Lessons Learned Integrating Open Source Software in a Commercial Life Sciences Workflow Product

Dr. Scott Markel

Principal Bioinformatics Architect

smarkel@scitegic.com

Overview

- Community effort
- Testing and bug fixes
- Operating systems
- Cascading changes
- Sequence data issues
- Program parameters and error messages
- Irritating little things
- Conclusion

Bioinformatics and Open Source Tools/Databases

- Many open source and publicly available tools and databases
- Address both the common and specialized needs of our community
- Include *de facto* standards like NCBI's GenBank file format and BLAST program
- Our customers expect access to all of these

So what happens when a commercial life sciences workflow product makes heavy use of open source and publicly available programs?

Community Effort

- Open source communities have been especially helpful during our development efforts
- Open source advantages are well-known, including
  - Mailing lists that provide quick answers
  - Software that has been tested, i.e., common things have been tested by many people (uncommon things may not have been tested much at all)
  - Code that can be modified locally, if necessary
- Sometimes the community is very small, e.g., André Blavier and EMBOSSwin

Testing and Bug Fixes

- Customers' product expectations: stability and professional quality
  - even when the third party tools we use aren't
- We sometimes need to correct a bug in our version immediately, simultaneously proposing the fix to the community
- Change our fix if the third party tool is modified differently by the community
- Regression tests are helpful when upgrading an open source program
- High throughput workflow products are good at finding exceptional cases, providing feedback and test cases

Operating Systems

- Product requirement: run on both Windows and Linux
- Many third party programs have a distinct Unix bias, causing a challenge on Windows
  - _program_list from Bio::Factory::EMBOSS exits if the operating system is either Windows or Macintosh

```
sub _program_list {
    my ($self) = @_;
    if( $^O =~ /MSWIN/i ||
      $^O =~ /Mac/i ) { return; }
...
}
```

Operating Systems (cont.)

- Command line invocation of programs through Perl can also be a challenge
  - Several different approaches
    - system function
    - backticks
    - pipes
  - BioPerl uses all three
  - All can have different behavior on Windows vs. Linux
  - Cygwin libraries can help by allowing Unix-like executables to run in a Windows environment
  - Win32::Process::Create can also be used, but causes a logic branch that we try to avoid

Cascading Changes

- NCBI changed the intermediate pages returned during online BLAST runs
  - Bio::Tools::Run::RemoteBlast could no longer distinguish between intermediate pages and the final result
- RefSeq's IDs are now longer
  - Result file parser in Bio::AlignIO::emboss truncates them

Sequence Data Issues

- Differing uses and expectations regarding upper and lower case sequence data
- Masking is represented in different ways
- Not all programs can handle ambiguous bases/residues; Selenocysteine (U) also causes problems
  - Some programs address this on their own, e.g., BLASTp and U → X
  - Others, e.g., EMBOSS' hmoment, fail

Program Parameters and Error Messages

- Programs can have cryptic parameter values, e.g., genetic code numbers that need to be replaced with names
- Some programs, e.g., Primer3, have large numbers of parameters
  - We split these into basic and advanced sets so that novice users aren't overwhelmed with detail
- Error messages should be meaningful
  - Tell users what went wrong and what to do about it
  - If a third party tool merely provides a stack trace, we may have to replace or augment it to make our software more user friendly

Irritating Little Things

- HMMER's hmmalign reformats NCBI's standard FASTA ID format
  - gi|38083732|ref|XP_357594.1 → gi_38083732_ref_XP_357594_1
- ctrl-A characters in NCBI's nonredundant database are illegal chars in any XML encoding
- Swiss-Prot entry P03393 has the gene name “ENV”, wreaking havoc since the string gets used as a hash name in BioPerl
- Fuzzy locations in GenBank entries are rarely used, but need to be correct when they arise

Irritating Little Things (cont.)

- ClustalW and FASTA input
  - FASTA IDs are truncated
  - Long FASTA descriptions overflow an allocated string and end up being used as sequence data
- BioPerl inconsistencies, e.g., Bio::SeqI function names and return values
  - get_SeqFeatures returns an array of Bio::SeqFeatureI objects
  - annotation returns a Bio::AnnotationCollectionI, which provides access to the array of annotations

Conclusion

We're strong believers in open source and publicly available tools and see the necessity of integrating them into a commercial product. Our community thrives on them, distinguishing us from other communities that rely more heavily on proprietary software, e.g., cheminformatics.