Biopython project update
(and the Python ecology for bioinformatics)

Tiago Antão[1] and Peter Cock[2,3]

1 - Liverpool School Of Tropical Medicine, UK
2 - SCRI, Dundee, UK
3 - MOAC Doctoral Training Centre, University of Warwick, UK

LSTM

scri living technology

Credits and caveats

- 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, Peter Cock
- Contributors who report bugs & join in the mailing list discussions
- Caveat: Person in front of you minor author (new population genetics module)

Overview

- Python
- Biopython
- The Python ecology for bioinformatics
  - Matplotlib
  - NumPy and SciPy
  - Jython/IronPython
  - ...Python as a complete solution for computational biology research

Python

- High-level, OO, free software (the usual goodies)
- Cultural traits of the Python community (applies to both the core language and many Python projects) – The Python ethos
  - Smooth learning curve – powerful yet tamable by new users
  - Focus on readability, maintainability
  - Good documentation
  - Examples will follow

Biopython
Available features

- 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)
- BioSQL support
- Population genetics (Bio.PopGen) – New module

Bio.Entrez
Examples

- Searching the taxonomy database

```
> from Bio import Entrez
> handle = Entrez.esearch(
    db="Taxonomy",
    term="Nanoarchaeum equitans")
> record = Entrez.read(handle)
                          #parse XML
> record["IdList"]
['158330']
```

- Fetch the species lineage

```
> handle = Entrez.efetch(
    db="Taxonomy",
    id="158330",
    retmode='xml')
> records = Entrez.read(handle)
> records[0]['Lineage']
'cellular organisms; Archaea;
Nanoarchaeota; Nanoarchaeum'
```

- Entrez interface supports all the API functions
- Will convert the XML output to typical Python structures

Bio.AlignIO

- File format agnostic
- Reads and writes: PFAM/Stockholm, Clustal, PHYLIP (strict), and FASTA
- Also reads several formats including Nexus

```
> from Bio import AlignIO
> alignment = AlignIO.read(open("PF09395_seed.sth"),"stockholm")
> print alignment
SingleLetterAlphabet() alignment with 14 rows and 77 columns
GFGTYCPTTCGVADYLQRYKPDMDKKLDDMEQDLEEIANLTRGA...NML Q7ZVG7_BRARE/37-110
...
RFGSYCPTTCGIADFLSTYQTXVDKDLQVLEDILNQAENKTSEA...KML O02689_TAPIN/1-77
RFGSYCPTMCGIAGFLSTYQNTVEKDLQNLEGILHQVENKTSEA...KML O02688_PIG/1-77
RFGSYCPTTCGVADFLSNYQTSVDKDLQNLEGILYQVENKTSEA...RMM O02672_9CETA/1-77
RFGSYCPTTCGIADFLSNYQTSVDKDLQDFEDILHRAENQTSEA...KMM O02682_EQUPR/1-77
```

Bio.PopGen

- Philosophy: Don't reinvent the wheel if there are good alternatives around
- Mainly a (smart) wrapper for existing applications
  - SimCoal2 for coalescent, Fdist for selection detection
  - LDNe (Ne estimation) coming soon
- Supports GenePop file format (standard in the “industry”)
- Multi-core aware! Even includes a naïve task scheduler

Bio.PopGen
Future and Alternatives

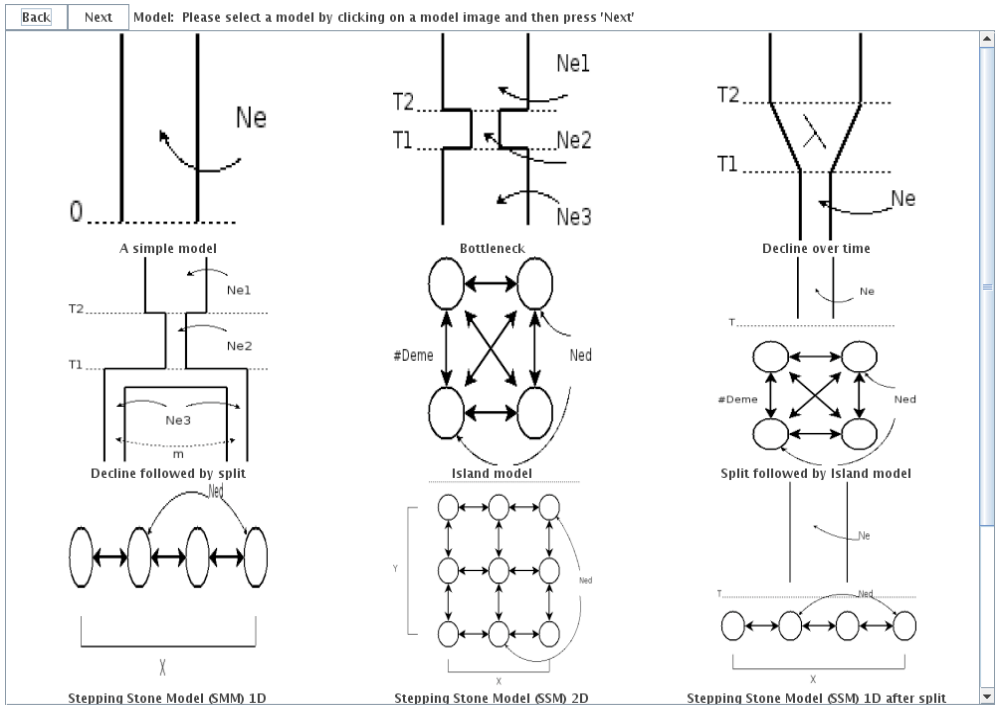
- PopGen Statistics (Fst, LD, EHH, Tajima D, HWE, ...) - Stalled for now, still not clear how to handle statistics (using SciPy or replicate code to avoid adding another dependency)
- Other Python solutions for population genetics
  - SimuPOP for individual based simulation
  - PyPop for some statistics
  - Bio.PopGen + PyPop + SimuPOP = Python's solution to (most) population genetics problems

Jython and IronPython
Interacting with your favourite VM

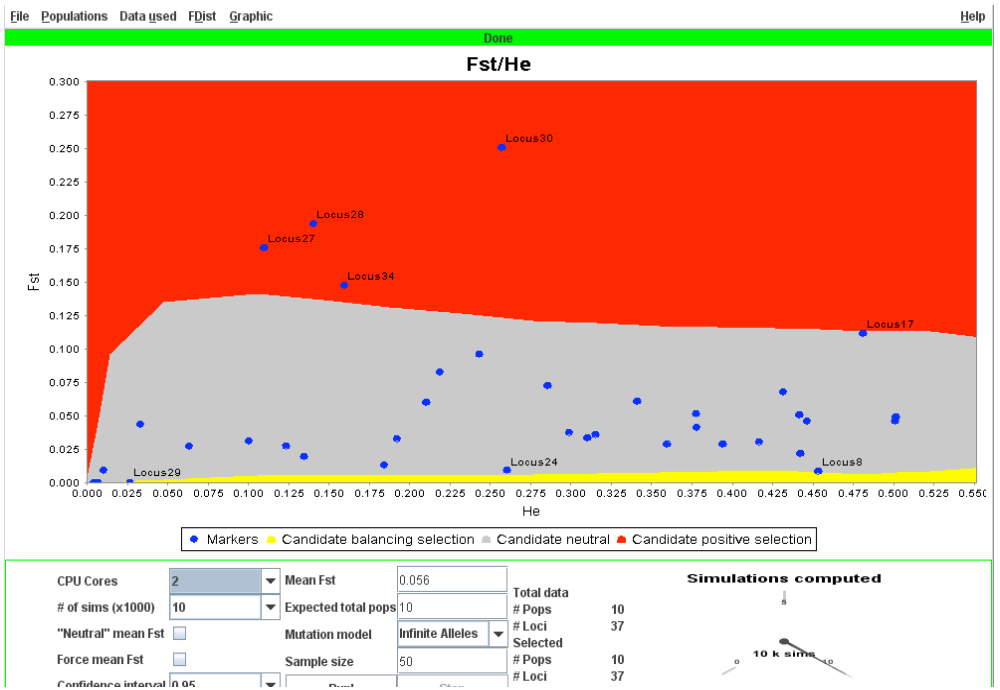
- Biopython can be partially used inside JVM and .NET environments
- Only parts of Biopython available (functionality that links to C code will not work)
- On JVM, BioJava might be a better option...
- Most applications done with Bio.PopGen are actually JavaWebStart Applications

Biopython on Java

- Example applications from Bio.PopGen



Coalescent simulation



Molecular adaptation detection using a Fst-outlier approach

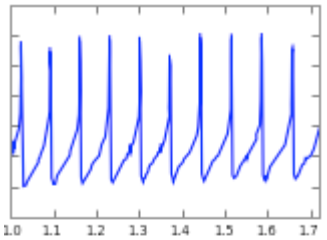
Biopython
Wrapping up

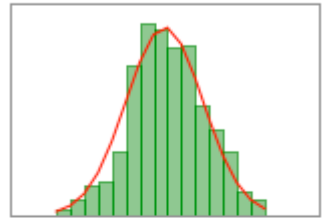
- Contributions welcomed
  - Code, documentation, bug reporting
- Suggestions welcomed
  - What functionality would you like to see?
- Short term goals
  - Moving to Subversion
  - Supporting more file formats in Bio.SeqIO and Bio.AlignIO
  - Numeric to numpy migration
  - Make Sequence objects more string like and OO

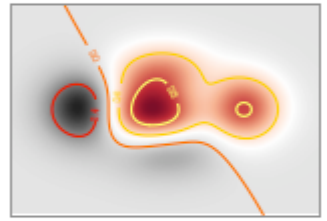
Matplotlib

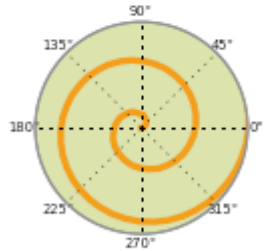
Easy and powerful charting

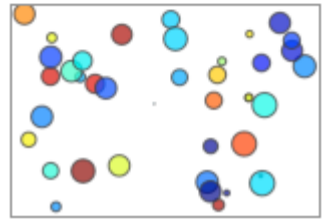
- Easy to use
- Documentation, including cookbooks
- Sensible defaults
- Powerful visualization aides







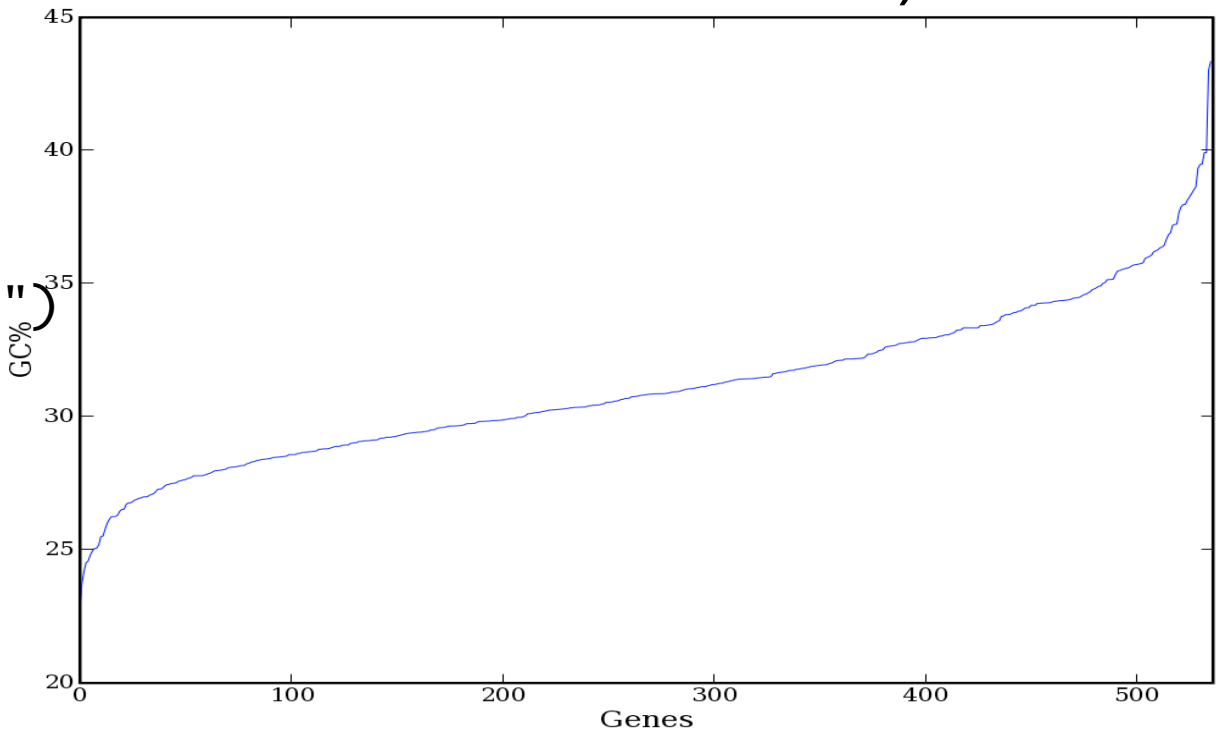




Matplotlib
GC-percentage example

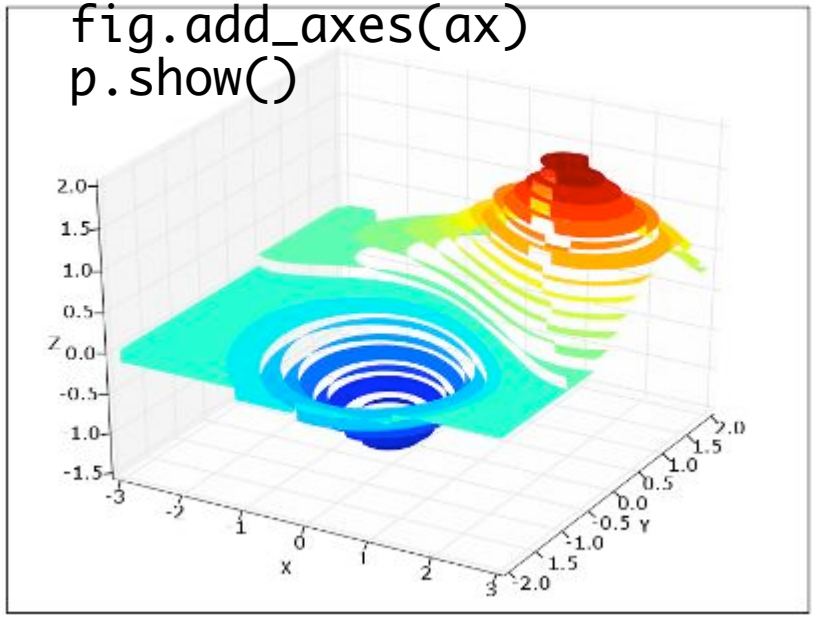
```
from Bio import SeqIO
from Bio.SeqUtils import GC
from pylab import *

data = [GC(rec.seq) for rec in
 SeqIO.parse(open("NC_005213.ffn"),
              "fasta")]

data.sort()

plot(data)
xlim([0,len(data)])
xlabel("Genes")
ylabel("GC%")
savefig("gc_plot.png")
show()
```

- Zoom, save built-in on show
- Straightforward code (judge for yourself, all code is left)



Matplotlib - 3D

```
u=r_[0:2*pi:100j]
v=r_[0:pi:100j]
x=10*outer(cos(u),sin(v))
y=10*outer(sin(u),sin(v))
z=10*outer(ones(size(u)),cos(v))
fig=p.figure()
ax = p3.Axes3D(fig)
ax.contour3D(x,y,z)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
fig.add_axes(ax)
p.show()
```



- User controlled 3D interface included (zoom, rotation, ...)
- Full code shown above the image
- 3D undergoing changes in matplotlib

NumPy and SciPy

- N-dimensional arrays
- Basic linear algebra functions
- Basic Fourier transforms
- Sophisticated random number capabilities

- Statistics
- Optimization
- Numerical integration
- Linear algebra
- Fourier transforms
- Signal processing
- Image processing
- Genetic algorithms
- ODE solvers

Other applications and libraries

- SWIG – C/C++ interaction
- Zope, Django and many other web frameworks
- Plone – Content Management
- ReportLab – PDF generation (used in Biopython)
- MPI for Python – Parallel programming
- SymPy – Symbolic Mathematics
- Python/R interface (e.g. microarrays)
- Pygr – Graph database interfaces with emphasis on bioinformatics
- PysCeS – Simulation of cellular systems
- SloppyCell – Simulation and analysis of biomolecular networks
- ...

Questions?