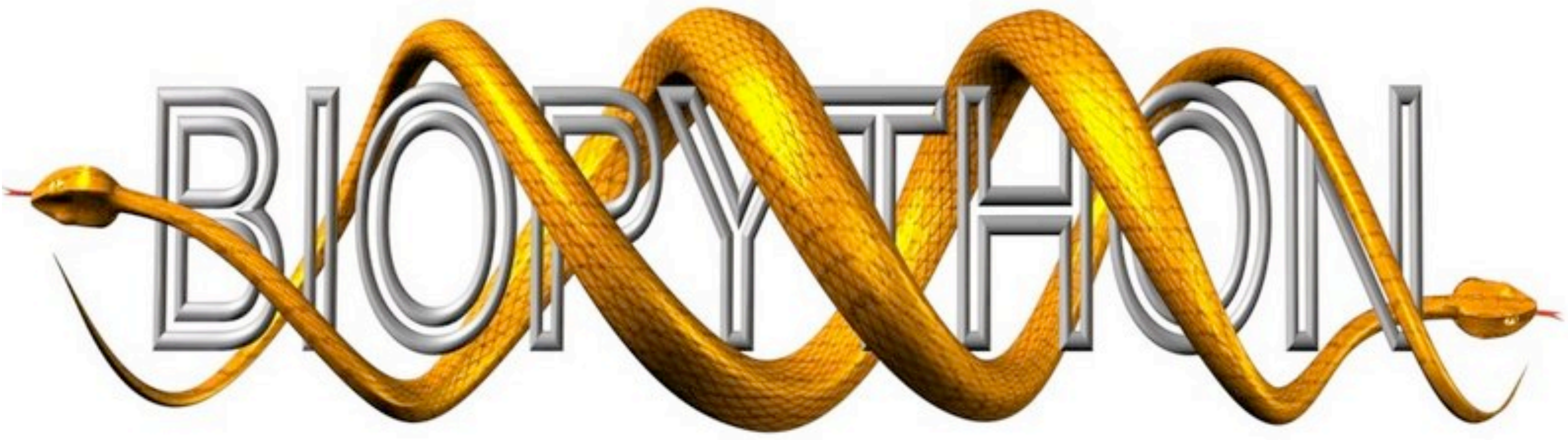
The 8th annual
Bioinformatics Open Source Conference
(BOSC 2007)
18th July, Vienna, Austria



Biopython Project Update

Peter Cock,
MOAC Doctoral Training Centre,
University of Warwick, UK

Talk Outline

- What is python?
- What is Biopython?
- Short history
- Project organisation
- What can you do with it?
- How can you contribute?
- Acknowledgements

What is Python?

- High level programming language
- Object orientated
- Open Source, free ($$$)
- Cross platform:
  Linux, Windows, Mac OS X, …
- Extensible in C, C++, …

What is Biopython?

- Set of libraries for computational biology
- Open Source, free ($$$)
- Cross platform:
  Linux, Windows, Mac OS X, …
- Sibling project to BioPerl, BioRuby, BioJava, …

Popularity by Google Hits

Python	98 million	Biopython	252,000
Perl	101 million	BioPerl	610,000
Ruby	101 million	BioRuby	122,000
Java	289 million	BioJava	185,000

- Both Perl and Python are strong at text
- Python may have the edge for numerical work (with the Numerical python libraries)

Biopython history

- 1999 : Started by Jeff Chang & Andrew Dalke
- 2000 : Biopython 0.90, first release
- 2001 : Biopython 1.00, “semi-complete”
- 2002 : Biopython 1.10, “semi-stable”
- 2003 : Biopython 1.20, 1.21, 1.22 and 1.23
- 2004 : Biopython 1.24 and 1.30
- 2005 : Biopython 1.40 and 1.41
- 2006 : Biopython 1.42
- 2007 : Biopython 1.43

Biopython Project Organisation

- Releases:
  - No fixed schedule
  - Currently once or twice a year
  - Work from a stable CVS base
- Bugs:
  - Online bugzilla
  - Some small changes handled on mailing list
- Tests:
  - Many based on unittest python library
  - Also simple scripts where output is verified

What can you do with Biopython?

- Read, write & manipulate sequences
- Restriction enzymes
- BLAST (local and online)
- Web databases (e.g. NCBI’s EUtils)
- Call command line tools (e.g. clustalw)
- Clustering (Bio.Cluster)
- Phylogenetics (Bio.Nexus)
- Protein Structures (Bio.PDB)

Manipulating Sequences

- Use Biopython’s Seq object, holds:
  - Sequence data (string like)
  - Alphabet (can include list of letters)
- Alphabet allows type checking, preventing errors like appending DNA to Protein

Manipulating Sequences

```
from Bio.Seq import Seq
from Bio.Alphabet.IUPAC import unambiguous_dna

my_dna=Seq('CTAAACATCCTTCAT', unambiguous_dna)
print 'Original:'
print my_dna
print 'Reverse complement:'
print my_dna.reverse_complement()
```

```
Original:
Seq('CTAAACATCCTTCAT', IUPACUnambiguousDNA())
Reverse complement:
Seq('ATGAAGGATGTTTAG', IUPACUnambiguousDNA())
```

Translating Sequences

```
from Bio import Translate
bact_trans=Translate.unambiguous_dna_by_id[11]

print 'Forward translation'
print bact_trans.translate(my_dna)
print 'Reverse complement translation'
print bact_trans.translate( \
                   my_dna.reverse_complement())
```

```
Forward translation
Seq('LNILH', HasStopCodon(IUPACProtein(), '*'))
Reverse complement translation
Seq('MKDV*', HasStopCodon(IUPACProtein(), '*'))
```

Sequence Input/Output

- Bio.SeqIO is new in Biopython 1.43
- Inspired by BioPerl’s SeqIO
- Works with SeqRecord objects
  (not format specific representations)
- Builds on existing Biopython parsers

SeqIO – Sequence Input

```
from Bio import SeqIO
handle = open('ls_orchid.fasta')
format = 'fasta'
for rec in SeqIO.parse(handle, format) :
    print "%s, len %i" % (rec.id, len(rec.seq))
    print rec.seq[:40].tostring() + "..."
handle.close()
```

```
gi|2765658|emb|Z78533.1|CIZ78533, len 740
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCAT...
gi|2765657|emb|Z78532.1|CCZ78532, len 753
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCAT...
...
```

SeqIO – Sequence Input

```
from Bio import SeqIO
handle = open('ls_orchid.gbk')
format = 'genbank'
for rec in SeqIO.parse(handle, format) :
    print "%s, len %i" % (rec.id, len(rec.seq))
    print rec.seq[:40].tostring() + "..."
handle.close()
```

```
Z78533.1, len 740
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCAT...
Z78532.1, len 753
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCAT...
...
```

SeqIO – Extracting Data

```
from Bio import SeqIO
handle = open('ls_orchid.gbk')
format = 'genbank'
from sets import Set
print Set([rec.annotations['organism'] \
       for rec in SeqIO.parse(handle, format)])
handle.close()
```

```
Set(['Cypripedium acaule', 'Paphiopedilum
primulinum', 'Phragmipedium lindenii',
'Paphiopedilum papuanum', 'Paphiopedilum
stonei', 'Paphiopedilum urbanianum',
'Paphiopedilum dianthum', ...])
```

SeqIO – Filtering Output

```
i_handle = open('ls_orchid.gbk')
o_handle = open('small_orchid.faa', 'w')
SeqIO.write([rec for rec in \
     SeqIO.parse(i_handle, 'genbank') \
     if len(rec.seq) < 600], o_handle, 'fasta')
i_handle.close()
o_handle.close()
```

```
>Z78481.1 P.insigne 5.8S rRNA gene and ITS1 ...
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTT...
>Z78480.1 P.gratrixianum 5.8S rRNA gene and ...
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTT...
...
```

3D Structures

- Bio.Nexus was added in Biopython 1.30 by Frank Kauff and Cymon Cox
- Reads Nexus alignments and trees
- Also parses Newick format trees

Newick Tree Parsing

```
(Bovine:0.69395,(Gibbon:0.36079,(Orang:0.33636,
(Gorilla:0.17147,(Chimp:0.19268, Human:
0.11927):0.08386):0.06124):0.15057):
0.54939,Mouse:1.21460):0.10;
```

```
from Bio.Nexus.Trees import Tree
tree_str = open("simple.tree").read()
tree_obj = Tree(tree_str)
print tree_obj
```

```
tree a_tree = (Bovine,(Gibbon,(Orang,(Gorilla,
(Chimp,Human)))),Mouse);
```

Newick Tree Parsing

```
tree_obj.display()
```

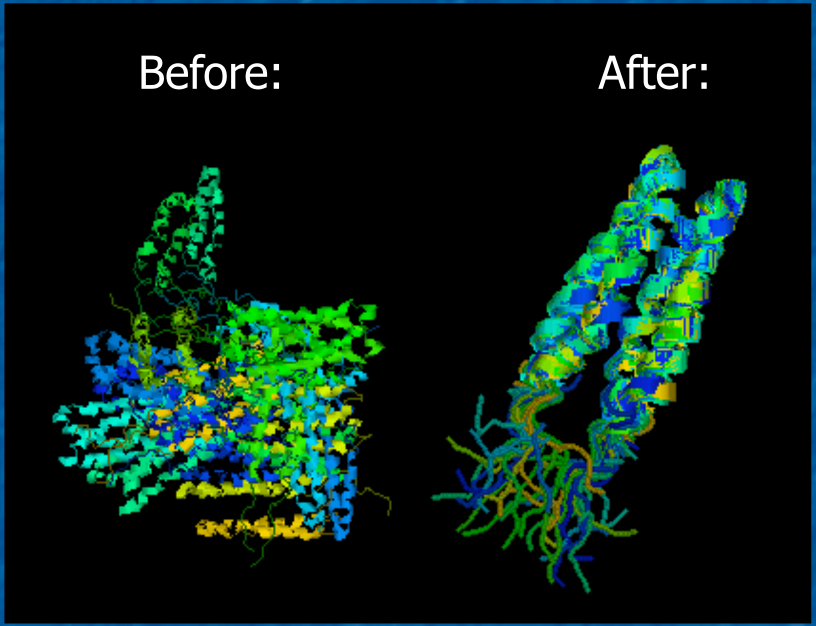
```
#    taxon     prev  succ      brlen     blen (sum) support
0    -         None  [1,2,11]  0.0       0.0        -
1    Bovine    0     []        0.69395   0.69395    -
2    -         0     [3,4]     0.54939   0.54939    -
3    Gibbon    2     []        0.36079   0.91018    -
4    -         2     [5,6]     0.15057   0.69996    -
5    Orang     4     []        0.33636   1.03632    -
6    -         4     [7,8]     0.06124   0.7612     -
7    Gorilla   6     []        0.17147   0.93267    -
8    -         6     [9,10]    0.08386   0.84506    -
9    Chimp     8     []        0.19268   1.03774    -
10   Human     8     []        0.11927   0.96433    -
11   Mouse     0     []        1.2146    1.2146     -

Root: 0
```

3D Structures

- Bio.PDB was added in Biopython 1.24 by Thomas Hamelryck
- Reads PDB and CIF format files

Working with 3D Structures



This example (online) uses Bio.PDB to align 21 alternative X-Ray crystal structures for PDB structure 1JOY.

http://www.warwick.ac.uk/go/peter_cock/python/protein_superposition/

Population Genetics (planned)

- Tiago Antão (with Ralph Haygood) plans to start a Population Genetics module

See also:

- PyPop: Python for Population Genetics
Alex Lancaster et al. (2003)
www.pypop.org

Areas for Improvement

- Documentation!
- I’m interested in sequences & alignments:
  - Seq objects – more like strings?
  - Alignment objects – more like arrays?
  - SeqIO – support for more formats
  - AlignIO? – alignment equivalent to SeqIO
- Move from Numeric to NumPy
- Move from CVS to SVN?

How can you Contribute?

- Users:
  - Discussions on the mailing list
  - Report bugs
  - Documentation improvement
- Coders:
  - Suggest bug fixes
  - New/extended test cases
  - Adopt modules with no current ‘owner’
  - New modules

Biopython Acknowledgements

- Open Bioinformatics Foundation or O|B|F for web hosting, CVS servers, mailing list
- Biopython developers, including:
  Jeff Chang, Andrew Dalke, Brad Chapman, Iddo Friedberg, Michiel de Hoon, Frank Kauff, Cymon Cox, Thomas Hamelryck, me
- Contributors who report bugs & join in the mailing list discussions

Personal Acknowledgements

- Everyone for listening
- Open Bioinformatics Foundation or O|B|F for the BOSC 2007 invitation
- Iddo Friedberg & Michiel de Hoon for their encouragement
- The EPSRC for my PhD funding via the MOAC Doctoral Training Centre, Warwick http://www.warwick.ac.uk/go/moac

Questions?