Bioinformatique

Un des objectifs majeurs de la bioinformatique réside dans l'étude automatique de séquences, principalement de l'ADN et de protéines,...

Ces séquences sont accessibles librement et publiquement, notamment par ces deux sources :

Snippet de Wikipédia: ***UniProt***

> **UniProt** est une base de données de séquences de protéines. Son nom dérive de la contraction de *Universal Protein Resource* (base de données **uni**verselle de **pro**téines). C'est une base de données ouverte, stable et accessible en ligne, elle est issue de la consolidation de l'ensemble des données produites par la communauté scientifique. UniProt est une base annotée, hiérarchisée où chaque séquence est accompagnée d'un ensemble riche de métadonnées et de liens vers de nombreuses autres bases de données : bibliographiques, phylogénétiques, nucléotidiques... Outre la séquence en acides aminés des protéines, UniProt fournit des informations sur leur fonction et leur structure ainsi que des liens vers d'autres bases de données.
>
> UniProt combine les données des bases Swiss-Prot, TrEMBL et Protein Information Resource (PIR) et est mise à jour régulièrement. Ses données reposent entre autres sur le serveur ExPASy de l'Institut suisse de bioinformatique et celui de l'EBI. Ces serveurs proposent en particulier la recherche de séquences homologues dans la base au moyen d'outils d'alignement de séquences comme FASTA ou BLAST.

Voir aussi le site https://www.uniprot.org/

fr:GenBank

Voir aussi le site https://www.ncbi.nlm.nih.gov/genbank/

Biopython est une librairie de programmes en langage Python dédiée à l'étude de séquences (ADN, ARN, protéines). Pour utiliser cette librairie, elle doit-être installée au préalable, par exemple :

- Avec la distribution Anaconda, via l'interface Anaconda-Navigator ou par la commande suivante : conda install -c conda-forge biopython

Consulter les références proposées en fin de page !

Compter les nucléotides d'une séquence ADN

Counting_DNA_Nucleotides-01.py

```
#!/usr/bin/env python
```

```
# -*- coding: utf-8 -*-
"""
On dispose d'un exemple de chaîne ADN (constituée des symboles 'A',
'C', 'G', 'T')
Le programme utilise plusieurs techniques pour donner les nombres
d'occurrences respectifs des différentes bases
"""
adn =
"AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC
"

# utilisation d'une liste et de la méthode .count()
bases = ["A","C","G","T"]
for base in bases:
    print(adn.count(base),)
print()

# Variante :
for c in 'ACGT':
    print(adn.count(c),)
print()

# variante un peu moins lisible
out = []
for c in 'ACGT':
    out.append(str(adn.count(c)))
print(' '.join(out))

# utilisation de la technique "list comprehension"
count = [adn.count(c) for c in 'ACGT']
for val in count:
    print(val,)
print()

# autre "list comprehension", avec impression formatée → version "one
line"
print("%d %d %d %d" % tuple([adn.count(X) for X in "ACGT"]))

# count "à la main", sans utilisation de fonctions/librairie
ACGT = "ACGT"
count = [0,0,0,0]
for c in adn:
    for i in range(len(ACGT)):
        if c == ACGT[i]:
            count[i] +=1
for val in count:
    print(val,)
print()

# count "à la main", avec .index()
```

```
ACGT = "ACGT"
count = [0,0,0,0]
for c in adn:
    count[ACGT.index(c)] += 1
for val in count:
    print(val,)
print()

# utilisation de la librairie collections
from collections import defaultdict
ncount = defaultdict(int)
for c in adn:
    ncount[c] += 1
print(ncount['A'], ncount['C'], ncount['G'], ncount['T'])

# collections.Counter
from collections import Counter
for k,v in sorted(Counter(adn).items()):
    print(v,)
print()

# avec un dictionnaire
freq = {'A': 0, 'C': 0, 'G': 0, 'T': 0}
for c in adn:
    freq[c] += 1
print(freq['A'], freq['C'], freq['G'], freq['T'])

# avec un dictionnaire et count(), impression différente
dico={}
for base in bases:
    dico[base] = adn.count(base)
for key,val in dico.items():
    print("{} = {}".format(key, val))
```

Trouver un motif

+ lecture de fichier

Finding_a_Protein_Motif-01.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
La description complète et les caractéristiques d'une protéine
particulière peuvent être obtenues via l'ID "uniprot_id" de la "UniProt
database", en insérant la référence dans ce lien :
http://www.uniprot.org/uniprot/uniprot_id
```

```
On peut aussi obtenir la séquence peptidique au format FASTA via le
lien :
http://www.uniprot.org/uniprot/uniprot_id.fasta
"""

from Bio import SeqIO
from Bio import ExPASy
from Bio import SeqIO

dic = {"UUU":"F", "UUC":"F", "UUA":"L", "UUG":"L",
    "UCU":"S", "UCC":"S", "UCA":"S", "UCG":"S",
    "UAU":"Y", "UAC":"Y", "UAA":"STOP", "UAG":"STOP",
    "UGU":"C", "UGC":"C", "UGA":"STOP", "UGG":"W",
    "CUU":"L", "CUC":"L", "CUA":"L", "CUG":"L",
    "CCU":"P", "CCC":"P", "CCA":"P", "CCG":"P",
    "CAU":"H", "CAC":"H", "CAA":"Q", "CAG":"Q",
    "CGU":"R", "CGC":"R", "CGA":"R", "CGG":"R",
    "AUU":"I", "AUC":"I", "AUA":"I", "AUG":"M",
    "ACU":"T", "ACC":"T", "ACA":"T", "ACG":"T",
    "AAU":"N", "AAC":"N", "AAA":"K", "AAG":"K",
    "AGU":"S", "AGC":"S", "AGA":"R", "AGG":"R",
    "GUU":"V", "GUC":"V", "GUA":"V", "GUG":"V",
    "GCU":"A", "GCC":"A", "GCA":"A", "GCG":"A",
    "GAU":"D", "GAC":"D", "GAA":"E", "GAG":"E",
    "GGU":"G", "GGC":"G", "GGA":"G", "GGG":"G",}

aminoacids = ''.join(sorted(list(set([v for k,v in dic.items() if v !=
"STOP"]))))
print(aminoacids)

# UniProt Protein Database access IDs
proteins = ['A2Z669', 'B5ZC00', 'P07204_TRBM_HUMAN',
'P20840_SAG1_YEAST']

handle = ExPASy.get_sprot_raw(proteins[0])
seq_record = SeqIO.read(handle, "swiss")
handle.close()
print()
print(seq_record)
```

Références

- Biopython (librairie python de bioinformatique)
- https://en.wikipedia.org/wiki/Bioinformatics
- https://en.wikipedia.org/wiki/Open_Bioinformatics_Foundation
- https://en.wikipedia.org/wiki/FASTA_format
- https://en.wikipedia.org/wiki/List_of_open-source_bioinformatics_software

- cours introductif sur biopython :
  - Introduction to Biopython VIB bioinformatics core, Kristian Rother, en particulier ce tutoriel
- Bioinformatics Algorithms
- Articles de la revue “Science in School” :
  - Bioinformatics with pen and paper: building a phylogenetic tree Cleopatra Kozlowski, 07/12/2010
  - Using biological databases to teach evolution and biochemistry, Germán Tenorio, 02/06/2014
- documentation sur les arbres phylogénétiques : https://biopython.org/wiki/Phylo
- Rosalind, plateforme d'apprentissage de la programmation en bioinformatique
  - Glossaire de bioinformatique
- Catalog – Stepik cours et challenges en programmation, avec des activités en bioinformatique
  - Bioinformatics Algorithms – Stepik (cours introductif)
  - Bioinformatics Institute – Stepik (“institut virtuel” russe sur l'apprentissage de la bioinformatique)
  - Bioinformatics Contest 2017 – Stepik concours de programmation 2017
  - Bioinformatics Contest 2018 – Stepik concours de programmation 2018
  - Bioinformatics Contest 2019 – Stepik concours de programmation 2019
- http://www.amberbiology.com/ & Python for the Life Sciences – A gentle introduction to Python for life scientists programmation privilégiant les modules standards de Python (pas le module biopython par exemple)
- Bioinformatics with Python Cookbook livre utilisant beaucoup la librairie biopython
- GenBank
- références sur la lecture de fichiers :
  - http://www.uniprot.org/help/programmatic_access#id_mapping_python_example
  - http://www.python-simple.com/python-biopython/Lecture-ecriture-sequences.php