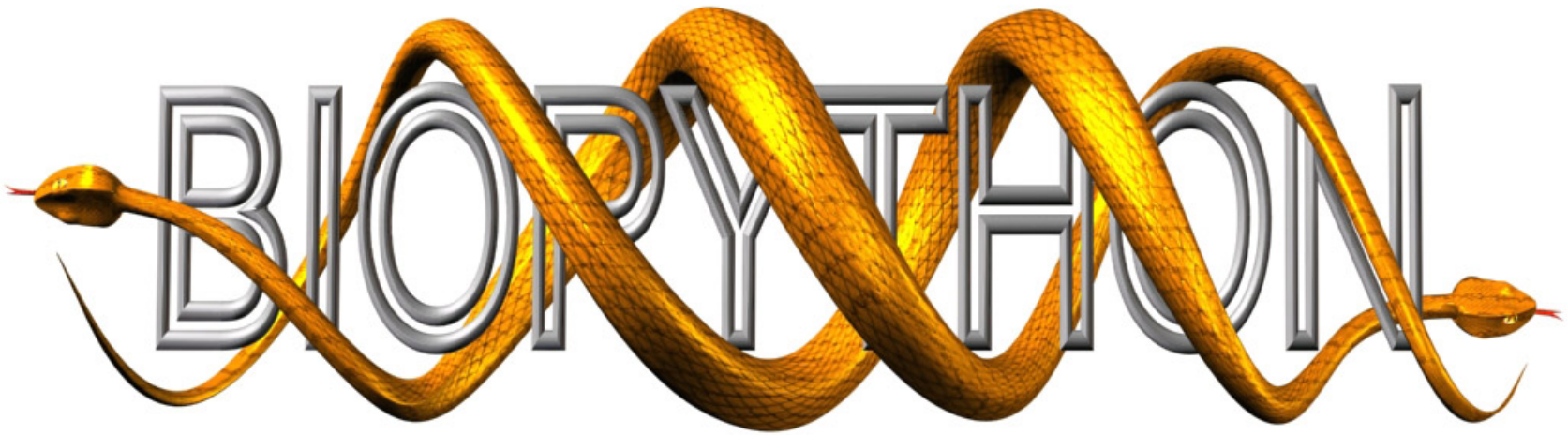
Scriptsprachen

Biopython

Sascha Winter

Lehrstuhl fuer Bioinformatik
Friedrich-Schiller-Universitaet Jena
sascha.winter@uni-jena.de

13.08.2014


BIOPYTHON

For what?

- Blast output
- Clustalw
- FASTA
- GenBank
- and more...

http://biopython.org/DIST/docs/tutorial/Tutorial.html

Read FASTA

```
from Bio import SeqIO

for seq_record in SeqIO.parse(datei, "fasta"):
    print seq_record.id
    print repr(seq_record.seq)
    print len(seq_record)
```

Read GenBank

```
from Bio import SeqIO

for seq_record in SeqIO.parse(datei, "genbank"):
    print seq_record.id
    print repr(seq_record.seq)
    print len(seq_record)
```

Random Access

```
records = list(SeqIO.parse(datei, "genbank"))
# liste

records = \
    SeqIO.to_dict(SeqIO.parse(datei, "genbank"))
# dict, id is key

# Take much memory on large files

records = SeqIO.index(datei, "genbank")
# dict like, but only reading when accessing
# only for some file formats
```

Write

```
from Bio import SeqIO

records = list(SeqIO.parse(datei, "genbank"))

with open(file, "w") as handle:
    SeqIO.write(records, handle, "fasta")

Bio.SeqIO.convert(...)
# direct format conversion
```

Sequences

Sequences

```
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC

my_seq = Seq("AGTACACTGGT") # Better with alphabet

my_seq = Seq("AGTACACTGGT", IUPAC.unambiguous_dna)

my_prot = Seq("AGTACACTGGT", IUPAC.protein)

my_prot[4:8]  # Like strings
Seq('CACT', IUPACProtein())
```

Sequences

```
protein_seq = Seq("EVRNAK", IUPAC.protein)
protein_seq2 = Seq("KANRVE", IUPAC.protein)

protein_seq + protein_seq2
Seq('EVRNAKKANRVE', IUPACProtein())

dna_seq = Seq("ACGT", IUPAC.unambiguous_dna)

protein_seq + dna_seq # TypeError

protein_seq.alphabet = generic_alphabet
```

Sequences

```
dna_seq.upper()

dna_seq.lower()

dna_seq.count("G")

dna_seq.complement()
dna_seq.reverse_complement()
# Use alphabet, Proteins have no complement
```

Sequences

```
coding_dna.transcribe()
messenger_rna.back_transcribe()

coding_dna.translate()
Seq('MAIVMGR*KGAR*', \
    HasStopCodon(IUPACProtein(), '*'))

coding_dna.translate\
    (table="Vertebrate Mitochondrial")

coding_dna.translate(to_stop=True)
Seq('MAIVMGR', IUPACProtein())
```

Sequences

```
seq1 = Seq("ACGT", IUPAC.unambiguous_dna)
seq2 = Seq("ACGT", IUPAC.unambiguous_dna)

seq1 == seq2 # Tests if they are same object
False        # But will change in the future

str(seq1) == str(seq2) # Better use sequence string
True                   # as dict key
```

Annotations

SeqRecord

```
from Bio.SeqRecord import SeqRecord
.seq # the sequence, a Seq
.id # id string
.description # human readable
.features # list of SeqFeature
```

And some more

SeqFeature

```
.type # 'CDS', 'gene', ...

.qualifiers # dict, GenBank feature table

sub_record = record[4300:4800]
# Keeps only features, that are fully in the slice
```

Location

```
.location # The location
# Start and End position + strand

# CompoundLocation for join locations
```

Location

```
start_pos = SeqFeature.AfterPosition(5)

end_pos = SeqFeature.BetweenPosition(9, \
    left=8, right=9)

my_location = SeqFeature.FeatureLocation(start_pos,
    end_pos)
[>5:(8^9)]

SeqFeature.ExactPosition

SeqFeature.BeforePosition

SeqFeature.OneOfPosition

SeqFeature.UnknownPosition
```

Location

```
# New Biopython
CompundLocation

# Old Biopython
# list of features:
sub = [SeqFeature(FeatureLocation(1,5), type="CDS",
     SeqFeature(FeatureLocation(10,15), type="CDS",

# to a joint feature
seq_feat = SeqFeature(FeatureLocation(1,15), type="

seq_feat.sub_features = sub
seq_feat.location_operator = "join"
```

Alignments

IO

```
from Bio import AlignIO

alignment = AlignIO.read(file, "stockholm")
# Single alignment

alignments = AlignIO.parse("resampled.phy",\
    "phylip")
# Many Alignments

AlignIO.parse(handle, "fasta", seq_count=2)
# Alignments from fasta, have to have same length
# seq_count = # sequences per alignment
```

IO

```
from Bio import AlignIO

AlignIO.write(my_alignments, file, "phylip")

AlignIO.convert("PF05371_seed.sth", "stockholm",\
    "PF05371_seed.aln", "clustal")
```

IO

```
for record in alignment: # Each row as SeqRecord
    print record

alignment[2].seq[6] == alignment[2,6]
# Get single position
```

NCBI Access

NCBI Access

```
from Bio import Entrez
Entrez.email = "A.N.Other@example.com"
# Informations about problems, no direct ip ban

handle = Entrez.einfo()
record = Entrez.read(handle)
# dict DbList -> ['pubmed', 'protein', ...]

# Bio.Entrez.Parser.ValidationError:
record = Entrez.read(handle,validate=False)
```

NCBI Access

```
handle = Entrez.einfo(db="pubmed")
record = Entrez.read(handle)

for field in record["DbInfo"]["FieldList"]:
    print "%(Name)s, %(FullName)s,\
        %(Description)s" % field
# list of all search fields
```

NCBI Access

```
handle = Entrez.esearch(db="nucleotide",\
    term="Cypripedioideae[Orgn] AND matK[Gene]")

record = Entrez.read(handle)

record["Count"]
'25'

record["IdList"]
['126789333', '37222967', '37222966',\
    '37222965', ..., '61585492']
```

Fetch GB Record

```
from Bio import Entrez

Entrez.email = "A.N.Other@example.com"

handle = Entrez.efetch(db="nucleotide",\
    id="186972394", rettype="gb", retmode="text")

print handle.read()

# rettype="fasta" for fasta
```

Fetch GB Record

```
from Bio import Entrez

Entrez.email = "A.N.Other@example.com"

handle = Entrez.efetch(db="nucleotide",\
    id="186972394", rettype="gb", retmode="text")

record = SeqIO.read(handle, "genbank")

handle.close()
```