Bio::SearchIO HO

modify his BPlite (*B*last *P*arser) Bio::Tools::BPlite module into Bioperl. This is of course in a sea of BLAST parsers that have been written by numerous people, but we will only cover the ones associated directly with the Bioperl project in this document. One of the reasons for writing yet another BLAST parser in the form of Bio::SearchIO is that even though both Bio::Tools::Blast and Bio::Tools::BPlite did their job correctly, and could parse WU-BLAST and NCBI-BLAST output, they did not adequately genericize what they were doing. By this we mean everything was written around the BLAST format and was not easily applicable to parsing, say, FastA alignments or a new alignment format. One of the powerful features of the Object-Oriented framework in Bioperl is the ability to read in, say, a sequence file in different formats or from different data sources like a database or XML-flatfile, and have the program code process the sequences objects in the same manner. We wanted to have

```
                ",Length=",     $hsp->length('total'),
                ",Percent_id=", $hsp->percent_identity, "\n";
          }
        }
      }
    }
  }
```

The example above shows just a fe

```
   0.267   0.0410    0.140

Matrix: BLOSUM62
```

On one hand it appears to be a complication, but by entering the worlds of the AlignIO and SimpleAlign objects you now hav

Our simple table of methods does not show all available arguments or returned values for all the SearchIO methods. The best place to explore any method in detail is *http://doc.bioperl.org* which provides the HTML versions of the Perl POD (Plain Old Documentation) that is embedded in every well-written Perl module - there's also a list of modules at the bottom of this HOWTO. Other sources of code are the e

one sends a start event, then some data, then an end event. This process is analagous to a finite state machine in

The simplest way to output data in HTML format is as follows.

```
my $writerhtml = new Bio::SearchIO::Writer::HTMLResultWriter();
my $outhtml = new Bio::SearchIO(-writer => $writerhtml,
-file   => ">searchio.html");
# get a result from Bio::SearchIO parsing or build it up in memory
$outhtml->write_result($result);
```

If you wanted to get the output as a staTm(a thrt tan >writd it oud toa)Tj/54 10 Tf()Tj/F4 10 Tf(le,e simpy uset the follo)Tj1 0 0 14

```
      my ($class,@args) = @_;
      my $self = $class->SUPER::new(@args); # chained contructor

      # process the 1 additional argument this object supports
      my ($ownarg1) = $self->_rearrange([OWNARG1],@args);

      return $self; # remember to pass the object reference back out
   }

   sub realign_hsp {
      my ($self) = @_;
      # implement my special realign method here
   }
```

The above code gives you a skeleton of how to start to implement your object. To register it so that it is used when the SearchIO system makes HSPs you just need to call a couple of functions. The code below outlines them.

```
use Bio::SearchIO;
use Bio::Search::HSP::HSPFactory;
use Bio::Search::Hit::HitFactory;

# setup the blast parser, you can do this with and SearchIO parser however
my $searchio = new Bio::SearchIO(-file => $blastfile,
                                 -format =>'blast');
# build HSP factory with a certain type of HSPs to make
# the default is Bio::Search::HSP::GenericHSP
my $hspfact = new Bio::Search::HSP::HSPFactory(-type =>
                'Bio::Search::HSP::RealignHSP');
# if you wanted to replace the Hit factory you can do this as well
# additionally there is an analagous
# Bio::Search::Result::ResultFactory for setting custom Result objects
my $hitfact = new Bio::Search::Hit::HitFactory(-type =>
```

Here is an example of such a lightweight listener - FastHitEventBuilder - it just throws away the HSPs and only builds Result and Hit objects.

```
use Bio::SearchIO;
use Bio::SearchIO::FastHitEventBuilder;

my $searchio = new Bio::SearchIO(-format => $format, -file => $file);

$searchio->attach_EventHandler(new
Bio::SearchIO::FastHitEventBuilder);

while( my $r = $searchio->next_result ) {
  while( my $h = $r->next_hit ) {
    # Hits will NOT have HSPs
    print $h->significance,"\n";
  }
}
```

You could also build your o

S e a r c h I O / W r i t e r / H i t T a b l e W r i t e r . p m
[http://doc.bioperl.org/releases/bioperl-1.4/Bio/SearchIO/Writer/HitTableWriter.html]
S e a r c h I O / W r i t e r / H S P T