The Bioperl Project: A look ahead

Jason E Stajich
Program in Genetics,
Departnent of Molecular Genetics & Microbiology
Duke University, Durham, NC
jason@bioperl.org

1 August 2002

State of the Onion Ring

- Bioperl 1.0 stable series released (currently 1.0.2)
  - 400+ modules, 3200+ tests
  - Collaboration of 30+ developers over 7 yrs
- 1.1 Dev Release, August,
- 1.2 slated for late Fall 2002
- Active project
  - 10-12k unique website visitor per month, 90k hits per month
  - Average of 250 bioperl-l messages per month
  - Average of 325 bioperl-guts-l msgs per month since Jan 2001 ($>$ 90% are commits)

Areas Covered

- Sequences, Features: Bio::Seq, Bio::SeqFeatureI
- Sequence file parsing: Bio::SeqIO
- Sequence Databases: Bio::DB::GenBank, Bio::DB::EMBL
- Local Indexed Sequence DBs: Bio::Index::Fasta, Bio::Index::EMBL
- Pairwise aligmment parsing: Bio::SearchIO, Bio::Search
- Multiple Sequence Alignmengs: Bio::AlignIO, Bio::SimpleAlign, Bio::Align
- Phylogenetic Trees: Bio::Tree, Bio::TreeIO
- Bibliographic data and DBs: Bio::Biblio, Bio::Annotation
- Sequence Variations: Bio::Variation, Bio::LiveSeq
- Maps and Markers: Bio::Map
- Sequence Clustes: Bio::Clutser
- DAS & Feature Database : Bio::Das, Bio::DB::GFF (SQL)

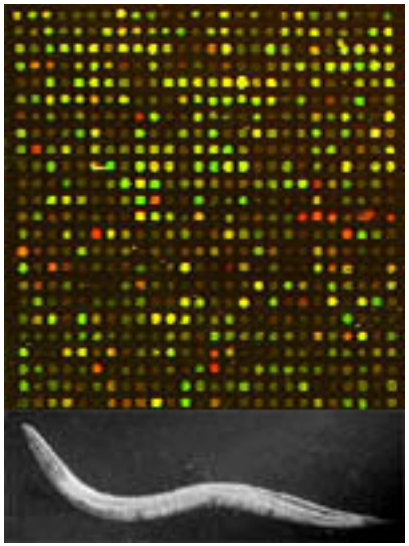
- Protein Structure parsing: Bio::Structure
- Sequence Feature Graphical Rendering: Bio::Graphics
- Common analyses application wrappes: Bio::Tools::Run
- Miscellaneous Result Parsing and Tools: Bio::Tools

So... Is the Toolkit feature complete?

No way! Many many exciting areas of activity

- Annotation Database integration
- Taxonomy
- Full DAS implementation, Easy UI
- Population Genetics
- Genotype & Phenotype information
- Expression data
- Protein Structure
- Proteomics
- Comparative methods
- Ontologies
- INSERT YOUR FAVORITE PET PROJECT HERE

So what else is next: This is the post-genomic era!!


Let's get started, its time for the post-genomics, meta-super-omic age.

Of course, don't forget about the parsing

"My biggest recurring issue has been the unreliability of genbank record parsing"

"The bioperl could offer better module to read NCBI or EMBL files"

"genbank parsing–the parser looks good so far, but it would be great if there was a way to parse those (obnoxious) records that refer to other accession numbers"

"BioPerl support for the NCBI-friendly retrieval of large numbers of genbank records and stronger parsing of genbank records. I would like to be able to read in a genbank record with SeqIO, and then write it out again in genbank format and have the before and after records diff cleanly. "

Or the documentation

- "I think the online documentation is somewhat lacking content, structure and user-friendliness. Some modules and sparingly documented, especially for first-time users, like Bio::SearchIO. The inclusion of more code examples would be more helpful for beginners."
- "If it's possible to organise the documentation a bit more effectively that would be great - no single document seems to have all the information needed to do any particular thing. E.g., for blasting you need the tutorial, example scripts, module documentation, pasteur institute course etc How about a beginners mailing list?"
- "...documentation needs work though"
- "Try to make it as easy as possible for mere mortals to use."

Or the installation

- "More support for Windows!"
- "I have spent the last day and a half trying to get the bioperl 1.0 version installed on my IRIX system."
- "Installing BioPerl and getting all the dependancies correct seems to be a black art. I have been working for a whole day to get BioPerl to install and pass all its test - so far no cigar. It really needs simplyfying"

Bioperl survey: 100+ responses to date

- The Good
  - We're making people's lives easier
  - People want to contribute code to the project
  - The quality of the modules is good enough for production work
  - Bugs are fixed quite quickly, releases are very clean overall

The Bad

- The Bad
  - It could be easier to install
  - Documentation gets about a B-/C+ from most people
  - Most users are not comfortable with reading an API
  - Tutorial on how to extend the toolkit is not always clear

The Ugly

- Things still aren't intuitive to many people!
- Consistency of method names, interfaces is not always clear
- It is hard for lots of people to contribute because they are scared of **Object Oriented Perl** and that the **Bioperl way** seems too complicated (Hint: it's not, we'll see later on).

Survey: Warm fuzzies

- "The development team is doing a great job. Most useful things are normally coded hundreds of times and almost always badly. Bioperl is really a great tool for avoiding that in bioinformatics. It has already saved me hundreds of hours (and probably saved me from a nervous breakdown while parsing EMBL files). I have found the object model and the code relatively easy to understand and use which is not the case for many packages. Thank you all."
- "...better then sliced bread, and I love it! But, I have been having trouble finding out what is available"
- "If you could put more description for each function with a short mini example, that would be really helpful. Since this is open source, this might be hard."
- "You guys did a great job!" "Outstanding Work."
- "Keep up the great work!! You make a lot of people's jobs easier."

Survey: User wishlist

- A tree module that can handle gene ontology
- Phylogenetics and molecular evolution
- Ties to clinical/phenotypic data
- Structural genomics
- Gene expression
- SRS Heuristic algorithms to link molecules from different databases.
- Pathway Database Structure
- Glyco-ingormatics: representations/visualization of glycans
- RNA Secondary Structure Prediction.
- Access to FPC data

The push and pull of Open Source development

- We have "customers", but we're not a company
- The more people that use the toolkit, the more useful it becomes (if you contribute ideas,bug reports,patches back...)
- All contributions are volunteers
- Fixing a bug for someone is a learning process, but it takes up your time
- 100% of reason the project is where it is: PRIDE

Managing an open source project

- Good leadership from the Core developers, give feedback, insure code is audited, write code, fix bugs no one else will.
- Deputize and promote people who know what they are doing to manage a section of the project
- Insure users get feedback. Answer messages on the mailing list. Mailing list also gives developers feedback.
- Working code wins. Working code for someone's project always wins.
- Code review to insure consistency across project.

Proposals to realize some of this

- Project charter to define what the project is about, how one gets access to contribute code, define the Core's role, how one becomes a Core developer.
- Use RFCs for large sets of modules/projects, request that people describe what they are doing, give prototype examples
- Assign roles for release master, auditors, documenters, web content manager, web designers. Give people credit in the documentation, on the website for doing this.
- Keep track of modules that have been code reviewed. Give people a place to mark what code they have reviewed.
- The same for bugs and bug fixers.
- The same for documenters and doc fixers.
- Give these people appropriate credit. Start a hall of fame webboard listing who has done what.

What's the plan for documentation?

- Website redesign. Make it easier for novices to find the basic. The API docs are not cutting it. More intro, overview docs.
- Move the biodesign.pod and other overview documents to an easy to find place.
- HOWTOs (http://bioperl.org/HOWTOs)
  - Similar to Linux Documentation Project (LDP)
  - User narratives, code reviews, lengthy module tutorials
  - Collected in one spot
  - Keep them up to date...
- Tutorial - continue to expand, perhaps modularize into segments. Make it into bite-sized chunks for people. Pasteur tutorial is good example another tutorial.

Writing a Bioperl module

Myth: Writing modules for Bioperl requires black magic, or Saint status at Perlmonks.

Truth: Writing a Bioperl module is just like writing any Perl module, just try and use our existing structure for exceptions,errors, inheritance, etc.

- All interfaces inherit from Bio::Root::RootI
- All objects inherit from an interface and Bio::Root::Root
- Use the shared constructor

```
sub new {
    my ($class,@args) = @_;
    my $self = $class->SUPER::new(@args);
}
```

- Make it general if you can: OMIM parser should talk about Phenotypes not just OMIM terms, Unigene parser should talk about sequence clusters not just Unigene clusters.

- Write tests! Follow design in t/XX.t directory
- It is an iterative process. Don't fret about making it perfect the first time. This is Perl.
- Try and use an OO design - if this is scary describe the basic things you want to do with your data.
- Seek feedback from the list. Few problems are unique or never seen before.
- Reuse existing Bioperl modules for the domains where it is appropriate (i.e. Don't invent a new Sequence object unless absolutely necessary)

New modules, a walkthrough: SearchIO

```
# Find all the BLAST hits which are at least 75% similar and
# 100 aa long
use Bio::SearchIO;
use Bio::DB::GenBank;
use Bio::SeqIO;
my @hits;
my $in = new Bio::SearchIO(-format => 'blast',
           -file   => 'report.bls');
while(my $result = $in->next_result ) {
  while( my $hit = $result->next_result ) {
     while( my $hsp = $hit->next_hsp ) {
       if( $hsp->length > 100 && $hsp->percent_identity >= 75) {
           push @hits, $hit->name();
            print join(', ', ("Query=". $result->query_name,
```

```
                                "hit=".$hit->name,
                                "pid=". $hsp->percentage_identity,
                                "length=". $hsp->length('hsp'),
                                "qlen= ". $result->query_length,
                                "hlen= ". $hit->length)), "\n";
                last;
        }
      }
    }
}
# Retrieve their sequences from the database
my $db = new Bio::DB::GenBank();
my $hitfile = new Bio::SeqIO(-file => '>hitfile', -format => 'fasta');
foreach my $hit ( @hits ) {
        my $hitseq = $db->get_Seq_by_acc($hit);
        if( $hitseq ){ $hitfile->write_seq($hitseq) }
        else { print STDERR "Cannot find seq $hit at GenBank\n"}
}
```

Really new modules, a walkthrough: ProtML

```
use Bio::Tools::Run::Phylo::Molphy::ProtML;
use Bio::AlignIO;
# Do this for lots of genes!!
my $protml = Bio::Tools::Run::Phylo::Molphy::ProtML
                 (-models => 'jtt',
                  -search => 'quick',
                  -other  => [ '-information', '-w' ]);
# sub in Bio::Tools::Run::Alignment::Clustalw instead
# to run the alignment now
my $alnin = new Bio::AlignIO(-format => 'clustalw',
                             -file   => 'cluster.aln');
my $aln = $in->next_aln;
$protml->alignment($aln);
my ($rc,$results) = $protml->run();
```

```
my $res = $results->next_result;
my $t = $res->next_tree;

my @group = ( $t->find_node('Human'), # find node with this name
              $t->find_node('Fly'),
              $t->find_node('Mouse'));
my $outgrp  = $t->find_node('Worm');
if( $tree->is_monophyletic(-nodes   => \@group,
                           -outgroup=> $outgrp ) {
    print "Group is monophyletic\n";
} else {
    print "Group is not monophyletic\n";
}
```

Eventually we'll have the same interface for nucml, ProtPars, njdist, maybe even PAUP.

Useful tidbits

```
# reimplement Bill Pearson's mrtrans
use Bio::AlignIO;
use Bio::SeqIO;
use Bio::Align::Utilities qw(aa_to_dna_aln);
my $in = new Bio::AlignIO(-format => 'clustalw', -file => 'prot.aln');
my $protaln = $in->next_aln;
my $seqio = new Bio::SeqIO(-format => 'fasta', -file => 'dnaseqs.fa');
my %dna;
while( my $seq = $seqio->next_seq() ) {
$dna{$seq->id} = $seq; # id needs to be the same as that in prot.aln
       # do transformations here if necessary
}
my $dna_aln = aa_to_dna_aln($protaln,\%dna);
my $out = new Bio::AlignIO(-format=> 'clustalw');
$out->write_aln($dna_aln);
```

Coming Attractions

(One of these could be your baby if you want it!) Not necessarily 1.2 material

- Bio::Functional
  - Expression objects with integrated Ontology data { Microarray, SAGE, Northern }.
  - Objects like: Probe, Target, Result, Experiment, Library, Tag.
  - Steal liberally from existing projects or put interface on top.
- Ontologies & DAG objects deal with all the new ontologies
- Pathways
- Maps {FPC, Marker Based}
- Assembly {AGP, Alignment}
- Phenotype information, integrate with Pedigree information
- Relation Database access and storage of these various objects
- Code audit, crusy corner cleanup

Acknowledgements

Bioperl Core (Hilmar Lapp, Ewan Birney, Heikki Lehväslaiho, Lincoln Stein)

Chris Dagdigian

http://bioperl.org/Core/Latest/AUTHORS

O|B|F

Wyeth®