

Filtres, graines et BLAST

Hélène Touzet

`helene.touzet@univ-lille.fr`

CNRS, Bonsai, CRIStAL

Algorithmes d'alignement



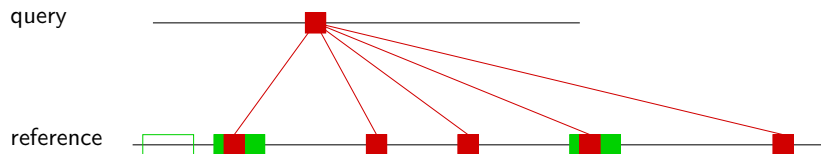
Algorithmes d'alignement

- algorithmes exacts par programmation dynamique
 - sensibilité + +, temps de calcul - -
 - adaptés à la comparaison de deux gènes, par exemple
- algorithmes heuristiques à base de filtre
 - sensibilité - -, temps de calcul + +
 - adaptés à la comparaison de génomes, à la recherche de séquences dans une base de données, à l'alignement de lectures de séquençage (mapping)

Filtre



Filtre : le paradigme "seed-and-extend"



seed



extend

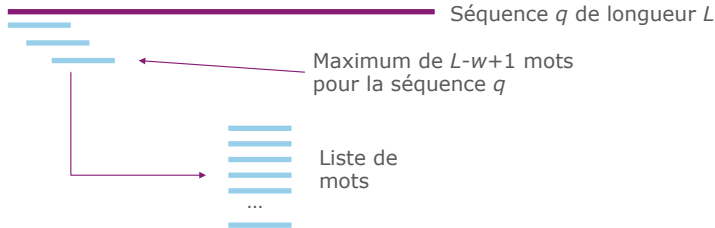


- graine (seed)
 - courte région de forte similarité
 - **choix de la forme de la graine** : k -mers, graines espacées, mots avec erreurs
 - **choix de l'échantillonnage** : toutes les positions, sous-ensemble de positions ...
- extension
 - reconstruction d'un alignement plus long autour de la graine
 - programmation dynamique
- indexation
 - indexation de la **requête**
 - indexation de la **référence**
 - **choix de la structure d'indexation** : table, table de hachage, arbre des suffixes, tableau des suffixes, FM-index, ...
- BLAST, Clustal Ω , Bowtie2, etc.

Rappels sur BLAST

- alignement local
- il est trop coûteux en temps de faire tous les alignements locaux possibles : BLAST ne s'intéresse qu'aux séquences avec un fort taux de similarité
- heuristique : méthode de calcul qui fournit rapidement une solution réalisable, mais pas nécessairement optimale. Une heuristique peut manquer des résultats.

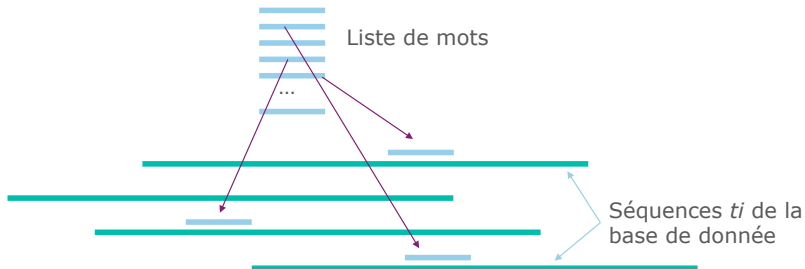
Étape 1 : recensement de tous les mots de longueur w qui composent la séquence requête (*k-mer*, *seed*, *k-word*, *anchor*)



w est un paramètre du programme

par exemple : $w = 11$ pour les séquences nucléiques, $w = 3$ pour les séquences protéiques

Étape 2 : extraction des séquences de la banque de données qui contiennent des mots communs



Étape 3 : Extension de ces points d'ancrage de proche en proche



`https://blast.ncbi.nlm.nih.gov`


Program Selection

Optimize for

☒ Highly similar sequences (megablast)

☐ More dissimilar sequences (discontiguous megablast)

☐ Somewhat similar sequences (blastn)

Choose a BLAST algorithm 

>whoami


```
taaactttctcgatcattattcagagtttctagttgctctagtggttaattttaactccga  
ttctagataatactctcgaaaaacaatggttccttctccttgttcaagtatgctccaaaa  
catatcattatgggttcacaaaaccatttcctataacatctaatagtatTTTTGTGGATAA  
aagatactcctgattttctagattaattggaaacggctgtatttGTGACCTTTTTTGTA  
actacataagtccttaaataaatgaaggattaacccaaaaccattgttatatgagtcct  
agtttcacactgtaagcttaacatttcctcatagtttataccaatatatatggatttaac  
aggatcttctatcctcgtctgcaacttatctttaccaaacttagtacatatccatttgggt  
aacttgcttcataaaactccctatcccgttctcttccattgcattctcatgtctaattat  
cccgtgttcaactactcgagtaatacattcctttttcatttttagctacttcaagtgtgca  
tggtttctcgccatattcaagctcaatttccttttccgctttgccaagatactttttaag
```

source : M. Schatz

Program Selection

Optimize for

- ☒ Highly similar sequences (megablast)
- ☐ More dissimilar sequences (discontiguous megablast)
- ☐ Somewhat similar sequences (blastn)

Choose a BLAST algorithm 

megablast : mot de longueur 28 par défaut (de 16 à 256 en option)
Convient pour des séquences d'identité $> 95\%$ **graine contigue**

discontiguous megablast : mot à trous, de longueur 11 (12 en option)
Tolérant aux substitutions. Comparaison inter-espèces **graine espacée**

BlastN : mot de longueur 11 par défaut (7 ou 15 en option)
Convient pour des séquences d'identité $> 85\%$ **graine contigue**


General Parameters

Max target sequences

100 ▾

Select the maximum number of aligned sequences to display 

Short queries

☒ Automatically adjust parameters for short input sequences 

Expect threshold

0.05



Word size

28 ▾

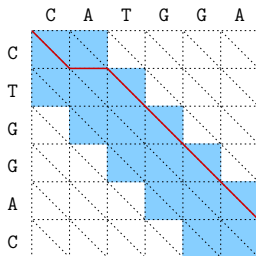


The length of the seed that initiates an alignment. [more...](#)

Seed-and-extend : la phase d'extension

```
... CGTCGTG CACTGCACG CATGGA ...  
      |||||  
... TCCACGT CACTGCACG CTGGAC ...  
    <----- seed ----->
```

- construction de deux alignements



```
C  A  T  G  G  A  -  
|  |  |  |  |  |  
C  -  T  G  G  A  C
```

alignement avec au plus ℓ indels

ℓ -strip = $\{(x, y); |x - y| \leq \ell\}$

Complexité : $O(\ell n)$

- l'implémentation par programmation dynamique peut être rendue plus efficace avec une implémentation SIMD

Alignement Felis Catus/ Nyctereute

```
1  ttcttcctaccctgccccgtcatgtgctgctctactgggccacg | 145  ggcgagc.....  
   |||||  
1  ttcttcctaccctgccccgtcatgtgctgctctactgggccacg | 181  ggc.agcccgagcggcacccccggcccgcccccggacggcac  
  
46  ttccggggcctgcggcgctgggaggcggtcgccaggccaagctg | 152  .....  
   |||||  
46  ttccggggcctgcggcgctgggaggcggtcggtcgccaggccaagctg | 225  ccccgatgacacccccgagccacccccctgcccccgcccccgcc  
  
91  cactgccggggcctcgctcgccccagcgccccggcccaccgccc | 153  ccccgagccgctcgcccccccgagccggtcccagccgagccgccc  
   ||| ||||| | || | |||||  
91  cacggccgggacaccgcgagaccagcgccccggcccgcacccc | 270  ccccgagccgccccggcccccgcccgaccctgaggagccag  
  
136  cccga.ggt.....c | 198  gcggcaggcaccaggaggagcgcgccaagatcacggccggga  
   ||||| ||| |  
136  cccgacggtacccccggccccccgcccccgacggcgagccccgac | 315  gtggcagccacgcaagcggagacgcgccaagatcacggccggga  
  
243  gcgcaaggccatgagggtcctgccggtggtggtc  
   |||||  
360  gcgcaaggccatgagggtcctgccggtggtggtc
```

Alignement local optimal



Les graines contigues, les k -mers

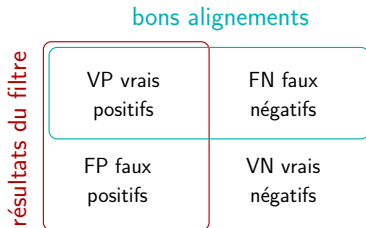
Graines contigues : les k -mers

```
11101111101111101111
AUCAGUGCAAAUGCUC-AAGA
| | | | | | | | | |
AUC-GUGCAUAUGCUCGAAGA
    11111 11111
        11111
```

- alignement : mot sur l'alphabet $\{0, 1\}$ (1 pour les matches, 0 pour les erreurs)
- k -mer : mot conservé de longueur k , 1^k

Comment évaluer la qualité d'un filtre ?

- temps de calcul
- qualité du résultat



VP	FP	$VP \times VN - FP \times FN$
VP+FN	VP+FP	$\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}$
Sensibilité	Spécificité	Coefficient de corrélation de Matthews

- graine/filtre sans perte : sensibilité 100%
- graine/filtre avec perte : pas de garantie de trouver l'intégralité des alignements pertinents

Tous les alignements de longueur au moins 100 avec au plus 10 erreurs contiennent 8 identités successives.
(autrement dit : 1^8 est une graine sans perte pour cet ensemble d'alignements)

Vrai ou faux?

La graine 1¹¹ reconnaît tous les alignements de longueur 100 avec un pourcentage d'identité supérieur ou égal à 90%

Vrai ou faux ?

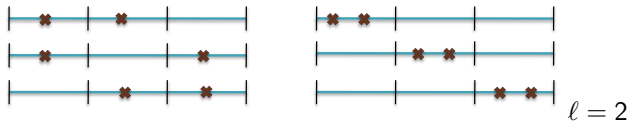
- un alignement d'une séquence de longueur m avec au plus ℓ différences doit contenir un mot exact commun de longueur supérieure ou égale à

$$\frac{m}{\ell + 1}$$

(Baeza-Yates and Perleberg, 1996)

- principe du pigeonnier : ℓ pigeons ne peuvent pas remplir $\ell + 1$ pigeonniers

On découpe l'alignement en $\ell + 1$ blocs de longueur $\frac{m}{\ell + 1}$



Comment calculer la sensibilité d'une graine k -mer

- k -mer : 1^k
- modèle pour le type d'alignements visés
 p , pourcentage d'identité ($0 \leq p \leq 1$)
 $\mathcal{A}(m, p)$: ensemble des mots de $\{0, 1\}^m$ avec une proportion de p caractères 1

ACTGACTG	ACTGACTG
TCTGACTG	AATGACTG
01111111	10111111

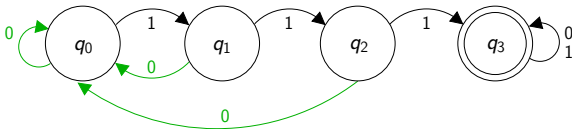
- sensibilité : proportion de mots de $\mathcal{A}(m, p)$ reconnus par la graine 1^k

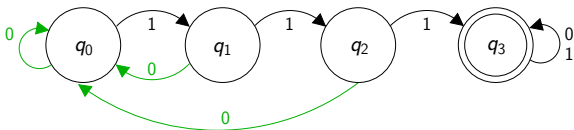
Automate fini déterministe pour le langage $\{0,1\}^* 1^k \{0,1\}^*$

- $k + 1$ états : q_0, \dots, q_k
- état initial : q_0
- état final, acceptant : q_k
- transitions

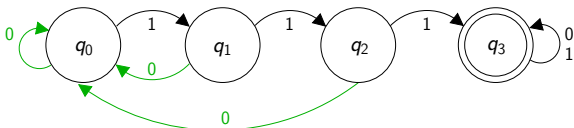
	q_0	q_1	q_2	\dots	q_ℓ	\dots	q_{k-1}	q_k
0	q_0	q_0	q_0		q_0		q_0	q_k
1	q_1	q_2	q_3		$q_{\ell+1}$		q_k	q_k

Exemple pour la graine 111 (3-mer)





- $P(i, q, x)$: probabilité d'atteindre l'état q après avoir lu un mot u de longueur i dont la dernière lettre est x
- $P(i, q) = P(i, q, 0) + P(i, q, 1)$: probabilité d'atteindre l'état q après i lettres



- $P(i, q, x)$: probabilité d'atteindre l'état q après avoir lu un mot u de longueur i dont la dernière lettre est x
- $P(i, q) = P(i, q, 0) + P(i, q, 1)$: probabilité d'atteindre l'état q après i lettres

$$\begin{aligned}
 P(i, 0, 0) &= (1 - p)[P(i - 1, 0) + P(i - 1, 1) + \dots + P(i - 1, k - 1)] \\
 P(i, 0, 1) &= 0 \\
 P(i, q, 0) &= 0 && \text{if } 0 < q < k \\
 P(i, q, 1) &= p P(i - 1, q - 1) \\
 P(i, k, 0) &= (1 - p)P(i - 1, k) \\
 P(i, k, 1) &= p [P(i - 1, k - 1) + P(i - 1, k)]
 \end{aligned}$$

$$P(i, 0, 0) = (1 - p)[P(i - 1, 0) + P(i - 1, 1) + \dots + P(i - 1, k - 1)]$$

$$P(i, 0, 1) = 0$$

$$P(i, q, 0) = 0 \quad \text{if } 0 < q < k$$

$$P(i, q, 1) = p P(i - 1, q - 1)$$

$$P(i, k, 0) = (1 - p)P(i - 1, k)$$

$$P(i, k, 1) = p [P(i - 1, k - 1) + P(i - 1, k)]$$

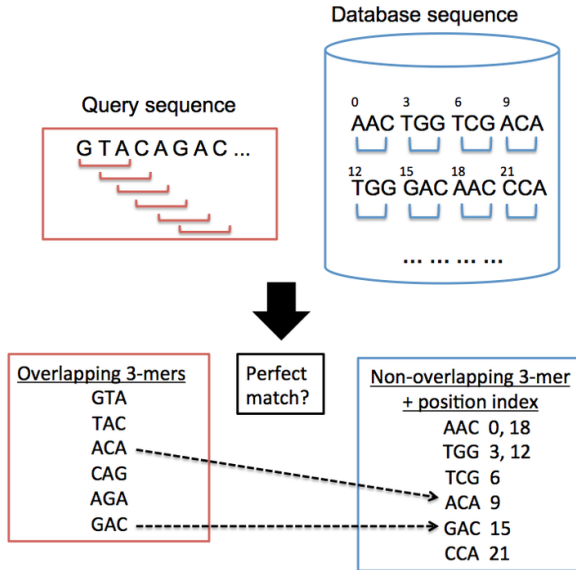
implémentation : programmation dynamique



BLAT

BLAST-like alignment tool

- application visée : localiser un gène dans un génome
- développé dans le cadre du séquençage du génome humain (2002)
- découpage du génome en k -mers non chevauchants (pavage)
- privilégie la rapidité, au détriment de la sensibilité, en réduisant la taille de l'index
- adapté à la recherche de séquence avec plus de 95% d'identité et de longueur supérieure à 25 bases



Le génome est découpé en k -mers non chevauchants (pavage)



UNIVERSITY OF CALIFORNIA

SANTA CRUZ

Genomics
Institute



Genome Browser



Genomes

Genome Browser

Tools

Mirrors

Downloads

My Data

Projects



Blat

Table Browser

Variant Annotation
Integrator

Data Integrator

Gene Interactions

Gene Sorter

Genome Graphs

In-Silico PCR

LiftOver

VisiGene

Other Utilities

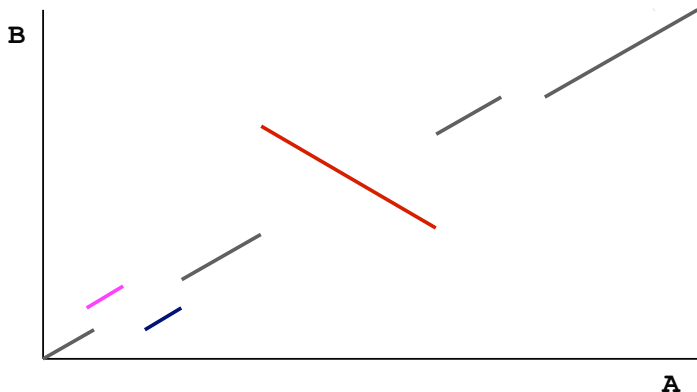
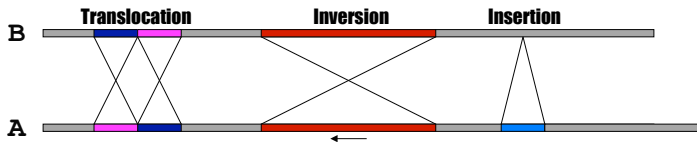

```
> Felis catus DRD4 gene for dopamine receptor D4, partial cds (276bp)
ttcttcctaccctgcccgtcatgctgctgctctactgggccacgttccggggcctgcgg
cgctgggaggcggctcgccaggccaagctgcactgccggggcgctcgctcgcccagcggc
cccggcccaccgccccccgaggtcggcgagcccccgacgccgtcgcgcccccgacgcc
gtcccagccgagccgcccggcgagcagccagcaggaggaggcgcgccaagatcacccggcgg
gagcgcaaggccatgagggtcctgccggtggtggtc
```

- <http://genome.ucsc.edu>
- génomes à tester : cat, dog, pig

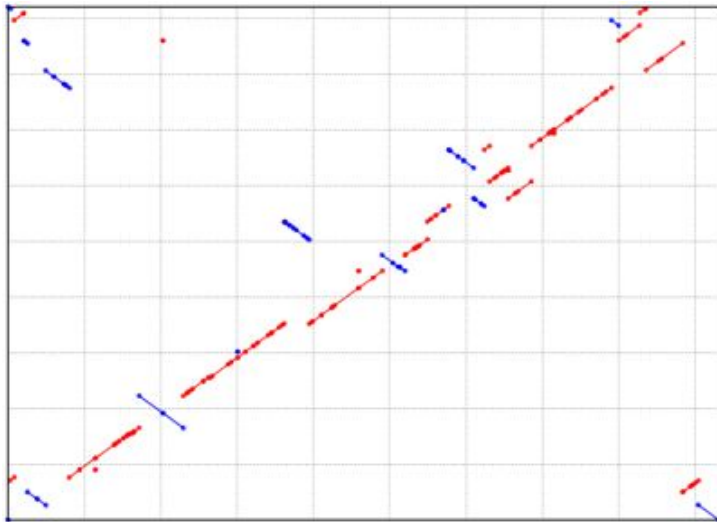
MUMmer

Maximal Unique Matcher

- application visées : comparaison de génomes
- développé à partir de 2004, en 2018 (Perl → C++)
- MUM (graines) : séquence exacte commune entre les deux génomes, unique et maximale
 - unique : présente en un seul exemplaire dans le génome de référence
 - maximal : on ne peut plus ajouter de bases, à droite ou à gauche
 - implémentation : arbre des suffixes, puis suffix array
- <https://mummer4.github.io/index.html>



<http://mummer.sourceforge.net/manual/AlignmentTypes.pdf>

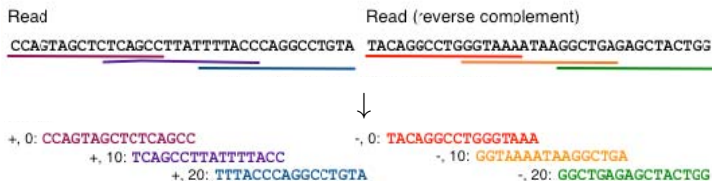


Yersinia pestis CO92 vs. *Yersinia pestis* KIM

- similarité nucléotidique élevée (>99,5%)
- nombreux remaniements et répétitions

Bowtie2

- application visée : aligner des lectures de séquençage (short reads) sur un génome de référence
- chaque read est découpé en k -mers de longueur 20 à 25, toutes les 3-5 positions



	seed	query positions	reference positions	extension	index
BlastN	11-mer	all	all	systematic	look-up table
BLAT	22-mer	all	tiling	partial	look-up table
Bowtie2	22-mer	1/10	all	partial	FM-index + BWT

	seed	query positions	reference positions	extension	index
BlastN	11-mer	all	all	systematic	look-up table
BLAT	22-mer	all	tiling	partial	look-up table
Bowtie2	22-mer	1/10	all	partial	FM-index + BWT

BLAST : Accurate up to 85% identity - all alignments

BLAT : Accurate up to 95% identity

Bowtie2 : Accurate up to 95% identity - single alignment

- a Il existe un alignement de longueur ℓ contenant y erreurs tel que tous les sous-alignements de longueur a contiennent au moins une erreur
- b Tous les alignements de longueur ℓ avec y erreurs sont reconnus par la graine k -mer de longueur a
- c La sensibilité de la graine k -mer de longueur a pour les alignements de longueur ℓ avec y erreurs est égale à 100%
- d Si un mot V sur l'alphabet $\{0,1\}^*$ est de longueur ℓ et qu'il contient y fois le caractère 0, alors le mot 1^a est un facteur de V
- e 1^a est une graine avec perte pour l'ensemble des alignements de longueur ℓ avec y erreurs
- f La graine 1^a est sans perte pour les alignements de longueur ℓ avec y erreurs



Les graines espacées

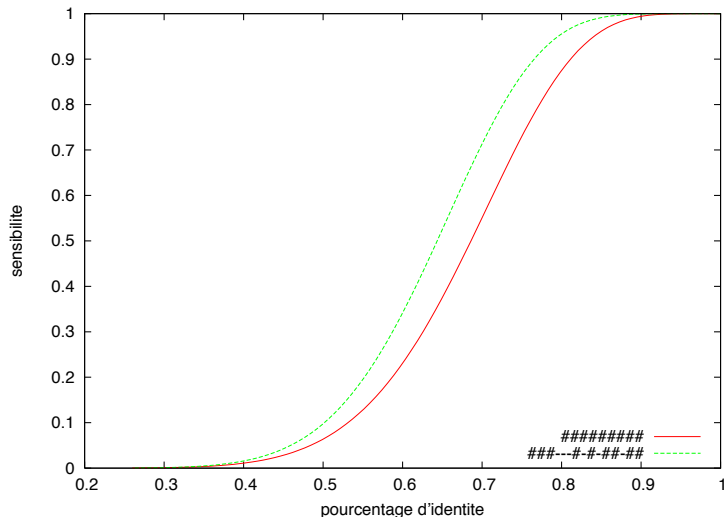
Filtres à base de graines espacées

- Graine : mot sur l'alphabet $\{1, *\}$
 - 1 \rightarrow match
 - * \rightarrow match ou mismatch

AUCAGUGCAA <u>A</u> UGCUC <u>A</u> AGA	AUCAGUGCAA <u>A</u> UGCUC <u>A</u> AGA
AUCAAUGCAG <u>A</u> UGC <u>G</u> CAAGA	AUCAGUGUAAUUGCUC <u>A</u> AGA
111*1*11	111*1*11

- Exemples : discontinuous megablast, LAST

Sensibilité : graine espacée vs graine contigue



graine contigue : 11111111, graine espacée : 111***1*11*11
tous les alignements de longueur 64 (courtesy Laurent Noé)

Pourquoi cela marche ?

111111	111*1*11
111111	111*1*11

la probabilité de deux occurrences successives est inférieure avec
les graines espacées

AUCAGUGCAAAUGCUCAAGA
| | | | | | | | | | | | | | | |
AUCAGUGCAAAUGCUCAAGA


```

111111
 111111
   111111
    111111
     111111
      111111
       111111
        111111
         111111
          111111
           111111
            111111
             111111
              111111
               111111
                111111

```

```

111*1*11
  111*1*11
    111*1*11
      111*1*11
        111*1*11
          111*1*11
            111*1*11
              111*1*11
                111*1*11
                  111*1*11
                    111*1*11
                      111*1*11

```


[illegible][illegible]

The diagram illustrates the construction of a Huffman tree. It shows a series of rows of '1's and '0's, where the first row is '111111' and subsequent rows show the merging of pairs of '1's into '0's. The final row is '000000'. The '1's are colored orange, and the '0's are colored gray.

[illegible]

[illegible]

[illegible]

A 10x10 grid of 1s and 0s. The 1s form a staircase pattern, with the number of 1s per row decreasing from 10 in the first row to 1 in the tenth row. The 1s are arranged in a way that they form a solid shape on the left side of the grid. The 1s in the first row are at columns 1-10. In the second row, columns 1-9 are 1, and column 10 is 0. In the third row, columns 1-8 are 1, and columns 9-10 are 0. This pattern continues until the tenth row, where only column 1 is 1, and columns 2-10 are 0. The 1s are arranged in a way that they form a solid shape on the left side of the grid. The 1s in the first row are at columns 1-10. In the second row, columns 1-9 are 1, and column 10 is 0. In the third row, columns 1-8 are 1, and columns 9-10 are 0. This pattern continues until the tenth row, where only column 1 is 1, and columns 2-10 are 0. The 1s are arranged in a way that they form a solid shape on the left side of the grid.

[illegible]

[illegible][illegible]

[illegible][illegible]


```

1 1 1 1 1 1 0 0 0 0
0 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 0 1 1 1 1 1 0
0 0 0 0 0 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 1 1 1
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 1

```

[illegible]

Comment calculer la sensibilité d'une graine ?

Données

- une graine espacée π
 - w , son poids (nombre de 1)
 - m , son étendue (nombre de 1 et nombre de *)
- un modèle pour les alignements
 - p , le pourcentage d'identité ($0 \leq p \leq 1$)
 - n , la longueur

Question

Quelle est la proportion des alignements qui sont détectés par la graine π ?



Les graines avec erreurs

Les graines avec erreurs

- erreurs : substitution, voire délétion/insertion
- graine
 - m : longueur de la graine
 - ℓ : nombre d'erreurs maximum autorisés
- recherche de la graine : programmation dynamique, ℓ -strip
- exemple : `kalign` pour l'alignement multiple, `bowtie2` (1 mismatch dans la graine)

Extrait de la documentation de Bowtie2

The trade-off between speed and sensitivity/accuracy can be adjusted by setting the seed length (-L), the interval between extracted seeds (-i), and the number of mismatches permitted per seed (-N).

For more sensitive alignment, set these parameters to (a) make the seeds closer together, (b) make the seeds shorter, and/or (c) allow more mismatches.

Conclusion sur les approches « seed-and-extend »

- rôle prédominant des k -mers dans le choix des graines
- la sensibilité et la rapidité vont dépendre de
 - la longueur du k -mer
 - l'écart entre les k -mers
- le choix de ces valeurs par le développeur du programme va dépendre de l'application visée
- non traité dans ce cours : l'indexation des séquences

Utiliser les graines sans étape d'extension

Exemple 1 : distance à base de comptage de k -mers

	ATGTGTG	CATGTG
Query sequences	x ATGTGTG	y CATGTG
Word size: 3	W_3^x ATG TGT GTG TGT GTG	W_3^y CAT ATG TGT GTG
Union of two sets	$W_3 = W_3^x \cup W_3^y$ CAT ATG TGT GTG	
Word counts	c_3^x 0 1 2 2	c_3^y 1 1 1 1

(0,1,2,2) (1,1,1,1)

Comment comparer les deux vecteurs $(0, 1, 2, 2)$ et $(1, 1, 1, 1)$?

- distance de Manhattan

$$|0 - 1| + |1 - 1| + |2 - 1| + |2 - 1|$$

- distance Euclidienne

$$\sqrt{(0 - 1)^2 + (1 - 1)^2 + (2 - 1)^2 + (2 - 1)^2}$$

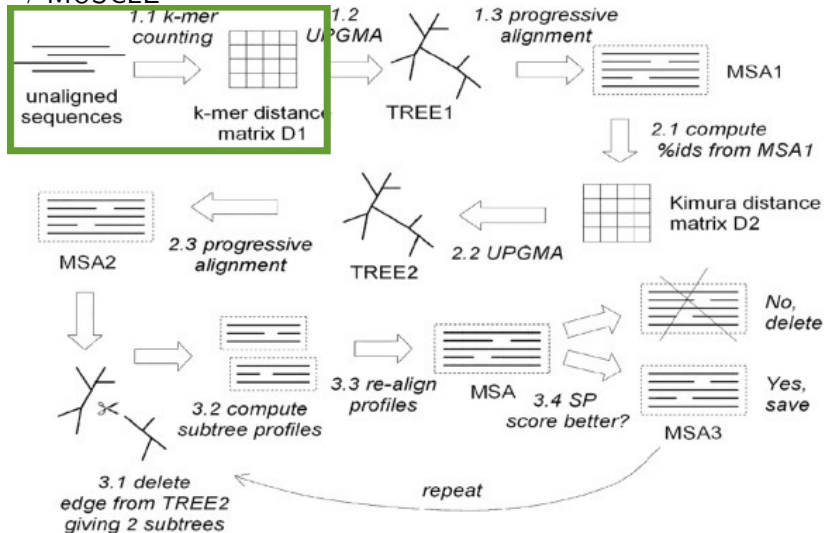
- distance d_2

$$(0 - 1)^2 + (1 - 1)^2 + (2 - 1)^2 + (2 - 1)^2$$

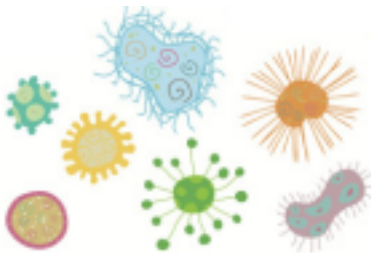
- distance de Mahalanobis

Application à l'alignement multiple

→ MUSCLE



Exemple 2 : k -mers discriminants



Classification taxonomique en métagénomique

- ensemble de lectures de séquençage (short reads) pour une communauté microbienne
- quelles sont les espèces présentes ?

- recensement de tous les k -mers présents dans la base de données de génomes connus
- assignation de chaque k -mer au plus petit ancêtre commun (LCA) de toutes les espèces contenant ce k -mer.

Wood and Salzberg *Genome Biology* 2014, **15**:R46
<http://genomebiology.com/2014/15/3/R46>

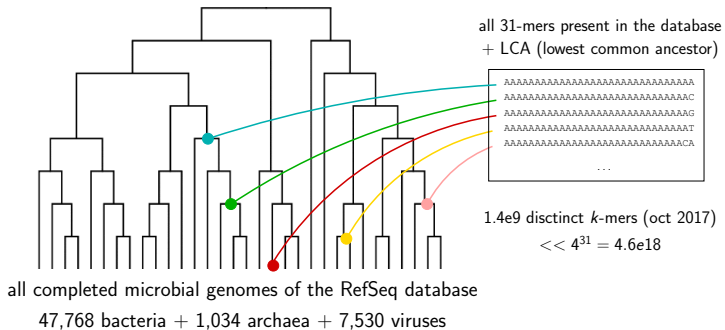


METHOD

Open Access

Kraken: ultrafast metagenomic sequence classification using exact alignments

Derrick E Wood^{1,2*} and Steven L Salzberg^{2,3}

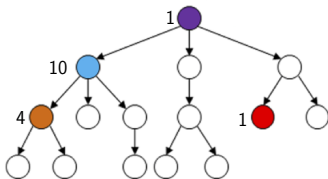


Precomputed database

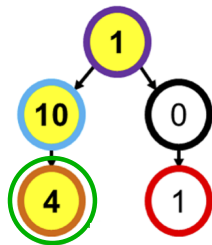
1. short read \rightarrow overlapping k-mers



2. identification of the LCA in the taxonomy for each k-mer



3. assignment of the read



Read assignment