Introduction to Bioperl

Gaolin Zheng

Feb. 14th, 2006

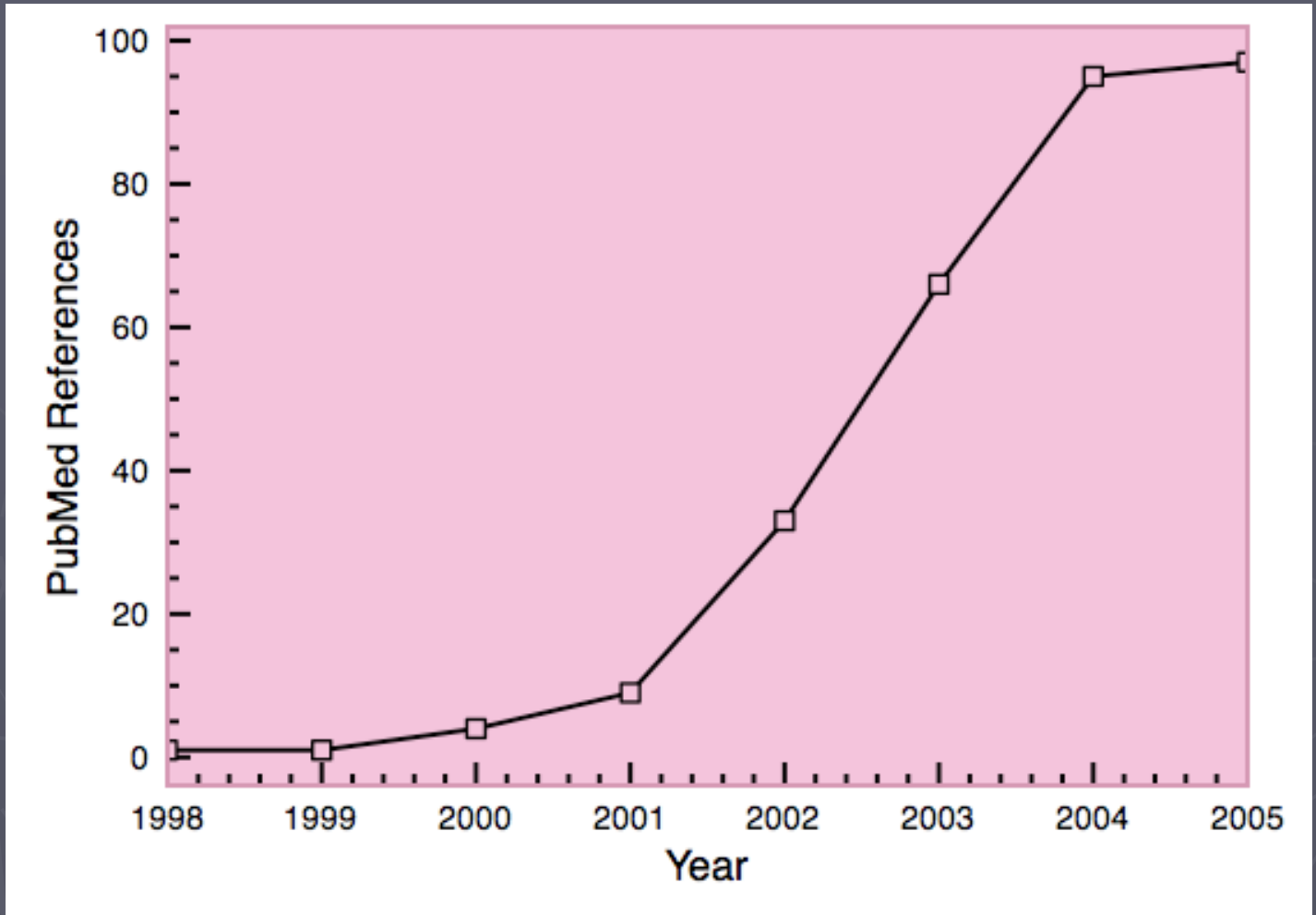
What is Bioperl?

- BioPerl is a tookit of perl modules useful in building bioinformatics solutions in Perl.
- BioPerl is open-source.

A Brief History of Bioperl

- 1996, Germany. Award-winning VSNS-BCD biocomputing courses.
- 2000, release 0.7.
- 2002, Bioperl 1.0

How Widely is Bioperl Being Used?



What Can You Do with Bioperl?

- Accessing sequence data from local and remote databases
- Transforming formats of database/ file records
- Manipulating individual sequences
- Searching for similar sequences
- Creating and manipulating sequence alignments
- Searching for genes and other structures on genomic DNA
- Developing machine readable sequence annotations

Major Bioperl Objects

- Sequence objects
- Location objects
- Interface objects
- Implementation objects

Sequence Objects

- PrimarySeq object
  - Stripped downed sequence object for efficiency
- RichSeq object
  - With annotations
- LocatableSeq object
  - It is a Seq object which is part of a multiple sequence align.
- LargeSeq object
  - Special type of Seq object used for handling very long sequences (e.g. > 100 MB).
- ……

Location Objects

- designed to be associated with a Sequence Feature object in order to show where the feature is on a longer sequence.
- Can also be standalone objects used to described positions.
- Complications
  - Some objects have multiple locations or sub-locations (e.g. a gene's exons may have multiple start and stop locations).
  - In unfinished genomes, the precise locations of features is not known with certainty.

Interface and Implementation Objects

- An interface is solely the definition of what methods one can call on an object, without any knowledge of how it is implemented.
- An implementation is an actual, working implementation of an object.

Our First Example

Convert a swissprot file to a fasta file

```
#!/usr/bin/perl
use strict;
use Bio::SeqIO;
# This program convert a swiss prot file to a fasta file

my $in = Bio::SeqIO->newFh(-file => '<data/seqs.sp',
                           -format => 'swiss');

my $out = Bio::SeqIO->newFh(-file => '>seqs.fasta',
                            -format => 'fasta');

print $out $_ while<$in>;

exit;
```

Our Second Example

Load a sequence from a remote server

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

$database = new Bio::DB::SwissProt;

$seq = $database->get_Seq_by_id('MALK_ECOLI');

my $out = Bio::SeqIO->newFh(-fh => \*STDOUT,
                            -format => 'fasta');

print $out $seq;

exit;
```

Our Third Example

Read an alignment file using AlignIO class

```
#!/usr/bin/perl -w
use strict;
use Bio::AlignIO;
my $in = new Bio::AlignIO(-file => '<data/infile.aln',
                          -format => 'clustalw');

# returns an alignI (alignment interface)
my $aln = $in->next_aln();
print "same length of all sequences: ",
($aln->is_flush()) ? "yes" : "no", "\n";
print "alignment length: ", $aln->length, "\n";
printf "identity: %.2f %%\n", $aln->percentage_identity();
printf "identity of conserved columns: %.2f %%\n",
$aln->overall_percentage_identity();
```

Our Fourth Example
A standalone blast

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
use Bio::Tools::Run::StandAloneBlast;
my $seq_in = Bio::SeqIO -> new(-file => '<data/prot1.fasta',
                               -format => 'fasta');
my $query = $seq_in -> next_seq();
my $factory = Bio::Tools::Run::StandAloneBlast -> new('program' => 'blastp',
                                                      'database' => 'swissprot',
                                                      _READMETHOD => 'Blast');
my $blast_report = $factory->blastall($query);
my $result = $blast_report->next_result();
while (my $hit = $result->next_hit()){
  print "\thit name: ", $hit->name(), "Significance: ", $hit->significance(), "\n";
  while (my $hsp = $hit->next_hsp()){
    print "E: ", $hsp->evalue(), "frac_identical: ", $hsp->frac_identical(), "\n";
  }

}
exit;
```