BioPerl I

Outline

- Overview of BioPerl
- Accessing and manipulating sequence and annotation data
  - Features, Annotations, Sequence data
- Processing sequence database search results (BLAST)
- Manipulating multiple sequence alignments

Big Picture

- BioPerl is a Perl toolkit for building programs
- In general it is focused on the data (Sequences, Alignments, Trees) more than implementation of algorithms
  - Primarily sequence focused based on contributors interests
- Since 1995 has been an open source collaboration with many different institutes and individuals

A significant portion of bioinformatics is just converting data from one format to another

Jumping in

- Processing a sequence file
  - Read in FASTA file
  - Count how many sequences are in the file
  - Count how many bases are in the file
  - Count how many bases are in the file, ignoring certain characters (Stop codons)
  - See if a particular sequence motif is present

Reading a FASTA file

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
my $file = 'basidio_fungi_20050923.aa';
my $in = Bio::SeqIO->new(-format => 'fasta',
                         -file   => $file);
my ($seqcount,$basecount,$basecount_nostops);
while ( my $seq = $in->next_seq) {
 $seqcount++; # count the number of sequences
 $basecount += $seq->length; # count how many bases in the whole db
 my $str = $seq->seq; # get the sequence as a string
 $str =~ s/\*//g;     # remove all '*' from sequence
 $basecount_nostops += length($str); # count bases from this string
}
print "In db $file ther are $seqcount sequences, and $basecount bases
($basecount_nostops ignoring *)\n";
printf "In db %s ther are %d sequences, and %d bases (%d ignoring *)\n",
  $file, $seqcount, $basecount, $basecount_nostops;
```

```
$ perl read_seq.pl
In db basidio_fungi_20050923.aa ther are 76650 sequences, and 35621971 bases
(35545008 ignoring *)
```

Finding Motifs

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
my $file = 'basidio_fungi_20050923.aa';
my $motif = '^[^C]+(C[^C]+){4}[^C]*$'; # CX^NCX^NCX^NC
#my $motif = '^[^C]+(C[^C]){4}[^C]*$'; # CXCXCXC
# my $motif = '(C[^C]{2,}){2,}'; # (CX^N)^N
my $in = Bio::SeqIO->new(-format => 'fasta',
                         -file   => $file);
 my $motif_count = 0;
 while ( my $seq = $in->next_seq) {
  my $str = $seq->seq; # get the sequence as a string
  if ( $str =~ /$motif/i ) {
    $motif_count++; # count number of sequences that have this motif
  }
 }
 printf "%d sequences have the motif $motif\n",$motif_count;
```

```
$ perl read_seq.pl
4 sequences have the motif ^[^C]+(C[^C]){4}[^C]*$
```

Sequence File Formats

- Lots of sequences file formats to capture both preferences of application designers and additional information
  - Minimal: FASTA, GCG
  - Base quality: FASTQ, FASTA+QUAL
  - Annotation: GenBank, EMBL, SwissProt, FASTQ
  - XML: BSML, AGAVE, NCBIXML, TIGRXML, CHADO

SeqIO Magic

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
my $file = 'wormpep.aa';
my $in = Bio::SeqIO->new(-format => 'genbank',
                         -file   => $file);
my ($seqcount,$basecount,$basecount_nostops);
while ( my $seq = $in->next_seq) {
 $seqcount++; # count the number of sequences
 $basecount += $seq->length; # count how many bases in the whole db
 my $str = $seq->seq; # get the sequence as a string
 $str =~ s/\*//g;     # remove all '*' from sequence
 $basecount_nostops += length($str); # count bases from this string
}
printf "In db %s ther are %d sequences, and %d bases (%d ignoring *)\n",
  $file, $seqcount, $basecount, $basecount_nostops;
```

Bio::SeqIO

- Used to read and write sequences in different formats
- Two methods: `next_seq` and `write_seq`
- next_seq will keep returning seqs till end of file (or stream)
- write_seq can take one sequence or a list of seqs to writes out sequence(s) in specified format

Writing Sequences: Convert format

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;

my $informat = 'genbank';
my $outformat= 'fasta';
my $in = Bio::SeqIO->new(-format => $informat,   # input handle
                         -file   => 'seqs.gbk');
my $out = Bio::SeqIO->new(-format => $outformat, # output handle
                          -file   => '>seqs.fas');
while( my $seq = $in->next_seq ) {
 $out->write_seq($seq);
}
```

$ perl write_seqs.pl

Writing Sequences: Convert format

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
use Getopt::Long;
my ($informat,$outformat) = ('genbank','fasta');
my ($infile,$outfile);
GetOptions('if:s'  => \$informat,
           'of:s'  => \$outformat,
           'i|input:s' => \$infile,
           'o|output:s'=> \$outfile);
die "need input and output filenames\n" unless $infile && $outfile;
my $in = Bio::SeqIO->new(-format => $informat,   # input handle
                         -file   => $infile);
my $out = Bio::SeqIO->new(-format => $outformat, # output handle
                          -file   => ">$outfile");
while( my $seq = $in->next_seq ) {
 $out->write_seq($seq);
}
```

$ perl convert_seqs.pl -i seqs.gbk -o seqs.fa

UNIX magic + SeqIO

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
my $fh
open( IN, "zcat database.nt.gz |") || die "$!"; # open handle
my $inz = Bio::SeqIO->new(-format => 'fasta', -fh => \*IN);
while( my $seq= $in->next_seq ) {
 # process sequences
}
```

```
#compress on the fly
open(my $ofh => "| gzip -c > newdb.fa.gz ") || die "$!"; # open handle
my $in = Bio::SeqIO->new(-format => 'fasta', -file => 'seqs.fas');
my $out = Bio::SeqIO->new(-format => 'fasta', -fh => $ofh);
while( my $seq= $in->next_seq ) {
 $out->write_seq($seq);
}
```

Writing Sequences: Convert format

```
use Bio::SeqIO;
use Getopt::Long
my ($informat,$outformat) = ('genbank','fasta');
my ($infile,$outfile);
my $length = 0;
GetOptions('if:s'      => \$informat,
           'of:s'      => \$outformat,
           'i|input:s' => \$infile,
           'o|output:s'=> \$outfile,
           'l|length:i'=> \$length);
die "need input and output filenames\n" unless $infile && $outfile;
my $in = Bio::SeqIO->new(-format => $informat,   # input handle
                         -file   => $infile);
my $out = Bio::SeqIO->new(-format => $outformat, # output handle
                          -file   => ">$outfile");
while( my $seq = $in->next_seq ) {
 next unless $seq->length > $length;
 $out->write_seq($seq);
}
```

$ perl convert_seqs.pl -i seqs.gbk -o seqs_trim.fa

What is in a Sequence object?

- Implementations in Bio::PrimarySeq, Bio::Seq, Bio::Seq::RichSeq
- Sequences have methods for seq info `length`, `seq`, and extracting a part of it `subseq`,`trunc`
- Associated information like name (`display_id`), comments (`description`), `accession_number`, GI number (`primary_id`)
- DNA Methods for reverse complementation (`revcom`), translation to protein (`translate`)

Additional associated information

- Features:
  - `get_SeqFeatures`, `add_SeqFeature`, `remove_SeqFeatures`
- Access the annotations in AnnotationCollection
  - `annotation` - Comments, Authors, DBLinks, Other Simple Tag/Values
- Get extra (GenBank) information
  - `secondary_accession` numbers, `division`, `pid`, `seq_version`, dates (created, last updated), keywords

Can create a sequence on the fly

```
#!/usr/bin/perl -w
use strict;
use Bio::Seq;
use Bio::SeqIO;
my $seq = Bio::Seq->new(-seq         => 'ATGAATGATGAA',
                        -display_id => 'example',
                        -description=> 'My first example sequence');
my $out = Bio::SeqIO->new(-format => 'fasta');
$out->write_seq($seq);
print "Id is ",$seq->display_id, "\n";
print "Length is ", $seq->length, "\n";
print "Seq is                  ",$seq->seq,"\n";
print "Reverse complement is ", $seq->revcom->seq,"\n";
print "Translation is ", $seq->translate->seq, "\n";
print "Subseq of 3..6 is ", $seq->subseq(3,6), "\n";
print "Trunc/revcom of 3..6 is ", $seq->trunc(3,6)->revcom->seq, "\n";
```

Seq on the fly: output

```
$ perl seq_on_the_fly.pl
>example My first example sequence
ATGAATGATGAA
Id is example
Length is 12
Seq is                    ATGAATGATGAA
Reverse complement is TTCATCATTCAT
Translation is MNDE
Subseq of 3..6 is GAAT
Trunc/revcom of 3..6 is ATTC
```

Navigating the documentation

- perldoc Bio::Seq
- http://bioperl.org/
  - See HOWTOs and Tutorial
  - See DeObfuscator

Sequences and Features

- Sequences can have annotation about them
  - Authors of the record; Accession Number; Version, Date created; Cross-references to other DBs
  - Location of the Genes, Exons, mutations, phosphorylation sites.
- *Features* are annotations on sequence with a location.
- *Annotations* are annotations associated to sequence.
- Feature: Bio::SeqFeature::Generic
- Annotation: Bio::Annotation::DBLink

Features

- Bio::SeqFeature namespace
  - Attached to sequences with a location
    - Bio::Location objects handle: start, end, strand
  - Tags-Value pairs can be associated with a feature

Getting data out of a feature

```
#     source          1..10001
#                     /organism="Homo sapiens"
#                     /mol_type="genomic DNA"
#                     /db_xref="taxon:9606"
#                     /chromosome="1"

print $feature->primary_tag," ",$feature->start, "..",$feature->end,"\n";
print "tags\n";
for my $tag ( sort $feature->get_all_tags ) {
 my @values = $feature->get_tag_values($tag);
    print " $tag:\t", join(",", @values),"\n";
}
```

```
source 1..10001
tags
 chromosome 1
 db_type    taxon:9606
 mol_type   genomic DNA
 organism   Homo sapiens
```

print_features.pl

Getting data out of a feature

```
#     mRNA            complement(join(3024..4108,4110..4258,4357..4533,
#                     5985..6225,6324..6641))
#                     /gene="LOC127086"
#                     /product="similar to ATP-dependent DNA helicase II, 70 kDa
#                     subunit (Lupus Ku autoantigen protein p70) (Ku70) (70 kDa
#                     subunit of Ku antigen) (Thyroid-lupus autoantigen) (TLAA)
#                     (CTC box binding factor 75 kDa subunit) (CTCBF) (CTC75)"
#                     /note="Derived by automated computational analysis using
#                     gene prediction method: GNOMON."
#                     /transcript_id="XM_060320.3"
#                     /db_xref="GI:37539614"
#                     /db_xref="GeneID:127086"
#                     /db_xref="InterimID:127086"

print $feature->primary_tag," ",$feature->start, "..",$feature->end,"\n";
print "tags\n";
for my $tag ( sort $feature->get_all_tags ) {
 my @values = $feature->get_tag_values($tag);
    print " $tag:\t", join(",", @values),"\n";
}
```

print_features.pl

```
mRNA 3024..6641
 db_xref:  GI:37539614,GeneID:127086,InterimID:127086
 gene:  LOC127086
 note:  Derived by automated computational analysis using gene pred ...
 product:  similar to ATP-dependent DNA helicase II, 70 kDa subunit ...
```

Getting data out of a feature

```
#     CDS             complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
#                     /gene="LOC127086"
#                     /note="overriding stop codons"
#                     /codon_start=1
#                     /transl_except=(pos:complement(6444..6446),aa:OTHER)
#                     /transl_except=(pos:complement(4224..4226),aa:OTHER)
#                     /transl_except=(pos:complement(4067..4069),aa:OTHER)
#                     /transl_except=(pos:complement(4049..4051),aa:OTHER)
#                     /transl_except=(pos:complement(4046..4048),aa:OTHER)
#                     /transl_except=(pos:complement(3791..3793),aa:OTHER)
#                     /transl_except=(pos:complement(3678..3680),aa:OTHER)
#                     /transl_except=(pos:complement(3036..3038),aa:OTHER)
#                     /protein_id="XP_060320.3"
#                     /db_xref="GI:37539615"
#                     /db_xref="GeneID:127086"
#                     /db_xref="InterimID:127086

print $feature->primary_tag," ",$feature->location->to_FTstring(),"\n";
print "tags\n";
for my $tag ( sort $feature->get_all_tags ) {
 my @values = $feature->get_tag_values($tag);
    print " $tag:\t", join(",", @values),"\n";
}
```

print_features2.pl

```
mRNA complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
  db_xref:  GI:37539614,GeneID:127086,InterimID:127086
  gene:  LOC127086
  note:  Derived by automated computational analysis using gene pred ...
  product:  similar to ATP-dependent DNA helicase II, 70 kDa subunit ...
```

Unwinding a "Split" location

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
my $file = 'NT_021877.gbk';
my $in = Bio::SeqIO->new(-format => 'genbank',
             -file   => $file);
my $cds_out = Bio::SeqIO->new(-format => 'fasta',
                  -file   => ">$file.CDS");
my $exon_out = Bio::SeqIO->new(-format => 'fasta',
                  -file   => ">$file.CDS_exons");
while ( my $seq = $in->next_seq ) {
  print $seq->display_id, " features:\n";
  for my $feature ( $seq->get_SeqFeatures ) {
    print " ",$feature->primary_tag," ",$feature->location->to_FTstring(),"\n";
    next unless ( $feature->primary_tag eq 'CDS');
    my ($name) = $feature->get_tag_values('gene'); # careful, what if it doesn't exist?
    my $exonct = 1;
    for my $exon ($feature->location->each_Location ) {
      print "  ",$exon->start, "..",$exon->end,"\n";
      print "  ",$exon->to_FTstring,"\n";
      my $exonseq = $seq->trunc($exon);
      $exonseq->display_id($name.".exon".$exonct++);
      $exon_out->write_seq($exonseq);
    }
    my $spliced = $feature->spliced_seq;
    $spliced->display_id($name);
    $cds_out->write_seq($spliced);
  }
}
```

Going to save whole CDS and each exon seqs

each_Location: Iterate through each Sub Location in a Split Loc
Printing 2 times for each Exon here.

spliced_seq: get all the pieces stitched together

location_manipulate.pl

Split Location Manipulation

```
NT_021877 features:
 source 1..10001
 source <1..>10001
 gene complement(3024..6641)
 mRNA
complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
 CDS
complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
  3024..4108
  complement(3024..4108)
  4110..4258
  complement(4110..4258)
  4357..4533
  complement(4357..4533)
  5985..6225
  complement(5985..6225)
  6324..6641
  complement(6324..6641)
```

location_manipulate.pl

Split Location Manipulation

```
>LOC127086
ATGCCCAGGGAAGACAGGGTGACCTGGAAGCCCAACTACTTCCTTAAGATCATCCAACTT
TTGGATGATTATCCGAAATGTTTCATCATGGGAGCAGACAATATGGGCTCTAAGCAGATG
CAGCAGATCCGCATATCCCTCTGCAGGAAGGCCATGGTGCTGCTAGGCAAGAACACCACG
ATGGTCAAGGCCATCTGAAGGCACCTGGAAAACAACCCAGCTCTAGAGAAACTGTTGCCT
CATATTCAGGGGAATGTGGGCTTTGTGTTCATCAAGGAGGACCTCACTGAGATCAGGGAC
CTGCTGCTGGCCAACAAGGTTTTAGGCATCACCACTAAAATCTCCAGGGGCGCCACTGAA
ATCCTGAGTGATGTGCAGCTGATCAAGACTGGAGACAAAGTGGGAGCCAGCGAAGCCACA

>LOC127086.exon1 complement(3024..4108)
GAGGAATCCAGAAAGCTAGAAGACCTGTTGAGGCAGGTTTGAGCCAAGGAGATCAGTTAG
TGAACACTCAGCAGGTTAAAGCTGAAGCTCAATAAAGATATAGTGCTCTCTGTGGGCATT
TATAATCCGATCCAGAAGGCTCTCAAGCCTCCTCCAATAAAGCCCTATCGAGAAATAGAT
GAATCAGTGAAAACCAAGACCTGGATATTTAATGTAAATACAGGCAGTTGGCTTCTGTCT
AGAGATACCAAGAGGTCTCAGATCTATGGAAGGCGTCAGATTATACTGGAGAAAGAGGAA
....
>LOC127086.exon2 complement(4110..4258)
CCAGCTGGGCCAGGACCAAAGCCAATAATCACTGAGATACAGGCATCTTCCTTGACTTGA
TGCACCTGAAGAAAACTGAGGGCTTTGATATACCTTTCTTCTACAGAGATATCACCAGCA
TAGCAGAGGATGAGGACCCCAGGGCTCAC
```

NT_021877.gbk.CDS

NT_021877.gbk.CDS_exon

```
complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
```

location_manipulate.pl

Ordering the exons

```
use Bio::SeqIO;
my $file = 'NT_021877.gbk';
my $in = Bio::SeqIO->new(-format => 'genbank',
               -file   => $file);
my $cds_out = Bio::SeqIO->new(-format => 'fasta',
                   -file   => ">$file.CDS");
my $exon_out = Bio::SeqIO->new(-format => 'fasta',
                   -file   => ">$file.CDS_exons");
my $exon_pep_out = Bio::SeqIO->new(-format => 'fasta',
                    -file   => ">$file.CDS_exons_pep");
while ( my $seq = $in->next_seq ) {
  print $seq->display_id, " features:\n";
  for my $feature ( $seq->get_SeqFeatures ) {
    print " ",$feature->primary_tag," ",$feature->location->to_FTstring(),"\n";
    next unless ( $feature->primary_tag eq 'CDS');
    my ($name) = $feature->get_tag_values('gene'); # careful, what if it doesn't exist?
    my $exonct = 1;
    for my $exon (sort { $a->start * $feature->strand <=> $b->start * $feature->strand }
           $feature->location->each_Location ) {
      print "  ",$exon->start, "..",$exon->end,"\n";
      print "  ",$exon->to_FTstring,"\n";
      my $exonseq = $seq->trunc($exon);
      $exonseq->display_id($name.".exon".$exonct++);
      $exonseq->description($exon->to_FTstring);
      $exon_out->write_seq($exonseq);
      $exon_pep_out->write_seq($exonseq->translate);
    }
    my $spliced = $feature->spliced_seq;
    $spliced->display_id($name);
    $cds_out->write_seq($spliced);
  }
}
```

Split Location Manipulation

```
stajich@milliways $ perl location_manipulate2.pl
NT_021877 features:
 source 1..10001
 source <1..>10001
 gene complement(3024..6641)
 mRNA
complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
 CDS
complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
  complement(6324..6641)
  complement(5985..6225)
  complement(4357..4533)
  complement(4110..4258)
  complement(3024..4108)
```

location_manipulate2.pl

Split Location Manipulation

```
complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
>LOC127086
ATGCCCAGGGAAGACAGGGTGACCTGGAAGCCCAACTACTTCCTTAAGATCATCCAACTT
TTGGATGATTATCCGAAATGTTTCATCATGGGAGCAGACAATATGGGCTCTAAGCAGATG
CAGCAGATCCGCATATCCCTCTGCAGGAAGGCCATGGTGCTGCTAGGCAAGAACACCACG
ATGGTCAAGGCCATCTGAAGGCACCTGGAAAACAACCCAGCTCTAGAGAAACTGTTGCCT
CATATTCAGGGGAATGTGGGCTTTGTGTTCATCAAGGAGGACCTCACTGAGATCAGGGAC
CTGCTGCTGGCCAACAAGGTTTTAGGCATCACCACTAAAATCTCCAGGGGCGCCACTGAA
ATCCTGAGTGATGTGCAGCTGATCAAGACTGGAGACAAAGTGGGAGCCAGCGAAGCCACA

>LOC127086.exon1 complement(6324..6641)
ATGCCCAGGGAAGACAGGGTGACCTGGAAGCCCAACTACTTCCTTAAGATCATCCAACTT
TTGGATGATTATCCGAAATGTTTCATCATGGGAGCAGACAATATGGGCTCTAAGCAGATG
CAGCAGATCCGCATATCCCTCTGCAGGAAGGCCATGGTGCTGCTAGGCAAGAACACCACG
ATGGTCAAGGCCATCTGAAGGCACCTGGAAAACAACCCAGCTCTAGAGAAACTGTTGCCT
CATATTCAGGGGAATGTGGGCTTTGTGTTCATCAAGGAGGACCTCACTGAGATCAGGGAC
CTGCTGCTGGCCAACAAG
>LOC127086.exon2 complement(5985..6225)
GTTTTAGGCATCACCACTAAAATCTCCAGGGGCGCCACTGAAATCCTGAGTGATGTGCAG
CTGATCAAGACTGGAGACAAAGTGGGAGCCAGCGAAGCCACACTGCTGAACATCTCTCCC
...
>LOC127086.exon1 complement(6324..6641)
MPREDRVTWKPNYFLKIIQLLDDYPKCFIMGADNMGSKQMQQIRISLCRKAMVLLGKNTT
MVKAI*RHLENNPALEKLLPHIQGNVGFVFIKEDLTEIRDLLLANK
>LOC127086.exon2 complement(5985..6225)
VLGITTKISRGATEILSDVQLIKTGDKVGASEATLLNISPFFGLVIQQVFDNGSIYNPEG
LDITEETAFSLSGECLRCFQ
```

NT_021877.gbk.CDS

NT_021877.gbk.CDS_exon

NT_021877.gbk.CDS_exon_pep

location_manipulate.pl

Translation of CDS is OKAY

```
>LOC127086.exon1 complement(6324..6641)
MPREDRVTWKPNYFLKIIQLLDDYPKCFIMGADNMGSKQMQQIRISLCRKAMVLLGKNTT
MVKAI*RHLENNPALEKLLPHIQGNVGFVFIKEDLTEIRDLLLANK
>LOC127086.exon2 complement(5985..6225)
VLGITTKISRGATEILSDVQLIKTGDKVGASEATLLNISPFFGLVIQQVFDNGSIYNPEG
LDITEETAFSLSGECLRCFQ

#      CDS             complement(join(3024..4108,4110..4258,4357..4533,5985..6225,6324..6641))
#                      /gene="LOC127086"
#                      /note="overriding stop codons"
#                      /codon_start=1
#                      /transl_except=(pos:complement(6444..6446),aa:OTHER)
#                      /transl_except=(pos:complement(4224..4226),aa:OTHER)
#                      /transl_except=(pos:complement(4067..4069),aa:OTHER)
#                      /transl_except=(pos:complement(4049..4051),aa:OTHER)
#                      /transl_except=(pos:complement(4046..4048),aa:OTHER)
#                      /transl_except=(pos:complement(3791..3793),aa:OTHER)
#                      /transl_except=(pos:complement(3678..3680),aa:OTHER)
#                      /transl_except=(pos:complement(3036..3038),aa:OTHER)
#                      /protein_id="XP_060320.3"
#                      /db_xref="GI:37539615"
#                      /db_xref="GeneID:127086"
#                      /db_xref="InterimID:127086
```

Annotations

- Information about the whole sequence usually
- Somewhat arbitrary separation that was made in BioPerl past
- Accessed through Bio::AnnotationCollection

Swissprot Record

```
ID   GCDH_CAEEL              Reviewed;         409 AA.
AC   Q20772;
DT   01-NOV-1997, integrated into UniProtKB/Swiss-Prot.
DT   01-NOV-1996, sequence version 1.
DT   31-OCT-2006, entry version 47.
DE   Probable glutaryl-CoA dehydrogenase, mitochondrial
precursor
DE   (EC 1.3.99.7) (GCD).
GN   ORFNames=F54D5.7;
OS   Caenorhabditis elegans.
OC   Eukaryota; Metazoa; Nematoda; Chromadorea; Rhabditida;
Rhabditoidea;
OC   Rhabditidae; Peloderinae; Caenorhabditis.
OX   NCBI_TaxID=6239;
RN   [1]
RP   NUCLEOTIDE SEQUENCE [LARGE SCALE GENOMIC DNA].
RC   STRAIN=Bristol N2;
RX   MEDLINE=99069613; PubMed=9851916; DOI=10.1126/science.
282.5396.2012;
RG   The C. elegans sequencing consortium;
RT   "Genome sequence of the nematode C. elegans: a platform
for
RT   investigating biology.";
RL   Science 282:2012-2018(1998).
```

```
DR   EMBL; Z66513; CAA91333.1; -; Genomic_DNA.
DR   PIR; T22647; T22647.
DR   UniGene; Cel.30446; -.
DR   HSSP; Q06319; 1BUC.
DR   IntAct; Q20772; -.
DR   Ensembl; F54D5.7; Caenorhabditis elegans.
DR   KEGG; cel:F54D5.7; -.
DR   WormBase; WBGene00010052; F54D5.7.
DR   WormPep; F54D5.7; CE03411.
DR   GO; GO:0005515; F:protein binding; IPI:IntAct.
DR   InterPro; IPR006089; Acyl_CoA_DH.
DR   InterPro; IPR006091; Acyl_CoA_DH/ox_M.
DR   InterPro; IPR006090; Acyl_CoA_DH_1.
DR   InterPro; IPR006092; Acyl_CoA_DH_N.
DR   InterPro; IPR009075; AcylCo_DH/ox_C.
DR   InterPro; IPR013786; AcylCoA_DH/ox_N.
DR   InterPro; IPR009100; AcylCoA_DH/ox_NM.
DR   InterPro; IPR013764; AcylCoA_DH_1/2_C.
DR   Pfam; PF00441; Acyl-CoA_dh_1; 1.
DR   Pfam; PF02770; Acyl-CoA_dh_M; 1.
DR   Pfam; PF02771; Acyl-CoA_dh_N; 1.
DR   PROSITE; PS00072; ACYL_COA_DH_1; FALSE_NEG.
DR   PROSITE; PS00073; ACYL_COA_DH_2; 1.
```

Annotation Collection access

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
my $in = Bio::SeqIO->new(-format => 'swiss',
                         -file   => 'rel9.swiss');
while( my $seq = $in->next_seq ) {
 my $collection = $seq->annotation;
 my @types = $collection->get_all_annotation_keys;
 print "types are @types\n";
 my @dblinks = $collection->get_Annotations('dblink');
 for my $dblink ( @dblinks ) {
   printf "%s:%s \n", $dblink->database, $dblink->primary_id .
     (defined $dblink->version ? ".".$dblink->version : '');
 }
}
```

types are keyword comment reference date_changed seq_update dblink gene_name
EMBL:Z66513
PIR:T22647
UniGene:Cel.30446
HSSP:Q06319
IntAct:Q20772
Ensembl:F54D5.7
KEGG:cel:F54D5.7
WormBase:WBGene00010052
WormPep:F54D5.7
GO:GO:0005515
InterPro:IPR006089
InterPro:IPR006091
InterPro:IPR006090
InterPro:IPR006092
InterPro:IPR009075
InterPro:IPR013786
InterPro:IPR009100
InterPro:IPR013764
Pfam:PF00441
Pfam:PF02770
Pfam:PF02771
PROSITE:PS00072
PROSITE:PS00073

Sequence Databases

- GenBank, Swissprot, EMBL all provide sequence data
- Can download large sets via FTP or web to work with sequence locally
- Ideal also for indexing genomic sequences and obtaining subregions of genome

Local Sequence Databases

Locally Indexed

- Bio::DB::Fasta
  - FASTA only format supported
  - Very fast. Respects most of Bio::Seq interface
- Bio::DB::Flat
  - Genbank, Swissprot, EMBL
- Sequence data: Bio::Index::GenBank, Bio::Index::EMBL, Bio::Index::Swissprot
- BLAST reports: Bio::Index::BLAST

Bio::DB::Fasta

db_fasta.pl

```
#!/usr/bin/perl -w
use strict;
use Bio::DB::Fasta;

my $dbfile = 'basidio_fungi_20050923.aa';
my $db = Bio::DB::Fasta->new($dbfile);

# retrieve a sequence
my $id = 'cneo_JEC21_TIGR:CNA07700';
my $seq = $db->get_Seq_by_acc($id);
if ( $seq ) {
  print "seq was ",$seq->seq,"\n";
} else {
  warn("Cannot find $id\n");
}
```

Bio::DB::Fasta custom ID

db_fasta_customid.pl

```
#!/usr/bin/perl -w
use strict;
use Bio::DB::Fasta;
my $dbfile = 'basidio_fungi_20050923.aa';
my $db = Bio::DB::Fasta->new($dbfile, -makeid => \&my_makeid);

# retrieve a sequence
for my $id ( qw(CNA07700 cneo_JEC21_TIGR:CNA07700) ) {
  my $seq = $db->get_Seq_by_acc($id);
  if ( $seq ) {
    print "seq was ",$seq->seq,"\n";
  } else {
    warn("Cannot find $id\n");
  }
}
sub my_makeid {
  my $id = shift;
  if ( $id =~ /^>[^:]+:(\S+)/ ) {
    return $1;
  } elsif  ($id =~ /^>(\S+)/) {
    return $1;
  } else {
    warn("cannot parse ID for $id\n");
  }
}
```

Bio::DB::Fasta for Genomes

```
#!/usr/bin/perl -w
use strict;
use Bio::DB::Fasta;
my $dbfile = 'saccharomyces_cerevisiae_S288C.fasta';
my $db = Bio::DB::Fasta->new($dbfile);

# retrieve a sub part of a chromosome as a STRING
my $piece = $db->seq('chrX',10002 => 12032);
my $piece_rev = $db->seq('chrX', 22102 => 20100);
```

Remote Databases

- NCBI: Bio::DB::GenBank and Bio::DB::GenPept
  - Bio::DB::Query::GenBank allows Entrez Queries to be run and retrieved
- Other dbs: Bio::DB::EMBL, Bio::DB::SwissProt

Bio::DB Remote DB query

```
use strict;
use Bio::DB::GenBank;
use Bio::SeqIO;

my $out = Bio::SeqIO->new(-format => 'genbank');
my $dbh = Bio::DB::GenBank->new;
my $query = 'MUSIGHBA1';
my $seq = $dbh->get_Seq_by_acc($query);
if( $seq ) {
 $out->write_seq($seq);
} else {
 warn("cannot find sequence $query\n");
}
```

genbank_query.pl

```
LOCUS       MUSIGHBA1                408 bp    mRNA    linear   ROD 27-APR-1993
DEFINITION  Mouse Ig active H-chain V-region from MOPC21, subgroup VH-II,
            mRNA.
ACCESSION   J00522
VERSION     J00522.1  GI:195052
KEYWORDS    constant region; immunoglobulin heavy chain; processed gene; variable region; variable region subgroup
VH-II.
SOURCE      Mus musculus (house mouse)
  ORGANISM  Mus musculus
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Glires; Rodentia;
            Sciurognathi; Muroidea; Muridae; Murinae; Mus.
REFERENCE   1  (bases 1 to 408)
  AUTHORS   Bothwell,A.L., Paskind,M., Reth,M., Imanishi-Kari,T., Rajewsky,K.
            and Baltimore,D.
  TITLE     Heavy chain variable region contribution to the NPb family of
```

Bio::DB::Fasta Caveats

- Not really suitable for indexing more than 1-2 M sequences
  - Does not work well for NextGen sequence files
    - Due to BerkeleyDB I assume
- FASTA files need to be consistent WRT line length. Can pre-process the file first with

```
$ bp_sreformat -if fasta -of fasta -i DB -o DB.new
$ mv DB.new DB
```