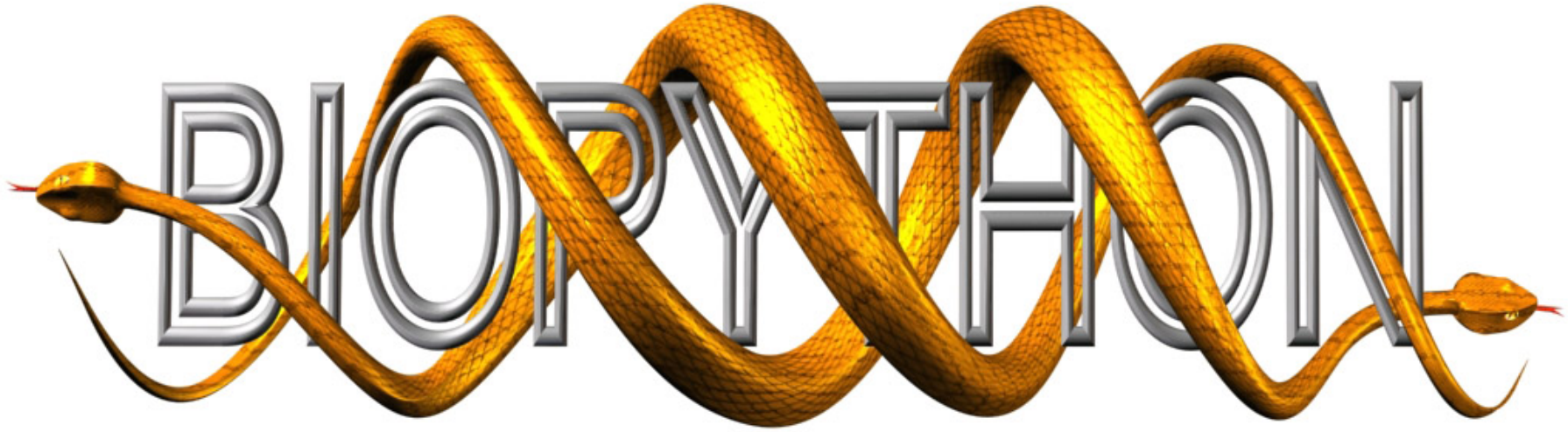


Biopython @ EuroSciPy 2010

EuroSciPy

Annual European Conference for Scientists using Python

Peter Cock, Plant Pathology, SCRI, Dundee, UK

EuroSciPy 2010, 3rd European meeting on Python in Science

Ecole Normale Supérieure, Paris, France, 10 July 2010

O|B|F

scri living technology

Open Bioinformatics Foundation (OBF)

The OBF supports:

- BioPerl
- Biopython
- BioJava
- BioRuby
- BioSQL
- EMBOSS
- …

BioPerl

O|B|F

scri
living technology

Contents

- Brief introduction to Biopython & history
- Examples:

  Sequence manipulation
  3D Biological structures

- Current and future projects
- Developers, git and github, …

scri
living technology

Biopython

- Free, open source library for bioinformatics
- Supported by Open Bioinformatics Foundation
- Runs on Windows, Linux, Mac OS X, etc
- International team of volunteer developers
- Currently about four releases per year
- Extensive “Biopython Tutorial & Cookbook”
- See www.biopython.org for details

scri
living technology

Biopython's Ten Year (and a bit) History

- 1999 • Started
- 2000 • First release
- 2001 • Biopython 1.00

  …
- 2007 • Biopython 1.43, …
- 2008 • Biopython 1.45, …
- 2009 • Biopython 1.50, …
  - • Application note
- 2010 • Biopython 1.54, …

BIOINFORMATICS **APPLICATIONS NOTE** *Vol. 25 no. 11 2009, pages 1422–1423*
doi:10.1093/bioinformatics/btp163

Sequence analysis

Biopython: freely available Python tools for computational molecular biology and bioinformatics

Peter J. A. Cock[1,*], Tiago Antao[2], Jeffrey T. Chang[3], Brad A. Chapman[4], Cymon J. Cox[5], Andrew Dalke[6], Iddo Friedberg[7], Thomas Hamelryck[8], Frank Kauff[9], Bartek Wilczynski[10,11] and Michiel J. L. de Hoon[12]

[1]Plant Pathology, SCRI, Invergowrie, Dundee, DD2 5DA, [2]Liverpool School of Tropical Medicine, Liverpool, L3 5QA, UK, [3]Institute for Genome Sciences and Policy, Duke University Medical Center, Durham, NC, [4]Department of Molecular Biology, Simches Research Center, Massachusetts General Hospital, Boston, MA 02114, USA, [5]Centro de Ciências do Mar, Universidade do Algarve, Faro, Portugal, [6]Andrew Dalke Scientific, AB, Gothenburg, Sweden, [7]California Institute for Telecommunications and Information Technology, University of California, San Diego, 9500 Gilman Dr., La Jolla, CA 92093-0446, USA, [8]Bioinformatics Center, Department of Biology, University of Copenhagen, Ole Maaloes Vej 5, 2200 Copenhagen N, Denmark, [9]Molecular Phylogenetics, Department of Biology, TU Kaiserslautern, 67653 Kaiserslautern, UK, [10]EMBL Heidelberg, Meyerhofstraße 1, 69117 Heidelberg, Germany, [11]Institute of Informatics, University of Warsaw, Poland and [12]RIKEN Omics Science Center, 1-7-22 Suehiro-cho, Tsurumi-ku, Yokohama-shi, Kanagawa-ken, 230-0045, Japan

Received and revised on March 11, 2009; accepted on March 16, 2009

Advance Access publication March 20, 2009

Associate Editor: Dmitrij Frishman

ABSTRACT

Summary: The Biopython project is a mature open source international collaboration of volunteer developers, providing Python libraries for a wide range of bioinformatics problems. Biopython includes modules for reading and writing different sequence file formats and multiple sequence alignments, dealing with 3D macromolecular structures, interacting with common tools such as BLAST, ClustalW and EMBOSS, accessing key online databases, as well as providing numerical methods for statistical learning.

Availability: Biopython is freely available, with documentation and source code at www.biopython.org under the Biopython license.

Contact: All queries should be directed to the Biopython mailing lists, see www.biopython.org/wiki/Mailing_lists; peter.cock@scri.ac.uk.

1 INTRODUCTION

Python (www.python.org) and Biopython are freely available open source tools, available for all the major operating systems. Python is a very high-level programming language, in widespread commercial and academic use. It features an easy to learn syntax, object-oriented programming capabilities and a wide array of libraries. Python can interface to optimized code written in C, C++ or even FORTRAN, and together with the Numerical Python project `numpy` (Oliphant, 2006), makes a good choice for scientific programming (Oliphant, 2007). Python has even been used in the numerically demanding field of molecular dynamics (Hinsen, 2000). There are also high-quality plotting libraries such as `matplotlib` (matplotlib.sourceforge.net) available.

*To whom correspondence should be addressed.

Since its founding in 1999 (Chapman and Chang, 2000), Biopython has grown into a large collection of modules, described briefly below, intended for computational biology or bioinformatics programmers to use in scripts or incorporate into their own software. Our web site lists over 100 publications using or citing Biopython.

The Open Bioinformatics Foundation (OBF, www.open-bio.org) hosts our web site, source code repository, bug tracking database and email mailing lists, and also supports the related BioPerl (Stajich *et al.*, 2002), BioJava (Holland *et al.*, 2008), BioRuby (www.bioruby.org) and BioSQL (www.biosql.org) projects.

2 BIOPYTHON FEATURES

The `Seq` object is Biopython's core sequence representation. It behaves very much like a Python string but with the addition of an alphabet (allowing explicit declaration of a protein sequence for example) and some key biologically relevant methods. For example,

```
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import generic_dna
>>> gene = Seq("ATGAAAGCAATTTTCGTACTG"
...            "AAAGGTTGGTGGCGCACTTGA",
...            generic_dna)
>>> print gene.transcribe()
AUGAAAGCAAUUUUCGUACUGAAAGGUUGGUGGCGCACUUGA
>>> print gene.translate(table=11)
MKAIFVLKGWWRT*
```

Sequence annotation is represented using `SeqRecord` objects which augment a `Seq` object with properties such as the record name, identifier and description and space for additional key/value terms. The `SeqRecord` can also hold a list of `SeqFeature`

scri living technology

Examples

scri
living technology

Sequence vs Sequence

- In biology the word “sequence” generally means an ordered collection of letters representing a directed molecular chain

  DNA usually A, C, G and T

  RNA usually A, C, G and U

  Proteins usually 20 single letter codes

- Python strings are often a good model
- Biopython has a Seq object…

scri
living technology

String like methods for Seq objects

- Seq has an alphabet (DNA, RNA or Protein)

```
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import generic_dna
>>> dna = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC", generic_dna)
>>> print dna, dna.alphabet
GATCGATGGGCCTATATAGGATCGAAAATCGC DNAAlphabet()

>>> len(dna)
32
>>> dna.count('C')
6
>>> dna.find("TATAT")
12
>>> print dna[:12] + "-----" + dna[17:]
GATCGATGGGCC-----AGGATCGAAAATCGC

>>> print dna.lower()
gatcgatgggcctatataggatcgaaaatcgc
```

Explicit declaration of the alphabet (sequence type).

scri living technology

Biological methods for sequences

- DNA to RNA to Protein - “The Central Dogma”

```
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import generic_dna
>>> dna = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC", generic_dna)
>>> print dna, dna.alphabet
GATCGATGGGCCTATATAGGATCGAAAATCGC DNAAlphabet()

>>> print dna.complement()
CTAGCTACCCGGATATATCCTAGCTTTTAGCG
>>> print dna.reverse_complement()
GCGATTTTCGATCCTATATAGGCCCATCGATC

>>> rna = dna.transcribe()
>>> print rna, rna.alphabet
GAUCGAUGGGCCUAUAUAGGAUCGAAAAUCGC RNAAlphabet()

>>> protein = rna.translate()
>>> print protein, protein.alphabet
DRWAYIGSKI ExtendedIUPACProtein()
```

scri
living technology

Sequence File Manipulation

- Manipulating nucleotide and protein sequences is a common task in Bioinformatics
- Manipulating plain text sequence files is too
- There are *lots* of different file formats ☹
- Motivation for common object and API

scri
living technology

Reading a FASTA file with Bio.SeqIO

```
>FL3B07415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAA
GTACGAGCAGATACTCCAATCGCAATTGTTTCTTCATTTAAAATTAGCTCGTCGCCACCT
TCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTT
GGATGATATTTAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTC
ATCTTATTTATCGTTAAGCCA
>FL3B07415I7AFR
CATTAACTAA...
```

```
from Bio import SeqIO

for rec in SeqIO.parse("phage.fasta", "fasta") :
    print rec.id, len(rec.seq), rec.seq[:10]+"..."
```

```
FL3B07415JACDX 261 TTAATTTTAT...
FL3B07415I7AFR 267 CATTAACTAA...
FL3B07415JCAY5 136 TTTCTTTTCT...
FL3B07415JB41R 208 CTCTTTTATG...
FL3B07415I6HKB 268 GGTATTTGAA...
FL3B07415I63UC 219 AACATGTGAG...
...
```

Focus on the filename and format (“fasta”)…

scri living technology

Reading a FASTQ file with Bio.SeqIO

```
@FL3BO7415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAAGTACGAGCAGATACTCCAATCGCAATTGTTTCTTC
ATTTAAAATTAGCTCGTCGCCACCTTCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTTGGATGATATT
TAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTCATCTTATTTATCGTTAAGCCA
+
BBBB2262=1111FFGGGHHHHIIIIIIIIIIIIIIIIIIIIIIFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGGFFFFFFFFFFFFFFFGB
BBCFFFFFFFFFFFFFFFFFFFFFGGGGGGGIIIIIIIGGGIIIGGGIIGGGG@AAAAA?===@@@???
@FL3BO7415I7AFR
CATTAACTAA...
```

```
from Bio import SeqIO

for rec in SeqIO.parse("phage.fastq", "fastq") :
    print rec.id, len(rec.seq), rec.seq[:10]+"..."
    print rec.letter_annotations["phred_quality"][:10], "..."
```

```
FL3BO7415JACDX 261 TTAATTTTAT...
[33, 33, 33, 33, 17, 17, 21, 17, 28, 16] ...
FL3BO7415I7AFR 267 CATTAACTAA...
[37, 37, 37, 37, 37, 37, 37, 37, 38, 38] ...
FL3BO7415JCAY5 136 TTTCTTTTCT...
[37, 37, 36, 36, 29, 29, 29, 29, 36, 37] ...
FL3BO7415JB41R 208 CTCTTTTATG...
[37, 37, 37, 38, 38, 38, 38, 38, 37, 37] ...
FL3BO7415I6HKB 268 GGTATTTGAA...
[37, 37, 37, 37, 34, 34, 34, 37, 37, 37] ...
FL3BO7415I63UC 219 AACATGTGAG...
[37, 37, 37, 37, 37, 37, 37, 37, 37, 37] ...
...
```

Just filename and format changed ("fasta" to "fastq")

scri living technology

Sequence File Manipulation

- Common object for sequence file entries, SeqRecord, for Seq plus annotation like ID
- Sequence file API based on iterators
- Memory efficient!
- Scales to millions of reads as seen in current sequencing platforms (Roche, Illumina, etc)

scri
living technology

Trimming a FASTA file with Bio.SeqIO

```
>FL3B07415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAA
GTACGAGCAGATACTCCAATCGCAATTGTTTCTTCATTTAAAATTAGCTCGTCGCCACCT
TCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTT
GGATGATATTTAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTC
ATCTTATTTATCGTTAAGCCA
>FL3B07415I7AFR
CATTAACTAA...
```

```
from Bio import SeqIO

recs = (r[:10] for r in SeqIO.parse("phage.fasta", "fasta")

SeqIO.write(recs, "long.fasta", "fasta")
```

```
>FL3B07415JACDX
TTAATTTTAT
>FL3B07415I7AFR
CATTAACTAA
>FL3B07415JCAY5
TTTCTTTTCT
>FL3B07415JB41R
CTCTTTTATG
...
```

Generator expression

scri living technology

Filtering a FASTA file with Bio.SeqIO

```
>FL3B07415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAA
GTACGAGCAGATACTCCAATCGCAATTGTTTCTTCATTTAAAATTAGCTCGTCGCCACCT
TCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTT
GGATGATATTTAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTC
ATCTTATTTATCGTTAAGCCA
>FL3B07415I7AFR
CATTAACTAA...
```

```
from Bio import SeqIO
recs = (r for r in SeqIO.parse("phage.fasta", "fasta") if len(r)>200)
SeqIO.write(recs, "long.fasta", "fasta")
```

```
>FL3B07415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAA
GTACGAGCAGATACTCCAATCGCAATTGTTTCTTCATTTAAAATTAGCTCGTCGCCACCT
TCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTT
GGATGATATTTAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTC
ATCTTATTTATCGTTAAGCCA
>FL3B07415I7AFR
CATTAACTAA...
```

Generator expression

scri living technology

Filtering and converting FASTQ to FASTA

```
@FL3BO7415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAAGTACGAGCAGATACTCCAATCGCAATTGTTTCTTC
ATTTAAAATTAGCTCGTCGCCACCTTCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTTGGATGATATT
TAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTCATCTTATTTATCGTTAAGCCA
+
BBBB2262=1111FFGGGHHHHIIIIIIIIIIIIIIIIIIIIIIFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGGFFFFFFFFFFFFFFFGB
BBCFFFFFFFFFFFFFFFFFFFFFFGGGGGGGIIIIIIIGGGIIIGGGIIGGGG@AAAAA?===@@@???
@FL3BO7415I7AFR
CATTAACTAA...
```

```
from Bio import SeqIO

recs = (r for r in SeqIO.parse("phage.fastq", "fastq") if len(r)>200)

SeqIO.write(recs, "long.fasta", "fasta")
```

```
>FL3BO7415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAA
GTACGAGCAGATACTCCAATCGCAATTGTTTCTTCATTTAAAATTAGCTCGTCGCCACCT
TCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTT
GGATGATATTTAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTC
ATCTTATTTATCGTTAAGCCA
>FL3BO7415I7AFR
CATTAACTAA...
```

Generator expression

scri living technology

General sequence file conversion

- Separate parse and write calls (as before):

```
from Bio import SeqIO

recs = SeqIO.parse("roche.sff", "sff")
SeqIO.write(recs, "reads.fastq", "fastq")
```

- Shorthand convert call (for the typical case):

```
from Bio import SeqIO

SeqIO.convert("roche.sff", "sff", "reads.fastq", "fastq")
```

- Simple to switch file formats:

```
from Bio import SeqIO

SeqIO.convert("roche.sff", "sff", "phage.fasta", "fasta")
```

- Some conversions are optimized

scri
living technology

Analysing a FASTA file with Bio.SeqIO

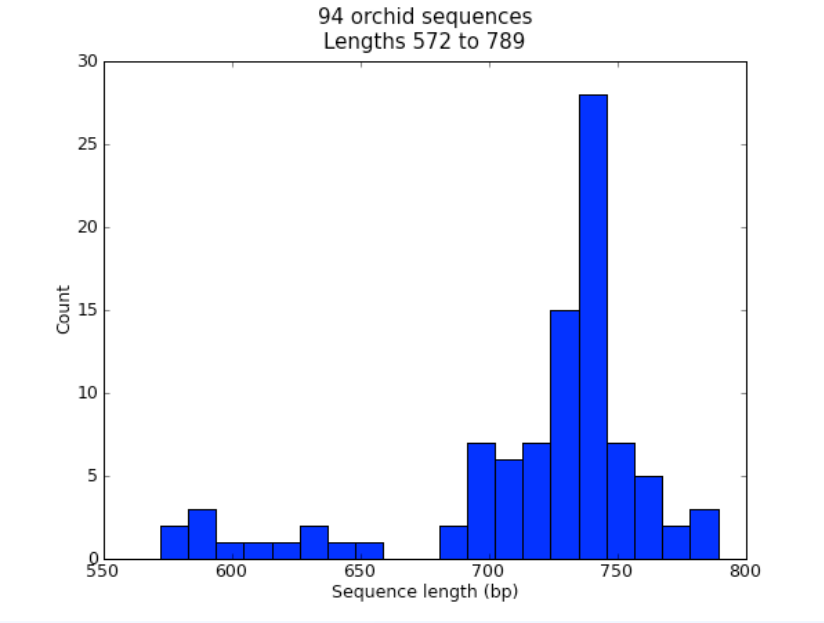
```
>>> from Bio import SeqIO
>>> sizes = [len(r) for r in SeqIO.parse("ls_orchid.fasta", "fasta")]
>>> len(sizes), min(sizes), max(sizes)
(94, 572, 789)
>>> sizes
[740, 753, 748, 744, 733, 718, 730, 704, 740, 709, 700, 726, ..., 592]
```

List comprehension

Analysing a FASTA file with Bio.SeqIO

```
from Bio import SeqIO
sizes = [len(r) for r in SeqIO.parse("ls_orchid.fasta", "fasta")]

import pylab
pylab.hist(sizes, bins=20)
pylab.title("%i orchid sequences\nLengths %i to %i" \
            % (len(sizes),min(sizes),max(sizes)))
pylab.xlabel("Sequence length (bp)")
pylab.ylabel("Count")
pylab.show()
```



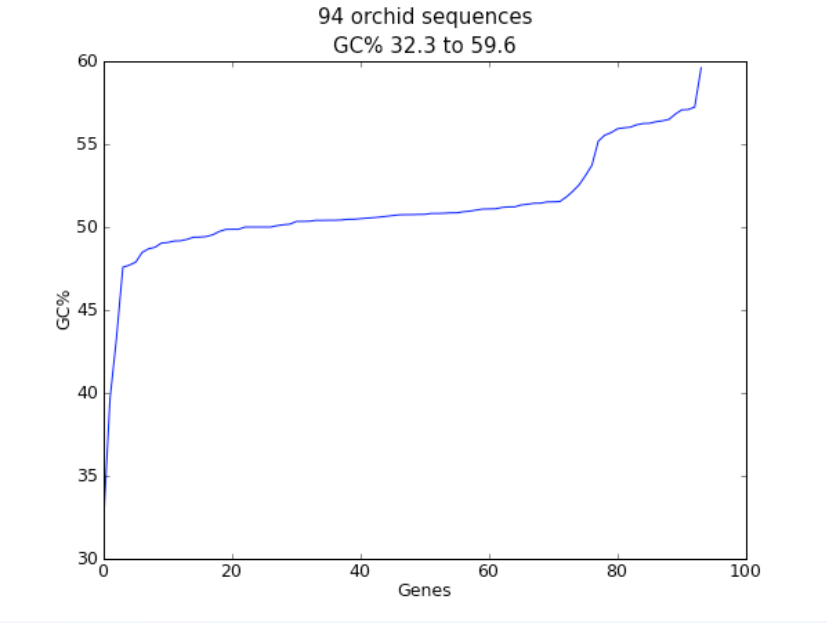
Plot with pylab
(aka matplotlib)

scri
living technology

Analysing a FASTA file with Bio.SeqIO

```
from Bio import SeqIO
from Bio.SeqUtils import GC
val = sorted(GC(r.seq) for r in SeqIO.parse("ls_orchid.fasta","fasta"))

import pylab
pylab.plot(val)
pylab.title("%i orchid sequences\nGC%% %0.1f to %0.1f" \
            % (len(val), min(val), max(val)))
pylab.xlabel("Genes")
pylab.ylabel("GC%")
pylab.show()
```



Calculate percentage of DNA sequence using the letters G or C (biologically important)

scri living technology

Querying online database – e.g. NCBI

```
>>> from Bio import Entrez
>>> Entrez.email = "A.N.Other@example.com"     # Tell NCBI who you are
>>> record = Entrez.read(Entrez.egquery(term="biopython"))
>>> for row in record["eGQueryResult"]:
...     print row["DbName"], row["Count”]
...
pubmed 10
pmc 109
journals 0
mesh 0
books 0
omim 0
omia 0
ncbisearch 0
nuccore 0
...
```

Using NCBI XML parser

numpy gets 74 hits in PubMedCentral, scipy gets 5 in PubMed and 91 in PMC

scri living technology

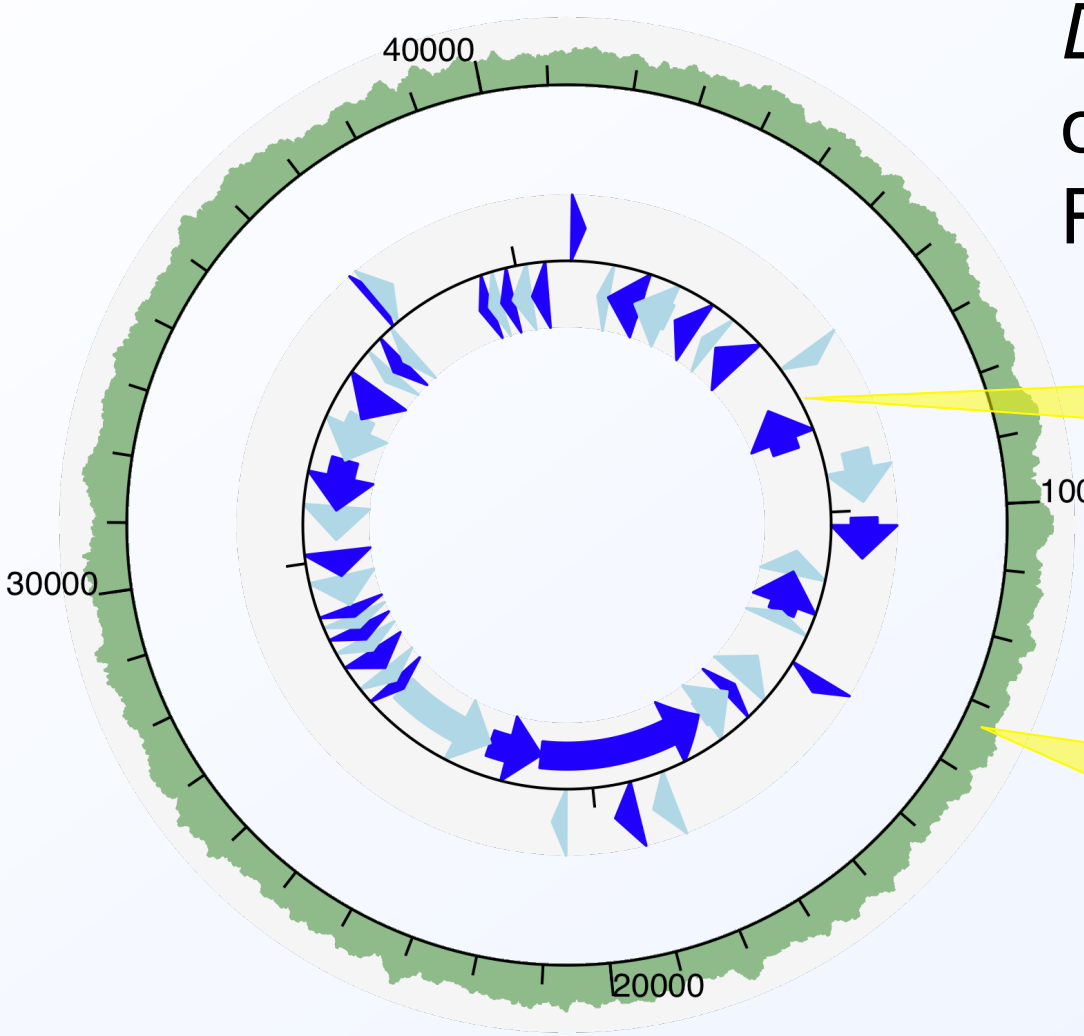
Manipulating 3D Biological structures

- Spacial alignment (using NumPy internally)

- See http://www.warwick.ac.uk/go/peter_cock/python/protein_superposition/
- Visualisation using OpenRasMol

scri living technology

Circular GenomeDiagram



De novo assembly of 42kb phage from Roche 454 data

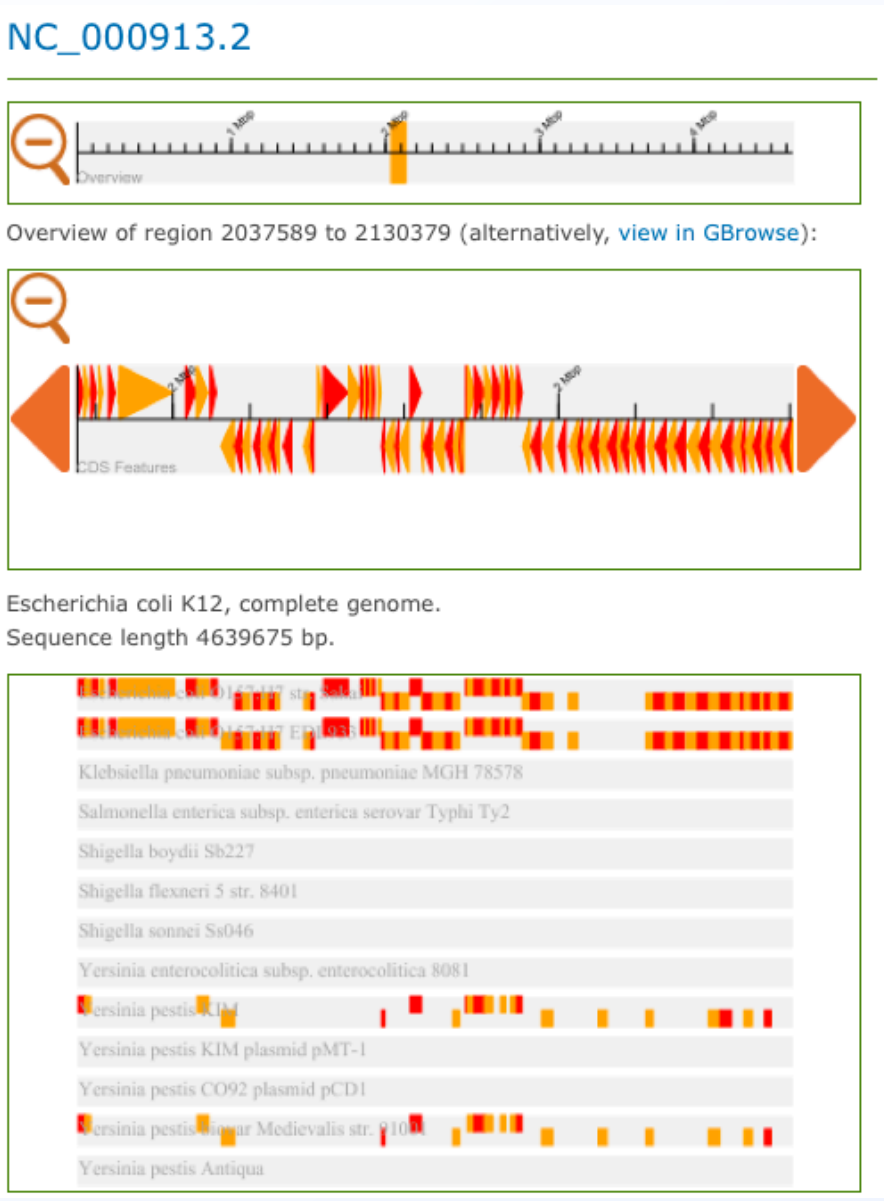
"Feature Track" showing ORFs

Scale tick marks

"Barchart Track" of read depth (~100, scale max 200)

scri
living technology

Linear - GenomeDiagram



Screenshot from an in-house web server using:

- Biopython
- BioSQL
- ReportLab
- SQLAlchemy
- Turbogears

scri living technology

Other functionality not discussed

- Calling and parsing BLAST (local and online)
- Call command line tools (e.g. clustalw)
- Restriction enzymes
- Multiple Sequence Alignments
- Clustering (Bio.Cluster)
- Phylogenetics (Bio.Phylo, Bio.Nexus)
- BioSQL support (common schema)
- Population genetics (Bio.PopGen)
- …

scri
living technology

Current and Future Work

- Python 3 support (now NumPy is almost there)
- Google Summer of Code 2009:

  Nick Matzkes, Biogeography

  (Erik Talevich , phyloXML, already merged)
- Google Summer of Code 2010:

  João Rodrigues, extending Bio.PDB module
- Lots of other stuff!

scri
living technology

Development

- Moved from CVS to git a year ago
- Hosted on github.com at http://github.com/biopython/biopython
- Over 50 people have made a branch
- New features are now routinely developed on public branches
- Still work from a main stable branch

scri
living technology

What do I personally use (Bio)python for?

- Scripting command line tools
- Basic sequence manipulation
- Preparing input files for genome assembly
- Analysis of genome assembly coverage etc
- Working with gene annotation
- Visualising genomic information
- Calling R scripts with rpy or rpy2
- …

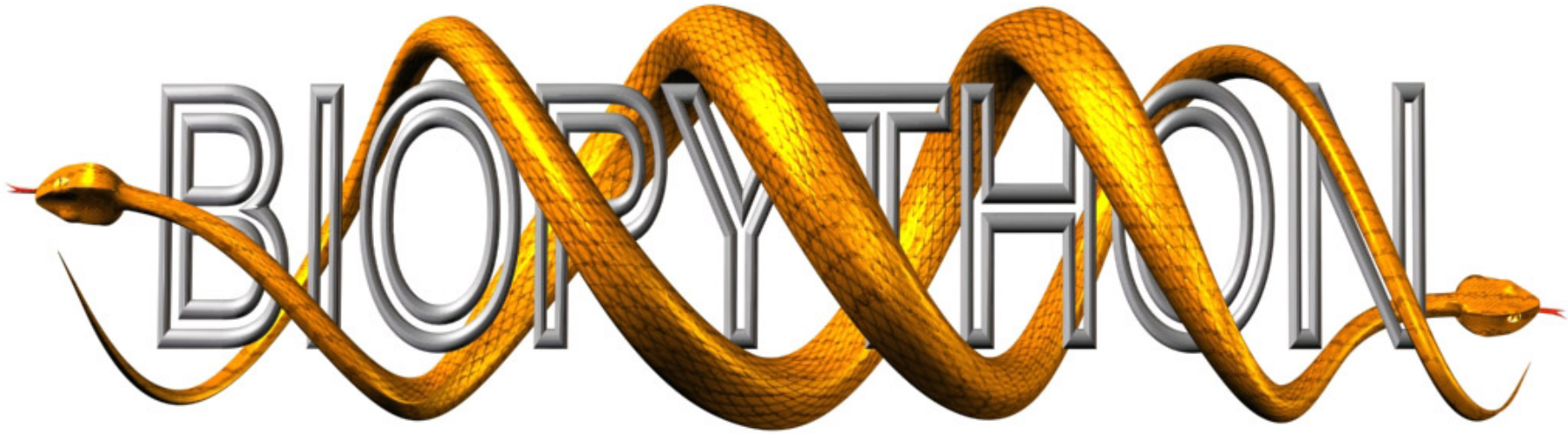
scri
living technology

Acknowledgements

- Other Biopython contributors & developers!
- Open Bioinformatics Foundation (OBF) supports Biopython (and BioPerl etc)
- My Biopython work was/is supported by:
  - EPSRC funded PhD (MOAC DTC, University of Warwick, UK)
  - SCRI (Scottish Crop Research Institute), who also paid my conference fees and travel to be here

What next?

- Sign up to our mailing list?



- Homepage www.biopython.org

scri
living technology