Exemple d’utilisation de Biopython

Analyse et comparaison de génomes procaryotes

Jennifer Becq

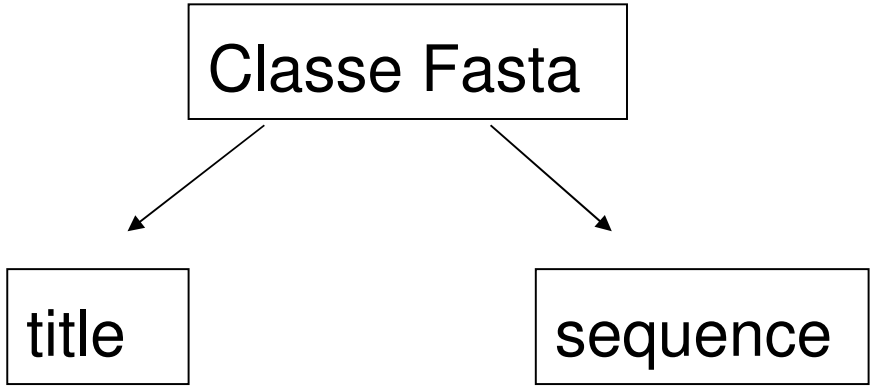
Chargement des séquences

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

Importation des modules de manipulation des séquences

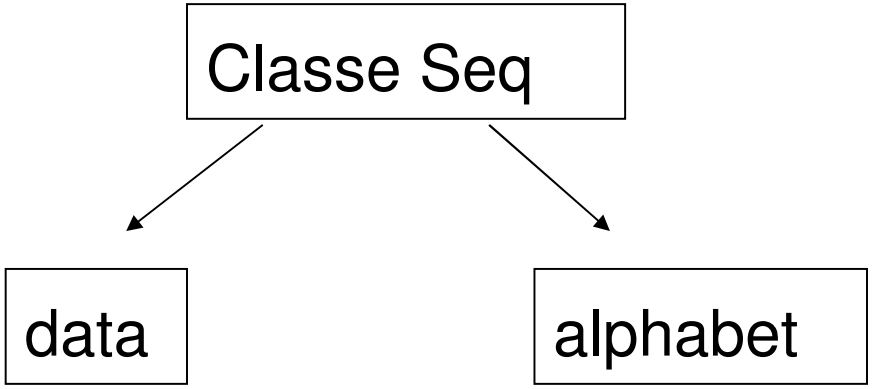
```
>>> f_ parser = Fasta.RecordParser()
>>> filin = open('maseq.fasta', 'r')
>>> lignes = string.join(filin.readlines(),'')
>>> mon_fasta = f_parser.parse(filin)
>>> mon_fasta = f_parser.parse_str(lignes)
```

Initialisation du parseur fasta / chargement de l'objet « fasta »




```
>>> mon_fasta.title
'NC_00961'
>>>mon_fasta.sequence
'aaacattcctggatca'
```

Chargement des séquences



IUPACProtein

ExtendedIUPACProtein

IUPACUnambiguousDNA

IUPACAmbiguousDNA

ExtendedIUPACDNA

id pour RNA...

```
>>> myseq =
Seq.Seq(mon_fasta.sequence,IUPAC.ambiguous_dna)
```

Formatage classe 'Seq' pour un traitement postérieur

Manipulations des séquences

1. Manipulation classique des séquences :
    - fragments
    - complémentaire
    - transcription
    - traduction

```
>>> ma_sous_seq = ma_seq[65:146]

>>> ma_seq_rev = ma_seq.reverse_complement(ma_seq)

>>> from Bio import Transcribe
>>> transcriber=Transcribe.unambiguous_transcriber
>>> mon_arn=transcriber.transcribe(ma_seq)
```

Manipulations des séquences

1. Manipulation classique des séquences :
    - fragments
    - complémentaire
    - transcription
    - traduction
2. Création d'un nouvel objet fasta

```
>>> mon_fasta = Fasta.Record()
>>> mon_fasta.sequence = mon_arn.data
>>> mon_fasta.title = 'mon arn 001'
```

Manipulations des séquences

1. Manipulation classique des séquences :
    - fragments
    - complémentaire
    - transcription
    - traduction
2. Création d'un nouvel objet fasta
3. Ecriture dans un fichier

```
>>> filout = open('mon_fichier.fasta','w')
>>> filout.write(str(mon_fasta))
>>> filout.close()
```

Manipulations des séquences

1. Manipulation classique des séquences :
    - fragments
    - complémentaire
    - transcription
    - traduction
2. Création d'un nouvel objet fasta
3. Ecriture dans un fichier
4. Transformation en chaîne de caractères

```
seq_chaine = ma_seq.tostring()
```

Recherche d'un génome donneur potentiel (1)

des blasts...

Utilisation en ligne du blast du NCBI :

```
>>> from Bio.Blast import NCBIWWW
>>> filin = open('ma_seq.fasta', 'r')
>>> f_parser = Fasta.RecordParser()
>>> mon_fasta = f_parser.parse(filin)

#### Blast
>>> bl_result = NCBIWWW.qblast('blastn',
    'nr', mon_fasta, format_type = 'Text')
```

Importation du module

Chargement de la séquence en format fasta

Utilisation de la fonction « qblast » du module NCBIWWW

Les arguments de la fonction qblast :

program	**BLASTP, BLASTN, (BLASTX, TBLASTN or TBLASTX)**
database	**banque de données à utiliser ('nr','refseq',...)**
sequence	**la séquence à rechercher**
ncbi_gi	TRUE/[FALSE] selon si on veut l'identificateur 'gi'
descriptions	nb de descriptions à afficher [500]
alignments	nb d'alignements à afficher [500]
expect	valeur de cutoff de E [10.0]
matrix	matrice de substitution (PAM30/70, BLOSUM80/45)
filter	filtre [None] (à choisir 'seg' ou 'dust')
format_type	sortie ['XML'] ou 'Text', 'ASN.1', 'HTML'

Recherche de fichiers genbank :

```
>>> from Bio.EUtils import HistoryClient

>>> client = HistoryClient.HistoryClient()
```

Requête NCBI « *search* »

```
>>> liste_id = client.search("Archaea[ORGN] OR Bacteria[ORGN]",
db="genome")
```

Récupération des informations « efetch »

```
>>> for id in liste_id:
. . .       seq = id.efetch(retmode = "text", rettype = "gb")
. . .       gb = seq.read()
. . .       seq = id.efetch(retmode = "text", rettype = "fasta")
```

Parsing des fichiers genbank :

1. Récupération des informations utiles pour la base de données

• identifiant taxonomique	/db_xref=taxon:(*num*)
• nom *Genre espèce*	ORGANISM
• taxonomie complète	après ligne ORGANISM
• n° d'accession	ACCESSION | VERSION
• type d'ADN	/plasmid | /organelle=...
• nom de plasmide / souche	/strain | /plasmid
• séquence	entre ORIGIN et //

2. Traitement de la séquence (calcul des signatures)

3. Intégration à la base de données

Ma classe genbank :

Variables:

flat

nom

taxonomie

taxon

accession

type

origin

idorigin

Méthodes :

GetSeq

toFasta

__len__

__repr__

_cherche#Pattern

```
class genbank:
    def _chercheNom(self):
            ...
            return nom

    def __init__(self, lines=None):
            self.flat = lines
            self.nom = self._chercheNom()

    def __len__(self):
            return self._chercheLg()

    def toFasta(self):
            fasta =Fasta .Record()
            fasta.title = self.nom
            fasta.sequence = self.GetSeq()
            return fasta
```

```
>>> from MonModule import *
>>> mon_gb = genbank(lines=lignes_fic_gb)
>>> len(mon_gb)
>>> mon_fasta = mon_gb.toFasta()
```