Bioperl

Sofia Robb

What is Bioperl?

Collection of tools to help you get your work done

Open source, contributed by users

Used by GMOD, wormbase, flybase, me, you

http://www.bioperl.org

Why use BioPerl?

Code is already written.
Manipulate sequences.
Run programs (e.g., blast, clustalw and phylip).
Parsing program output (e.g., blast and alignments).
And much, much more. (http://www.bioperl.org/wiki/Bptutorial.pl)

Learning about bioperl

Manipulation of sequences from a file

Query a local fasta file

Creating a sequence record

File format conversions

Retrieving annotations

Parsing Blast output

Manipulating Multiple Alignments

Other Cool Things

Learning about Bioperl:

Navigating Bioperl website
Deobfuscator
Bioperl docs

www.bioperl.org Main Page

page | discussion | view source | history

BioPerl

main links
- Main Page
- Getting Started
- Downloads
- Installation
- Recent changes
- Random page

documentation
- Quick Start
- FAQ
- HOWTOs
- API Docs
- Scrapbook
- BioPerl Tutorial

HOWTOs

HOWTOs are narrative-based descriptions of BioPerl modules focusing more on a concept or a task than one specific module.

BioPerl HOWTOs

Beginners HOWTO

An introduction to BioPerl, including reading and writing sequence files, running and parsing BLAST, retrieving from databases, and more.

SeqIO HOWTO

Sequence file I/O, with many script examples.

SearchIO HOWTO

Parsing reports from sequence comparison programs like BLAST and writing custom reports.

Tiling HOWTO

Using search reports parsed by SearchIO to obtain robust overall alignment statistics

Feature-Annotation HOWTO

Reading and writing detailed data associated with sequences.

SimpleWebAnalysis HOWTO

Submitting sequence data to Web forms and retrieving results.

Flat Databases HOWTO

Indexing local sequence files for fast retrieval.

PAML HOWTO

Using the PAML package using BioPerl.

OBDA Access HOWTO

howto | discussion | view source | history

BioPerl

main links
- Main Page
- Getting Started
- Downloads
- Installation
- Recent changes
- Random page

documentation
- Quick Start
- FAQ
- HOWTOs
- API Docs
- Scrapbook
- BioPerl Tutorial
- Tutorials
- Deobfuscator
- Browse Modules

community
- News
- Mailing lists
- Supporting BioPerl

HOWTO:Beginners

Contents [hide]

BioPerl

ain links

- Main Page
- Getting Started
- Downloads
- Installation
- Recent changes
- Random page

ocumentation

- Quick Start
- FAQ
- HOWTOs
- API Docs
- Scrapbook
- BioPerl Tutorial
- Tutorials
- Deobfuscator
- Browse Modules

ommunity

- News
- Mailing lists
- Supporting BioPerl

Deobfuscator

Contents [hide]

What is the Deobfuscator?

The Deobfuscator was written to make it easier to determine the methods that are available from a given BioPerl module (a common BioPerl FAQ).

BioPerl is a highly object-oriented software package, with often multiple levels of inheritance. Although each individual module is usually well-documented for the methods specific to it, identifying the inherited methods is less straightforward.

The Deobfuscator indexes all of the BioPerl POD documentation, taking account of the inheritance tree (thanks to Class::Inspector), and then presents all of the methods available to each module through a searchable web interface.

Where can I find the Deobfuscator?

The Deobfuscator is currently available here, indexing *bioperl-live*.

Welcome to the BioPerl Deobfuscator [bioperl-live] what is it?

Search **class names** by string or Perl regex (examples: Bio::SeqIO, seq, fasta$)

blast [Submit Query]

OR select a class from the list:

Bio::SearchIO::blast	Event generator for event based parsing of blast reports
Bio::SearchIO::blast_pull	A parser for BLAST output
Bio::SearchIO::blasttable	Driver module for SearchIO for parsing NCBI -m 8/9 format
Bio::SearchIO::blastxml	A SearchIO implementation of NCBI Blast XML parsing.
Bio::SearchIO::megablast	a driver module for Bio::SearchIO to parse megablast reports (format 0)
Bio::Tools::Run::RemoteBlast	Object for remote execution of the NCBI Blast via HTTP
Bio::Tools::Run::StandAloneBlast	Object for the local execution of the NCBI BLAST program suite (blastall, blastpgp, bl2seq). There is experimental support for WU-Blast and NCBI rpsblast.
Bio::Tools::Run::StandAloneNCBIBlast	Object for the local execution of the NCBI BLAST program suite (blastall, blastpgp, bl2seq). With experimental support for NCBI rpsblast.

Deobfuscator

Bio::SearchIO::XML::BlastHandler	XML Handler for NCBI Blast XML parsing.
Bio::SearchIO::XML::PsiBlastHandler	XML Handler for NCBI Blast PSIBLAST XML parsing.

sort by method

methods for **Bio::Tools::Run::StandAloneBlast**

executable	Bio::Tools::Run::StandAloneBlast	string representing the full path to the exe	my $exe = $blastfactory->executable('blasta
finally	Bio::Root::Root	not documented	not documented
io	Bio::Tools::Run::WrapperBase	Bio::Root::IO object	$obj->io($newval)
new	Bio::Tools::Run::StandAloneBlast	Bio::Tools::Run::StandAloneNCBIBlast or StandAloneWUBlast	my $obj = Bio::Tools::Run::StandAloneBlast
no_param_checks	Bio::Tools::Run::WrapperBase	value of no_param_checks	$obj->no_param_checks($newva
otherwise	Bio::Root::Root	not documented	not documented
outfile_name	Bio::Tools::Run::WrapperBase	string	my $outfile = $wrapper->outfile_
program	Bio::Tools::Run::StandAloneBlast	not documented	not documented

doc.bioperl.org

Log in / create account

page | discussion | view source | history

BioPerl

main links
- Main Page
- Getting Started
- Downloads
- Installation
- Recent changes
- Random page

documentation
- Quick Start
- FAQ
- HOWTOs
- API Docs
- Scrapbook
- BioPerl Tutorial
- Tutorials
- Deobfuscator

API Docs

Contents [hide]

Online POD Documentation

POD Documentation is available for bioperl-live and past releases at doc.bioperl.org.

Alternatively you can enter the module name in the search box and see any contributed Wiki documentation for the module.

Documentation from the Deobfuscator

The Deobfuscator indexes all of the BioPerl POD documentation, taking account of the inheritance tree, and then presents all of the methods available to each module through a searchable web interface.

Documentation from the CPAN

BioPerl

Perldoc (Pdoc rendered) documentation for BioPerl Modules

Released Code

Official documentation for released code is available here:

- BioPerl 1.6.0, download the entire doc set here.
- BioPerl 1.5.2, download the entire doc set here.
- BioPerl 1.5.1, download the entire doc set here.
- BioPerl 1.4, download the entire doc set here.
- BioPerl 1.2.3, download the entire doc set here.
- BioPerl 1.2.2, download the entire doc set here.
- BioPerl 1.2, download the entire doc set here.
- BioPerl 1.0.2, download the entire doc set here.
- BioPerl 1.0.1, download the entire doc set here.
- BioPerl 1.0, download the entire doc set here.

Active Code

This documentation represents the active development code and is autogenerated daily from the SVN repository:

Module	Description
bioperl-live	BioPerl Core Code
bioperl-corba-server	BioPerl BioCORBA Server Toolkit (wraps bioperl objects as BioCORBA objects and runs them in an ORBit ORB)
bioperl-corba-client	Bioperl BioCORBA Client Toolkit (wraps BioCORBA objects as bioperl objects)

All Modules TOC All

bioperl-live
bioperl-live::Bio
bioperl-live::Bio::Align
bioperl-live::Bio::AlignIO
bioperl-

PhyloNetwork
PrimarySeq
PrimarySeqI
PullParserI
Range
RangeI
SearchDist
SearchIO
Seq
SeqAnalysisParserI
SeqFeatureI
SeqI
SeqIO
SeqUtils
SimpleAlign
SimpleAnalysisI

Bio SeqIO

Summary | Included libraries | Package variables | Synopsis | Description | General documentation | Methods

Toolbar

WebCvs

Summary

Bio::SeqIO - Handler for SeqIO Formats

Package variables

Privates (from "my" definitions)

```
%valid_alphabet_cache;
$entry = 0
```

Included modules

Bio::Factory::FTLocationFactory
Bio::Seq::SeqBuilder
Bio::Tools::GuessSeqFormat
Symbol

Inherit

Bio::Factory::SequenceStreamI Bio::Root::IO Bio::Root::Root

Synopsis

Bio::SeqIO module synopsis

doc.bioperl.org

Synopsis

```
    use Bio::SeqIO;

    $in  = Bio::SeqIO->new(-file => "inputfilename" ,
                           -format => 'Fasta');
    $out = Bio::SeqIO->new(-file => ">outputfilename" ,
                           -format => 'EMBL');

    while ( my $seq = $in->next_seq() ) {
            $out->write_seq($seq);
    }

  # Now, to actually get at the sequence object, use the standard Bio::Seq
  # methods (look at Bio::Seq if you don't know what they are)

    use Bio::SeqIO;

    $in  = Bio::SeqIO->new(-file => "inputfilename" ,
                           -format => 'genbank');

    while ( my $seq = $in->next_seq() ) {
       print "Sequence ",$seq->id, " first 10 bases ",
             $seq->subseq(1,10), "\n";
    }

  # The SeqIO system does have a filehandle binding. Most people find this
```

Bio::SeqIO module description

doc.bioperl.org

Description

Bio::SeqIO is a handler module for the formats in the SeqIO set (eg, Bio::SeqIO::fasta). It is the officially sanctioned way of getting at the format objects, which most people should use.
The **Bio::SeqIO** system can be thought of like biological file handles. They are attached to filehandles with smart formatting rules (eg, genbank format, or EMBL format, or binary trace file format) and can either read or write sequence objects (Bio::Seq objects, or more correctly, Bio::SeqI implementing objects, of which Bio::Seq is one such object). If you want to know what to do with a Bio::Seq object, read **Bio::Seq**.
The idea is that you request a stream object for a particular format. All the stream objects have a notion of an internal file that is read from or written to. A particular SeqIO object instance is configured for either input or output. A specific example of a stream object is the Bio::SeqIO::fasta object.
Each stream object has functions

```
    $stream->next_seq();
```

and

```
    $stream->write_seq($seq);
```

Bio::SeqIO method list

doc.bioperl.org

Methods		
new	Description	Code
newFh	Description	Code
fh	Description	Code
_initialize	No description	Code
next_seq	Description	Code
write_seq	Description	Code
alphabet	Description	Code
_load_format_module	Description	Code
_concatenate_lines	Description	Code
_filehandle	Description	Code
_guess_format	Description	Code
DESTROY	No description	Code
TIEHANDLE	Description	Code
READLINE	No description	Code

Bio::SeqIO new method description

doc.bioperl.org

Methods description

new code next Top

```
Title    : new
Usage    : $stream = Bio::SeqIO->new(-file => $filename,
                                     -format => 'Format')
Function: Returns a new sequence stream
Returns : A Bio::SeqIO stream initialised with the appropriate format
Args     : Named parameters:
             -file => $filename
             -fh => filehandle to attach to
             -format => format

           Additional arguments may be used to set factories and
           builders involved in the sequence object creation. None of
           these must be provided, they all have reasonable defaults.
             -seqfactory   the Bio::Factory::SequenceFactoryI object
             -locfactory   the Bio::Factory::LocationFactoryI object
             -objbuilder   the Bio::Factory::ObjectBuilderI object
```

See Bio::SeqIO::Handler

Manipulation of sequences from a file

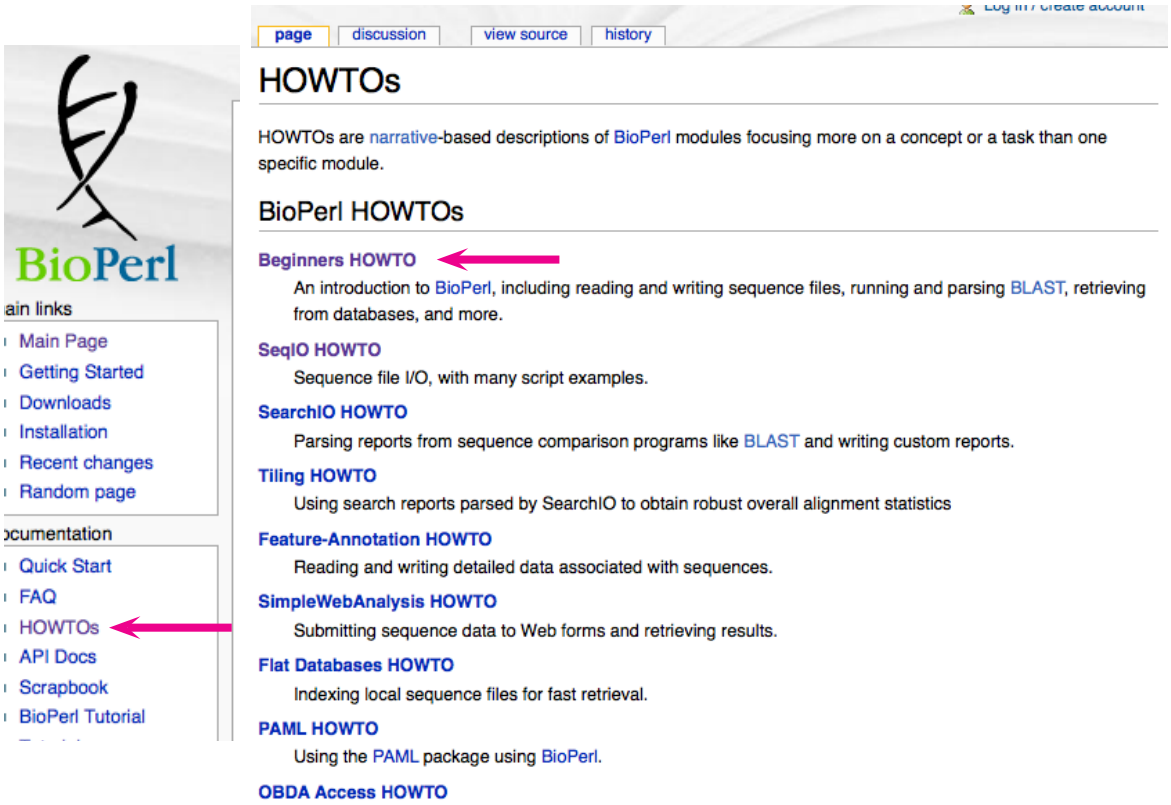
Problem:

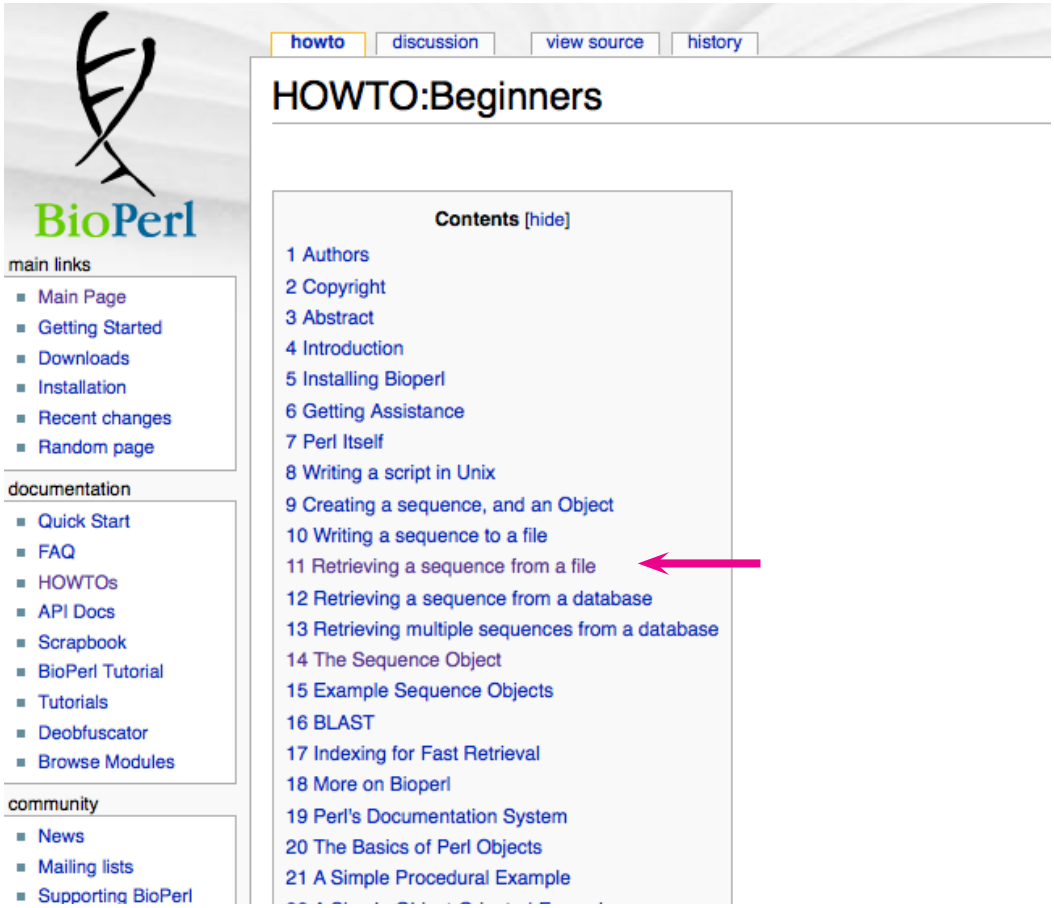
You have a sequence file and you want to do something to each sequence.

What do you do first?

HowTo:

http://www.bioperl.org/wiki/HOWTOs


page
discussion
view source
history
BioPerl
ain links
Main Page
Getting Started
Downloads
Installation
Recent changes
Random page
ocumentation
Quick Start
FAQ
HOWTOs
API Docs
Scrapbook
BioPerl Tutorial
HOWTOs
HOWTOs are narrative-based descriptions of BioPerl modules focusing more on a concept or a task than one specific module.
BioPerl HOWTOs
Beginners HOWTO
An introduction to BioPerl, including reading and writing sequence files, running and parsing BLAST, retrieving from databases, and more.
SeqIO HOWTO
Sequence file I/O, with many script examples.
SearchIO HOWTO
Parsing reports from sequence comparison programs like BLAST and writing custom reports.
Tiling HOWTO
Using search reports parsed by SearchIO to obtain robust overall alignment statistics
Feature-Annotation HOWTO
Reading and writing detailed data associated with sequences.
SimpleWebAnalysis HOWTO
Submitting sequence data to Web forms and retrieving results.
Flat Databases HOWTO
Indexing local sequence files for fast retrieval.
PAML HOWTO
Using the PAML package using BioPerl.
OBDA Access HOWTO


howto
discussion
view source
history
BioPerl
main links
Main Page
Getting Started
Downloads
Installation
Recent changes
Random page
documentation
Quick Start
FAQ
HOWTOs
API Docs
Scrapbook
BioPerl Tutorial
Tutorials
Deobfuscator
Browse Modules
community
News
Mailing lists
Supporting BioPerl
HOWTO:Beginners
Contents [hide]
1 Authors
2 Copyright
3 Abstract
4 Introduction
5 Installing Bioperl
6 Getting Assistance
7 Perl Itself
8 Writing a script in Unix
9 Creating a sequence, and an Object
10 Writing a sequence to a file
11 Retrieving a sequence from a file
12 Retrieving a sequence from a database
13 Retrieving multiple sequences from a database
14 The Sequence Object
15 Example Sequence Objects
16 BLAST
17 Indexing for Fast Retrieval
18 More on Bioperl
19 Perl's Documentation System
20 The Basics of Perl Objects
21 A Simple Procedural Example

Retrieving a sequence from a file

One beginner's mistake is to not use Bio::SeqIO when working with sequence files. This is understandable in some respects. You may have read about Perl's `open` function, and Bioperl's way of retrieving sequences may look odd and overly complicated, at first. But don't use `open`! Using open immediately forces you to do the parsing of the sequence file and this can get complicated very quickly. Trust the SeqIO object, it's built to open and parse all the common sequence formats, it can read and write to files, and it's built to operate with all the other Bioperl modules that you will want to use.

Let's read the file we created previously, "sequence.fasta", using SeqIO. The syntax will look familiar:

```
#!/bin/perl -w

use Bio::SeqIO;

$seqio_obj = Bio::SeqIO->new(-file => "sequence.fasta", -format => "fasta" );
```

One difference is immediately apparent: there is no **>** character. Just as with with the `open()` function this means we'll be reading from the "sequence.fasta" file. Let's add the key line, where we actually retrieve the Sequence object from the file using the `next_seq` method:

```
#!/bin/perl -w

use Bio::SeqIO;

$seqio_obj = Bio::SeqIO->new(-file => "sequence.fasta", -format => "fasta" );

$seq_obj = $seqio_obj->next_seq;
```

BioPerl

main links

- Main Page
- Getting Started
- Downloads
- Installation
- Recent changes
- Random page

documentation

- Quick Start
- FAQ
- HOWTOs
- API Docs
- Scrapbook
- BioPerl Tutorial

page | discussion | view source | history

HOWTOs

HOWTOs are narrative-based descriptions of BioPerl modules focusing more on a concept or a task than one specific module.

BioPerl HOWTOs

Beginners HOWTO
An introduction to BioPerl, including reading and writing sequence files, running and parsing BLAST, retrieving from databases, and more.

SeqIO HOWTO
Sequence file I/O, with many script examples.

SearchIO HOWTO
Parsing reports from sequence comparison programs like BLAST and writing custom reports.

Tiling HOWTO
Using search reports parsed by SearchIO to obtain robust overall alignment statistics

Feature-Annotation HOWTO
Reading and writing detailed data associated with sequences.

SimpleWebAnalysis HOWTO
Submitting sequence data to Web forms and retrieving results.

Flat Databases HOWTO
Indexing local sequence files for fast retrieval.

PAML HOWTO
Using the PAML package using BioPerl.

OBDA Access HOWTO

BioPerl

ain links

- Main Page
- Getting Started
- Downloads
- Installation
- Recent changes
- Random page

ocumentation

- Quick Start
- FAQ
- HOWTOs
- API Docs
- Scrapbook
- BioPerl Tutorial
- Tutorials
- Deobfuscator
- Browse Modules

mmunity

- News
- Mailing lists
- Supporting BioPerl
- BioPerl Media
- Hot Topics

howto | discussion | view source | history

HOWTO:SeqIO

This HOWTO will teach you about the Bio::SeqIO system for reading and writing sequences of various format

Contents [hide]

The basics

This section assumes you've never seen BioPerl before, perhaps you're a biologist trying to get some informa something about this hot topic, "bioinformatics". Your first script may want to get some information from a file c

A piece of advice: always use the module Bio::SeqIO! Here's what the first lines of your script might look like:

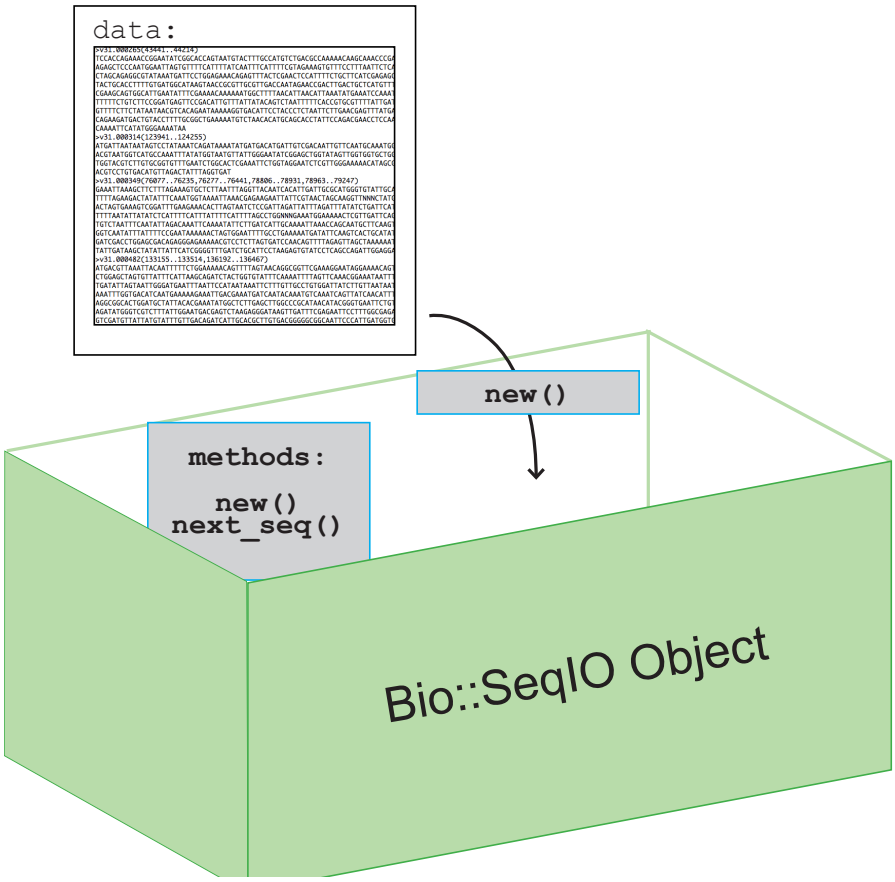
```
#!/bin/perl

use strict;
use Bio::SeqIO;

my $file = shift; # get the file name, somehow
my $seqio_object = Bio::SeqIO->new(-file => $file);
my $seq object = $seqio object->next seq;
```

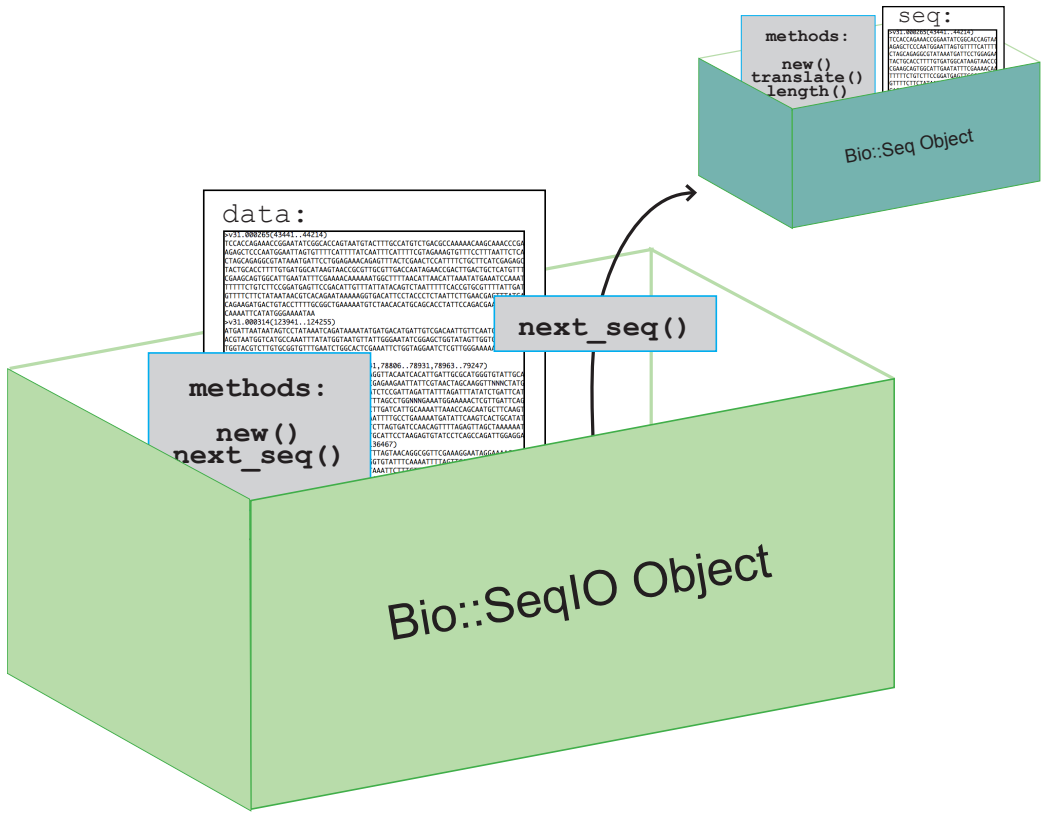
```
#!/usr/bin/perl -w
#file: inFasta_loop.pl
use strict;
use Bio::SeqIO;

my $file = shift;

my $seqIO_object = Bio::SeqIO->new(
                        -file => $file,
                        -format => 'fasta',
                    );

while (my $seq_object = $seqIO_object->next_seq){
        #do stuff to each sequence in the fasta
}
```

What is a SeqIO object?
What is a Seq object?

Objects

Objects are like boxes that hold
your data and
tools (methods) for your data






```perl
#!/usr/bin/perl -w
#file: inFasta_loop.pl
use strict;
use Bio::SeqIO;

# get fasta filename from user input
my $file = shift;

# create a SeqIO obj with $file as filename
# $seqIO_object contains all the individual sequence
#       that are in the file named $file
my $seqIO_object = Bio::SeqIO->new(
                        -file => $file,
                        -format => 'fasta',
                    );

# using while loop and next_seq method to "get to"
#   and create a Seq obj for each individual sequence
#   in the SeqIO obj of many sequences
while (my $seq_object = $seqIO_object->next_seq){
        #do stuff to each sequence in the fasta
}
```

1. Get a file name from user input (@ARGV) and stores in $file

2. Create a new seqIO object in $seqIO_object, using filename $file and format 'fasta'

3. Create a second seqIO object in $out using format 'fasta'

4. Loop thru each seq object in $seqIO_object storing information from the object in variables.

5. Print out the stored information

6. Print out $seq_object using the method or tool 'write_seq()' and the seqIO object $out.

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

my $file = shift;
my $seqIO_object = Bio::SeqIO->new(
                -file => $file,
                -format => 'fasta',
                        );
my $out_seqIO_Obj = Bio::SeqIO->new(-format => 'fasta');

while (my $seq_object = $seqIO_object->next_seq){
      my $id = $seq_object->id;
      my $desc = $seq_object->desc;
      my $seqString = $seq_object->seq;
      my $revComp = $seq_object->revcom;
      my $alphabet = $seq_object-> alphabet;
      my $translation_seq_obj   = $seq_object-> translate;
      my $translation  = $translation_seq_obj -> seq;
      my $seqLen = $seq_object->length;

      print "translation: $translation\n";
      print "alphapet: $alphabet\n";
      print "seqLen: $seqLen\n";

      #prints to STDOUT
      $out_seqIO_Obj->write_seq($seq_object);
}
```

fasta input:

```
>seqName seq description is blah blah blah
AGGCTCAATTTAGTTTTCCTTGTCCTTATTTTAAAAGGTGTCCAGTG
TGATGTGCAGCTGGTGGAGTCTGGGGGAGGCTTAGTGCAGCCTGGAG
GGTCCCGGAAACTCTCCTGTGCAGCCTCTGGATTCACTTTCAGTAGC
TTTGGAATGCACTGGGTTCGTCAGGCTCCAGAGAAGGGGCTGGAGTG
GGTCGCATACATTAGTAGTGGCAGTAGTACCCTCCACTATGCAGACA
CAGTGAAGGGCCGATTCACCATCTCAAGAGACAATCCCAAGAACACC
CTGTTCCTGCAAATGACCAGTCTAAGGTCTGAGGACACGGCCATGTA
TTACTGTGCAAGATGGGGTAACTACCCTTACTATGCTATGGACTACT
GGGGTCAA
```

output:

```
translation: RLNLVFLVLILKGVQCDVQLVESGGGLVQPGGSRKLSCAASGFTFSSF
GMHWVRQAPEKGLEWVAYISSGSSTLHYADTVKGRFTISRDNPKNTLFLQMTSLRSEDTAM
YYCARWGNYPYYAMDYWGQGTSVTVSS
alphapet: dna
seqLen: 408
>seqName seq description is blah blah blah
AGGCTCAATTTAGTTTTCCTTGTCCTTATTTTAAAAGGTGTCCAGTGTGATGTGCAGCTG
GTGGAGTCTGGGGGAGGCTTAGTGCAGCCTGGAGGGTCCCGGAAACTCTCCTGTGCAGCC
TCTGGATTCACTTTCAGTAGCTTTGGAATGCACTGGGTTCGTCAGGCTCCAGAGAAGGGG
CTGGAGTGGGTCGCATACATTAGTAGTGGCAGTAGTACCCTCCACTATGCAGACACAGTG
AAGGGCCGATTCACCATCTCAAGAGACAATCCCAAGAACACCCTGTTCCTGCAAATGACC
AGTCTAAGGTCTGAGGACACGGCCATGTATTACTGTGCAAGATGGGGTAACTACCCTTAC
TATGCTATGGACTACTGGGGTCAAGGAACCTCAGTCACCGTCTCCTCA
```

Table from http://www.bioperl.org/wiki/HOWTO:Beginners

List of seq object methods

Table 1: Sequence Object Methods

Name	Returns	Example	Note
new	Sequence object	$so = Bio::Seq->new(-seq => "MPQRAS")	create a new one, see Bio::Seq for more
seq	sequence string	$seq = $so->seq	get or set the sequence
display_id	identifier	$so->display_id("NP_123456")	get or set an identifier
primary_id	identifier	$so->primary_id(12345)	get or set an identifier
desc	description	$so->desc("Example 1")	get or set a description
accession	identifier	$acc = $so->accession	get or set an identifier
length	length, a number	$len = $so->length	get the length
alphabet	alphabet	$so->alphabet('dna')	get or set the alphabet ('dna','rna','protein')
subseq	sequence string	$string = $seq_obj->subseq(10,40)	Arguments are start and end
trunc	Sequence object	$so2 = $so1->trunc(10,40)	Arguments are start and end
revcom	Sequence object	$so2 = $so1->revcom	Reverse complement
translate	protein Sequence object	$prot_obj = $dna_obj->translate	See the Bioperl Tutorial for more
species	Species object	$species_obj = $so->species	See Bio::Species for more

#file: fn-fasta_outGenBank.pl

Change 'format' in the new() method from 'fasta' to 'genbank' to change the way the SeqIO object $out is displayed in STDOUT.

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

my $file = shift;
my $seqIO_object = Bio::SeqIO->new(
                                -file => $file,
                                -format => 'fasta',
                                );

my $out_seqIO_Obj= Bio::SeqIO->new(-format => 'genbank');

while (my $seq_object = $seqIO_object->next_seq){
      $out_seqIO_Obj>write_seq($seq_object); #prints to STDOUT
}
```

```
LOCUS       seqName                 408 bp    dna     linear   UNK
DEFINITION  seq description is blah blah blah
ACCESSION   unknown
FEATURES             Location/Qualifiers
BASE COUNT       95 a     98 c    111 g    104 t
ORIGIN
        1 aggctcaatt tagttttcct tgtccttatt ttaaaaggtg tccagtgtga tgtgcagctg
       61 gtggagtctg ggggaggctt agtgcagcct ggagggtccc ggaaactctc ctgtgcagcc
      121 tctggattca ctttcagtag ctttggaatg cactgggttc gtcaggctcc agagaagggg
      181 ctggagtggg tcgcatacat tagtagtggc agtagtaccc tccactatgc agacacagtg
      241 aagggccgat tcaccatctc aagagacaat cccaagaaca ccctgttcct gcaaatgacc
      301 agtctaaggt ctgaggacac ggccatgtat tactgtgcaa gatggggtaa ctacccttac
      361 tatgctatgg actactgggg tcaaggaacc tcagtcaccg tctcctca
//
```

Query a local fasta file

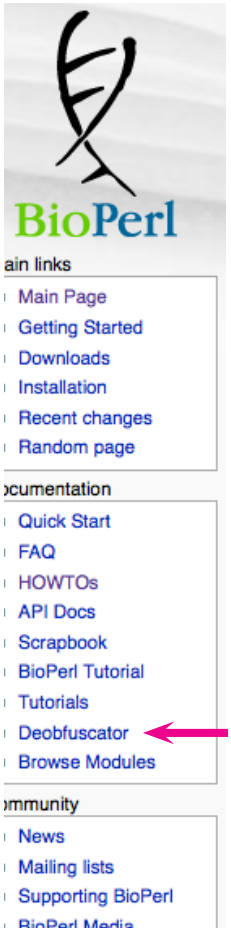
<u>Query a local fasta file</u>

You have a fasta file that contains many records.

You want to retrieve a specific record.

You do not want to loop through all records until you find the correct record.

Use Bio::DB::Fasta.



BioPerl

ain links

- Main Page
- Getting Started
- Downloads
- Installation
- Recent changes
- Random page

ocumentation

- Quick Start
- FAQ
- HOWTOs
- API Docs
- Scrapbook
- BioPerl Tutorial
- Tutorials
- Deobfuscator
- Browse Modules

ommunity

- News
- Mailing lists
- Supporting BioPerl
- BioPerl Media

Deobfuscator

Contents [hide]

What is the Deobfuscator?

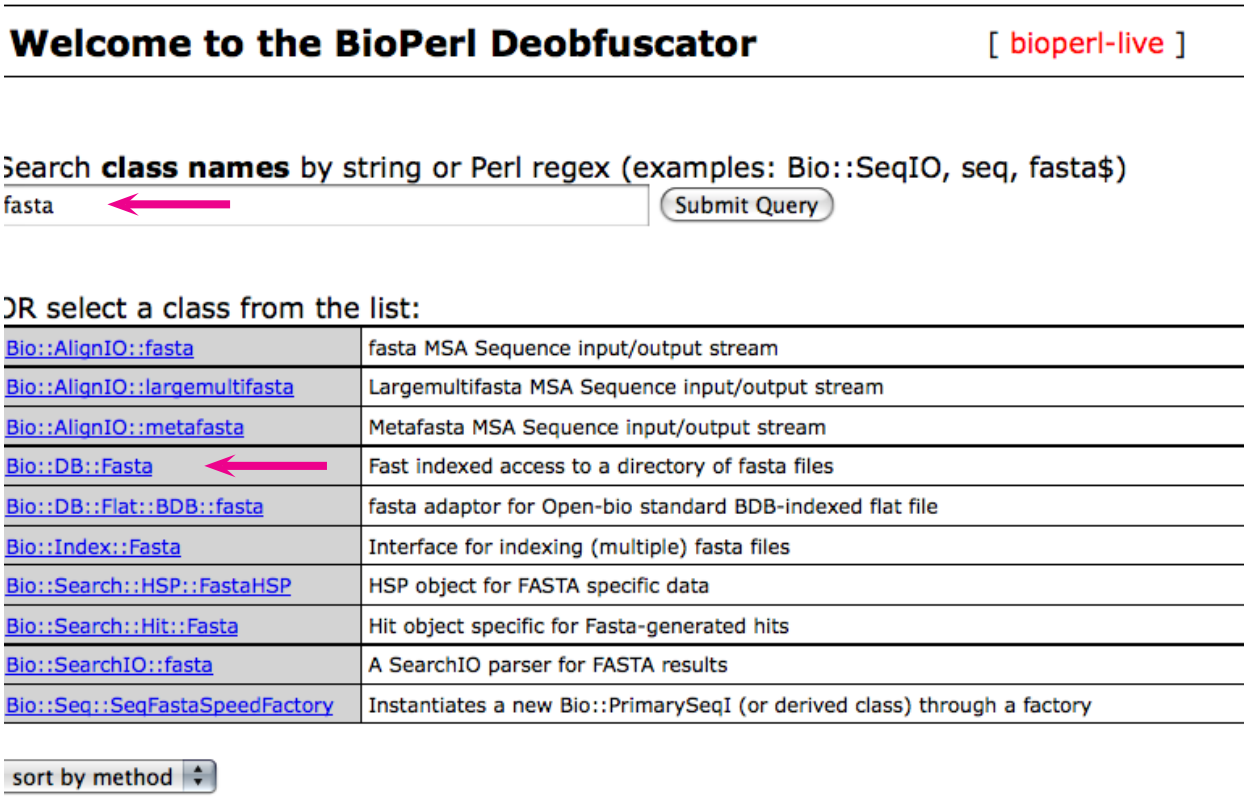
The Deobfuscator was written to make it easier to determine the methods that are available from a given BioPerl module (a common BioPerl FAQ).

BioPerl is a highly object-oriented software package, with often multiple levels of inheritance. Although each individual module is usually well-documented for the methods specific to it, identifying the inherited methods is less straightforward.

The Deobfuscator indexes all of the BioPerl POD documentation, taking account of the inheritance tree (thanks to Class::Inspector), and then presents all of the methods available to each module through a searchable web interface.

Where can I find the Deobfuscator?

The Deobfuscator is currently available here, indexing *bioperl-live*.



Welcome to the BioPerl Deobfuscator

[bioperl-live]

Search **class names** by string or Perl regex (examples: Bio::SeqIO, seq, fasta$)

fasta

Submit Query

OR select a class from the list:

Bio::AlignIO::fasta	fasta MSA Sequence input/output stream
Bio::AlignIO::largemultifasta	Largemultifasta MSA Sequence input/output stream
Bio::AlignIO::metafasta	Metafasta MSA Sequence input/output stream
Bio::DB::Fasta	Fast indexed access to a directory of fasta files
Bio::DB::Flat::BDB::fasta	fasta adaptor for Open-bio standard BDB-indexed flat file
Bio::Index::Fasta	Interface for indexing (multiple) fasta files
Bio::Search::HSP::FastaHSP	HSP object for FASTA specific data
Bio::Search::Hit::Fasta	Hit object specific for Fasta-generated hits
Bio::SearchIO::fasta	A SearchIO parser for FASTA results
Bio::Seq::SeqFastaSpeedFactory	Instantiates a new Bio::PrimarySeqI (or derived class) through a factory

sort by method

Bio::AlignIO::metafasta	Metafasta MSA Sequence input/output stream
Bio::DB::Fasta	Fast indexed access to a directory of fasta files
Bio::DB::Flat::BDB::fasta	fasta adaptor for Open-bio standard BDB-indexed flat file
Bio::Index::Fasta	Interface for indexing (multiple) fasta files
Bio::Search::HSP::FastaHSP	HSP object for FASTA specific data
Bio::Search::Hit::Fasta	Hit object specific for Fasta-generated hits
Bio::SearchIO::fasta	A SearchIO parser for FASTA results
Bio::Seq::SeqFastaSpeedFactory	Instantiates a new Bio::PrimarySeqI (or derived class) through a factory

sort by method

methods for **Bio::DB::Fasta**

Method	Class	Returns	Usage
alphabet	Bio::DB::Fasta	not documented	not documented
basename	Bio::DB::Fasta	not documented	not documented
calculate_offsets	Bio::DB::Fasta	not documented	not documented
caloffset	Bio::DB::Fasta	not documented	not documented
carp	Bio::Root::RootI	not documented	not documented
CLEAR	Bio::DB::Fasta	not documented	not documented
confess	Bio::Root::RootI	not documented	not documented
dbmargs	Bio::DB::Fasta	not documented	not documented
debug	Bio::Root::Root	none	$obj->debug("This is debugging output");

Other packages in the module: Bio::DB::Fasta Bio::PrimarySeq::Fasta

Summary | Included libraries | Package variables | Synopsis | Description

Toolbar

WebCvs

Summary

Bio::DB::Fasta -- Fast indexed access to a directory of fasta files

Package variables

No package variables defined.

Included modules

AnyDBM_File
Fcntl
File::Basename qw (basename dirname)
IO::File

Inherit

Bio::DB::SeqI Bio::Root::Root

Synopsis

```
use Bio::DB::Fasta;

# create database from directory of fasta files
my $db      = Bio::DB::Fasta->new('/path/to/fasta/files');
```

Can also find these pages at http://doc.bioperl.org/bioperl-live/

Bio::DB::fasta module synopsis

doc.bioperl.org

Synopsis

```
use Bio::DB::Fasta;

# create database from directory of fasta files
my $db      = Bio::DB::Fasta->new('/path/to/fasta/files');

# simple access (for those without Bioperl)
my $seq      = $db->seq('CHROMOSOME_I',4_000_000 => 4_100_000);
my $revseq   = $db->seq('CHROMOSOME_I',4_100_000 => 4_000_000);
my @ids      = $db->ids;
my $length   = $db->length('CHROMOSOME_I');
my $alphabet = $db->alphabet('CHROMOSOME_I');
my $header   = $db->header('CHROMOSOME_I');

# Bioperl-style access
my $db      = Bio::DB::Fasta->new('/path/to/fasta/files');

my $obj     = $db->get_Seq_by_id('CHROMOSOME_I');
my $seq     = $obj->seq; # sequence string
my $subseq  = $obj->subseq(4_000_000 => 4_100_000); # string
my $trunc   = $obj->trunc(4_000_000 => 4_100_000); # seq object
my $length  = $obj->length;
# (etc)

# Bio::SeqIO-style access
my $stream  = Bio::DB::Fasta->new('/path/to/files')->get_PrimarySeq_stream;
while (my $seq = $stream->next_seq) {
  # Bio::PrimarySeqI stuff
}
```

Bio::DB::fasta module description

doc.bioperl.org

Description

Bio::DB::Fasta provides indexed access to one or more Fasta files. It provides random access to each sequence entry, and to subsequences within each entry, allowing you to retrieve portions of very large sequences without bringing the entire sequence into memory.

When you initialize the module, you point it at a single fasta file or a directory of multiple such files. The first time it is run, the module generates an index of the contents of the file or directory using the AnyDBM module (Berkeley DB* preferred, followed by GDBM_File, NDBM_File, and SDBM_File). Thereafter it uses the index file to find the file and offset for any requested sequence. If one of the source fasta files is updated, the module reindexes just that one file. (You can also force reindexing manually). For improved performance, the module keeps a cache of open filehandles, closing less-recently used ones when the cache is full.

The fasta files may contain any combination of nucleotide and protein sequences; during indexing the module guesses the molecular type. Entries may have any line length up to 65,536 characters, and different line lengths are allowed in the same file. However, within a sequence entry, all lines must be the same length except for the last.

Bio::DB::fasta method description

doc.bioperl.org

get_Seq_by_id **code** **prev**

```
Title   : get_Seq_by_id
Usage   : my $seq = $db->get_Seq_by_id($id)
Function: Bio::DB::RandomAccessI method implemented
Returns : Bio::PrimarySeqI object
Args    : id
```

Query a local fasta file

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

my $dbfile = 'uniprot_sprot.fasta';
my $db_obj = Bio::DB::Fasta->new($dbfile);

# retrieve a sequence
my $id = 'sp|Q13547|HDAC1_HUMAN';
my $seq_obj = $db_obj->get_Seq_by_id($id);

if ( $seq_obj ) {
        print "seq: ",$seq_obj->seq,"\n";
} else {
        warn("Cannot find $id\n");
}
```

output

```
seq: MAQTQGTRRKVCYYYDGDVGNYYYGQGHPMKPHRIRMTHNLLLNYGLYRKMEIYRPHKANAE
EMTKYHSDDYIKFLRSIRPDNMSEYSKQMQRFNVGEDCPVFDGLFEFCQLSTGGSVASAVKLNKQQT
DIAVNWAGGLHHAKKSEASGFCYVNDIVLAILELLKYHQRVLYIDIDIHHGDGVEEAFYTTDRVMTV
SFHKYGEYFPGTGDLRDIGAGKGKYYAVNYPLRDGIDDESYEAIFKPVMSKVMEMFQPSAVVLQCGS
DSLSGDRLGCFNLTIKGHAKCVEFVKSFNLPMLMLGGGGYTIRNVARCWTYETAVALDTEIPNELPY
NDYFEYFGPDFKLHISPSNMTNQNTNEYLEKIKQRLFENLRMLPHAPGVQMQAIPEDAIPEESGDED
EDDPDKRISICSSDKRIACEEEFSDSEEEGEGGRKNSSNFKKAKRVKTEDEKEKDPEEKKEVTEEEK
```

Creating a sequence record

<u>Creating a sequence record</u>

You have a sequence and want to create a Seq object on the fly.

Use Bio::Seq.

Create a sequence record on the fly.

#file:createSeqOnFly.pl

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

my $seqObj = Bio::Seq->new(-seq => 'ATGAATGATGAA',
                  -display_id => 'seq_example',
                  -description=> 'this seq is awesome');

my $out_seqIO_Obj = Bio::SeqIO->new(-format => 'fasta');
$out_seqIO_Obj->write_seq($seqObj);

print "Id: ",$seqObj->display_id, "\n";
print "Length: ", $seqObj->length, "\n";
print "Seq: ",$seqObj->seq,"\n";
print "Subseq (3..6): ", $seqObj->subseq(3,6), "\n";
print "Translation: ", $seqObj->translate->seq, "\n";
```

1. Create a new seq object

2. Create and print a new seqIO object in fasta format using $seqObj

3. Get features of $seqObj by using seqObj methods

Notice the coupling of methods.

Output

```
>seq_example this seq is awesome
ATGAATGATGAA
Id: seq_example
Length: 12
Seq: ATGAATGATGAA
Subseq (3..6): GAAT
Translation: MNDE
```

File format conversions

<u>File format conversions</u>

You have GenBank files and want to extract only the sequence in fasta format.

Use Bio::SeqIO.

Formats

BioPerl's SeqIO system understands lot of formats and can interconvert all of them. Here is a current listing of formats, as of version 1.5.

Table 1: Bio::SeqIO modules and formats supported

Name	Description	File extension	Module
abi	ABI tracefile	ab[i1]	Bio::SeqIO::abi
ace	Ace database	ace	Bio::SeqIO::ace
agave	AGAVE XML		Bio::SeqIO::agave
alf	ALF tracefile	alf	Bio::SeqIO::alf
asciitree	write-only, to visualize features		Bio::SeqIO::asciitree
bsml	BSML, using XML::DOM	bsml	Bio::SeqIO::bsml
bsml_sax	BSML, using XML::SAX		Bio::SeqIO::bsml_sax
chadoxml	CHADO sequence format		Bio::SeqIO::chadoxml
chaos	CHAOS sequence format		Bio::SeqIO::chaos
chaosxml	Chaos XML		Bio::SeqIO::chaosxml
ctf	CTF tracefile	ctf	Bio::SeqIO::ctf

http://www.bioperl.org/wiki/HOWTO:SeqIO

```
OCUS       MUSIGHBA1                408 bp    mRNA    linear   ROD 27-APR-1993
EFINITION  Mouse Ig active H-chain V-region from MOPC21, subgroup VH-II,
           mRNA.
CCESSION   J00522
ERSION     J00522.1  GI:195052
EYWORDS    constant region; immunoglobulin heavy chain; processed gene; variable re-
ion; variable region subgroup VH-II.
OURCE      Mus musculus (house mouse).
 ORGANISM  Mus musculus
           Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
           Mammalia; Eutheria; Euarchontoglires; Glires; Rodentia;
           Sciurognathi; Muroidea; Muridae; Murinae; Mus.
EFERENCE   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
           antibodies: somatic mutation evident in a gamma 2a variable region
 JOURNAL   Cell 24 (3), 625-637 (1981)
  PUBMED   6788376
OMMENT     Original source text: Mouse C57Bl/6 myeloma MOPC21, cDNA to mRNA,
           clone pAB-gamma-1-4. [1] studies the response in C57Bl/6 mice to
           NP proteins. It is called the b-NP response because this mouse
           strain carries the b-IgH haplotype. See other entries for b-NP
           response for more comments.
EATURES             Location/Qualifiers
    source          1..408
                    /db_xref="taxon:10090"
                    /mol_type="mRNA"
                    /organism="Mus musculus"
    CDS             <1..>408
                    /db_xref="GI:195055"
                    /codon_start=1
                    /protein_id="AAD15290.1"
                    /translation="RLNLVFLVLILKGVQCDVQLVESGGGLVQPGGSRKLSCAASGFT
                    FSSFGMHWVRQAPEKGLEWVAYISSGSSTLHYADTVKGRFTISRDNPKNTLFLQMTSL
                    RSEDTAMYYCARWGNYPYYAMDYWGQGTSVTVSS"
                    /note="Ig H-chain V-region from MOPC21"
    sig_peptide     <1..48
    mat_peptide     49..>408
                    /product="Ig H-chain V-region from MOPC21 mature peptide"
    misc_recomb     343..344
                    /note="V-region end/D-region start (+/- 1bp)"
    misc_recomb     356..357
                    /note="D-region end/J-region start"
ASE COUNT       95 a     98 c    111 g    104 t
RIGIN      57 bp upstream of PvuII site, chromosome 12.
       1 aggctcaatt tagttttcct tgtccttatt ttaaaaggtg tccagtgtga tgtgcagctg
      61 gtggagtctg ggggaggctt agtgcagcct ggagggtccc ggaaactctc ctgtgcagcc
     121 tctggattca ctttcagtag ctttggaatg cactgggttc gtcaggctcc agagaagggg
     181 ctggagtggg tcgcatacat tagtagtggc agtagtaccc tccactatgc agacacagtg
     241 aagggccgat tcaccatctc aagagacaat cccaagaaca ccctgttcct gcaaatgacc
     301 agtctaaggt ctgaggacac ggccatgtat tactgtgcaa gatggggtaa ctacccttac
     361 tatgctatgg actactgggg tcaaggaacc tcagtcaccg tctcctca
```

= GenBank Format

Fasta Format

```
>MUSIGHBA1 Mouse Ig active H-chain V-region from MOPC21,
subgroup VH-II, mRNA.
AGGCTCAATTTAGTTTTCCTTGTCCTTATTTTAAAAGGTGTCCAGTGTGATGTGCAGCTG
GTGGAGTCTGGGGGAGGCTTAGTGCAGCCTGGAGGGTCCCGGAAACTCTCCTGTGCAGCC
TCTGGATTCACTTTCAGTAGCTTTGGAATGCACTGGGTTCGTCAGGCTCCAGAGAAGGGG
CTGGAGTGGGTCGCATACATTAGTAGTGGCAGTAGTACCCTCCACTATGCAGACACAGTG
AAGGGCCGATTCACCATCTCAAGAGACAATCCCAAGAACACCCTGTTCCTGCAAATGACC
AGTCTAAGGTCTGAGGACACGGCCATGTATTACTGTGCAAGATGGGGTAACTACCCTTAC
TATGCTATGGACTACTGGGGTCAAGGAACCTCAGTCACCGTCTCCTCA
```

Convert from GenBank to fasta.

#file:convert_genbank2fasta.pl

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

my ($informat,$outformat) = ('genbank','fasta');
my ($infile,$outfile) = @ARGV;

my $in_seqIO_Obj = Bio::SeqIO->new(
        -format => $informat,
        -file => $infile,
        );
my $out_seqIO_Obj = Bio::SeqIO->new(
        -format => $outformat,
        -file => ">$outfile"
        );

while ( my $seqObj = $in_seqIO_Obj->next_seq ) {
    $out_seqIO_Obj->write_seq($seqObj);
}
```

Retrieving annotations

Retrieving annotations

You have GenBank files and want to retrieve annotations.

Use Bio::SeqIO.

Sample GenBank file with Features/Annotations

```
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 re-
gion; 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
            antibodies: somatic mutation evident in a gamma 2a variable region
  JOURNAL   Cell 24 (3), 625-637 (1981)
   PUBMED   6788376
COMMENT     Original source text: Mouse C57Bl/6 myeloma MOPC21, cDNA to mRNA,
            clone pAB-gamma-1-4. [1] studies the response in C57Bl/6 mice to
            NP proteins. It is called the b-NP response because this mouse
            strain carries the b-IgH haplotype. See other entries for b-NP
            response for more comments.
FEATURES             Location/Qualifiers
     source          1..408
                     /db_xref="taxon:10090"
                     /mol_type="mRNA"
                     /organism="Mus musculus"
     CDS             <1..>408
                     /db_xref="GI:195055"
                     /codon_start=1
                     /protein_id="AAD15290.1"
                     /translation="RLNLVFLVLILKGVQCDVQLVESGGGLVQPGGSRKLSCAASGFT
                     FSSFGMHWVRQAPEKGLEWVAYISSGSSTLHYADTVKGRFTISRDNPKNTLFLQMTSL
                     RSEDTAMYYCARWGNYPYYAMDYWGQGTSVTVSS"
                     /note="Ig H-chain V-region from MOPC21"
     sig_peptide     <1..48
     mat_peptide     49..>408
                     /product="Ig H-chain V-region from MOPC21 mature peptide"
     misc_recomb     343..344
                     /note="V-region end/D-region start (+/- 1bp)"
     misc_recomb     356..357
                     /note="D-region end/J-region start"
BASE COUNT       95 a     98 c    111 g    104 t
ORIGIN      57 bp upstream of PvuII site, chromosome 12.
        1 aggctcaatt tagttttcct tgtccttatt ttaaaaggtg tccagtgtga tgtgcagctg
       61 gtggagtctg ggggaggctt agtgcagcct ggagggtccc ggaaactctc ctgtgcagcc
      121 tctggattca ctttcagtag ctttggaatg cactgggttc gtcaggctcc agagaagggg
      181 ctggagtggg tcgcatacat tagtagtggc agtagtaccc tccactatgc agacacagtg
      241 aagggccgat tcaccatctc aagagacaat cccaagaaca ccctgttcct gcaaatgacc
      301 agtctaaggt ctgaggacac ggccatgtat tactgtgcaa gatggggtaa ctacccttac
```

```
FEATURES             Location/Qualifiers
     source          1..408
                     /db_xref="taxon:10090"
                     /mol_type="mRNA"
                     /organism="Mus musculus"
     CDS             <1..>408
                     /db_xref="GI:195055"
                     /codon_start=1
                     /protein_id="AAD15290.1"
                     /translation="RLNLVFLVLILKGVQCDVQLVESGGGLVQPGGSRKLSCAASGFT
                     FSSFGMHWVRQAPEKGLEWVAYISSGSSTLHYADTVKGRFTISRDNPKNTLFLQMTSL
                     RSEDTAMYYCARWGNYPYYAMDYWGQGTSVTVSS"
                     /note="Ig H-chain V-region from MOPC21"
     sig_peptide     <1..48
     mat_peptide     49..>408
                     /product="Ig H-chain V-region from MOPC21 mature peptide"
     misc_recomb     343..344
                     /note="V-region end/D-region start (+/- 1bp)"
     misc_recomb     356..357
                     /note="D-region end/J-region start"
```

primary_tag

tag=value

Get annotations from a GenBank file

#file: get_annot_from_genbank.pl

```
use strict;
use Bio::SeqIO;

my $infile = shift;
my $seqIO = Bio::SeqIO->new(
        -file => $infile,
        -format => 'genbank',
    );
while (my $seqObj = $seqIO -> next_seq){
    my $name = $seqObj -> id;
    foreach my $feature_obj ($seqObj->get_SeqFeatures){
        my $primary_tag = $feature_obj->primary_tag;
        my ($start, $end) = ($feature_obj->start , $feature_obj->end);
        my $range = $start . ".." . $end;
        foreach my $tag ( sort $feature_obj->get_all_tags ) {
            my @values = $feature_obj->get_tag_values($tag);
            my $value_str = join ",", @values;
            print "$name($range)\t$primary_tag\t$tag:$value_str\n";
        }
    }
}
```

get_SeqFeature produces an array of Bio::SeqFeatureI objects

Output

```
MUSIGHBA1(1..408)        source   db_xref:taxon:10090
MUSIGHBA1(1..408)        source   mol_type:mRNA
MUSIGHBA1(1..408)        source   organism:Mus musculus
MUSIGHBA1(1..408)        CDS      codon_start:1
MUSIGHBA1(1..408)        CDS      db_xref:GI:195055
MUSIGHBA1(1..408)        CDS      note:Ig H-chain V-region from MOPC21
MUSIGHBA1(1..408)        CDS      protein_id:AAD15290.1
MUSIGHBA1(1..408)        CDS      translation:RLNLVFLVLILKGVQCDVQLVESGGGLVQPGGSRKLSCAASGFTFSSF
GMHWVRQAPEKGLEWVAYISSGSSTLHYADTVKGRFTISRDNPKNTLFLQMTSLRSEDTAMYYCARWGNYPYYAMDYWGQGTSVTVSS
MUSIGHBA1(49..408)       mat_peptide     product:Ig H-chain V-region from MOPC21 mature pep-
tide
MUSIGHBA1(343..344)      misc_recomb     note:V-region end/D-region start (+/- 1bp)
```

Manipulating Multiple Alignments

Use Bio::AlignIO

for parsing and writing multiple alignment file formats including:

fasta, phylip, nexus, clustalw, msf, mega, meme, pfam, psi, selex, stockholm.

Convert from fasta_aln to nexus

#file: multi_align_convert.pl

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

my $align_fasta = shift;
my $in_alignIO_obj = Bio::AlignIO->new(
                    -format => 'fasta',
                    -file => $align_fasta
                    );
my $out_alignIO_obj = Bio::AlignIO->new(
                    -format => 'nexus',
                    -file => ">$align_fasta.nex"
                    );
while( my $align_obj = $in_alignIO_obj->next_aln ){
        $out_alignIO_obj->write_aln($align_obj);
}
```

ıext_aln produces a
Bio::SimpleAlign object

Bio::SimpleAlign Object

Remove some sequences and rewrite the result

Extract or remove columns

Calculate consensus string and percent identity

Parsing BLAST Output

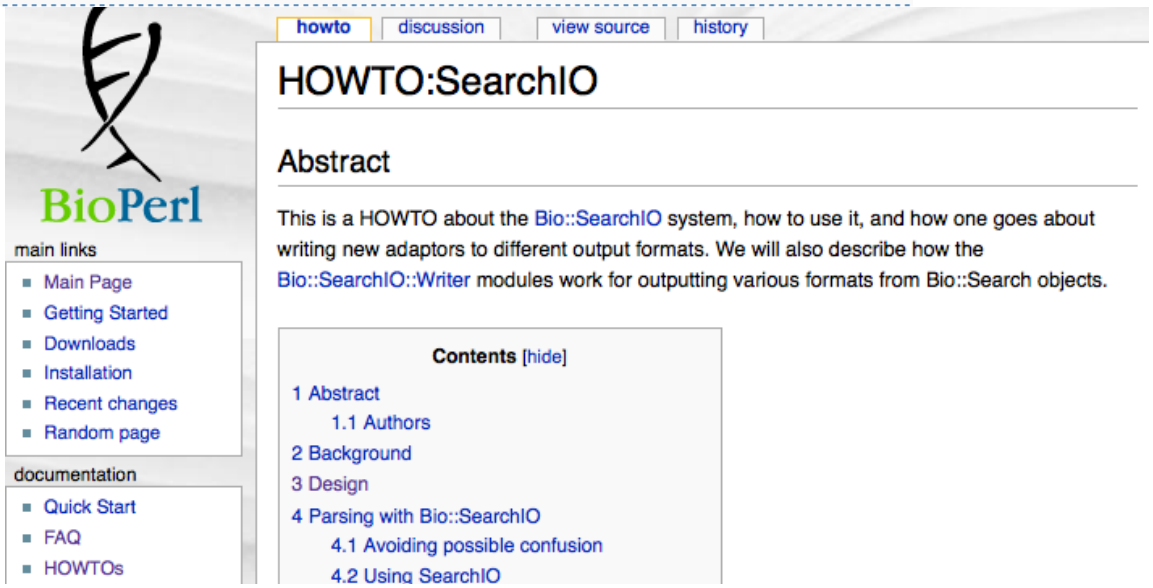
Parsing BLAST reports

Use Bio::SearchIO

Where do you start?

Here's an example of how one would use SearchIO to extract data from a BLAST report:

```
use Bio::SearchIO;
$report_obj = new Bio::SearchIO(-format => 'blast',
                                -file   => 'report.bls');
while( $result = $report_obj->next_result ) {
    while( $hit = $result->next_hit ) {
      while( $hsp = $hit->next_hsp ) {
         if ( $hsp->percent_identity > 75 ) {
           print "Hit\t", $hit->name, "\n", "Length\t", $hsp->length('total'),
                 "\n", "Percent_id\t", $hsp->percent_identity, "\n";
         }
       }
     }
}
```



NCBI BLAST Report

Result

```
BLASTX 2.2.12 [Aug-07-2005]


Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= smed-HDAC1-1
         (1213 letters)

Database: swissprot.aa
           427,028 sequences; 157,875,145 total letters

Searching..................................................done
```

Hit

```
                                                          Score    E
Sequences producing significant alignments:              (bits) Value

sp|P56517|HDAC1_CHICK RecName: Full=Histone deacetylase 1; Short...   535   e-151
```

HSP

```
>sp|P56517|HDAC1_CHICK RecName: Full=Histone deacetylase 1; Short=HD1
          Length = 480

 Score =  535 bits (1379), Expect = e-151
 Identities = 255/343 (74%), Positives = 292/343 (85%), Gaps = 1/343 (0%)
 Frame = +3

Query: 3    CPVFDGLFEFCQLSAGGSVASAVKLNKNKADIAINWSGGLHHAKKSEASGFCYVNDIVMG 182
            CPVFDGLFEFCQLSAGGSVASAVKLNK + DIA+NW+GGLHHAKKSEASGFCYVNDIV+
Sbjct: 100  CPVFDGLFEFCQLSAGGSVASAVKLNKQQTDIAVNWAGGLHHAKKSEASGFCYVNDIVLA 159

Query: 183  ILELLKYHERVLYVDIDIHHGDGVEEAFYTTDRVMTVSFHKYGEYFPXXXXXXXXXXXXX 362
            ILELLKYH+RVLY+DIDIHHGDGVEEAFYTTDRVMTVSFHKYGEYFP
Sbjct: 160  ILELLKYHQRVLYIDIDIHHGDGVEEAFYTTDRVMTVSFHKYGEYFPGTGDLRDIGAGKG 219

Query: 363  XNYAVNFPLRDGIDDESYESIFKPVVEKVIESFKPNAIVLQCGADSLSGDRLGCFNLSLK 542
             YAVN+PLRDGIDDESYE+IFKPV+ KV+E+F+P+A+VLQCG+DSLSGDRLGCFNL++K
Sbjct: 220  KYYAVNYPLRDGIDDESYEAIFKPVISKVMETFQPSAVVLQCGSDSLSGDRLGCFNLTIK 279

Query: 543  GHGKCVEYMRQQPIPLLMLGGGGYTIRNVARCWTYETALALGTTIPNELPYNDYYEYFTP 722
            GH KCVE+++   +P+LMLGGGGYTIRNVARCWTYETA+AL T IPNELPYNDY+EYF P
Sbjct: 280  GHAKCVEFVKSFNLPMLMLGGGGYTIRNVARCWTYETAVALDTEIPNELPYNDYFEYFGP 339

Query: 723  DFKLHISPSNMANQNTPEYLERMKQKLFENLRSIPHAPSVQMQDIPEDAMDIDDGEQMDN 902
            DFKLHISPSNM NQNT EYLE++KQ+LFENLR +PHAP VQMQ IPEDA+  D G++ +
Sbjct: 340  DFKLHISPSNMTNQNTNEYLEKIKQRLFENLRMLPHAPGVQMQPIPEDAVQEDSGDE-EE 398

Query: 903  ADPDKRISILASDKYREHEADLSDSEDEGD-NRKNVDCFKSKR 1028
             DP+KRISI  SDK    + + SDSEDEG+  RKNV  FK  +
Sbjct: 399  EDPEKRISIRNSDKRISCDEEFSDSEDEGEGGRKNVANFKKAK 441
```

Result

```
 Database: /common/data/swissprot.aa
    Posted date:  Oct 4, 2009  2:02 AM
  Number of letters in database: 157,875,145
  Number of sequences in database:  427,028

Lambda     K      H
   0.318    0.134    0.401

Gapped
Lambda     K      H
   0.267   0.0410    0.140


Matrix: BLOSUM62
Gap Penalties: Existence: 11, Extension: 1
Number of Hits to DB: 281,587,467
Number of Sequences: 427028
Number of extensions: 5577736
Number of successful extensions: 16223
Number of sequences better than 1.0e-10: 1
Number of HSP's better than  0.0 without gapping: 15290
Number of HSP's successfully gapped in prelim test: 0
Number of HSP's that attempted gapping in prelim test: 0
Number of HSP's gapped (non-prelim): 16078
length of database: 157,875,145
effective HSP length: 119
effective length of database: 107,058,813
effective search space used: 30404702892
frameshift window, decay const: 40,  0.1
T: 12
A: 40
X1: 16 ( 7.3 bits)
X2: 38 (14.6 bits)
X3: 64 (24.7 bits)
```

Bookmark it!!

See

http://www.bioperl.org/wiki/HOWTO:SearchIO

for a GREAT example of a blast report,

code to parse it,

a table of methods,

and the values the methods return.

Bio::SearchIO object for BLAST reports

```
#!/usr/bin/perl -w
use strict;
use Bio::SearchIO;
#file: blast_parser_intro.pl

my $blast_report = shift;

my $searchIO_obj = Bio::SearchIO->new(
                    -file => $blast_report,
                    -format => 'blast'
                   );
```

Result object and methods

#file: sample_Blast_parser_1.pl

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

my $blast_report = shift;

my $searchIO_obj = Bio::SearchIO->new(
                    -file => $blast_report,
                    -format => 'blast'
                    );

while (my $result_obj = $searchIO_obj ->next_result ) {
      my $program = $result_obj ->algorithm;
      my $queryName = $result_obj ->query_name;
      my $queryDesc = $result_obj ->query_description;
      my $queryLen = $result_obj ->query_length;
      print "program=$program\tqueryName=$queryName\t";
      print "queryDesc=$queryDesc\tqueryLen=$queryLen\n";
}
```

Output:

```
program=BLASTX  queryName=smed-HDAC1-1  queryDesc=histone deacetylase 1 queryLen=1213
```

http://www.bioperl.org/wiki/HOWTO:SearchIO

Object	Method	Example	Description
Result	algorithm	BLASTX	algorithm string
Result	algorithm_version	2.2.4 [Aug-26-2002]	algorithm version
Result	query_name	20521485|dbj|AP004641.2	query name
Result	query_accession	AP004641.2	query accession
Result	query_length	3059	query length
Result	query_description	Oryza sativa ... 977CE9AF checksum.	query description
Result	database_name	test.fa	database name
Result	database_letters	1291	number of residues in database
Result	database_entries	5	number of database entries
Result	available_statistics	effectivespaceused ... dbletters	statistics used
Result	available_parameters	gapext matrix allowgaps gapopen	parameters used
Result	num_hits	1	number of hits
Result	hits		List of all Bio::Search::Hit::GenericHit object(s) for this Result
Result	rewind		Reset the internal iterator that dictates where next_hit() is pointing, useful for re-iterating through the list of hits.

Hit object and methods

#file: sample_Blast_parser_2.pl

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

my $blast_report = shift;

my $searchIO_obj = Bio::SearchIO->new(
                    -file => $blast_report,
                    -format => 'blast'
                    );

while (my $result_obj = $searchIO_obj->next_result ) {
    while (my $hit_obj = $result_obj->next_hit){
        my $hitName = $hit_obj->name;
        my $hitAcc = $hit_obj->accession;
        my $hitLen = $hit_obj->length;
        my $hitSig = $hit_obj->significance;
        my $hitScore = $hit_obj->raw_score;

        print "hitName=$hitName\thitAcc=$hitAcc\thitLen=$hitLen\t";
        print "hitSig=$hitSig\thitScore=$hitScore\n";
    }
}
```

must get hit objects from a result object

Output:

hitName=sp|P56517|HDAC1_CHICK hitAcc=P56517 hitLen=480 hitSig=1e-151 hitScore=535

http://www.bioperl.org/wiki/HOWTO:SearchIO

Hit	name	443893|124775	hit name
Hit	length	331	Length of the Hit sequence
Hit	accession	443893	accession (usually when this is a genbank formatted id this will be an accession number- the part after the *gb* or *emb*)
Hit	description	LaForas sequence	hit description
Hit	algorithm	BLASTX	algorithm
Hit	raw_score	92	hit raw score
Hit	significance	2e-022	hit significance
Hit	bits	92.0	hit bits
Hit	hsps		List of all Bio::Search::HSP::GenericHSP object(s) for this Hit
Hit	num_hsps	1	number of HSPs in hit
Hit	locus	124775	locus name
Hit	accession_number	443893	accession number
Hit	rewind		Resets the internal counter for next_hsp() so that the iterator will begin at the beginning of the list

HSP object and methods

#file: sample_Blast_parser.pl

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

use Bio::SearchIO;

my $blast_report = shift;

my $searchIO_obj = Bio::SearchIO->new(
                    -file => $blast_report,
                    -format => 'blast'
                    );

while (my $result_obj = $searchIO_obj->next_result ) {
    while (my $hit_obj = $result_obj->next_hit){
        while (my $hsp_obj = $hit_obj ->next_hsp){
            my $evalue = $hsp_obj->evalue;
            my $hitString = $hsp_obj->hit_string;
            my $queryString = $hsp_obj->query_string;
            my $homologyString = $hsp_obj->homology_string;

            print "hsp evalue: $evalue\n";
            print "HIT     : ",substr($hitString,0,50),"\n";
            print "HOMOLOGY: ",substr($homologyString,0,50),"\n";
            print "QUERY   : ",substr($queryString,0,50),"\n";
        }
    }
}
```

must get hsp objects from a hit object

Output:

```
hsp evalue: 1e-151
HIT     : CPVFDGLFEFCQLSAGGSVASAVKLNKQQTDIAVNWAGGLHHAKKSEASG
HOMOLOGY: CPVFDGLFEFCQLSAGGSVASAVKLNK + DIA+NW+GGLHHAKKSEASG
```

http://www.bioperl.org/wiki/HOWTO:SearchIO

HSP	algorithm	BLASTX	algorithm
HSP	evalue	2e-022	e-value
HSP	expect	2e-022	alias for evalue()
HSP	frac_identical	0.884615384615385	Fraction identical
HSP	frac_conserved	0.923076923076923	fraction conserved (conservative and identical replacements aka "fraction similar") (only valid for Protein alignments will be same as frac_identical)
HSP	gaps	2	number of gaps
HSP	query_string	DMGRCSSG ..	query string from alignment
HSP	hit_string	DIVQNSS ...	hit string from alignment
HSP	homology_string	D+ SSGCN	string from alignment
HSP	length('total')	52	
HSP	length('hit')	50	
HSP	length('query')	15	
HSP	hsp_length	52	
HSP	frame	0	
HSP	num_conserved	48	
HSP	num_identical	46	
HSP	rank	1	
HSP	seq_inds('query','identical')	(96	
HSP	seq_inds('query','conserved-not-identical')	(96	

HSP	seq_inds('query','conserved')	(966,967,969,971,973,974,975, ...)	conserved or identical positions as array
HSP	seq_inds('hit','identical')	(197,202,203,204,205, ...)	identical positions as array
HSP	seq_inds('hit','conserved-not-identical')	(198,200)	conserved not identical positions as array
HSP	seq_inds('hit','conserved',1)	(197,202-246)	conserved or identical positions as array, with runs of con
HSP	score	227	score
HSP	bits	92.0	score in bits
HSP	range('query')	(2896,3051)	start and end as array
HSP	range('hit')	(197,246)	start and end as array
HSP	percent_identity	88.4615384615385	% identical
HSP	strand('hit')	1	strand of the hit
HSP	strand('query')	1	strand of the query
HSP	start('query')	2896	start position from alignment
HSP	end('query')	3051	end position from alignment
HSP	start('hit')	197	start position from alignment
HSP	end('hit')	246	end position from alignment
HSP	matches('hit')	(46,48)	number of identical and conserved as array
HSP	matches('query')	(46,48)	number of identical and conserved as array
HSP	get_aln	*sequence alignment*	Bio::SimpleAlign object
HSP	hsp_group	*Not available in this report*	Group field from WU-BLAST reports run with -topcombo

Other Cool Things

Whole set of wrappers for running Bioinformatics tools in bioperl-run

Run BLAST locally or submit remote jobs (through NCBI)

Run PAML - handles setup and take down of temporary files and directories

Run alignment progs through similar interfaces: TCoffee, MUSCLE, Clustalw

Relational Databases for sequence and features

Repository of scripts to do really cool things. (http://www.bioperl.org/wiki/Scripts)

How do you find all the available methods?

From the BioPerl HowTo we know to use Bio::SearchIO?

What Next????

“Follow the Objects” Using CPAN

Start by Searching for Bio::SearchIO

Read the methods, find the one that seems the best

Christopher Fields > BioPerl-1.6.901 > Bio::SearchIO

Module Version: 1.006901 Source

- NAME
- SYNOPSIS
- DESCRIPTION
- FEEDBACK
  - Mailing Lists
  - Support
  - Reporting Bugs
- AUTHOR - Jason Stajich & Steve Chervitz
- APPENDIX
  - new
  - newFh
  - fh
  - attach_EventHandler
  - eventHandler
  - next_result
  - write_result
  - write_report
  - writer

Remeber BioPerl How To uses next_result

next_result Returns another object. What methods belong to the Bio::Search::Result::ResultI object?

next_result

```
Title   : next_result
Usage   : $result = stream->next_result
Function: Reads the next ResultI object from the stream and returns it.

          Certain driver modules may encounter entries in the stream that
          are either misformatted or that use syntax not yet understood
          by the driver. If such an incident is recoverable, e.g., by
          dismissing a feature of a feature table or some other non-mandatory
          part of an entry, the driver will issue a warning. In the case
          of a non-recoverable situation an exception will be thrown.
          Do not assume that you can resume parsing the same stream after
          catching the exception. Note that you can always turn recoverable
          errors into exceptions by calling $stream->verbose(2) (see
          Bio::Root::RootI POD page).
Returns : A Bio::Search::Result::ResultI object
Args    : n/a
```

CPAN

Home · Authors · Recent · News · Mir

Bio::Search::Result::ResultI

in All CPAN Search

Results 1 - 10 of 43 Found

1 · 2 · 3 · 4 · 5 · Next >>

Bio::Search::Result::ResultI
Abstract interface to Search Result objects
BioPerl-1.6.901 - 18 May 2011 - Christopher Fields

Bio::Search::Result::GenericResult
Generic Implementation of Bio::Search::Result::ResultI interface applicable to most search re
BioPerl-1.6.901 - 18 May 2011 - Christopher Fields

Bio::Search::Result::PullResultI

CPAN

Home · Authors · Recent

in All CPAN Sea

Christopher Fields > BioPerl-1.6.901 > Bio::Search::Result::F

Module Version: 1.006901 Source

- NAME
- SYNOPSIS
- DESCRIPTION
- FEEDBACK
  - Mailing Lists
  - Support
  - Reporting Bugs
- AUTHOR
- COPYRIGHT
- DISCLAIMER
- APPENDIX
  - next_hit
  - sort_hits
  - _default sort_hits
  - query_name
  - query_accession
  - query_length
  - query_description
  - database_name
  - database_letters
  - database_entries

next_hit returns an object. What methods belong to Bio::Search::Hit::HitI object?

next_hit

```
Title   : next_hit
Usage   : while( $hit = $result->next_hit()) { ... }
Function: Returns the next available Hit object, representing potential
          matches between the query and various entities from the database.
Returns : a Bio::Search::Hit::HitI object or undef if there are no more.
Args    : none
```

CPAN

Home · Authors · Recent · News ·

Bio::Search::Hit::HitI

in All CPAN Search

Results 1 - 10 of 28 Found

1 · 2 · 3 · Next >>

Bio::Search::Hit::HitI

Interface for a hit in a similarity search result

BioPerl-1.6.901 - 18 May 2011 - Christopher Fields

Bio::Search::Hit::GenericHit

A generic implementation of the Bio::Search::Hit::HitI interface

BioPerl-1.6.901 - 18 May 2011 - Christopher Fields

CPAN

Home · Authors · Recent · News

in All CPAN Search

Christopher Fields > BioPerl-1.6.901 > Bio::Search::Hit::HitI

Module Version: 1.006901 Source

next_hsp returns an object. What methods belong to Bio::Search::HSP::HSPI object?

next_hsp

```
Title    : next_hsp
Usage    : while( $hsp = $obj->next_hsp()) { ... }
Function : Returns the next available High Scoring Pair
Example  :
Returns  : A <Bio::Search::HSP::HSPI> object or null if finished
Args     : none
```

hsps

CPAN

Home · Authors · Recent · News · Mir

in All CPAN Search

Christopher Fields > BioPerl-1.6.901 > Bio::Search::HSP::HSPI

Module Version: 1.006901 Source

Yea!! hit_string returns a string, not an object. Done!!

hit_string

```
Title   : hit_string
Usage   : my $hseq = $hsp->hit_string;
Function: Retrieves the hit sequence of this HSP as a string
Returns : string
Args    : none
```