# BIOPERL

## *About BIOPERL*

Bioperl is a collection of more than 500 Perl modules for bioinformatics that have been written and maintained by an international group of volunteers. BioPerl is free (under a very unrestrictive copyright), and its home is *http://www.bioperl.org*.

One of the most difficult things about BioPerl is getting started using it. This is due to a scarcity of good documentation (which is being rectified) as well as the sheer size of the BioPerl module library. The modules in BioPerl are written in the object-oriented style. Perl programmers who do not know object-oriented programming can still use the BioPerl modules with just a bit of extra information.

The BioPerl modules cover various areas of bioinformatics. Although BioPerl includes some example programs, it is not meant to be a collection of complete user-ready programs. Rather, it's implemented as a toolkit you can dip into for help when writing your own programs. Its goal is to provide good working solutions to common bioinformatics tasks and to speed your program development.

One of the best things about BioPerl is that it's an open source project, meaning that interested developers are invited to contribute by writing code or in other ways, and the code is available to anyone interested.

## *The Growth of BIOPERL*

The practice of freely distributing Perl software for bioinformatics over the Internet began around 1992. Gradually, Perl became more and more popular for biology applications. The release of Perl Version 5 and its support for object-oriented programming accelerated the development of reusable modules for biology at many research centers around the world.

The large-scale genome sequencing efforts, then underway, provided much of the impetus, as well as the talent and funding, for these efforts. The BioPerl project, officially organized in 1995, coalesced around one of these bioinformatics research groups that was doing a good job of organizing the collective volunteer effort such collaborative projects require. The BioPerl web site is frequently being updated and improved, and is the primary source of BioPerl code and documentation.

Today, the BioPerl project has grown to a point where it is both useful enough, and well enough documented, that it is a must for Perl programming in bioinformatics. The documentation includes a program `bptutorial.pl` that comes with BioPerl, which explains and demonstrates several areas of the project.

## *BIOPERL Problems*

BioPerl is still a work in progress, and it has some problems. The two main problems are:

First, the BioPerl documentation is incomplete. In fact, until fairly recently, there was no document that provided a tutorial introduction to the project. This has changed; the `bptutorial.pl` document is an excellent beginning, despite its occasional errors. This document cleverly combines a tutorial with quite a few example programs that you can run. Other documentation for BioPerl is also available, including Internet-based tutorials, forthcoming books, example programs, and journal articles. So, the situation has recently improved.

Second, BioPerl is big (over 500 modules), written by volunteers, and gradually evolving. The size of the project is a sign that BioPerl addresses many interesting and useful problems, but it also means that, for the new user of BioPerl, an overview of the available resources is a task in itself.

The majority of the BioPerl code is quite good, especially the most-used parts of it. However, the volunteer and evolving nature of BioPerl development means that some of the code is unfinished and not as well integrated with other parts of the project as one would like. Newer or less used modules may still need some shaking out by users in real-world situations. This is where you can make an initial contribution to the project: as you find problems, report them.

Many of the computing world's most successful programs are the result of the same kind of volunteer development as BioPerl (the Perl language itself and the Apache web server are two examples). BioPerl is well positioned to achieve a similarly central position in the field of bioinformatics.

## *Overview of Objects in the Module of BIOPERL (Applications)*

BioPerl is a big project and a fairly large collection of modules used in Bioinformatics. Some of these modules are standalone; others interact with each other in various ways.

Your first task in learning about BioPerl is to get an idea of the main subject areas the modules are designed to address. So to begin with, here is a brief overview of the main types of objects in BioPerl, collected in a few broadly defined groups.

*Sequences*

1. `Bio::Seq` is the main sequence object in BioPerl.
2. `Bio::PrimarySeq` is a sequence object without features.
3. `Bio::SeqIO` provides sequence file input and output.
4. `Bio::Tools::SeqStats` provides statistics on a sequence.
5. `Bio::LiveSeq::*` handles changing sequences.
6. `Bio::Seq::LargeSeq` provides support for very large sequences.

*Databases*

1. `Bio::DB::GenBank` provides GenBank access. Similar modules are available for several biological databases.
2. `Bio::Index::*` indexing and accessing local databases.
3. `Bio::Tools::Run::StandAloneBlast` runs BLAST on your local computer.
4. `Bio::Tools::Run::RemoteBlast` runs BLAST remotely.
5. `Bio::Tools::BPlite` parses BLAST reports.
6. `Bio::Tools::BPpsilite` parses `psiblast` reports.
7. `Bio::Tools::HMMER::Results` parses HMMER hidden Markov model results.

*Alignments*

1. `Bio::SimpleAlign` manipulates and displays simple multiple sequence alignments.
2. `Bio::UnivAln` manipulates and displays multiple sequence alignments.
3. `Bio::LocatableSeq` are sequence objects with start and end points for locating relative to other sequences or alignments.
4. `Bio::Tools::pSW` aligns two sequences with the Smith-Waterman algorithm.

5. `Bio::Tools::BPbl2seq` is a lightweight BLAST parser for pairwise sequence alignment using the BLAST algorithm.
6. `Bio::AlignIO` also aligns two sequences with BLAST.
7. `Bio::Clustalw` is an interface to the Clustalw multiple sequence alignment package.
8. `Bio::TCoffee` is an interface to the TCoffee multiple sequence alignment package.
9. `Bio::Variation::Allele` handles sets of alleles.
10. `Bio::Variation::SeqDiff` handles sets of mutations and variants.

*Features and genes on sequences*

1. `Bio::SeqFeature` is the sequence feature object in BioPerl.
2. `Bio::Tools::RestrictionEnzyme` locates restriction sites in sequence.
3. `Bio::Tools::Sigcleave` finds amino acid cleavage sites.
4. `Bio::Tools::OddCodes` rewrites amino acid sequences in abbreviated codes for specific statistical analysis (e.g., a hydrophobic/hydrophilic two-letter alphabet).
5. `Bio::Tools::SeqPattern` provides support for regular expression descriptions of sequence patterns.
6. `Bio::LocationI` provides an interface to location information for a sequence.
7. `Bio::Location::Simple` handles simple location information for a sequence, both as a single location and as a range.
8. `Bio::Location::Split` provides location information where the location may encompass multiple ranges, and even multiple sequences.
9. `Bio::Location::Fuzzy` provides location information that may be inexact.
10. `Bio::Tools::Genscan` is an interface to the gene finding program.
11. `Bio::Tools::Sim4::Results` (and Exon) is an interface to the gene exon finding program.
12. `Bio::Tools::ESTScan` is an interface to the gene finding program.
13. `Bio::Tools::MZEF` is an interface to the gene finding program.
14. `Bio::Tools::Grail` is an interface to the gene finding program.
15. `Bio::Tools::Genemark` is an interface to the gene finding program.
16. `Bio::Tools::EPCR` parses the output of ePCR program.

## *An example program using BIOPERL*

```
#!E:/Perl/bin/perl.exe
use Bio::Seq;
use Bio::SeqIO;
system('cls');
print "\n\n\n\n\n\n\n\n\n\n";
$sequence="GCGATGCGAGGATGA";
$seq = Bio::Seq->new( -id => "my_sequence", -seq  => $sequence ,-type => 'dna');
# the actual sequence is here
print "Your query sequence is $sequence\n\n";
# make a reverse of sequence
$seq_rev=reverse($sequence);
print "Reverse of your query sequence is $seq_rev\n\n";
# make a reverse complement sequence
$rev = $seq->revcom();
$actual_bases = $rev->seq();
print "Reverse Complement of your query sequence is $actual_bases\n\n";
# make a translation
$trans = $seq->translate();
# we could also write it as raw formatted output
$output = Bio::SeqIO->new('-format' => 'raw', -fh => \*STDOUT);
print "Translated sequence is ";
$output->write_seq($trans);
```