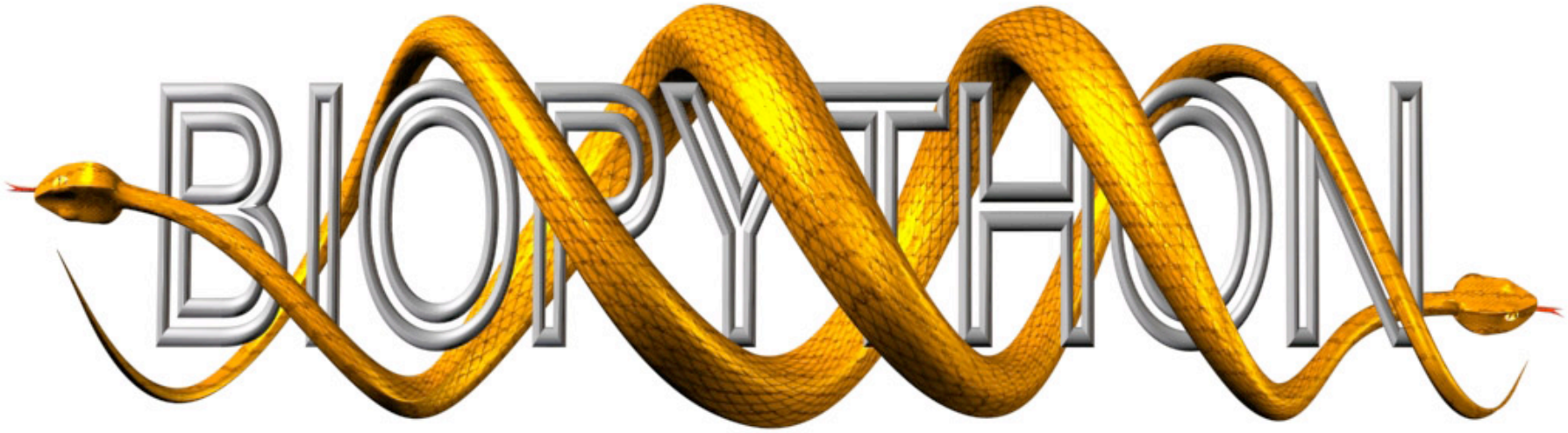


Second generation sequence data and Biopython

Peter Cock, NextGenBUG meeting 2 June 2009, held at SCRI, Dundee, UK

This talk was at a Scottish Bioinformatics User Group meeting. Explanatory comments like this try to cover what I said then. Peter.

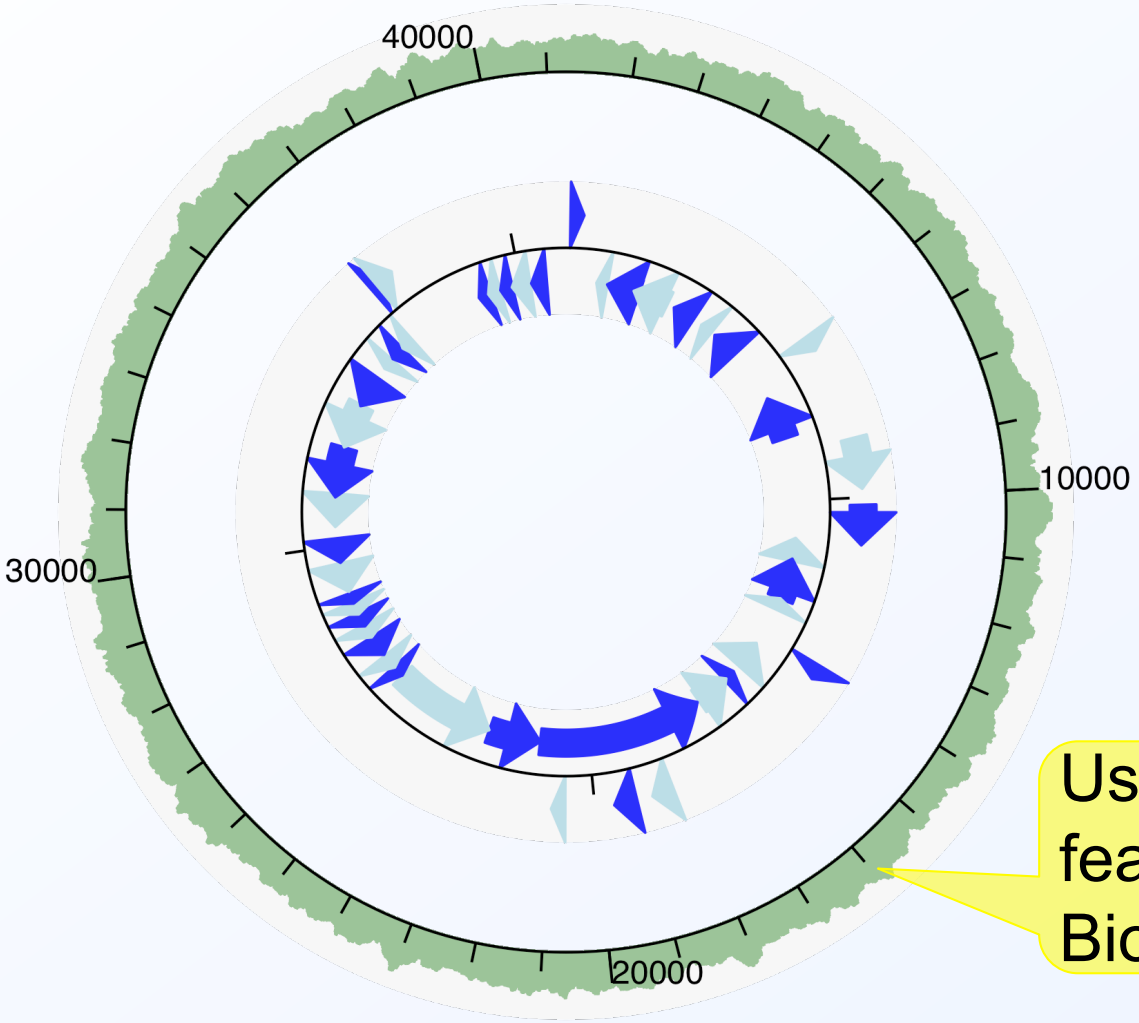
scri
living technology

Biopython

- Free, open source library for Python
  (like BioPerl, but for Python not Perl)
- Current release 1.50 supports Python 2.3 to 2.6
- Runs on Windows, Linux, Mac OS X, etc
- Extensive “Biopython Tutorial & Cookbook”
- Described in recent application note: Cock *et al.* 2009 *Bioinformatics 25(11), 1422-1423.*
- *See www.biopython.org for details*

scri
living technology

GenomeDiagram and Biopython



- Roche 454 read depth mapped onto 42kb phage
- Read depth ~100, scale max 200

Using GenomeDiagram features not yet in the Biopython Tutorial ...

scri
living technology

FASTA, QUAL and FASTQ

- Two main file formats used for read quality scores per base (in sequence space):
  - FASTA + QUAL, two paired files
  - FASTQ, single standalone files
- Solexa/Illumina have FASTQ variant using a different quality scaling (also supported in Biopython)

Now two variants – see Biopython 1.51

- More complicated in flow space (Roche 454), or colour space (SOLiD)

scri
living technology

Example FASTA and QUAL files

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

```
>FL3BO7415JACDX
33 33 33 33 17 17 21 17 28 16 16 16 16 37 37 38 38 38 39 39
39 39 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
40 40 40 40 40 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 38 38 38 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 38 38 38 38 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 38 33 33 33 34 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 38 38 38 38
38 38 38 40 40 40 40 40 40 40 38 38 38 40 40 40 38 38 38 40
40 38 38 38 38 31 32 32 32 32 32 30 28 28 28 31 31 31 30 30
30
>FL3BO7415I7AFR
...
```

- Records come in pairs using same identifier, e.g. FL3BO7415JACDX
- **Sequence** in FASTA file, e.g, TTAAT...CA
- **PHRED quality values** in QUAL file, e.g. 33, 33, …

scri
living technology

Corresponding example FASTQ file

```
@FL3BO7415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAAGTACGAGCAGATACTCCAATCGCAATTGTTTCTTC
ATTTAAAATTAGCTCGTCGCCACCTTCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTTGGATGATATT
TAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTCATCTTATTTATCGTTAAGCCA
+
BBBB2262=1111FFGGGHHHHIIIIIIIIIIIIIIIIIIIIIIIFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGGFFFFFFFFFFFFFFFGB
BBCFFFFFFFFFFFFFFFFFFFFFFFFGGGGGGGIIIIIIIGGGIIIGGGIIGGGG@AAAAA?===@@@???
@FL3BO7415I7AFR
...
```

- **Identifier**, e.g. FL3BO7415JACDX
- **Sequence**, e.g. TTAAT...CA
- **PHRED quality values**, e.g. 33, 33, 33, 33, 17, ..., 30, 30 using an ASCII encoding with offset 33.

scri living technology

Sequence input/output in Biopython

- Reading and writing sequence file formats:
  - FASTA
  - FASTQ (and the Solexa/Illumina variant)
  - QUAL
  - GenBank
  - *etc*

Actually there are two Solexa/Illumina FASTQ variants – see Biopython 1.51

- Stored as SeqRecord objects holding the sequence and annotations
- Can be used with a BioSQL database (a shared schema also used by BioPerl etc)

scri
living technology

Reading a FASTA file

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

```
from Bio import SeqIO

for rec in SeqIO.parse(open("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

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

```
from Bio import SeqIO

for rec in SeqIO.parse(open("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

FASTQ to FASTA

```
@FL3BO7415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAAGTACGAGCAGATACTCCAATCGCAATTGTTTCTTC
ATTTAAAATTAGCTCGTCGCCACCTTCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTTGGATGATATT
TAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTCATCTTATTTATCGTTAAGCCA
+
BBBB2262=1111FFGGGHHHHIIIIIIIIIIIIIIIIIIIIIIFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGGFFFFFFFFFFFFFFFGB
BBCFFFFFFFFFFFFFFFFFFFFFFFGGGGGGGIIIIIIIGGGIIIGGGIIGGGG@AAAAA?===@@@???
```

```
from Bio import SeqIO

records = SeqIO.parse(open("phage.fastq"), "fastq")

out_handle = open("phage.fasta", "w")
SeqIO.write(records, out_handle, "fasta")
out_handle.close()
```

```
>FL3BO7415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAA
GTACGAGCAGATACTCCAATCGCAATTGTTTCTTCATTTAAAATTAGCTCGTCGCCACCT
TCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTT
GGATGATATTTAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTC
ATCTTATTTATCGTTAAGCCA
```

Focus on output filename and format (“fastq”)…

scri
living technology

FASTQ to QUAL

```
@FL3BO7415JACDX
TTAATTTTATTTTGTCGGCTAAAGAGATTTTTAGCTAAACGTTCAATTGCTTTAGCTGAAGTACGAGCAGATACTCCAATCGCAATTGTTTCTTC
ATTTAAAATTAGCTCGTCGCCACCTTCAATTGGAAATTTATAATCACGATCTAACCAGATTGGTACATTATGTTTTGCAAATCTTGGATGATATT
TAATGATGTACTCCATGAATAATGATTCACGTCTACGCGCTGGTTCTCTCATCTTATTTATCGTTAAGCCA
+
BBBB2262=1111FFGGGHHHHIIIIIIIIIIIIIIIIIIIIIIFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFGGGGFFFFFFFFFFFFFFFGB
BBCFFFFFFFFFFFFFFFFFFFFFFGGGGGGGIIIIIIIGGGIIIGGGIIGGGG@AAAAA?===@@@???
```

```
from Bio import SeqIO

records = SeqIO.parse(open("phage.fastq"), "fastq")

out_handle = open("phage.qual", "w")
SeqIO.write(records, out_handle, "qual")
out_handle.close()
```

```
>FL3BO7415JACDX
33 33 33 33 17 17 21 17 28 16 16 16 16 37 37 38 38 38 39 39
39 39 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
40 40 40 40 40 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 38 38 38 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 38 38 38 38 37 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 38 33 33 33 34 37 37 37 37 37 37 37
37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 38 38 38 38
38 38 38 40 40 40 40 40 40 40 38 38 38 40 40 40 38 38 38 40
40 38 38 38 38 31 32 32 32 32 32 30 28 28 28 31 31 31 30 30
30
```

Just filename and format changed (“fastq” to “qual”)

scri living technology

Filtering a FASTQ file

```
from Bio import SeqIO

good_reads = (rec for rec in \
    SeqIO.parse(open("phage.fastq"), "fastq") if \
    min(rec.letter_annotations["phred_quality"])>=20)

out_handle = open("good_quality.fastq", "w")
SeqIO.write(good_reads, out_handle, "fastq")
out_handle.close()
```

This shows a very crude quality selection criteria for illustrative purposes.

This could be generalised of course.

scri
living technology

Trimming a FASTQ file

```
from Bio import SeqIO

trimmed= (rec[:40] for rec in \
          SeqIO.parse(open("phage.fastq"), "fastq"))

out_handle = open("read_starts.fastq", "w")
SeqIO.write(trimmed, out_handle, "fastq")
out_handle.close()
```

This shows a very crude trimming criteria for illustration (taking the first 40 bases of each read).

Alternatively, you might want to look for primer sequences to remove, or do quality trimming.

scri
living technology

Roche 454 SFF files

- Roche 454 sequencers produce SFF files
- Hold original flow space information *and* sequence space representation
- Roche tools allow simple SFF to FASTA and QUAL conversion (with or without trimming)
- Next release of Biopython may be able to read directly from a Roche 454 SFF file
- Would allow simple SFF file manipulation without the (Linux only) Roche tools

Biopython 1.52?

scri
living technology

Questions to audience

- What kinds of things to you do with FASTQ, FASTA+QUAL and SFF read files?

  I'm interested in possible use cases for Biopython

- Does anyone have any paired end data they can share for testing purposes?

  I'd like a small subset as a distributable test case

- If you work directly with SFF files, do you use the trimmed or untrimmed sequences?

scri
living technology

Acknowledgements

- Other Biopython contributors & developers
- Open Bioinformatics Foundation (OBF) supports Biopython (and BioPerl etc)
- My early contributions to Biopython were while funded by an EPSRC PhD studentship at the MOAC Doctoral Training Centre, University of Warwick.
- My recent contributions to Biopython (e.g. FASTQ and QUAL) were while employed by SCRI

scri
living technology

Questions from audience?

- If you think of something later, please ask me by email, or on the Biopython mailing list
- See www.biopython.org for details

Some real world examples from earlier talks at this meeting actually matched my examples.

Quality trimming at the start and ends was also mentioned as a common use case.

Note I avoided talking about the Solexa and Illumina FASTQ variants here.

scri
living technology