Biopython

Andrea Passerini
passerini@disi.unitn.it

Informatica

Biopython

Descrizione

- Il progetto Biopython è un'associazione di sviluppatori di codice Python liberamente disponibile per bioinformatica
- La homepage del progetto è `http://www.biopython.org`
- Il codice viene fornito come una serie di librerie Python
- Le librerie forniscono tra l'altro:
  - oggetti sequenza specificatamente sviluppati per sequenze biologiche
  - funzioni di parsing di formati di file comunemente usati in bioinformatica
  - funzioni per l'allineamento di sequenze
  - funzioni di interfacciamento ai principali database bioinformatici online
- Vedremo alcuni esempi delle funzionalità di Biopython. Altri probabilmente li scoprirete durante il corso di studi

Installazione

- Biopython può essere facilmente scaricato ed installato a partire dalla homepage del progetto
- Il modo più semplice per installarlo è con l'installer per pacchetti python

```
pip install biopython
```

- E' possibile installarlo anche usando il packet manager della distribuzione, ad esempio per distribuzioni Debian-like (tipo Ubuntu):

```
sudo apt-get install python-biopython
```

- E' utile installare anche la documentazione:

```
sudo apt-get install python-biopython-doc
```

Oggetti Biopython

Sequenze

- La sequenza è uno degli elementi centrali della bioinformatica
- Esistono vari tipi di sequenze: DNA, RNA, proteine, etc
- Ogni tipo ha il suo alfabeto di riferimento, e metodi specifici, nonché metodi per la conversione da un tipo all'altro (trascrizione, traduzione)
- Biopython fornisce un oggetto `Seq` che implementa tutti questi tipi di sequenza
- L'oggetto è definito nel modulo `Bio.Seq`

Sequenze

Oggetto `Seq`

- Una sequenza generica si crea con:

```
>>> from Bio.Seq import Seq
>>> s = Seq("ACCGTTTAAC")
>>> s
Seq('ACCGTTTAAC', Alphabet())
```

- Ad un oggetto sequenza è associato un alfabeto, che dipende dal tipo di sequenza in questione. Se non specificato, si usa un alfabeto generico

Sequenze

Alfabeti

- Specificare l'alfabeto di una sequenza permette di utilizzare i metodi specifici del tipo di sequenza (e.g. la stringa complemento del DNA)
- Gli alfabeti sono definiti nel modulo `Bio.Alphabet.IUPAC`
- Per creare una sequenza con uno specifico alfabeto, lo si scrive come secondo argomento:

```
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import IUPAC
>>> s = Seq("ACCGTTTAAC",
... IUPAC.unambiguous_dna)
>>> s
Seq('ACCGTTTAAC', IUPACUnambiguousDNA())
```

Sequenze

Alfabeti

- Gli alfabeti di base definiti nel modulo `IUPAC` :

  IUPAC.unambiguous_dna una sequenza di DNA completamente specificata

  IUPAC.unambiguous_rna una sequenza di RNA completamente specificata

  IUPAC.protein una sequenza proteica

Sequenze

Oggetti di tipo sequenza

- Le sequenze Biopython sono estensioni delle stringhe Python, con una serie di attributi e metodi specifici
- Sono quindi manipolabili con gli operatori definiti per le stringhe:

```
>>> from Bio.Seq import Seq
>>> s = Seq("ACCGTTTAAC")
>>> len(s)
10
>>> s[9]
'C'
>>> s[1:3] # sottosequenza Biopython
Seq('CC', Alphabet())
```

Sequenze

Oggetti immutabili

- Come le stringhe Python, le sequenze Biopython sono oggetti immutabili

```
>>> from Bio.Seq import Seq
>>> s = Seq("ACCGTTTAAC")
>>> s[9] = 'G'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: Seq instance has no
                attribute '__setitem__'
```

Somma di sequenze

- Alcuni operatori sono soggetti a restrizioni dovute al tipo di sequenza
- Ad esempio è possibile sommare sequenze dello stesso tipo o di tipo generico

```
>>> s
Seq('ACCGTTTAAC', Alphabet())
>>> s2 = Seq('AAACCCGATTA')
>>> s + s2
Seq('ACCGTTTAACAAACCCGATTA', Alphabet())
```

Somma di sequenze

- Ma sommare sequenze di tipo diverso non generico genera un errore

```
>>> s =  Seq('AAACCCGATTA', IUPAC.unambiguous_dna)
>>> s2 = Seq('AAADGHHHTEWW', IUPAC.protein)
>>> s + s2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/var/lib/python-support/python2.5/Bio/Seq.py"
                                    line 64, in __add__
    str(other.alphabet))
TypeError: ('incompatable alphabets',
'IUPACUnambiguousDNA()', 'IUPACProtein()')
```

Conversioni

- E' sempre possibile ottenere una stringa Python da una sequenza con la funzione `str()`:

```
>>> s =  Seq('AAACCCGATTA',
                IUPAC.unambiguous_dna)
>>> str(s)
'AAACCCGATTA'
```

Mutabilità

- Biopython fornisce anche oggetti sequenza `MutableSeq` di tipo mutabile
- E' possibile generare una sequenza mutabile da una immutabile con il metodo `tomutable()`

```
>>> sm = s.tomutable()
>>> sm
MutableSeq(array('c', 'AAACCCGATTA'),
           IUPACUnambiguousDNA())
>>> sm[2] = 'G'
>>> sm
MutableSeq(array('c', 'AAGCCCGATTA'),
           IUPACUnambiguousDNA())
```

Metodi specifici del tipo

Complementazione

- Nel caso di DNA ed RNA, è utile l'operazione di complementazione che rende la sequenza complementare
- Tale operazione è fornita dal metodo `complement`

```
>>> s =  Seq('CCGTTAAAAC',
... IUPAC.unambiguous_dna)
>>> s.complement()
Seq('GGCAATTTTG', IUPACUnambiguousDNA())
```

- Il `reverse_complement` fornisce il complemento della sequenza letta da destra verso sinistra

```
>>> r =  Seq('AAACCCGAUUA',
... IUPAC.unambiguous_rna)
>>> r.reverse_complement()
Seq('UAAUCGGGUUU', IUPACUnambiguousRNA())
```

Metodi specifici del tipo

Trascrizione

- La trascrizione è l'operazione con cui una stringa di DNA viene trascritta nel corrispondente RNA
- In Biopython, tale operazione viene eseguita dal metodo `transcribe`, che restituisce in uscita la corrispondente sequenza di RNA

```
>>> coding_dna =  Seq('CCGTTAAAAC',
... IUPAC.unambiguous_dna)
>>> coding_dna.transcribe()
Seq('CCGUUAAAAC', IUPACUnambiguousRNA())
```

Attenzione

`transcribe` semplicemente sostituisce *T* con *U*, quindi si applica sul DNA codificante. Per trascrivere da DNA template si deve fare:

```
template_dna.reverse_complement().transcribe()
```

Metodi specifici del tipo

Trascrizione

- Il metodo `back_transcribe` converte da RNA messaggero a DNA codificante:

```
>>> r =  Seq('CCGUUAAAAC',
... IUPAC.unambiguous_rna)
>>> r.back_transcribe()
Seq('CCGTTAAAAC', IUPACUnambiguousDNA())
```

- Applicare la trascrizione a sequenze del tipo sbagliato genera un errore:

```
>>> prot = Seq('MADAGHHFDCE', IUPAC.protein)
>>> prot.transcribe()
[...]
ValueError: Proteins cannot be transcribed!
```

Metodi specifici del tipo

Traduzione

- La traduzione è l'operazione con cui una stringa di RNA (messaggero) viene tradotta in una sequenza proteica
- In Biopython, tale operazione viene eseguita dal metodo `translate`

```
>>> rna = Seq('GCCAUUGUAAUGGGCCGC',
... IUPAC.unambiguous_rna)
>>> rna.translate()
Seq('AIVMGR', ExtendedIUPACProtein())
```

- Il metodo `translate` si può anche applicare direttamente a DNA codificante

```
>>> dna = Seq('GCCATTGTAATGGGCCGC',
... IUPAC.unambiguous_dna)
>>> dna.translate()
Seq('AIVMGR', ExtendedIUPACProtein())
```

Metodi specifici del tipo

Traduzione

- La traduzione si basa su una tabella di traduzione che associa triplette di DNA (e quindi corrispettivo RNA) a singoli aminoacidi.
- Esistono tabelle diverse a seconda del tipo di DNA in questione (oltre al DNA della cellula, anche alcuni organelli quali i mitcondri hanno un DNA proprio)
- E' possibile specificare tabelle diverse da quella standard come argomento di `translate`

```
>>> dna = Seq("GTAATGGGCCGCTGAAAGGGTGCC",
... IUPAC.unambiguous_dna)
>>> dna.translate()
Seq('VMGR*KGA', HasStopCodon(IUPACProtein(), '*'))
>>> dna.translate(table="Vertebrate Mitochondrial")
Seq('VMGRWKGA', IUPACProtein())
```

Metodi specifici del tipo

Traduzione

```
>>> dna = Seq("GTAATGGGCCGCTGAAAGGGTGCC",
... IUPAC.unambiguous_dna)
>>> dna.translate()
Seq('VMGR*KGA', HasStopCodon(IUPACProtein(), '*'))
```

- Il tipo della sequenza risultante specifica che è una proteina con un codone di terminazione
- Il codone di terminazione viene convertito in '*' e la traduzione continua con i nucleotidi successivi
- Per tradurre una sequenza solo fino al codone di terminazione (se presente) is usa l'argomento `to_stop`:

```
>>> dna.translate(to_stop=True)
Seq('VMGR', IUPACProtein())
```

Input ouput

Parsing di file

- Le sequenze biologiche vengono tipicamente scritte su file di testo arricchiti con annotazioni ed informazioni varie.
- Esistono vari formati di file di sequenze biologiche (ed altro materiale in biologia computazionale), tipicamente a seconda dell'entità che ha stabilito il formato e creato il database, e del tipo di informazioni in esso contenute.
- Le sequenze biologiche recuperabili dai grandi database disponibili online sono fornite in uno di questi formati, e riuscire a maneggiarli è indispensabile per poter lavorare con tali sequenze.

Formati comuni

fasta

è un formato molto semplice per scrivere elenchi di sequenze, ciascuna con una sua intestazione (preceduta da >). Utilizzato come formato di ingresso da molti programmi per analisi di sequenze, allineamento etc

```
>sp|P25730|FMS1_ECOLI CS1 fimbrial subunit A precursor (CS1 pilin)
MKLKKTIGAMALATLFATMGASAVEKTISVTASVDPTVDLLQSDGSALPNSVALTYSPAV
NNFEAHTINTVVHTNDSDKGVVVKLSADPVLSNVLNPTLQIPVSVNFAGKPLSTTGITID
SNDLNFASSGVNKVSSTQKLSIHADATRVTGGALTAGQYQGLVSIILTKST

>sp|P15488|FMS3_ECOLI CS3 fimbrial subunit A precursor (CS3 pilin)
MLKIKYLLIGLSLSAMSSYSLAAAGPTLTKELALNVLSPAALDATWAPQDNLTLSNTGVS
NTLVGVLTLSNTSIDTVSIASTNVSDTSKNGTVTFAHETNNSASFATTISTDNANITLDK
NAGNTIVKTTNGSQLPTNLPLKFITTEGNEHLVSGNYRANITITSTIK

>sp|P33781|FMS5_ECOLI CS5 fimbrial subunit precursor (CS5 pilin)
MKKNLLITSVLAMATVSGSVLAAVTNGQLTFNWQGVVPSAPVTQSSWAFVNGLDIPFTPG
TEQLNITLDSNKDITARSVKPYDFFIVPVSGNVTPGAPVTRDTSANINSVNAFLSSVPVS
NGFVGNKQLTLSTAVEAAKGEVAITLNGQALKVGSASPTVVTVASNKKESHISIDMNAKA
```

Formati comuni

genbank

è un formato più ricco, che contiene campi appositi per vari tipi di annotazioni, quali l'organismo in cui la sequenza si trova, i riferimenti bibliografici ad essa relativi, etc. E' associato al database omonimo, che è il principale deposito di sequenze genomiche annotate

```
LOCUS       SCU49845     5028 bp    DNA             PLN       21-JUN-1999
DEFINITION  Saccharomyces cerevisiae TCP1-beta gene, partial cds, and Axl2p
            (AXL2) and Rev7p (REV7) genes, complete cds.
ACCESSION   U49845
VERSION     U49845.1  GI:1293613
KEYWORDS    .
SOURCE      Saccharomyces cerevisiae (baker's yeast)
  ORGANISM  Saccharomyces cerevisiae
            Eukaryota; Fungi; Ascomycota; Saccharomycotina; Saccharomycetes;
            Saccharomycetales; Saccharomycetaceae; Saccharomyces.
REFERENCE   1  (bases 1 to 5028)
  AUTHORS   Torpey,L.E., Gibbs,P.E., Nelson,J. and Lawrence,C.W.
  TITLE     Cloning and sequence of REV7, a gene whose function is required for
            DNA damage-induced mutagenesis in Saccharomyces cerevisiae
  JOURNAL   Yeast 10 (11), 1503-1509 (1994)
  PUBMED    7871890
FEATURES             Location/Qualifiers
     source          1..5028
                     /organism="Saccharomyces cerevisiae"
```

Formati comuni

swiss

è il formato del database UniProt (ottenuto combinando Swiss-Prot, TrEMBL and PIR). E' il corrispettivo di GenBank per le proteine: il più grande database di sequenze proteiche annotate

```
ID   143B_HUMAN     STANDARD;      PRT;   245 AA.
AC   P31946;
DT   01-JUL-1993 (Rel. 26, Created)
DT   01-FEB-1996 (Rel. 33, Last sequence update)
DT   15-SEP-2003 (Rel. 42, Last annotation update)
DE   14-3-3 protein beta/alpha (Protein kinase C inhibitor protein-1)
DE   (KCIP-1) (Protein 1054).
GN   YWHAB.
OS   Homo sapiens (Human).
OC   Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
OC   Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
OX   NCBI_TaxID=9606;
RN   [1]
RP   SEQUENCE FROM N.A.
RC   TISSUE=Keratinocytes;
RX   MEDLINE=93294871; PubMed=8515476;
RA   Leffers H., Madsen P., Rasmussen H.H., Honore B., Andersen A.H.,
RA   Walbum E., Vandekerckhove J., Celis J.E.;
RT   "Molecular cloning and expression of the transformation sensitive
RT   epithelial marker stratifin. A member of a protein family that has
RT   been involved in the protein kinase C signalling pathway.";
```

Input

Parsing

- Le funzioni di input output sono fornite nel modulo `Bio.SeqIO`
- In particolare la funzione `parse` prende come ingresso un oggetto file e una stringa che indica il formato, e fornisce in uscita un oggetto iterabile
- L'oggetto fornito itera sui record del file, dove un record tipicamente corrisponde alle informazioni relative ad una sequenza
- i record letti dal file vengono memorizzati in oggetti `SeqRecord`, comprensivi di tutte le annotazioni

```
from Bio import SeqIO
f = open("sp.fasta")
for seq_record in SeqIO.parse(f, "fasta") :
    print(seq_record.id)
    print(seq_record.seq)
```

Oggetto `SeqRecord`

- L'oggetto `SeqRecord` contiene una serie di attributi (la maggior parte opzionali) relativi alla sequenza che rappresenta. I principali:

seq	la sequenza stessa, come oggetto `Seq` (obbligatorio)
id	l'id della sequenza, tipicamento usato per riferirvisi, recuperarla da dizionari, etc.
name	il nome della sequenza, più comprensibile dell'id
description	una descrizione testuale delle caratteristiche della sequenza

Oggetto `SeqRecord`: esempio

```
>>> from Bio import SeqIO
>>> f = open("sp.fasta")
>>> rec = next(SeqIO.parse(f,"fasta"))
>>> rec
SeqRecord(seq=Seq('MKLKKTIGAMALATLFATMGASAVEKTISVT
ASVDPTVDLLQSDGSALPNSVALTYSPAVNNFEAHTINTVVHTNDSDKGV
VVKLSADPVLSNVLNPTLQIPVSVNFAGKPLSTTGITIDSNDLNFASSGV
NKVSSTQKLSIHADATRVTGGALTAGQYQGLVSIILTKST',
SingleLetterAlphabet()), id='sp|P25730|FMS1_ECOLI',
name='sp|P25730|FMS1_ECOLI', description='sp|P2573
0|FMS1_ECOLI CS1 fimbrial subunit A precursor (CS1
pilin) - Escherichia coli, and Escherichia coli
O6.', dbxrefs=[])
>>> rec.id
'sp|P25730|FMS1_ECOLI'
>>> rec.name
'sp|P25730|FMS1_ECOLI'
```

Oggetto `SeqRecord`: esempio

```
>>> from Bio import SeqIO
>>> f = open("data.genbank")
>>> rec = next(SeqIO.parse(f, "genbank"))
>>> rec
SeqRecord(seq=Seq('GATCCTCCATATACAACGGTATCTCCACCTC
AGGTTTAGATCTCAACAACGGAACCATTGCCGACATGAGACAGTTAGGTA
TCGTCGAGAGTTACAAGCTAAAACGAGCAGTAGTCAGCTCTGCATCTGAA
......TGACTCAGATTCTAATTTTAAGCTATTCAATTTCTCTTTGATC',
IUPACAmbiguousDNA()), id='U49845.1', name='SCU49845',
description='Saccharomyces cerevisiae TCP1-beta gene,
partial cds, and Axl2p (AXL2) and Rev7p (REV7) genes,
complete cds.', dbxrefs=[])
```

Oggetto `SeqRecord`: attributo `annotations`

- L'attributo `annotations` contiene un dizionario delle annotazioni trovate nel fare il parsing del record da file
- Le annotazioni possono quindi essere recuperate dal dizionario specificandone il nome (che fa da chiave)

```
>>> rec.annotations.keys()
dict_keys(['sequence_version', 'source',
'taxonomy','keywords', 'references',
'accessions','data_file_division', 'date',
'organism','gi'])
>>> rec.annotations["source"]
"Saccharomyces cerevisiae (baker's yeast)"
>>> rec.annotations["taxonomy"]
['Eukaryota', 'Fungi', 'Ascomycota',
'Saccharomycotina', 'Saccharomycetes',
'Saccharomycetales', 'Saccharomycetaceae',
'Saccharomyces']
```

Dizionari di sequenze

- L'attributo `id` della sequenza deve essere (per definizione) univoco, e può essere usato come chiave per creare un dizionario di sequenze
- Il metodo `to_dict()` prende in ingresso un oggetto creato da `parse`, e restituisce un dizionario dei `SeqRecord` contenuti nel file relativo, con chiave data dall'id

```
>>> dict = SeqIO.to_dict(SeqIO.parse(f,
... "genbank"))
>>> dict.keys()
dict_keys(['U49845.1', 'AF090832.1'])
>>> dict['AF090832.1'].description
''Drosophila melanogaster muscle LIM protein
at 84B (Mlp84B) gene, complete cds'
```

Output

Scrivere sequenze su file

- Il metodo `write` permette di scrivere una sequenza su file
- Prende in ingresso un oggetto iterabile con oggetti `SeqRecord`, un oggetto file, ed il formato in cui scrivere il record.
- E' possibile specificare un formato diverso da quello del record (e `write` al momento supporta pochi formati)

```
f = open("data.dst","w")
>>> SeqIO.write([dict['U49845.1']], f, "fasta")
```

Nota

- `write` prende come primo argomento un contenitore di oggetti `SeqRecord` NON un oggetto `SeqRecord`
- ho quindi convertito al volo l'elemento del dizionario di chiave `'U49845.1'` in una lista racchiudendolo tra parentesi quadre