Master of Science in
Genomics and Bioinformatics

Lecture 9-1
Biopython

GNBF5010

Instructor: Jessie Y. Huang

Learning Objectives

- The `Biopython` package
- `Seq` objects and their methods
- `SeqRecord` objects and the data fields
- `SeqIO` for reading and writing sequence objects
- Direct access to GenBank with `Entrez.efetch`

biopython

- An integrated collection of modules for "biological computation"
  - DNA/protein sequences
  - sequence alignments
  - population genetics
  - molecular structures
- Also provides interfaces to common biological databases (e.g. GenBank) and to some common locally installed software (e.g. BLAST).
- Loosely based on BioPerl
- See more at https://biopython.org/

The Seq object

- A Seq object can be a DNA, RNA, or protein sequence
- Contains the sequence and a pre-defined alphabet
  - The alphabets are defined objects such as IUPACAmbiguousDNA or IUPACProtein, which are defined in the Bio.Alphabet module
  - A Seq object with a DNA alphabet has some different methods than one with an Amino Acid alphabet

```
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import IUPAC
>>> my_seq = Seq('AGTACACTGGT', IUPAC.unambiguous_dna)
>>> my_seq
Seq('AGTACACTGGT', IUPAC.unambiguous_dna())
>>> print(my_seq)
AGTACACTGGT
```

Seq objects have string methods

- Seq objects have methods that work just like string objects
- You can get the len() of a Seq, slice it, and count() specific letters in it

```
>>> my_seq = Seq('GATCGATGGGCCTATATAGGATCGAAAATCGC',
IUPAC.unambiguous_dna)
>>> len(my_seq)
32
>>> print(my_seq[6:9])
TGG
>>> my_seq.count("G")
9
```

Seq objects have special methods

- DNA Seq objects can translate its sequence into proteins

```
>>> coding_dna=Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG",
IUPAC.unambiguous_dna)
>>> coding_dna.translate()
Seq('MAIVMGR*KGAR*', HasStopCodon(IUPACProtein(), '*'))
>>> print(coding_dna.translate(table=2, to_stop=True))
MAIVMGRWKGAR
```

- Seq objects with a DNA alphabet have reverse_complement() method:

```
>>> my_seq = Seq('TTTAAAATGCGGG', IUPAC.unambiguous_dna)
>>> print(my_seq.reverse_complement())
CCCGCATTTTAAA
```

- The Bio.SeqUtils module has some useful methods, such as GC() to calculate % of G+C bases in a DNA sequence.

```
>>> from  Bio.SeqUtils  import  GC
>>> GC(my_seq)
46.875
```

See more at https://biopython.org/wiki/Seq

The SeqRecord object

- Like a database record (e.g. GenBank)
- Contains a Seq object and some annotation fields (or “attributes”) like seq, id, name, description, annotaion, features, and dbxrefs

```
>>> from Bio.Seq import Seq
>>> from Bio.SeqRecord import SeqRecord
>>> test_seq = Seq('GATC')
>>> test_record = SeqRecord(test_seq, id='xyz')
>>> test_record.description = 'This is only a test'
>>> print(test_record)
ID: xyz
Name: <unknown name>
Description: This is only a test
Number of features: 0
Seq('GATC', Alphabet())
>>> print(test_record.seq)
GATC
```

See more at https://biopython.org/wiki/SeqRecord12

The SeqIO object

- The all-purpose file read/write tool for SeqRecords
- Can read/write many file types: fasta, gbk, embl, and etc.
- Its read() and write() methods take filename as argument and no need to “open” file first

```
>>> from Bio import SeqIO
>>> gene = SeqIO.read("NC_005816.fna", "fasta")
>>> gene.id
'gi|45478711|ref|NC_005816.1|'
>>> gene.seq
Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...
CTG', SingleLetterAlphabet())
>>> len(gene.seq)
9609
```

Multiple FASTA records in one file

- A single FASTA format file can store many sequences
- `SeqIO.parse()` reads the records one by one and returns a list of SeqRecord objects

```
from Bio import SeqIO

with open("seqs.fas", "r")  as fh:
    seq_list = list(SeqIO.parse(fh, "fasta"))
print(seq_list[0].seq)   #shows the first sequence in the list
```

See more at https://biopython.org/wiki/SeqIO

Direct access to GenBank

- The Entrez module uses the NCBI Efetch service
- Efetch works on many NCBI databases including protein and PubMed literature citations
- The 'gb' data type (or genbank) contains much more annotation information, but rettype='fasta' also works
- With a few tweaks, this code could be used to download a list of GenBank ID's and save them as FASTA or GenBank files:

```
from Bio import Entrez
from Bio import SeqIO
Entrez.email = "myemail@cuhk.edu.hk" # NCBI requires your identity
handle = Entrez.efetch(db="nucleotide", id="186972394",
                       rettype="gb", retmode="text")
record = SeqIO.read(handle, "genbank")
```

```
ID: EU490707.1

Name: EU490707

Description: Selenipedium aequinoctiale maturase K (matK) gene, partial cds; chloroplast

Number of features: 3

/molecule_type=DNA

/topology=linear

/data_file_division=PLN

/date=26-JUL-2016

/accessions=['EU490707']

/sequence_version=1

/keywords=['']

/source=chloroplast Selenipedium aequinoctiale

/organism=Selenipedium aequinoctiale

/taxonomy=['Eukaryota', 'Viridiplantae', 'Streptophyta', 'Embryophyta', 'Tracheophyta',
'Spermatophyta', 'Magnoliophyta', 'Liliopsida', 'Asparagales', 'Orchidaceae',
'Cypripedioideae', 'Selenipedium']

/references=[Reference(title='Phylogenetic utility of ycf1 in orchids: a plastid gene
more variable than matK', ...), Reference(title='Direct Submission', ...)]

Seq('ATTTTTTACGAACCTGTGGAAATTTTTGGTTATGACAATAAATCTAGTTTAGTA...GAA', IUPACAmbiguousDNA())
```

References

- https://biopython.org/wiki/Documentation
- The Biopython tutorial & cookbook
  http://biopython.org/DIST/docs/tutorial/Tutorial.html
- http://fenyolab.org/presentations/Introduction_Biostatistics_Bioinformatics_2015/