

INTRODUCTION TO PYTHON: BIOPYTHON SUPPLEMENT

STEPHANIE SPIELMAN

BIG DATA IN BIOLOGY SUMMER SCHOOL, 2015

CENTER FOR COMPUTATIONAL BIOLOGY AND BIOINFORMATICS

UNIVERSITY OF TEXAS AT AUSTIN

BIOPYTHON IS A STANDARD BIOINFORMATICS TOOL

- **Convenient for parsing and manipulating
sequence data**

BIOPYTHON IS A STANDARD BIOINFORMATICS TOOL

- Convenient for parsing and manipulating sequence data
- Epic tutorial linked on the course website:
<http://biopython.org/DIST/docs/tutorial/Tutorial.html>
- And this wiki: <http://biopython.org/wiki/Biopython>

THE LANGUAGE OF BIOPYTHON

- **Sequences are stored as Seq objects**
 - Simply the sequence string and the alphabet

THE LANGUAGE OF BIOPYTHON

- **Sequences are stored as Seq objects**
 - Simply the sequence string and the alphabet

```
from Bio.Seq import Seq
from Bio.Alphabet import * # not always needed
```

```
# Define a Seq object
my_seq = Seq("ACTAGACAA")
my_seq = Seq("ACTAGACAA", IUPACAmbiguousDNA)
```

THE LANGUAGE OF BIOPYTHON

- **Sequences are stored as Seq objects**
 - Simply the sequence string and the alphabet

```
from Bio.Seq import Seq
from Bio.Alphabet import * # not always needed
```

```
# Define a Seq object
my_seq = Seq("ACTAGACAA")
my_seq = Seq("ACTAGACAA", IUPACAmbiguousDNA)
```

```
# Manipulate with various methods. Use dir() for more!
my_seq.translate()
my_seq.transcribe()
my_seq.complement()
my_seq.tomutable()
```

THE LANGUAGE OF BIOPYTHON

- Sequence *records* are stored as SeqRecord objects

THE LANGUAGE OF BIOPYTHON

- **Sequence *records* are stored as SeqRecord objects**

```
from Bio.SeqRecord import SeqRecord  
from Bio.Seq import Seq
```

```
# Define a SeqRecord object, with a Seq object  
my_seq = Seq("ACTAGACAA")  
my_seqrecord = SeqRecord(my_seq, id = "unique_id")
```


THE LANGUAGE OF BIOPYTHON

- **Sequence *records* are stored as SeqRecord objects**

```
from Bio.SeqRecord import SeqRecord
from Bio.Seq import Seq
```

```
# Define a SeqRecord object, with a Seq object
my_seq = Seq("ACTAGACAA")
my_seqrecord = SeqRecord(my_seq, id = "unique_id")
```

```
# Attributes and methods of SeqRecord objects (again,
dir() )
my_seqrecord.id
my_seqrecord.seq
my_seqrecord.description
```

SEQ OBJECTS ARE ***NOT*** STRINGS!

```
my_seq = Seq("ACTAGACAA")
```

```
# Can re-cast to a string, as needed  
my_string_seq = str(my_seq)
```

READING SEQUENCE DATA

- **Two input/output BioPython modules:**
 - SeqIO for sequence files
 - AlignIO for multiple sequence alignment files

READING SEQUENCE DATA

- **Two input/output BioPython modules:**
 - SeqIO for sequence files
 - AlignIO for multiple sequence alignment files
- **Two main functions for reading:**
 - .read() if file has one sequence/alignment
 - .parse() if file has multiple sequences/alignments

BIOPYTHON I/O SYNTAX

```
from Bio import AlignIO  
from Bio import SeqIO
```

```
# .read() and .parse() take 2 arguments:
```

```
# <AlignIO/SeqIO>.<read/parse>("filename", "format")
```

BIOPYTHON READS FILES AS SEQRECORD OBJECTS



BIOPYTHON READS FILES AS SEQRECORD OBJECTS

```
from Bio import AlignIO

# Read in an alignment file
alignment_records = AlignIO.read("aln.phy", "phylip")

# We can loop over records!
for record in alignment_records:
    print record.id
    print record.seq # Consider re-casting to str()
```

BIOPYTHON READS FILES AS SEQRECORD OBJECTS

```
from Bio import AlignIO

# Read in an alignment file
alignment_records = AlignIO.read("aln.phy", "phylip")

# We can loop over records!
for record in alignment_records:
    print record.id
    print record.seq # Consider re-casting to str()
```

Remember: import the Seq, SeqRecord modules to manipulate these!

BIOPYTHON READS FILES AS SEQRECORD OBJECTS

```
from Bio import SeqIO
```

```
# Read in an file with many unaligned sequences  
seq_records = list(SeqIO.read("seqs.fasta", "fasta"))
```

BIOPYTHON READS FILES AS SEQRECORD OBJECTS

```
from Bio import SeqIO

# Read in an file with many unaligned sequences
seq_records = list(SeqIO.read("seqs.fasta", "fasta"))

# Again, loop
for rec in seq_records:
    # commands
```

CONVERTING SEQUENCE DATA FORMAT

- Easily convert between sequence file formats with `.convert()`

CONVERTING SEQUENCE DATA FORMAT

- Easily convert between sequence file formats with `.convert()`

```
from Bio import AlignIO
```

```
# Input file is FASTA, but we want PHYLIP!
```

```
#AlignIO.convert(<infile>, <informat>, <outfile>,  
outformat>)
```

```
AlignIO.convert("in.fasta", "fasta", "out.phy", "phylip")
```

WRITING SEQUENCE DATA

- Use the `.write()` method to write SeqRecord object(s) to a file

WRITING SEQUENCE DATA

- Use the `.write()` method to write `SeqRecord` object(s) to a file

```
from Bio import SeqIO
```

```
SeqIO.write(<record(s)>, <outfile>, <outformat>)
```

WRITING SEQUENCE DATA

- Use the `.write()` method to write `SeqRecord` object(s) to a file

```
from Bio import SeqIO
```

```
SeqIO.write(<record(s)>, <outfile>, <outformat>)
```

- For various reasons, just use `SeqIO` for writing

WRITING A SINGLE SEQUENCE TO A FILE

```
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import SeqIO

# Define a SeqRecord object
seq_object = Seq("ACGTC")
seq_record = SeqRecord(seq_object, id = "my_id")

# Write it to a file
SeqIO.write(seq_record, "outfile.fasta", "fasta")
```


WRITING MULTIPLE SEQUENCES TO A FILE

```
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import SeqIO

# Save lots of SeqRecord objects in a list!
recs = []
for i in range(10):
    seq_object = Seq(<some sequence>)
    seq_record = SeqRecord(seq_object, id = <some_id>)
    recs.append(seq_record)

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

SCRIPTING EXERCISES

Goal:

- **Parse sequence data files and extract meaningful information**
- **Perform some calculations**

SCRIPT #1

Determine the average *GC-content* for all sequences in a given file

SCRIPT #1

Determine the average *GC-content* for all sequences in a given file

- Your script should include two functions:
 - A function for reading in a sequence file
 - A function for computing the GC-content for a *single sequence* (why?)

SCRIPT #2

Determine the *pairwise distance* between two sequences

SCRIPT #2

Determine the *pairwise distance* between two sequences

ACGTAAA
AGGTAAT

SCRIPT #2

Determine the *pairwise distance* between two sequences

ACGTAA
AGGTAAT

$$\text{Distance} = \# \text{diff} / \text{length} = 2 / 7 = 0.286$$

SCRIPT #2

Determine the *pairwise distance* between two sequences

- Your script should include two functions:
 - A function for reading in a sequence file
 - A function for computing pair-wise distances

SCRIPT #2

Determine the *pairwise distance* between two sequences

- Your script should include two functions:
 - A function for reading in a sequence file
 - A function for computing pair-wise distances
- Important considerations:
 - Allow for different sequence file formats
 - The sequences must be the same length

SCRIPT #2

Determine the *pairwise distance* between two sequences

- Your script should include two functions:
 - A function for reading in a sequence file
 - A function for computing pair-wise distances
- Important considerations:
 - Allow for different sequence file formats
 - The sequences must be the same length (how can we enforce this?)