Bio::SeqIO HOWTO

Ewan Birney, EBI `<birney-at-ebi.ac.uk>`

Darinn London, EBI `<dlondon-at-ebi.ac.uk>`

Brian Osborne, Cognia Corporation `<brian-at-cognia.com>`

This HOWTO tries to teach you about the SeqIO system for reading and writing sequences of various formats

TT

2. 10 second overview

Lots of bioinformatics inv

For some one of the initial perplexities of Bioperl is the variety of dif

```
                                '-format' => $format );
# Now that we have a seq stream,
# we need to tell it to give us a $seq.
# We do this using the 'next_seq' method of SeqIO.

while (my $seq = $inseq->next_seq) {
      print $seq->accession_number,"\n";
}
exit;
```

This script takes two arguments on the commandline, and input filename and the format of the input file. This is the basic way to access the data in a Genbank fi for the next call to next_seq. It kno

```
my $usage = "all2y.pl informat outfile outfileformat\n";
my $informat = shift or die $usage;
my $outfile = shift or die $usage;
```

```
use IO::String;
use Bio::SeqIO;
```

```
my $usage = "splitgb.pl infile\n";
my $infile = shift or die $usage;
```

```
                    exit;
```

And finally, you might want to make use of the SeqIO object in a perl one-liner. Perl one-liners are perl programs that make use of flags to the perl binary allowing you to run programs from the command-line without actually needing to write a script into a file. The -e flag tak

```
use Bio::SeqIO;

my $input_file = shift;
my $output_file = shift;

# we have to declare $seq_in and $seq_out before
# the eval block as we want to use them afterwards

my $seq_in;
my $seq_out;

eval {
 $seq_in  = Bio::SeqIO->new( -format => 'genbank',
                             -file => $input_file);

$seq_out = Bio::SeqIO->new( -format => 'fasta',
                            -file => ">$output_file");
};
if( $@ ) { # an error occurred
  print "Was not able to open files, sorry!\n";
  print "Full1r>sorsor$@sorry!\n";
```