi', 'swissprot',
'pdb-rcsb-cgi']
```

You can also get the parent from a child.

```
parent_entity=child_entity.get_parent()
```

3.11.1.1.1 Constructing a Structure object

A Structure object is produced by a PDBParser object:

```
from Bio.PDB.PDBParser import PDBParser

p=PDBParser(PERMISSIVE=1)

structure_id="1fat"

filename="pdb1fat.ent"

s=p.get_structure(structure_id, filename)
```

The PERMISSIVE flag indicates that a number of common problems (see 3.11.5.1) associated with PDB files will be ignored (but note that some atoms and/or residues will be missing). If the flag is not present a PDBConstructionException

Let's give some examples. Asn 10 with a blank insertion code would have residue id (`'' '', 10, '' ''`). Water 10 would have residue id (`''W'', 10, '' ''`). A glucose molecule (a hetero residue with residue name GLC) with sequence identifier 10 would have residue id (`''H`

```
a.get_anisou()     # anisotropic B factor

a.get_fullname()   # atom name (with spaces, e.g. ".CA.")
```

To represent the atom coordinates, siguij, anisotropic B factor and sigatm Numpy arrays are used.

3.11.2 Disorder

3.11.2.1 General approach

Disorder should be dealt with from two points of view: the atom and the residue points of view. In general,

3.11.5.1.2 Duplicate atoms Structure 1EJG contains a Ser/Pro point mutation in chain A at position 22. In turn, Ser 22 contains some disordered atoms. As expected, all atoms belonging to Ser 22 have a

3.11.6 Other features

Chapter 4

Advanced

4.1 Sequence Class

4.2 Regression Testing Framework

Biopython has a regression testing framework written Andrew Dalke and ported to PyUnit by Brad Chapman which helps us make sure the code is as bug-free as possible before going out.

4.2.1 Writing a Regression Test

Every module that goes into Biopython should have a test (and should also have documentation!). Let's say you've written a new module called Biospam – here is what you should do to make a regression test:

1. Write a script called `test_Biospam.py`
    - This script should live in the Tests directory

4.3 Parser Design

4.3.1 Design Overview

database
posted_date
num_letters_in_database
num_sequences_in_database
num_letters_searched RESERVED. Currently unused. I’ve never

```
enzyme_role
structure_db
structure_id
dblinks_db
dblinks_id
record_end
```

4.3.9 Fasta

The Fasta.py module works with FASTA-for(h)tla(witteh)tld1(h)3(seh)tlquwitehule da-1(t-414(Th)tlh)tl(ule)-2FASTauleydu

4103.4(Ma(wittl)-2ine4(ast)1(a)]TJ/F89.963Tf0-18.389420(Och)t55Th)tlh420(Service28(421(Referencnzh420(Matee)-eh)talz

record_originator
journal_subset
subheadings
secondary_source_id
source
title_abbreviation
title
transliterated_title
unique_identifier
volume_issue
year
pubmed_id

organelle
organism_classification
reference_number
reference_position
reference_comment
reference_cross_reference
reference_author
reference_title
reference_location
comment
database_cross_reference
keyword
featu3Fabitledaer

4.3.15 MetaTool

(d) `self.sum_letters`: a dictionary. `{i1: s1, i2: s2,...,in:sn}` where:

i.

Chapter 5

Where to go from here – contributing to Biopython

5.1 Maintaining a distribution for a platform

We try to release Biopython to make it as easy to install as possible for users. Thus, we try to provide the Biopython libraries in as many install formats as we can. Doing this from release to release can be a lot of work for developers, and sometimes requires them to maintain packages they are not all that familiar with. This section is meant to provide tipsTf0-a27(e)-347(c)-1(an)1(.)-486(Doir)-3-1(an)o955Td[0(Th)(o)27ns,a3347(l)1(me)-1e

Macintosh

Chapter 6

```
 with multiple lines.
>>> import cStringIO
>>> my_info_handle = cStringIO.StringIO(my_info)
>>> first_line = my_info_handle.readline()
>>> print first_line
A string

>>> second_line = my_info_handle.readline()
>>> print second_line
 with multiple lines.
```