Projet
Filtrage et assemblage de reads

Allan RINGEVAL
Zinara LIDAMAHASOLO

Filtrage

Organisation

- 1 classe
- 5 méthodes de classe

Conception (1)

- 5 arguments de classe :
  - fichier de reads
  - fichier du génome
  - paramètres
  - fichier de sorti pour reads du génome
  - fichier de sorti pour reads restant

Conception (2)

- Argument “paramètres” : tuple(taille de kmer, probabilité de faux positif)
- Filtre de Bloom pour l’indexation

Etapes (1)

1ère méthode de classe :

- Lecture du fichier du génome de SARS-Cov2
  - modules SeqIO et Seq des packages Bio et Bio.Seq
- Indexation du génome :
  - Création table : package bitarray, initialisation à 0
  - 2 tables : 1 pour génome brin positif, 1 pour brin négatif
  - Brin négatif : package Bio.Seq
  - Taille des tables = *- (nlnp) / (ln2)**2* (n : nb items, p : taux de faux positif)
- Remplissage des tables : appelle 2ème méthode de classe

Etapes (2)

2ème méthode de classe :

- Remplissage des tables :
  - Fonctions de hachage : package mmh3
  - Nombre fonctions de hachage = $(m/n) * ln2$ (m : taille de table, n : nb kmers)
  - Fonctionnement : change en 1 les 0

Etapes (3)

3ème méthode de classe :

- Détermine si un read est trouvé ou pas :
  - Découpe le reads en kmers
  - Décalage de chaque début de kmers de n nucléotides ($n = 20$ pour 25-mers)
  - Décalage uniquement chez les kmers du reads
  - Appelle 4ème méthode de classe pour chercher un kmer dans les 2 tables
  - Seuil : 25%

4ème méthode de classe :

- Cherche un kmer dans les tables

Etapes (4)

5ème méthode de classe :

- Lecture du fichier fastq de reads :
  - package gzip
  - module FastqGeneralIterator du package Bio.SeqIO.QualityIO
- Création d'un fichier fastq de sorti pour reads du génome
- Création d'un autre fichier fastq de sorti pour autres reads
- Pour chaque reads :
  - Appelle 3ème méthode pour savoir si reads est trouvé ou pas
  - Si dans brin + ⟶ fichier de sorti pour reads du génome
  - Si dans brin - ⟶ Reverse complement ⟶ fichier de sorti pour reads du génome
  - Si pas trouvé ⟶ fichier de sorti pour autres reads

Assemblage

Organisation

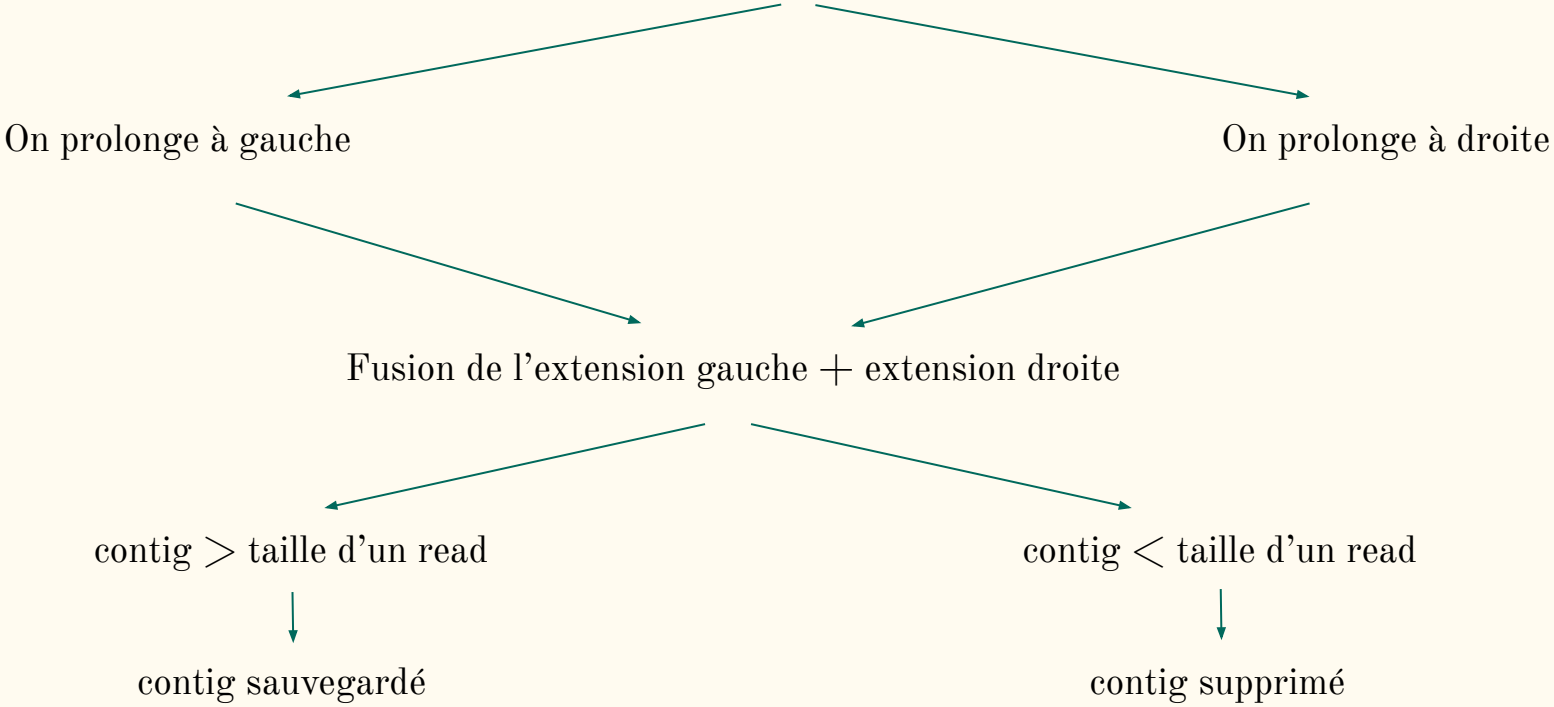
- 1 classe
- 10 méthodes de classe

Conception

- Structure de données :
  - Dictionnaire python
    - stockage des kmers
  - clé = kmer
  - valeur = tuple (Boolean,int)
    - 1er élément -> Kmer déjà utilisé ou non
    - 2ème élément -> nombre d'occurrence du kmer

Déroulement de l'algo

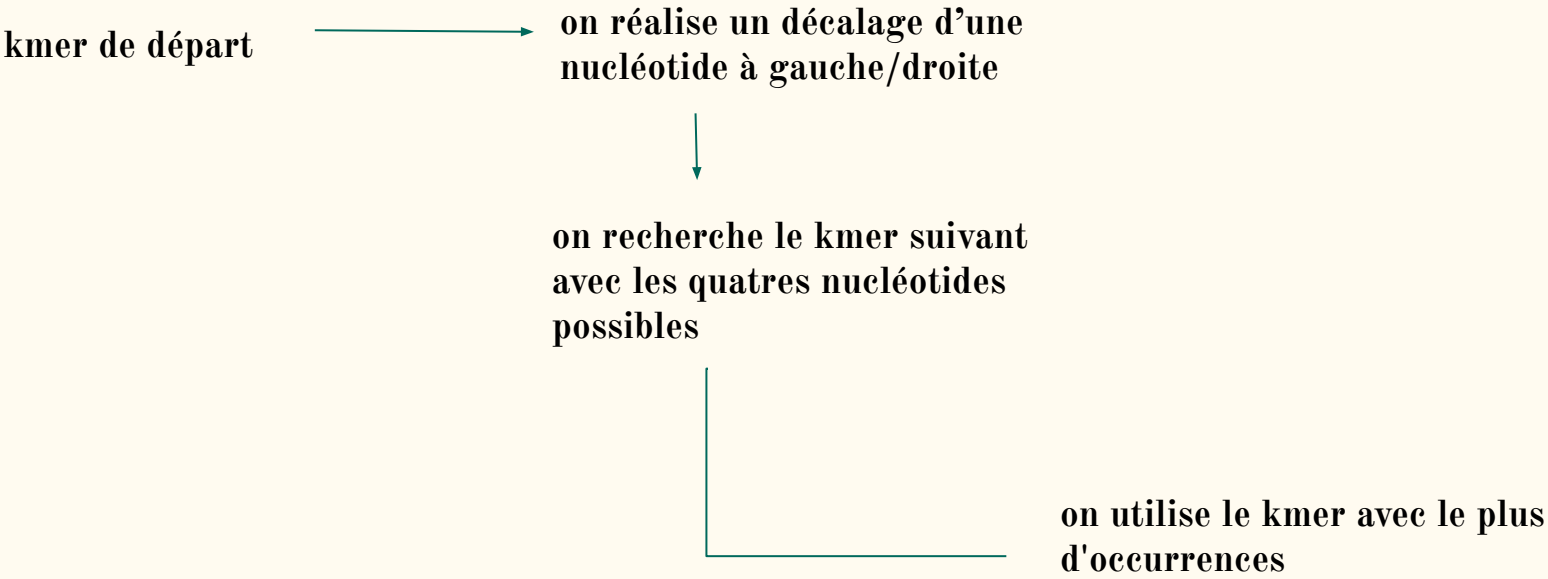
On cherche un kmer pas utilisé dans la table de hachage



Gestion d’erreurs

1ère étape : supprimer les kmers observés moins de trois fois

2ème étape : trouver le kmer suivant



Exécution du filtrage

- Taille de kmer : 25
- Taux de faux positif : 1%
- Décalage entre chaque début de kmer des reads : 20
- Durée d'exécution : 48 min
- Nombre de reads trouvé : 10 785

Exécution de l’assemblage

- Durée d'exécution : 3 sec
- Taille du contig : 29 886 nucléotides sur 29 903
- Taille de kmer : 35
- Supprimer kmer : observé moins de 3 fois

Alignement global (1)

```
#=======================================
#
# Aligned_sequences: 2
# 1: NC_045512.2
# 2: 0
# Matrix: EDNAFULL
# Gap_penalty: 10.0
# Extend_penalty: 0.5
#
# Length: 29903
# Identity:   29886/29903 (99.9%)
# Similarity: 29886/29903 (99.9%)
# Gaps:          17/29903 ( 0.1%)
# Score: 149430.0
#
#
#=======================================
```

```
     1 ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTC     50
               ||||||||||||||||||||||||||||||||||||||||||
     1 ---------TTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTC     41

    51 TTGTAGATCTGTTCTCTAAACGAACTTTAAAATCTGTGTGGCTGTCACTC    100
       ||||||||||||||||||||||||||||||||||||||||||||||||||
    42 TTGTAGATCTGTTCTCTAAACGAACTTTAAAATCTGTGTGGCTGTCACTC     91
```

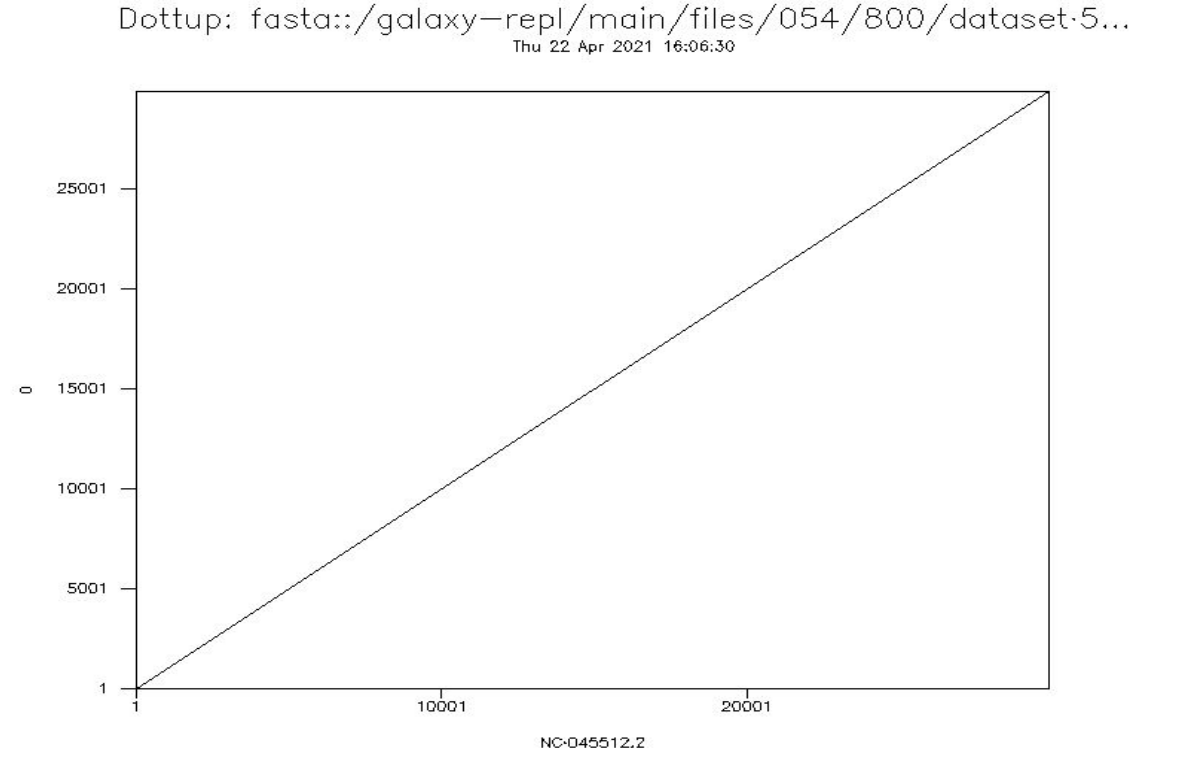
```
29851 TAGCTTCTTAGGAGAATGACAAAAAAAAAAAAAAAAAAAAAAAAAAAA  29900
      ||||||||||||||||||||||||||||||||||||||||||||||||
29842 TAGCTTCTTAGGAGAATGACAAAAAAAAAAAAAAAAAAAAAAAA-----  29886

29901 AAA  29903

29886 ---  29886
```

Galaxy / Needle

Alignement global (2)



Galaxy / Dottup

Merci pour votre attention !