Converting GenBank (or other formats) to Fasta

Brad Chapman (chapmanb@uga.edu)

March 17, 2007

Contents

1 Background and Purpose 1

2 Quick conversion using the FormatIO system 1

3 Specialized conversion using Fasta Record objects 2

4 Beyond GenBank to Fasta 3

1 Background and Purpose

One of the most common file formats used in biology is Fasta format, thus converting into this format for storage of sequences or preparation for input into a program is very common. Fasta format has become so popular because it is very simple. It consists of two parts, a title line which begins with a `>` character, and then a number of sequence lines. Here's an example Fasta sequence:

```
>gi|3318709|pdb|1A91|  Subunit C Of The F1fo Atp Synthase Of Escherichia Coli
MENLNMDLLYMAAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGL
GLYVMFAVA
```

This document will discuss ways in Biopython to convert a file of GenBank records into a file of Fasta sequences. Although Fasta format is simple, it can also be difficult to deal with because there are no standards for how information about a sequence will be presented on the title line. Because of this we'll discuss two ways to do conversions:

- A quick and easy way which assumes that you don't need specialized control over the way output is produced on the title line.
- A more controlled way which allows you to construct title lines to your specifications.

These examples assume some basic familiarity with constructing Biopython Parsers and Iterators, and some understanding of the `SeqRecord` object in Biopython.

2 Quick conversion using the FormatIO system

If you only want a quick and easy to type way to convert GenBank to Fasta format with reasonable title names, Biopython has an in-development format conversion mechanism. If you've used BioPerl, this mechanism is similar to that used in their `SeqIO` system. Basically, it involves converting GenBank records into standard Biopython `SeqRecord` objects, and then converting these `SeqRecord` objects back into a Fasta record in a file. As we mentioned, this mechanism is easy, so all of this work is done internally within Biopython code.

Now that we understand how things will be working internally, let's write the code that drives the conversion process. First, we open up a handle to read from the GenBank file, and a handle to write to the output Fasta file. Assuming you've somehow defined `input_file`, which is an existing file of GenBank records, and `output_file`, the name of the output file, this is just standard python:

```
input_handle = open(input_file)
output_handle = open(output_file, "w")
```

Now, we create a Biopython `FormatIO` object which will, using `SeqRecord` objects, be converting between GenBank format and Fasta format. First, we get the Biopython registry of formats:

```
from Bio import formats
```

The imported `formats` object contains a dictionary-like interface to biological file formats able to be dealt with through the `FormatIO` system. A simple `print formats` gives you a list of all the available formats:

```
formats, exporting 'blast', 'blastn', 'blastp', 'blastx', 'embl', 'embl/65',
'empty', 'fasta', 'genbank', 'genbank-records', 'genbank-release',
'ncbi-blastn', 'ncbi-blastp', 'ncbi-blastx', 'ncbi-tblastn', 'ncbi-tblastx',
'search', 'sequence', 'swissprot', 'swissprot/38', 'swissprot/40',
'tblastn', 'tblastx', 'wu-blastn', 'wu-blastp', 'wu-blastx'
```

The important thing to note is that we've got our `genbank` and `fasta` formats we want to deal with. With these formats, we can create a formatter with one line of code:

```
formatter = FormatIO("SeqRecord", formats["genbank"], formats["fasta"])
```

With the formatter in hand, the actual conversion is also a very simple single line of code, using our input and output handles defined above:

```
formatter.convert(input_handle, output_handle)
```

That's it – we've done the conversion. Assuming we have a GenBank file beginning like:

```
LOCUS       ATCOR66M      513 bp    mRNA            PLN       02-MAR-1992
DEFINITION  A.thaliana cor6.6 mRNA.
ACCESSION   X55053
VERSION     X55053.1  GI:16229
```

This will output Fasta that looks like:

```
>X55053.1 A.thaliana cor6.6 mRNA.
aacaaaacacacatcaaaaacgattttacaagaaaaaaatatctgaaaaatgtcagagaccaacaagaatgc
```

Thus, the id (accession number with versioning) and description are maintained in the final Fasta title.

3 Specialized conversion using Fasta Record objects

If the generally useful scheme described above for retaining sequence information in the Fasta title does not work for your purposes, it is always possible to hand create your own system for conversion between GenBank information and Fasta titles.

This involves using the standard Fasta `Record` object. The important thing about this object, for our purposes, is that if you have a Fasta record object and get the string representation of it (by printing or calling `str` on the object) it will output nicely formatted Fasta. Thus, we need to only set the `title` and `sequence` attributes of the Fasta record, and we can write out nice Fasta.

Now that the preliminaries are out of the way, we'll get down to doing the work. First, we create a GenBank Iterator which will read over the input file and return GenBank Record objects:

```
from Bio import GenBank

iterator = GenBank.Iterator(input_handle, GenBank.RecordParser())
```

Now, using the information we learned above about Fasta Records, we can write out a Fasta file with complete control over how the tile looks. In this example case, we'll just store information about the GI number, LOCUS name, version, and description in the title line. We then loop over the GenBank records and write out the Fasta Records one at a time:

```
from Bio import Fasta

for gb_rec in iterator:
    fasta_rec = Fasta.Record()
    fasta_rec.title = "%s|%s|%s %s" % \
     (gb_rec.gi, gb_rec.locus, gb_rec.version, gb_rec.definition)
    fasta_rec.sequence = gb_rec.sequence
    output_handle.write(str(fasta_rec) + "\n")
```

This will produce an output file that, given the GenBank we showed above, looks like:

```
>16229|ATCOR66M|X55053.1 A.thaliana cor6.6 mRNA.
AACAAAACACACATCAAAAACGATTTTACAAGAAAAAAATATCTGAAAAATGTCAGAGAC
```

To adjust what is written on the title line, all we need to change from the code above is the line specifying what makes up `fasta_rec.title`.

4 Beyond GenBank to Fasta

Although this example describes converting GenBank to Fasta, other conversions are possible using the same frameworks.

For the `FormatIO` system, conversions from Swissprot or EMBL format to Fasta should work just as easily by replacing the `formats["genbank"]` dictionary call above with `swissprot` or `embl`. Currently, the `FormatIO` system is still under development and writing other formats besides Fasta is not yet supported. Other output formats will hopefully be added in the future.

For the Fasta Record system, any input format can be used that is supported by parsing in Biopython. For output, the GenBank Record class also supports string output in GenBank flat file format.