Initiation à la bio-informatique
Module 2 : Alignement de séquences

Partie 1 :

Comparaison de séquences deux à deux

Comparaisons locales et matrices de score

Yasmina Mesloub & Ségolène Caboche

Université de Lille - TAG

(segolene.caboche@pasteur-lille.fr)

Pourquoi comparer des séquences ?

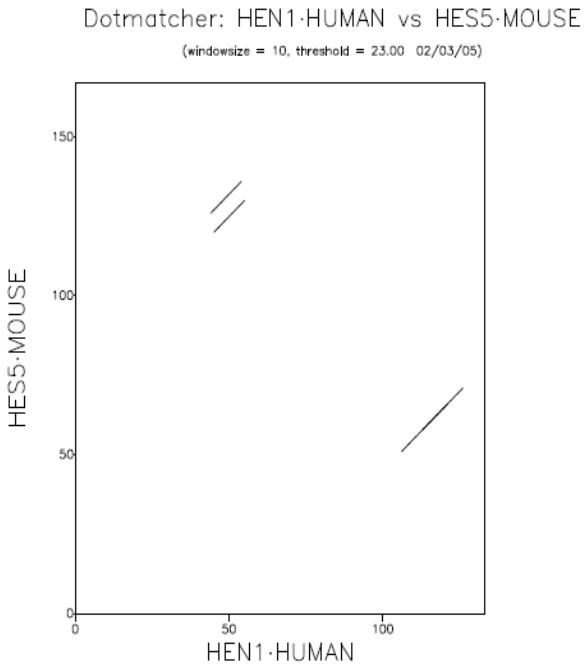
- recherche d'homologie (entre gènes)
- trouver des régions similaires (domaines protéiques)
- identifier les positions introns/exons (comparaison d'un gène à son ARNm)
- ...

```
AAB24882        TYHMCQFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPT 60
AAB24881        --------------------YECNQCGKAFAQHSSLKCHYRTHIGEKPYECNQCGKAFSK 40
                                     ****: .***:  * *:** * :****.:* ********..

AAB24882        PSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHTGEKPYE-CNQCGKAFAQ- 116
AAB24881        HSHLQCHKRTHTGEKPYECNQCGKAFSQHGLLQRHKRTHTGEKPYMNVINMVKPLHNS 98
                 **** *:***********:***:**.: .****************    :  *.: :
```

Dotplot

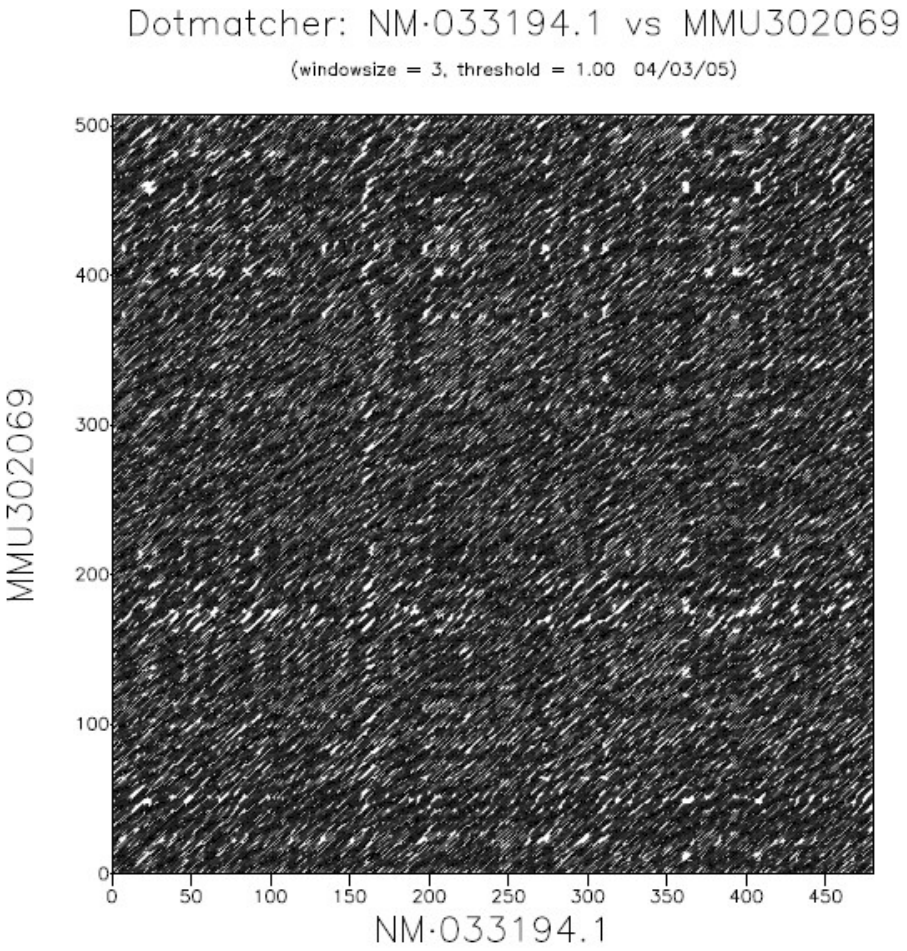
- un outil graphique pour la comparaison de séquences
- Principe :
  - mettre les séquences le long des axes d'une matrice
  - mettre un point là où il y a un match
- une diagonale (une suite de points en diagonale) ⇒ une région similaire



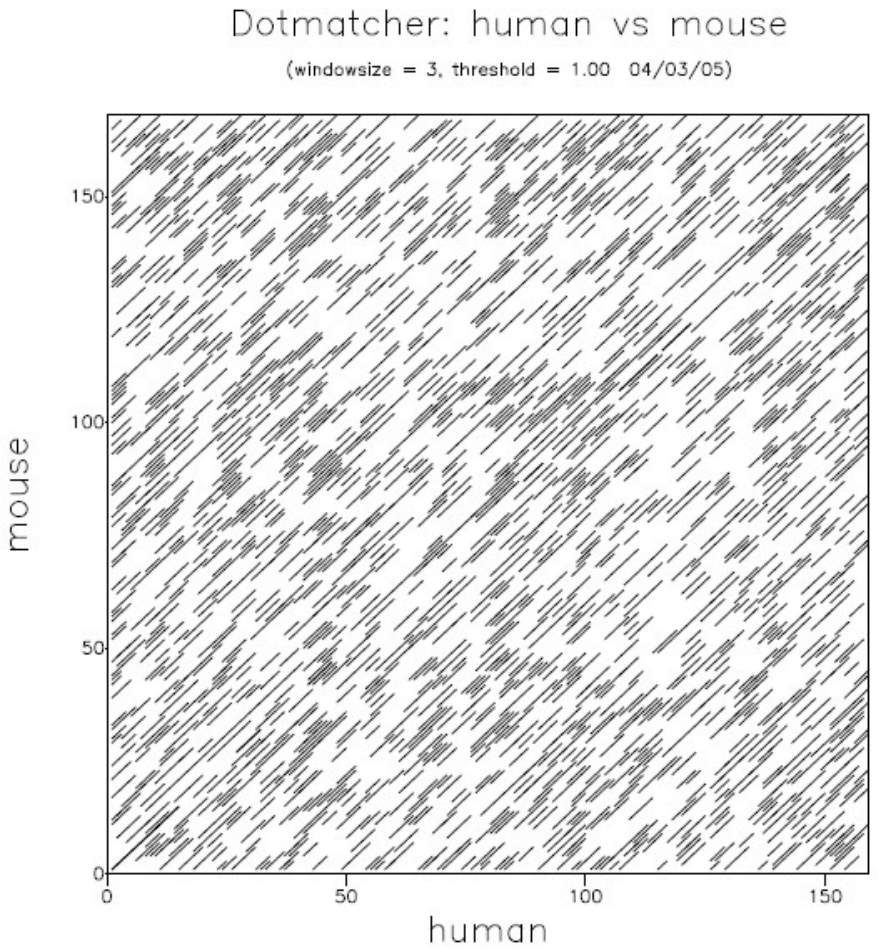
Dotplot : exemple

	G	A	T	C	T	A	C
G	*						
T			*		*		
T			*		*		
C				*			*
T			*		*		
G	*						
C				*			*
A		*				*	

Dotplot : exemple



deux genes ...



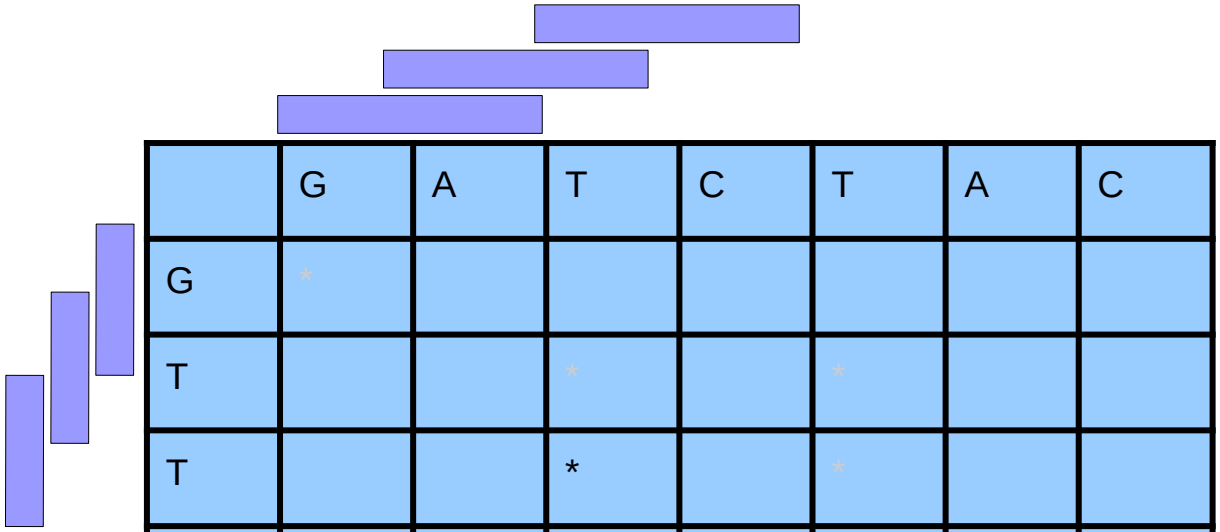
et leurs proteines.

Problème de bruit => filtrage trop faible

Dotplot : filtrage

- blocs d'identité (FASTA)
  - idée : ne représenter que des fenêtres exactes et de longueur fixe
  - exemple de programme : **dottup**
  - très sélectif et peu sensible
- blocs de similarité avec un seuil de score (Maizel & Lenk - 1981)
  - idée : ne représenter que des fenêtres possédant un score supérieur à un seuil
  - exemple de programme : **dotmatcher**
  - bonne sélectivité et bonne sensibilité
- filtrer les blocs pour éliminer ceux qui se chevauchent
  - idée : observez la ressemblance globale
  - exemple de programme : **dotpath**
- accorder un poids physico-chimique aux matchs (Staden - 1982)
  - variation de l'intensité, ne pas se contenter de 0 et 1
  - prise en compte des propriétés des acides aminés

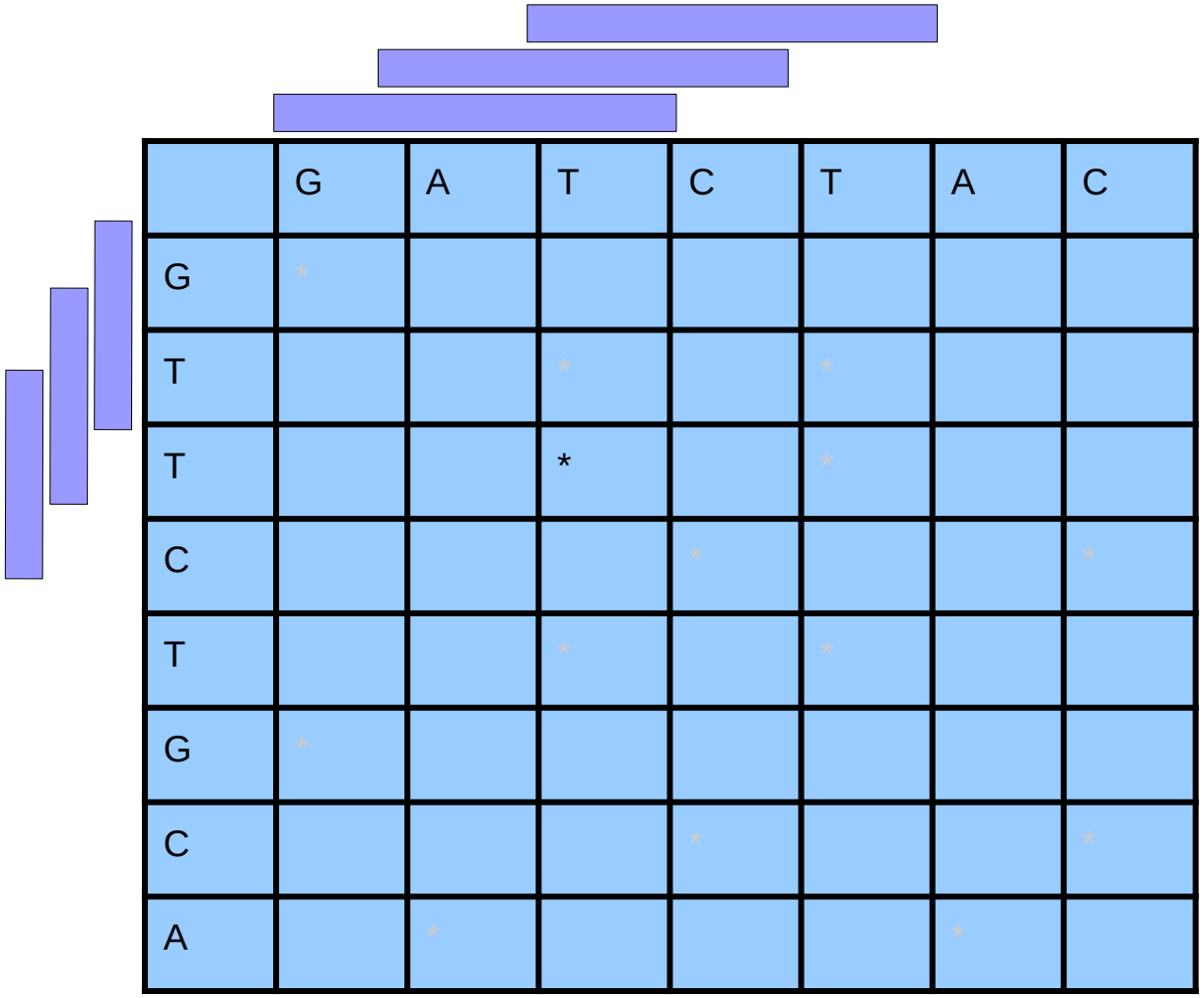
Dotplot : exemple



	G	A	T	C	T	A	C
G	*						
T			*		*		
T			*		*		
C				*			*
T			*		*		
G	*						
C				*			*
A		*				*	

=> fenêtre glissante de taille 2

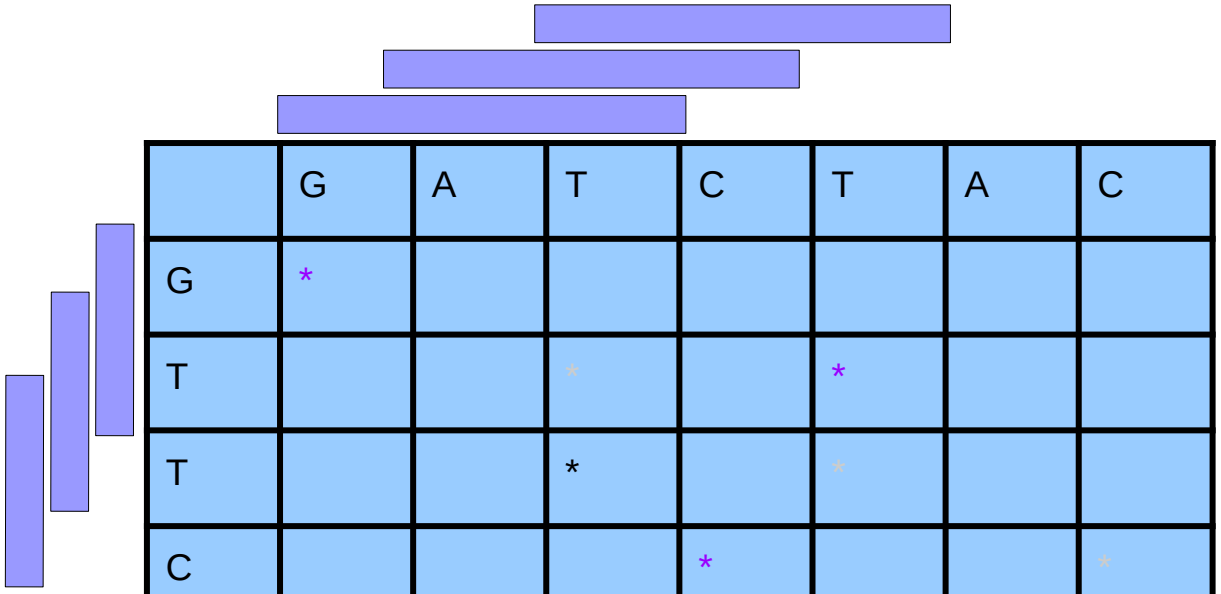
Dotplot : exemple



	G	A	T	C	T	A	C
G	*						
T			*		*		
T			*		*		
C				*			*
T			*		*		
G	*						
C				*			*
A		*				*	

=> fenêtre glissante de taille 3

Dotplot : exemple

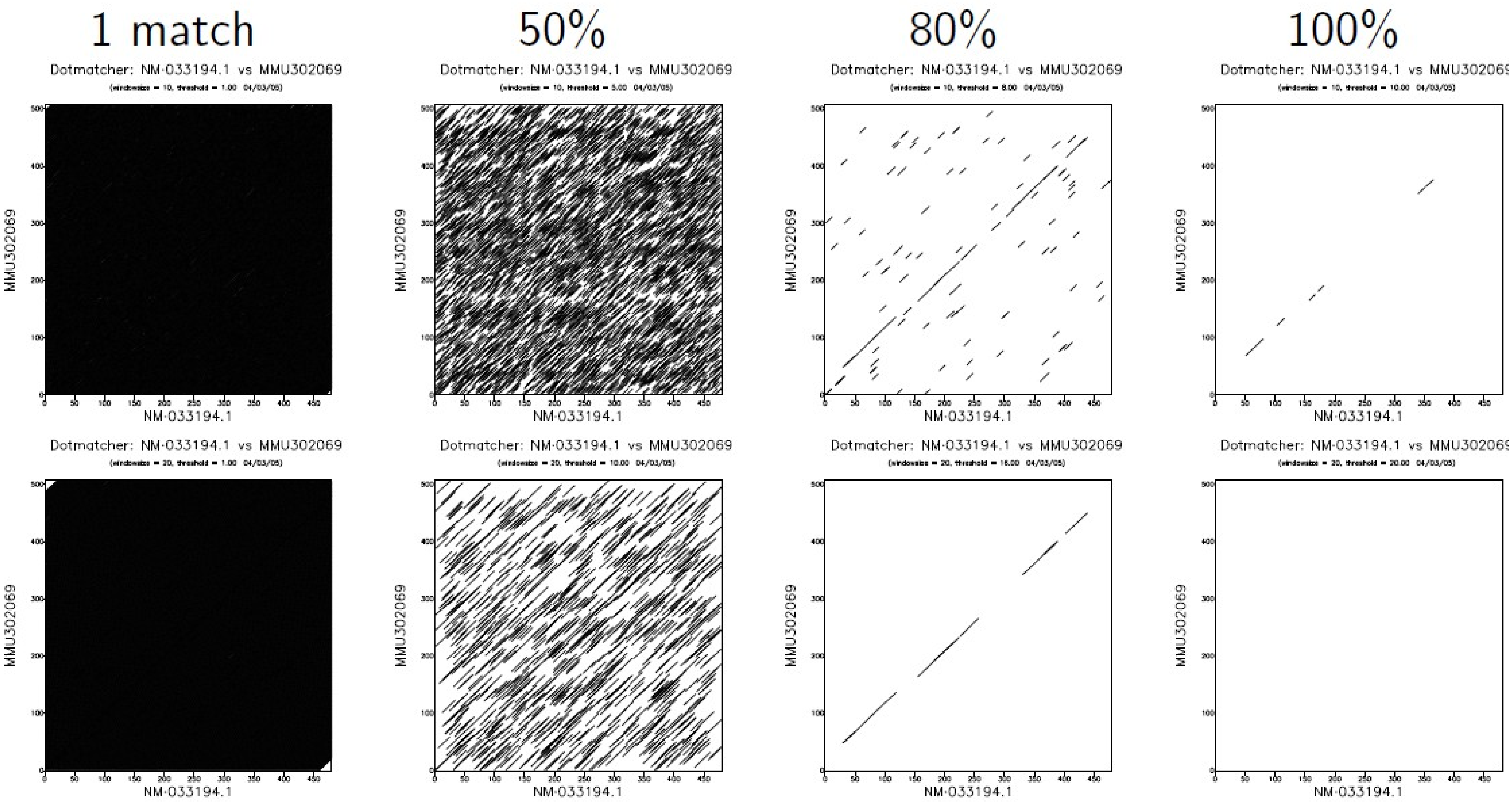


	G	A	T	C	T	A	C
G	*						
T			*		*		
T			*		*		
C				*			*
T			*		*		
G	*						
C				*			*
A		*				*	

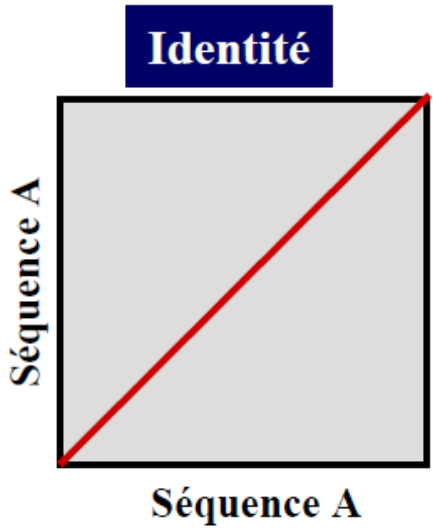
=> fenêtre glissante de taille 3, seuil ≥ 2/3

Dotplot : taille des mots et seuil

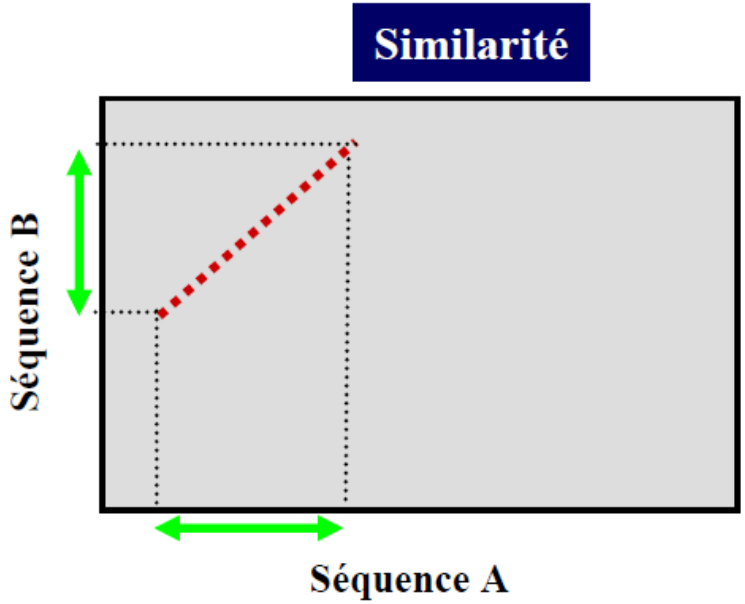
Taille des mots 10 et 20, seuil variant de 1 % à 100 %



Dotplot : interprétation

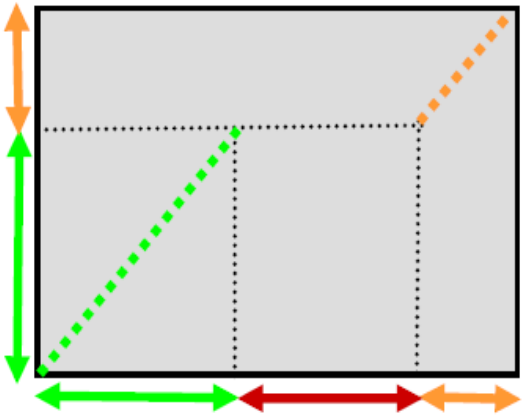


Une grande diagonale en cas d'identité parfaite : la séquence contre elle même

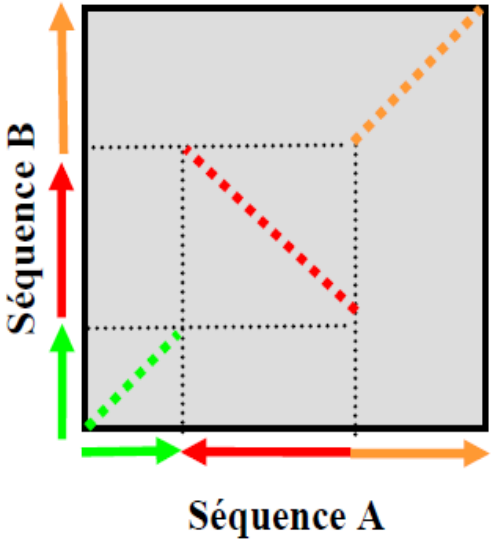


Les régions de similarité apparaissent comme des suites de points alignés ⇒ diagonales

Dotplot : interprétation

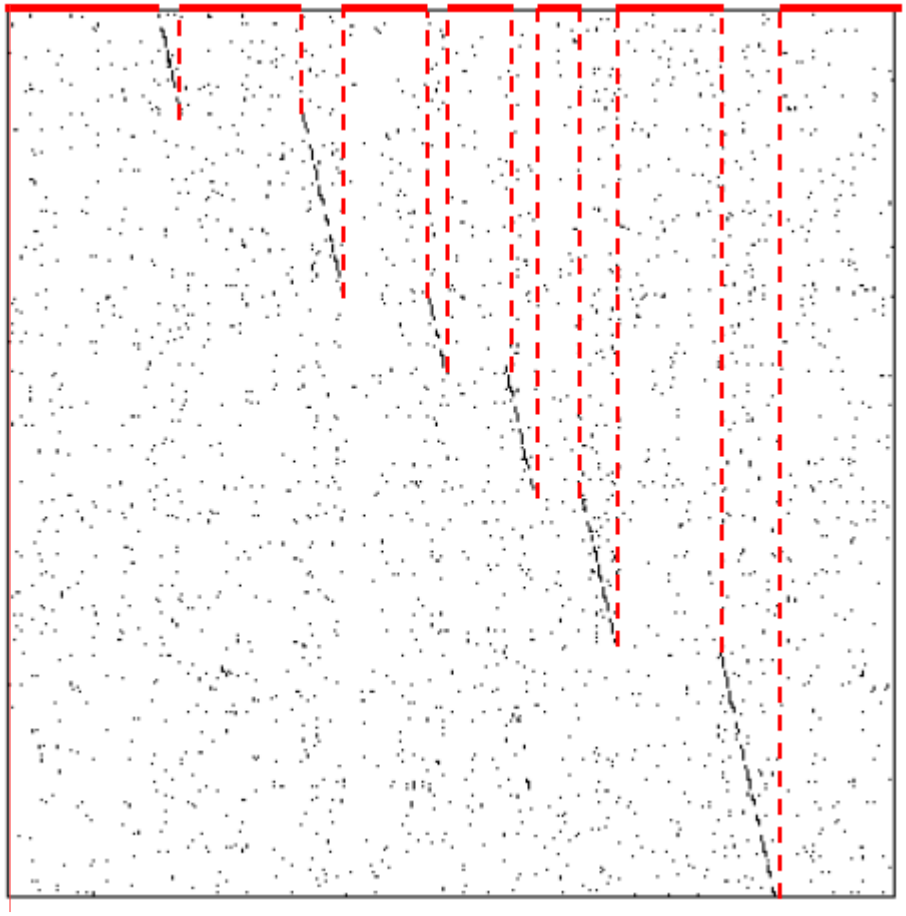


Un décalage par rapport à la diagonale indique une insertion ou une délétion dans l'une des séquences.



Une inversion de l'orientation d'une diagonale traduit une inversion d'une région d'ADN.

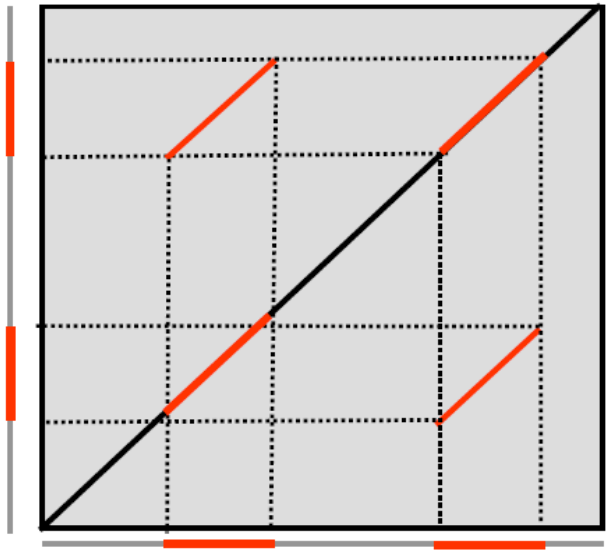
Dotplot : interprétation



horizontalement : séquence nucléaire du gène de l'actine de muscle de Pisaster ochraceus

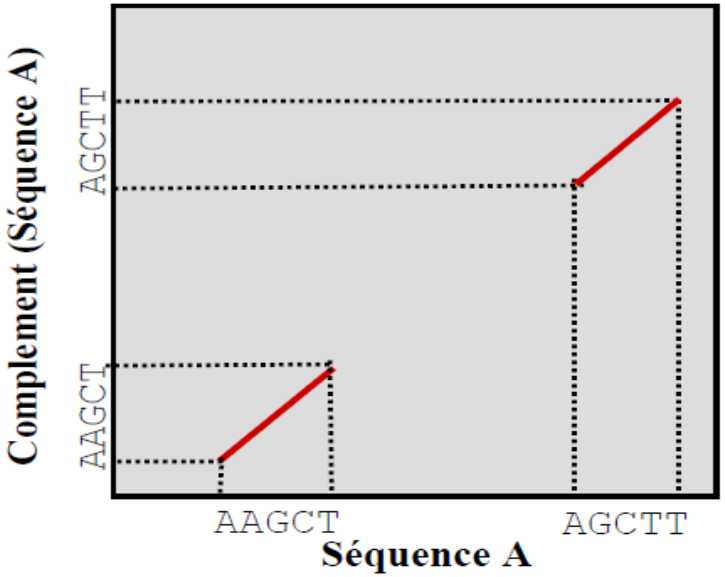
verticalement : cDNA de ce même gène

Dotplot : interprétation



Une séquence avec elle-même :

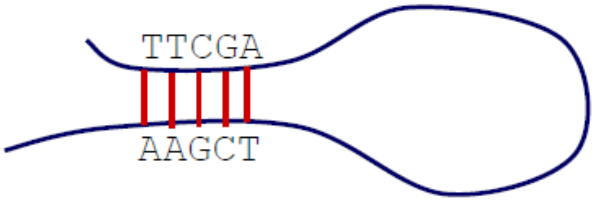
diagonales parallèles => présence de régions répétées



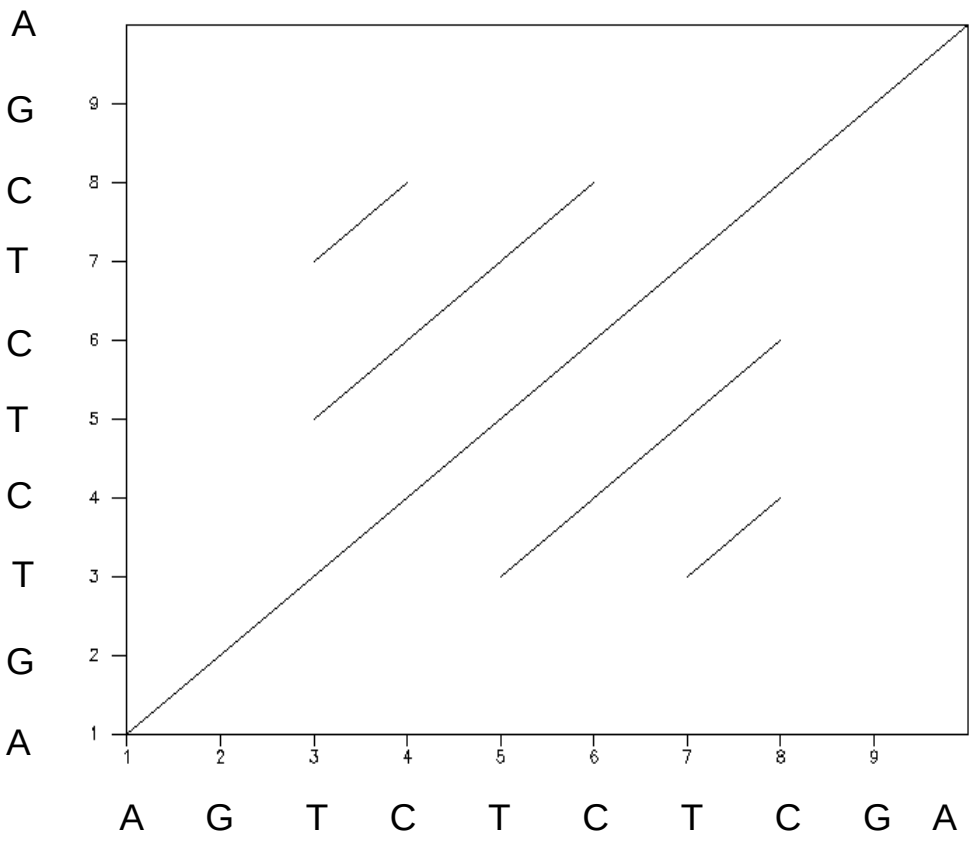
Une séquence avec sa séquence complémentaire :

Détection de régions complémentaires dans une séquence

(ex: structure secondaire d'ARN)



Dotplot : une seule séquence



Répétition en tandem

(micro/mini satellite)

Word size = 2

Dotplot : avantages et inconvénients

- les plus:
  - simple
  - très informatif
- les moins:
  - interprétation ⇒ pas de mesure objective
  - identification ⇒ pas de méthode de détection automatique

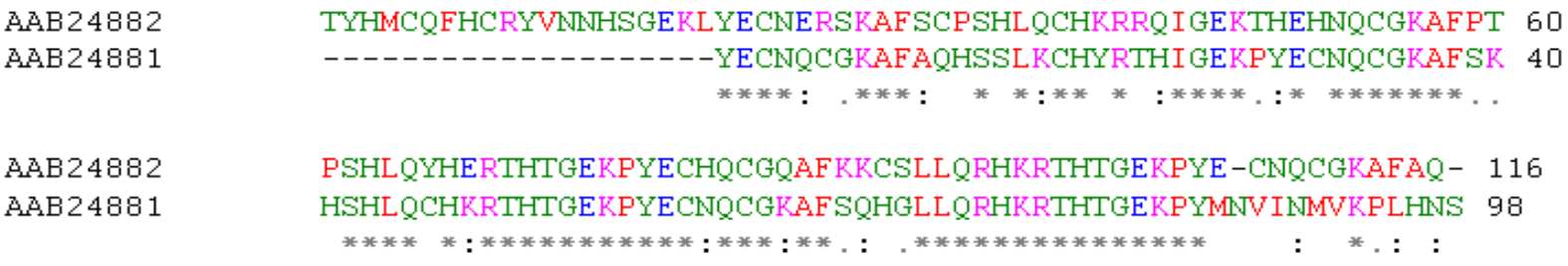
⇒ besoin d'une mesure quantitative de similarité

Dotplot : Exercices

Exercices 1 et 2

Alignement

- 3 types d'alignement
  - global → match sur deux séquences complètes
  - local → match sur des sous-séquences
  - semi-global → chevauchements



Alignement global

Exemple : heat shock protein beta 9

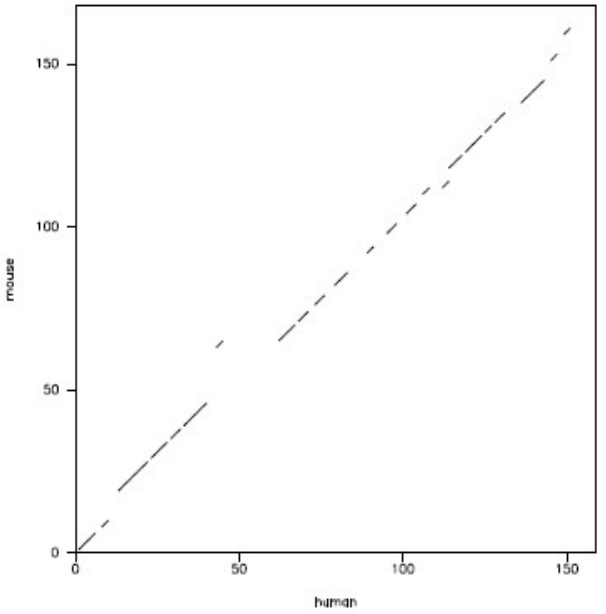
```
>human
MQRVGNTFSN ESRVASRCPS VGLAERNRVA TMPVRLLRDS PAAQEDNDHA RDGFQMKLDA
HGFAPEELVV QVDGQWLMVT GQQQLDVRDP ERVSYRMSQK VHRKMLPSNL SPTAMTCCLT
PSGQLWVRGQ CVALALPEAQ TGPSPRLGSL GSKASNLTR
>mouse
MQRVGSSFST GQREPGENRV ASRCPSVALA ERNQVATLPV RLLRDEVQGN GCEQPSFQIK
VDAQGFAPED LVVRIDGQNL TVTGQRQHES NDPSRGRYRM EQSVHRQMQL PPTLDPAAMT
CSLTPSGHLW LRGQNKCLPP PEAQTGQSQK PRRGGPKSSL QNESVKNP
```

```
human   1 MQRVGNTFS------NESRVASRCPSVGLAERNRVATMPVRLLRDSPAAQ
          |||||::||      .|:||||||||.|||||:|||:|||||||.   .
mouse   1 MQRVGSSFSTGQREPGENRVASRCPSVALAERNQVATLPVRLLRDE---V

human  45 EDNDHARDGFQMKLDAHGFAPEELVVQVDGQWLMVTGQQQLDVRDPERVS
          :.|...:..||:|:||.|||||:|||::|||.|.||||:|.:..||.|..
mouse  48 QGNGCEQPSFQIKVDAQGFAPEDLVVRIDGQNLTVTGQRQHESNDPSRGR

human  95 YRMSQKVHRKM-LPSNLSPTAMTCCLTPSGQLWVRGQCVALALPEAQTG
          |||.|.|||:| ||..|.|.||||.|||||.||:|||...|..|||||.
mouse  98 YRMEQSVHRQMQLPPTLDPAAMTCSLTPSGHLWLRGQNKCLPPPEAQTGQ

human 144 S--PRLGSLGSKASNLTR-----      159
          |  ||.|  |.|:|....
mouse 148 SQKPRRG--GPKSSLQNESVKNP      168
```

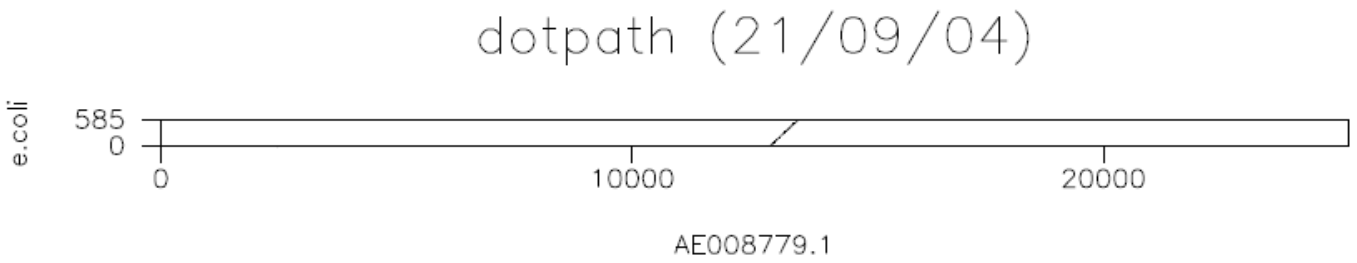


Similarité globale => alignement **global**

Alignement semi-global

AE008779.1 : *Salmonella typhimurium LT2, section 83 of 220 of the complete genome.* *25184 bp*

e.coli : *Escherichia coli peptidyl tRNA hydrolase.* *585 bp*




```
AE008779.1 12901 TCAGGACAAAAAACGTGGCAATTAAATTGATTGTTGGTCTGGCGAATCCC 13201 TGGATCTCCCTCCGGGCGTCGCGAAATTTAAACTTGGCGGCGGCCACGGC
                              |||.|.|||||||||||||.||.||||||||.||| |||||||.|||||.||||||||.|||||||.|.|||||.|||||.||.
e.coli         1              gtgacgattaaattgattgtcggcctggcgaacccc   287 tggatctgcctcctggcgtcgccaaatttaaattgggcggtggccatggt

AE008779.1 12951 GGTGCGGAATATGCCGCGACGCGACACAATGCAGGCGCATGGTACGTCGA 13251 GGCCACAATGGTCTGAAAGACATCATCAGCAAGCTGGGCAATAATCCCAA
                 |||||.|||||.|||||.||||||||.|||||.||.||.||||.|||.|| ||.||||||||.||||||||||||||||||.||..||||.|||||.||.||
e.coli         37 ggtgctgaatacgccgcaacgcgacataatgctggtgcctggttcgttga   337 ggtcacaatggactgaaagacatcatcagtaaattgggtaataaccctaa

AE008779.1 13001 TTTACTGGCGGAGCGCCTGCGCGCGCCGTTGCGTGAAGAGCCTAAATTCT 13301 CTTTCACCGATTACGCGTTGGAATTGGTCATCCAGGCGATAAAAATAAAG
                 .||||||||.||||||..|||||||.|||.||||.||||||.|||||||||| |||||||||.|||||.|.||||||.|||||||.||||||||||||||
e.coli         87 cttactggcagagcgtttgcgcgctccgctgcgcgaagaggctaaattct   387 ctttcaccgtttacgcatcggaatcggtcatccgggcgataaaaataaag

AE008779.1 13051 TTGGCTATACCTCACGCATCACGCTGGAAGGGGAAGATGTTCGCCTGCTG 13351 TTGTTGGTTTCGTGCTGGGTAAACCCCCTGTTTCTGAACAAAAATTAATT
                 ||||.||||||.||.||..||||.||.|.|||.||||||||.||||||.|. |||||.||||||.|||.|.||.||||||..||||||.||.||||||
e.coli       137 ttggttatacttcgcgagtcactcttggaggcgaagatgtccgcctgtta   437 ttgtcggttttgtgttaggcaaaccgcctgttagtgaacagaagttaatt

AE008779.1 13101 GTACCCACCACGTTCATGAACCTCAGTGGTAAAGCAGTTGGCGCAATGGC 13401 GATGAGGCCATTGACGAAGCGGCACGCTGTACGGAATTGTGGTTCAAAGA
                 ||.||.||.||.||.|||||.|||||.||.|||||.||||..||.|||||| |||||.||||||||||||||||.||.|||||.|||.|||||||.|.|||
e.coli       187 gtcccgactacatttatgaatctcagcggcaaagccgttgcggcgatggc   487 gatgaagccattgacgaagcggcgcgttgtactgaaatgtggtttacaga

AE008779.1 13151 CAGTTTTTACCGTATTCAGCCGGACGAAATTTTGGTCGCTCACGACGAGC 13451 GGGTCTGGCCAAAGCAACAAGCCGTTTGCATACCTTTAAGGCGCAATAAC
                 ||||||||.|||.|||.|.||||||||||||.||||.||.||||||||.| .||..||.||||||||||.|.|||.||||||..|||||||.|||||||||
e.coli       237 cagtttttccgcattaatccggacgaaattctggtggcccacgacgaac   537 tggcttgaccaaagcaacgaaccgattgcacgcctttaaagcgcaataa
```

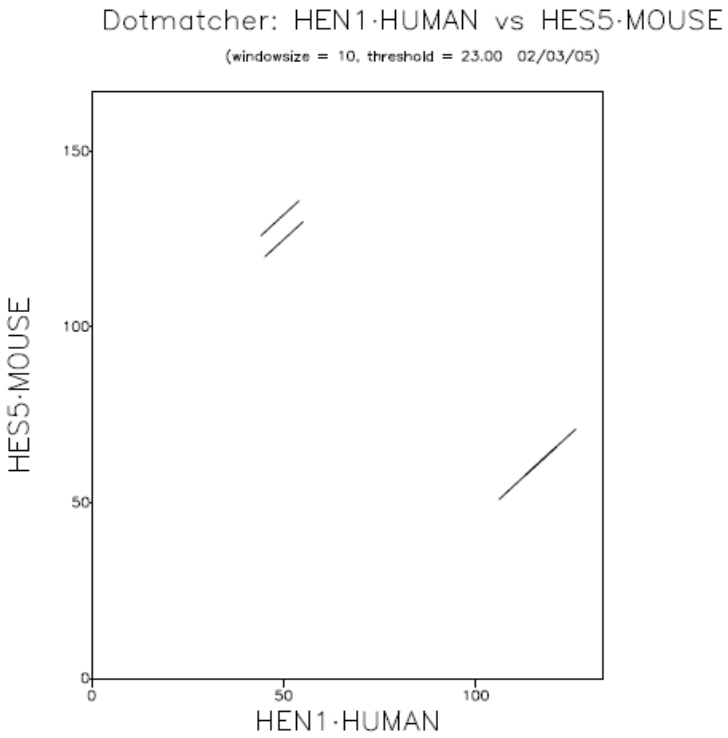
similarité globale sur une des deux séquences => alignement **semi-global**

Alignement local

Exemple : domaine conservé

```
>HEN1_HUMAN
MMLNSDTMELDLPPTHSETESGFSDCGGGAGPDGAGPGGPGGGQARGPEPGEPGRKDLQHLSREERRRRR
RATAKYRTAHATRERIRVEAFNLAFAELRKLLPTLPPDKKLSKIEILRLAICYISYLNHVLDV
>HES5_MOUSE
MAPSTVAVEMLSPKEKNRLRKPVVEKMRRDRINSSIEQLKLLLEQEFARHQPNSKLEKADILEMAVSYLK
HSKAFAAAAGPKSLHQDYSEGYSWCLQEAVQFLTLHAASDTQMKLLYHFQRPPAPAAPAKEPPAPGAAPQ
PARSSAKAAAAAVSTSRQPACGLWRPW
```

```
            110       120
HEN1_H PDKKLSKIEILRLAICYISY
       :..::.: .::..:. :...
HES5_M PNSKLEKADILEMAVSYLKH
             60        70
```



similarité sur une sous-partie => alignement **local**

Alignement

- données:
  - une paire de séquences (ADN / protéine)
  - une schéma de score : comment compter ce qui se ressemble ?
- but:
  - déterminer le degré de similarité (meilleur score)
  - montrer la similarité (meilleur alignement)
- décrit la ressemblance grâce à 3 opérations (mutations ponctuelles)
  - insertion
  - délétion
  - identité/substitution (match/mismatch)
- mesure la ressemblance en donnant un poids à chacune des opérations
  - poids positif (“récompense”) aux bonnes parties de l’alignement (appariement de deux lettres identiques ou proches)
  - poids négatif ou nul (“pénalité”) associé aux mauvaises (appariement de deux lettres non relatées, non-appariement)

Composantes d'un schéma de score

- score (ou poids) pour une identité/substitution :
  - matrice de similarité:
  - s(a, b) = score d'alignement des nucléotides a et b

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

- score (ou poids) d'un indel (insertion/délétion) :
  - score unitaire : -2 par exemple par indel

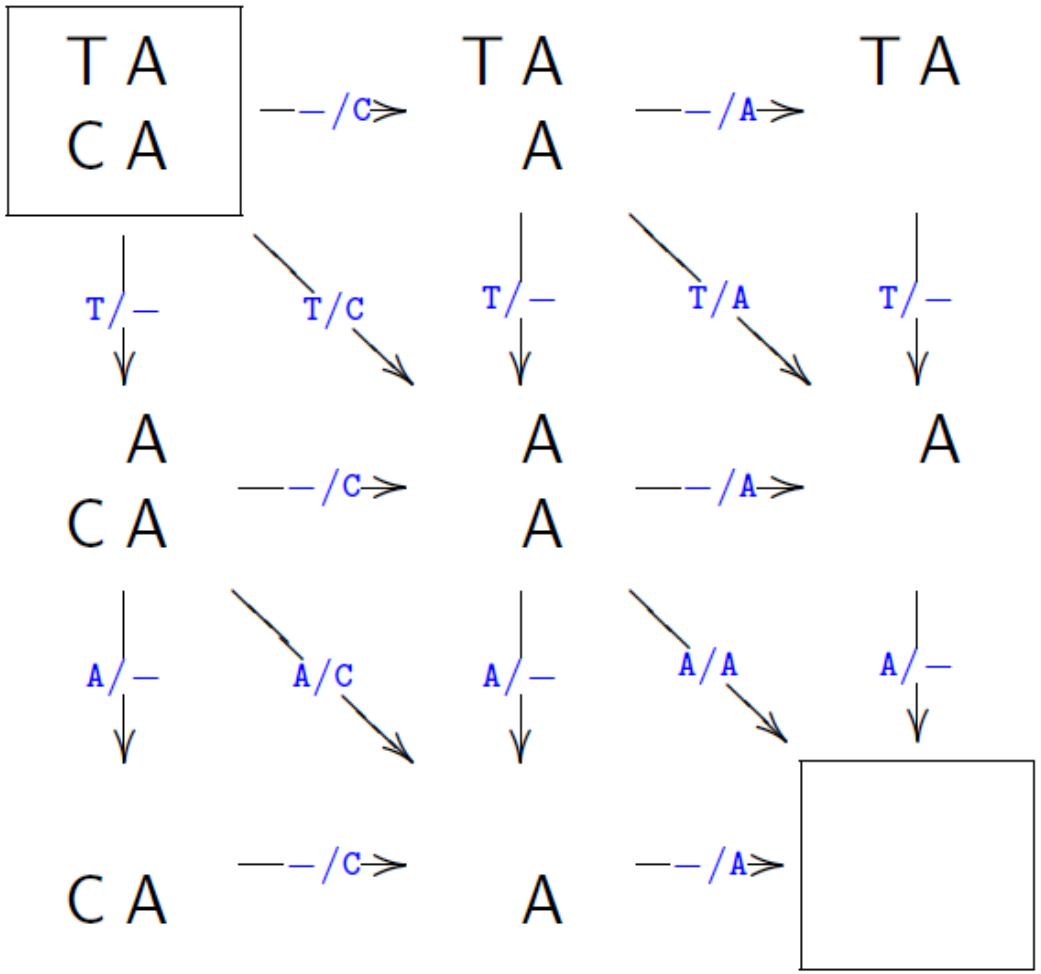
Composantes d'un schéma de score

- score de l'alignement = somme des scores des événements élémentaires
- Exemple :

```
A  A  C  G  T  A  C  G  A  T  A
|     |  |  |  |        |     |
A  -  C  G  T  A  -  A  A  G  A
-------------------------------
2 -2  2  2  2  2 -2  0  2  0  2 = 10
```

Quel est le meilleur alignement ?

- Soient 2 séquences de longueur n: alignement de longueur maximale 2n
- exemple avec les séquences TA et CA



TA
CA

TA-
-CA

TA-
C-A

-TA
CA-

-TA
C-A

T-A
-CA

T-A
C-A

TA--
--CA

--TA
CA--

T-A-
-C-A

-T-A
C-A-

-TA-
C--A

Quel est le meilleur alignement ?

- nombre max d'alignements (séquences de longueur n)

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \sim \frac{2^{2n}}{\sqrt{2\pi n}}$$

pour deux séquences de longueur 100 : 2.10^{57} alignements

- grâce à la représentation en tableau : complexité en temps et en espace O(n2) (proportionnel au produit de la longueur des séquences)
- pour deux séquences de longueur 100 : 10,000 opérations

Comment calculer le meilleur alignement ?

- Alignement Global : Needleman & Wunsch, 1970
- Recherche du meilleur alignement global

- **Programmation dynamique** : méthode algorithmique pour résoudre des problèmes d'optimisation
  - on débute par les solutions de sous-problèmes plus petits pour ensuite déduire progressivement les solutions de l'ensemble

- Les deux séquences sont placées sur les axes d'une **matrice de scores** d'alignement
  - une case donne le score optimal d'alignement entre les deux sous-séquences correspondantes

Algorithme de Needleman & Wunsch

Définition : Alignement par paires par programmation dynamique

Soient deux séquences S et T de longueurs respectives N et P. La recherche d'un alignement global optimal entre S et T suivant une fonction de score w est obtenu par construction d'une *matrice des scores optimaux d'alignement* M[0..N,0..P] telle que :

- initialisation :

$$M[0,0] = 0$$
$$M[i,0] = M[i-1,0] + w(x_i,-) \quad \text{pour tout i de 1 à N}$$
$$M[0,j] = M[0,j-1] + w(-,y_j) \quad \text{pour tout j de 1 à P}$$

- calcul du score optimal :

$$M[i,j] = \text{optimise} \begin{cases} M[i-1,j-1] + w(x_i,y_j) \\ M[i-1,j] + w(x_i,-) \\ M[i,j-1] + w(-,y_j) \end{cases}$$

où M[i,j] représente le score de l'alignement de S[1..i] avec T[1..j] et optimise est la fonction :

- *maximum* si w est une matrice de score maximisante
- *minimum* si w est une matrice de score minimisante

La matrice des directions est obtenue par les formules suivantes :

- initialisation :

$$Dir[0,0] = x$$
$$Dir[i,0] = \leftarrow \quad \text{pour tout i de 1 à N}$$
$$Dir[0,j] = \uparrow \quad \text{pour tout j de 1 à P}$$

- calcul des directions :

$$Dir[i,j] = \text{Union} \begin{cases} \nwarrow \text{si } M[i,j] = M[i-1,j-1] + w(x_i,y_j) \\ \uparrow \text{si } M[i,j] = M[i-1,j] + w(x_i,-) \\ \leftarrow \text{si } M[i,j] = M[i,j-1] + w(-,y_j) \end{cases}$$

Comment calculer le meilleur alignement ?

Exemple : alignement de ATT contre TTC

Initialisation

S(0,0)=0

S(1,0)=S(1-1,0)+s(A,-)

= S(0,0)+s(A,-)

= 0 -1 = -1

Dynamic programming and backtracking

source i

target j

		-		A		T		T	
-		0	-1	-1	-2	-2	-3	-3	
		-1	-1	-2	0	-3	-1	-4	
T		-1	-2	-1	-2	0	-1	-1	
		-2	-2	-2	0	-1	+1	-2	
T		-2	-3	-2	-3	0	-1	+1	
		-3	-3	-3	-3	-1	-1	0	
C		-3	-4	-3	-4	-1	-2	0	

	-	A	T	T
-	0	-1	-2	-3
T	-1	-1	0	-1
T	-2	-2	0	+1
C	-3	-3	-1	0

$S_{i-1,j-1}$ $+s(a_i,b_j)$	$S_{i,j-1}$ $+s(-,b_j)$
$S_{i-1,j}$ $+s(a_i,-)$	$S_{i,j}$

scoring scheme :

- $s(a_i,b_i) = +1$ if $a_i = b_i$
- $s(a_i,b_i) = -1$ if $a_i = b_i$
- $s(a_i,-) = -1$
- $s(-,b_i) = -1$

A T T -
- T T C

Comment calculer le meilleur alignement ?

Exemple : alignement de ATT contre TTC

Initialisation

S(0,0)=0

S(0,2)=S(0,2-1)+s(-,T)

= S(0,1)+s(A,-)

= -1-1 = -2

Dynamic programming and backtracking

source i

target j

		-		A		T		T	
-		0	-1	-1	-2	-2	-3	-3
		-1	-1	-2	0	-3	-1	-4
T		-1	-2	-1	-2	0	-1	-1
		-2	-2	-2	0	-1	+1	-2
T		-2	-3	-2	-3	0	-1	+1
		-3	-3	-3	-3	-1	-1	0
C		-3	-4	-3	-4	-1	-2	0

		-	A	T	T
-		0	-1	-2	-3
T		-1	-1	0	-1
T		-2	-2	0	+1
C		-3	-3	-1	0

$S_{i-1,j-1}$ $+s(a_i,b_j)$	$S_{i,j-1}$ $+s(-,b_j)$
$S_{i-1,j}$ $+s(a_i,-)$	$S_{i,j}$

scoring scheme :

- $s(a_i,b_i) = +1$ if $a_i = b_i$
- $s(a_i,b_i) = -1$ if $a_i = b_i$
- $s(a_i,-) = -1$
- $s(-,b_i) = -1$

A T T -
- T T C

Comment calculer le meilleur alignement ?

Exemple : alignement de ATT contre TTC

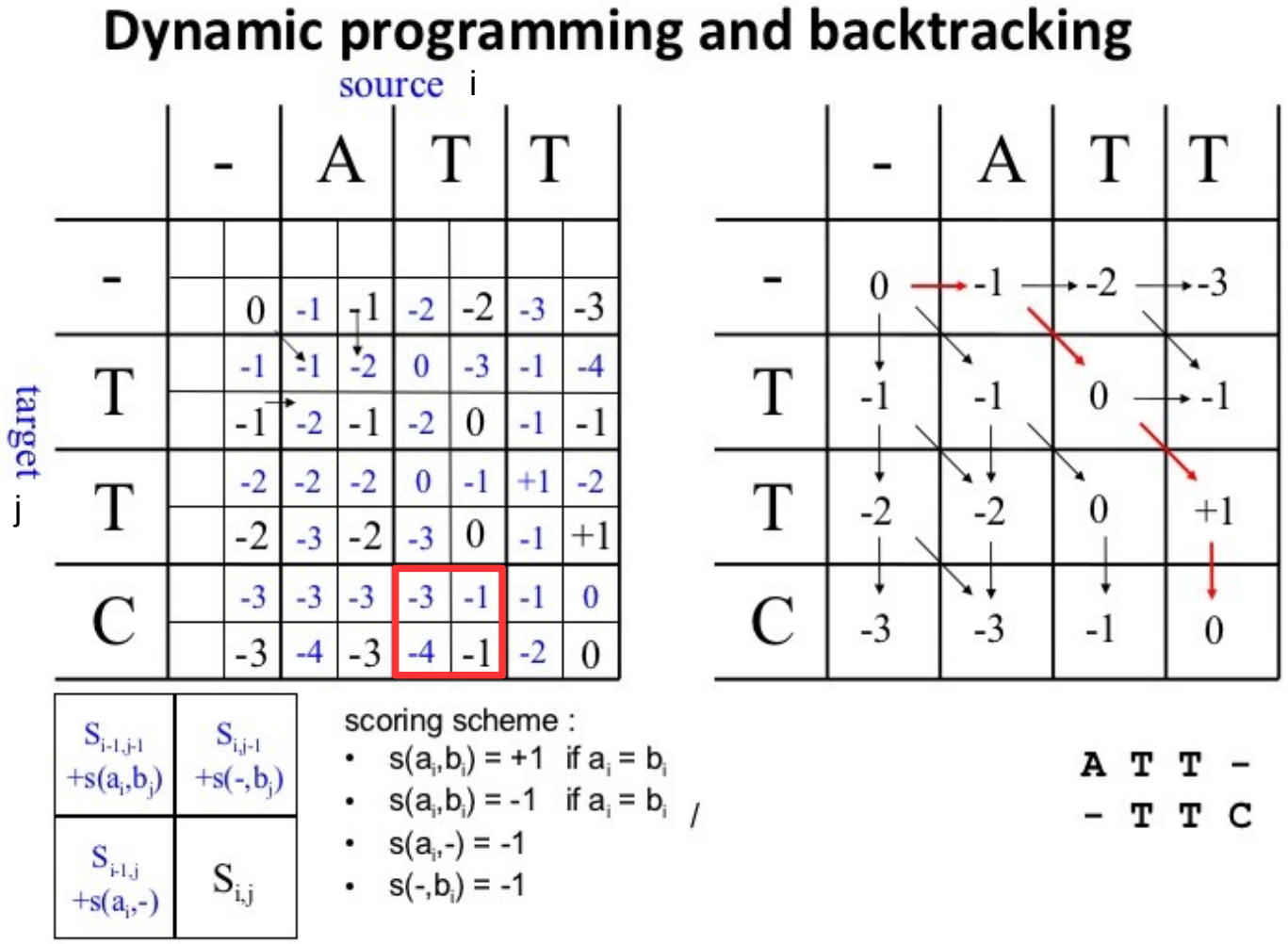
Calcul du score S(2,3)

=> S(1,2)+s(T,C)=-2-1=-3

=>S(2,2)+s(-,C)=0-1=-1

=>S(1,3)+s(T,-)=-3-1=-4

Max => S(2,3)=-1



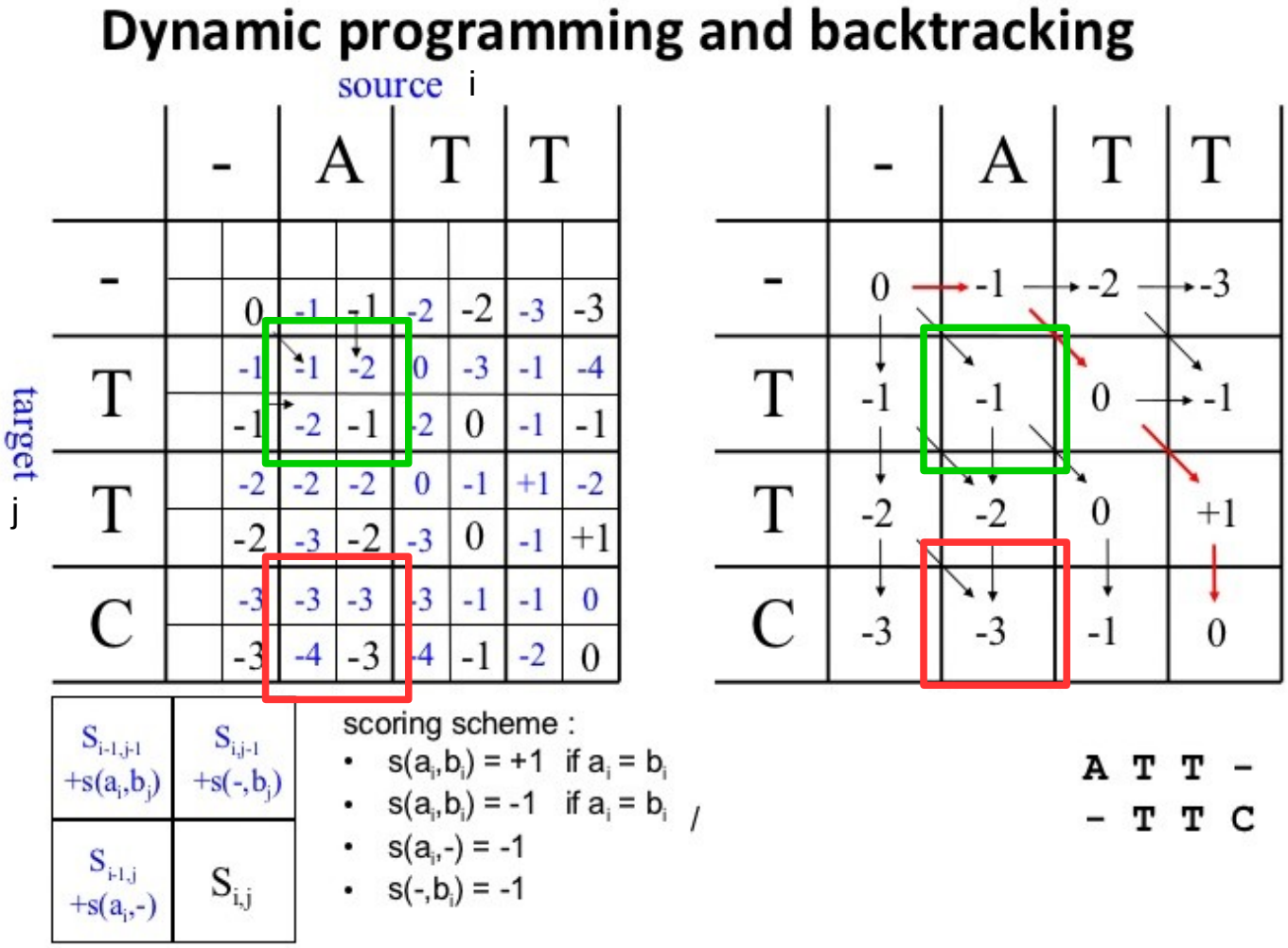
Comment calculer le meilleur alignement ?

Exemple : alignement de ATT contre TTC

Stockage des directions :

Pour chaque case,

la direction est stockée



Comment calculer le meilleur alignement ?

Exemple : alignement de ATT contre TTC

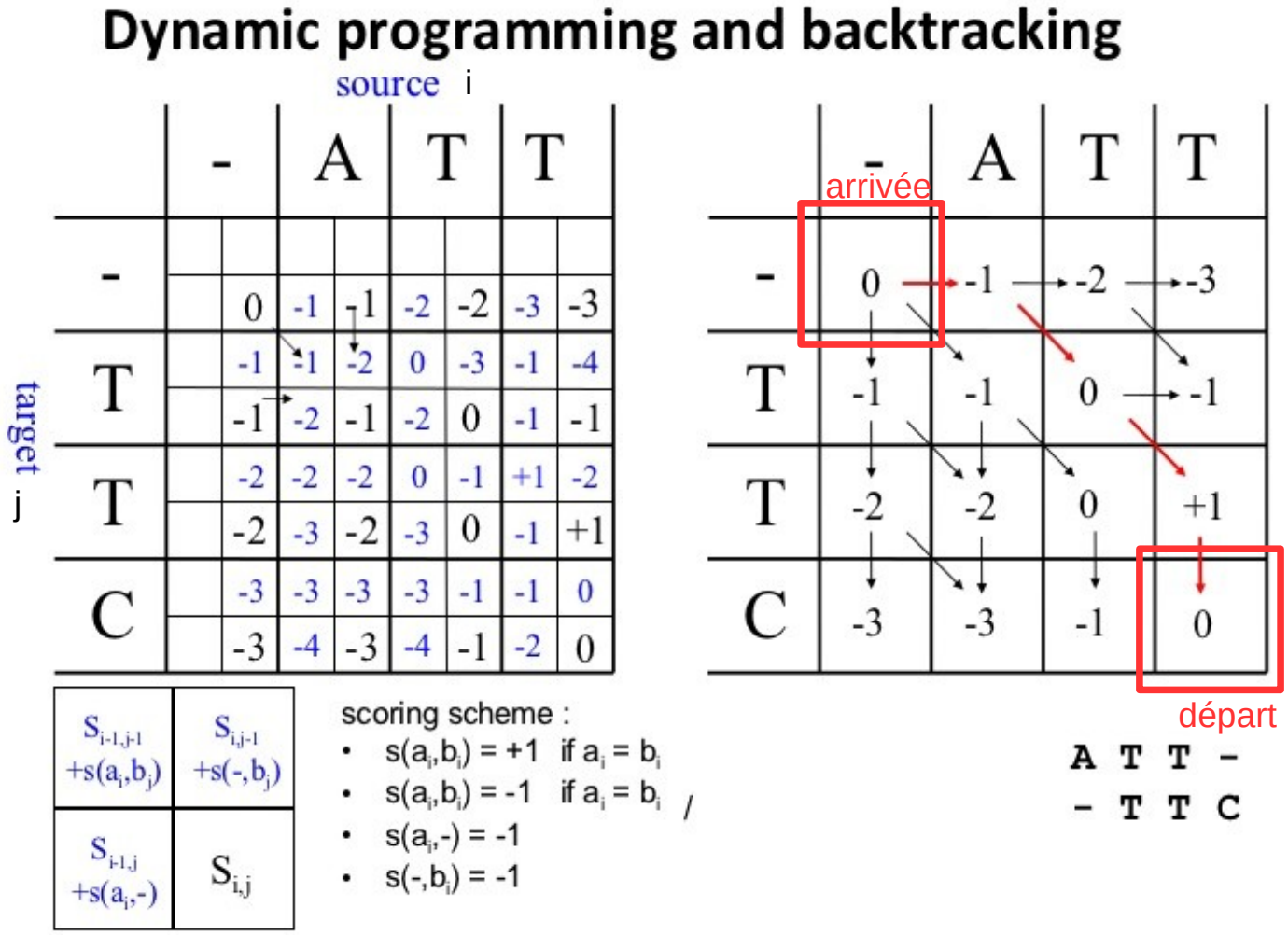
Backtracking :

Permet de reconstruire

le ou les alignements

optimaux pour le système

de score



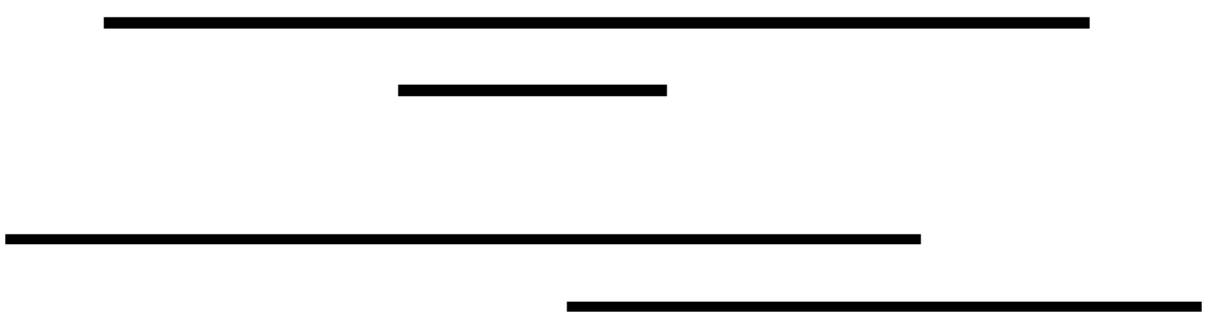
Comment calculer le meilleur alignement ?

- Alignement semi-global

```
CAGCACTTGGATTCTCGG
||||      | |     ||
CAGC-----G-T----GG

CAGCA-CTTGGATTCTCGG
   ||  |  |||
---CAGCGTGG--------
```

- ne pénalise pas les gaps (séries d'indels) aux extrémités sinon, similaire à l'alignement global
- permet de détecter des similarités de ce type:



Comment calculer le meilleur alignement ?

- Alignement local : algorithme de Smith & Waterman, 1981
- modification de l'algorithme de Needleman-Wunsch :
  - Initialisation = 0
  - si le score est négatif alors = 0 => si le meilleur alignement jusqu'à la position (i, j) aboutit à un score négatif on le stoppe et on recommence un nouvel alignement local en cette position

Définition : Alignement local par programmation dynamique

Soient deux séquences S et T de longueurs respectives N et P (avec N < P). La recherche d'un alignement local optimal entre S et T suivant une fonction de score w est obtenu par construction d'une *matrice des scores optimaux d'alignement* M[0..N,0..P] telle que :

- initialisation :

$$M[0,0] = 0$$
$$M[i,0] = 0 \quad \text{pour tout } i \text{ de } 1 \text{ à } N$$
$$M[0,j] = 0 \quad \text{pour tout } j \text{ de } 1 \text{ à } P$$

- calcul du score optimal :

$$M[i,j] = \max \begin{cases} M[i-1,j-1] + w(x_i,y_j) \\ M[i-1,j] + w(x_i,-) \\ M[i,j-1] + w(-,y_j) \\ 0 \end{cases}$$

où M[i,j] représente le score de l'alignement de S[1..i] avec T[1..j] et w est une matrice de score maximisante.

La matrice des directions est obtenue par les mêmes formules que celles définies pour l'alignement global.

Comment calculer le meilleur alignement ?

- L'alignement optimal ne part plus nécessairement de la cellule (n, m), mais de la cellule de plus grand score,et s'arrête à la première cellule contenant un zéro.

		A	E	K	P	C	A	Y	E	N	N	E	F
	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	1	0	0	0	0	0	1	0	0	1	0
K	0	0	0	2	1	0	0	0	0	0	0	0	0
C	0	0	0	1	1	2	1	0	0	0	0	0	0
A	0	1	0	0	0	1	3	2	1	0	0	0	0
Y	0	0	0	0	0	0	2	4	3	2	1	0	0
E	0	0	1	0	0	0	1	3	5	4	3	2	1
N	0	0	0	0	0	0	0	2	4	6	5	4	3
E	0	0	1	0	0	0	0	1	3	5	5	6	5
P	0	0	0	0	1	0	0	0	2	4	4	5	4
I	0	0	0	0	0	0	0	0	1	3	3	4	3
L	0	0	0	0	0	0	0	0	0	2	2	3	2
A	0	1	0	0	0	0	1	0	0	1	1	2	1

EKPCAYEN
EK-CAYEN

EKPCAYENNE
EK-CAYEN-E

EKPCAYENNE
EK-CAYE-NE

Retour sur le système de score

- le score unitaire pour une identité/substitution ne reflète pas la réalité :
  - certaines substitutions sont moins graves (ex : acides aminés ayant des propriétés physico-chimiques proches)
  - Certaines identités sont plus importantes (ex : l'alignement de 2 tryptophanes a plus de poids que des acides aminés plus communs)
- le score unitaire pour un gap ne reflète pas la réalité :
  - Au niveau de l'évolution, il vaut mieux avoir des gaps groupés

Pénalité associée aux gaps

- fonctions de gaps différentes ⇒ algorithmes différents
- la plus simple ⇒ fonction linéaire : $g \times l$
  - (l : longueur du gap)
- fonctions plus réalistes :
  - fonctions affines : $o + e \times l$
    - o : pénalité d'ouverture de gap
    - e : pénalité d'extension de gap
  - fonctions logarithmiques

Différents alignements possibles

	A	B	C
Match Cost	1	1	1
Mismatch Cost	-1	-1	-1
Gap Open Penalty (o)	0	1	4
Gap Extension Penalty (e)	0	0.1	0.1

AL1 :

```
AT-GCGGGACA-TG
|  |||   |  ||
A-GGCG---C-CTG
```

(7 matches, 0 mismatches, 5 ogaps, 2 egaps, score=7.0)

AL2 :

```
ATGCGGGACATG
|.|||   |.||
AGGCG---CCTG
```

(7 matches, 2 mismatches, 1 ogap, 2 egaps, score=3.8)

AL3 :

```
ATGCGGGACATG
   .||..|.||
---AGGCGCCTG
```

(5 matches, 4 mismatches, 1 ogap, 2egaps, score=1.0)

Comment bien choisir les pénalités de gaps?

- Peu de connaissance a priori
- Spécificité aux données
- Valeurs typiques pour une fonction de gap affine
  - $0.5 < o < 5.0$
  - $0.05 < e < 1.0$
- Toujours prendre (en valeur absolue) $o > 1/2$ substitution

Exercice

- Voici deux séquences artificielles AAGTCATTCGCACATCG et AACACATCG et trois alignements possibles:

```
AAGTCATTCGCACATCG    AAGTCATTCGCACATCG    AAGTCATTCGCACATCG
||  |||              ||  |||  ||          ||       |||||||
AACACATCG            AACACAT-CG           AA--------CACATCG
```

1 - Calculez le score de chaque alignement avec le jeu de paramètres proposé par défaut par la plupart des programmes d'alignement :

- alignement semi-global
- Appariement = 5 Mésappariement = -4
- Ouverture d'indel = -10 ; Extension d'indel = -0,5 ;

2- Quel sera l'alignement retenu par un programme d'alignement avec ces paramètres ?

3- Changer la valeur d'un seul paramètre pour que le premier alignement possède le meilleur score ; puis pour que ce soit le second

Les matrices de score

- Impliquées dans toutes les analyses par comparaison de séquences
- Représentent implicitement une théorie de l'évolution
- Résultats fortement dépendants de la matrice
- La compréhension d'une matrice ⇒ un bon choix

Similarité vs. distance

- Un élément de la matrice représente:
  - le coût du remplacement d'une base par une autre (**distance**)
  - la mesure de la **similarité** du remplacement

- Même principe de recherche :

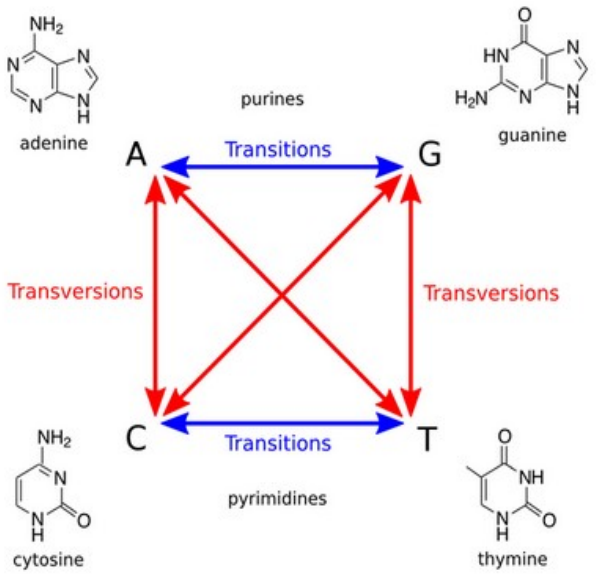
  maximiser un score ≡ minimiser une distance

- La matrice de distance et de similarité peuvent être déduites l'une de l'autre :

  Similarité = 1 - Distance

Les matrices de score pour l'ADN

- Transition : substitution sans changement de famille
- Transversion : avec changement de famillle



Identité (similarité)

	A	C	G	T
A	1	0	0	0
C	0	1	0	0
G	0	0	1	0
T	0	0	0	1

BLAST (similarité)

	A	C	G	T
A	1	-3	-3	-3
C	-3	1	-3	-3
G	-3	-3	1	-3
T	-3	-3	-3	1

Transition/Transversion

	A	C	G	T
A	5	-4	-3	-4
C	-4	5	-4	-3
G	-3	-4	5	-4
T	-4	-3	-4	5

Les matrices de scores pour les protéines

- Les scores sont calculées basé sur le **Log-odds ratio**
- Exprime le ratio entre:
  - la probabilité que deux résidus i et j soient alignés par descendance
  - la probabilité que ceux-ci soient alignés par chance

$$S_{ij} = \log \frac{q_{ij}}{p_i p_j}$$

- Explication :
  - q_{ij} = la fréquence que l'alignement de i et j soit observé dans des séquences homologues
  - p_i = la fréquence d'occurrence de i
  - un score est > 0 si une substitution est plus fréquente qu'attendue par hasard

Les matrices PAM "Point Accepted Mutation"

- Représente les échanges possibles ou acceptable d'un acide aminé par un autre lors de l'évolution (Dayhoff et al.,1978)

- Si deux séquences appartiennent au même processus évolutif, et qu'un acide aminé de l'une a été muté pour donner l'autre, alors on peut supposer que les deux acides aminés sont similaires :
  - les mutations sont dites acceptées
  - elles ont été conservées au cours de l'évolution de part leur caractère à ne pas altérer la fonction de la protéine

Les matrices PAM "Point Accepted Mutation"

- Les protéines évoluent via des successions de mutations ponctuelles indépendantes les unes des autres et acceptées dans la population

- Basées sur l'alignement global de ~1300 protéines conservées à plus de 85% appartenant à 71 familles de protéines

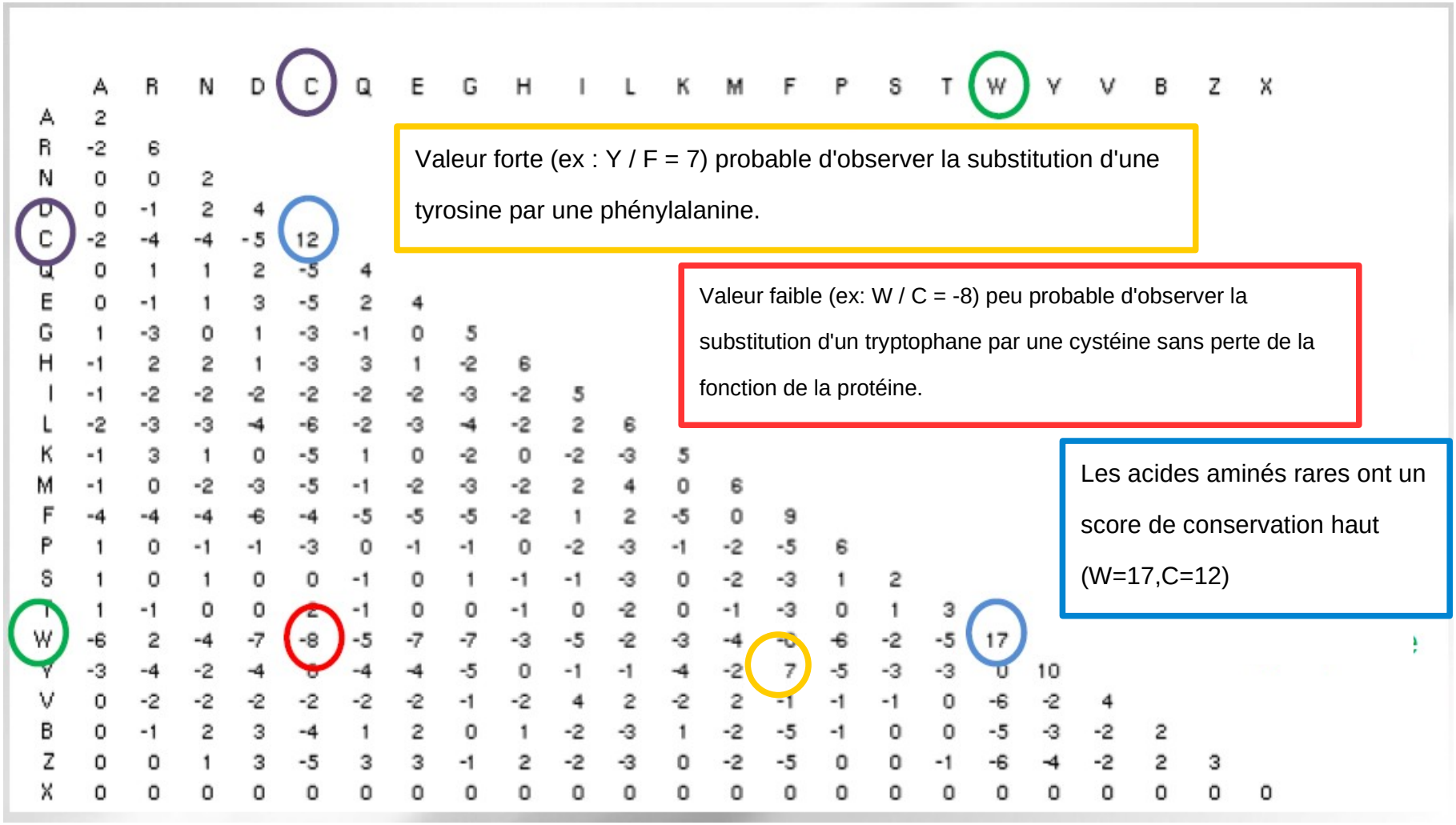
- Aujourd'hui actualisées : 16 130 séquences appartenant à 2 621 familles de protéines

Les matrices PAM "Point Accepted Mutation"

- PAM-1 :
  - donne la probabilité qu'une substitution soit acceptée pour 100 acides aminés
- PAM-X
  - multiplication X fois de cette matrice par elle-même permet d'analyser des distances d'évolution plus importantes
- PAM-250 :
  - Matrice de mutation standard de Dayhoff
  - Optimal pour l'alignement de séquences très éloignés

Les matrices PAM "Point Accepted Mutation"

- Exemple de la matrice PAM-250



	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X
A	2																						
R	-2	6																					
N	0	0	2																				
D	0	-1	2	4																			
C	-2	-4	-4	-5	12																		
Q	0	1	1	2	-5	4																	
E	0	-1	1	3	-5	2	4																
G	1	-3	0	1	-3	-1	0	5															
H	-1	2	2	1	-3	3	1	-2	6														
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5													
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6												
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5											
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6										
F	-4	-4	-4	-6	-4	-5	-5	-5	-2	1	2	-5	0	9									
P	1	0	-1	-1	-3	0	-1	-1	0	-2	-3	-1	-2	-5	6								
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2							
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3						
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17					
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10				
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4			
B	0	-1	2	3	-4	1	2	0	1	-2	-3	1	-2	-5	-1	0	0	-5	-3	-2	2		
Z	0	0	1	3	-5	3	3	-1	2	-2	-3	0	-2	-5	0	0	-1	-6	-4	-2	2	3	
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Valeur forte (ex : Y / F = 7) probable d'observer la substitution d'une tyrosine par une phénylalanine.

Valeur faible (ex: W / C = -8) peu probable d'observer la substitution d'un tryptophane par une cystéine sans perte de la fonction de la protéine.

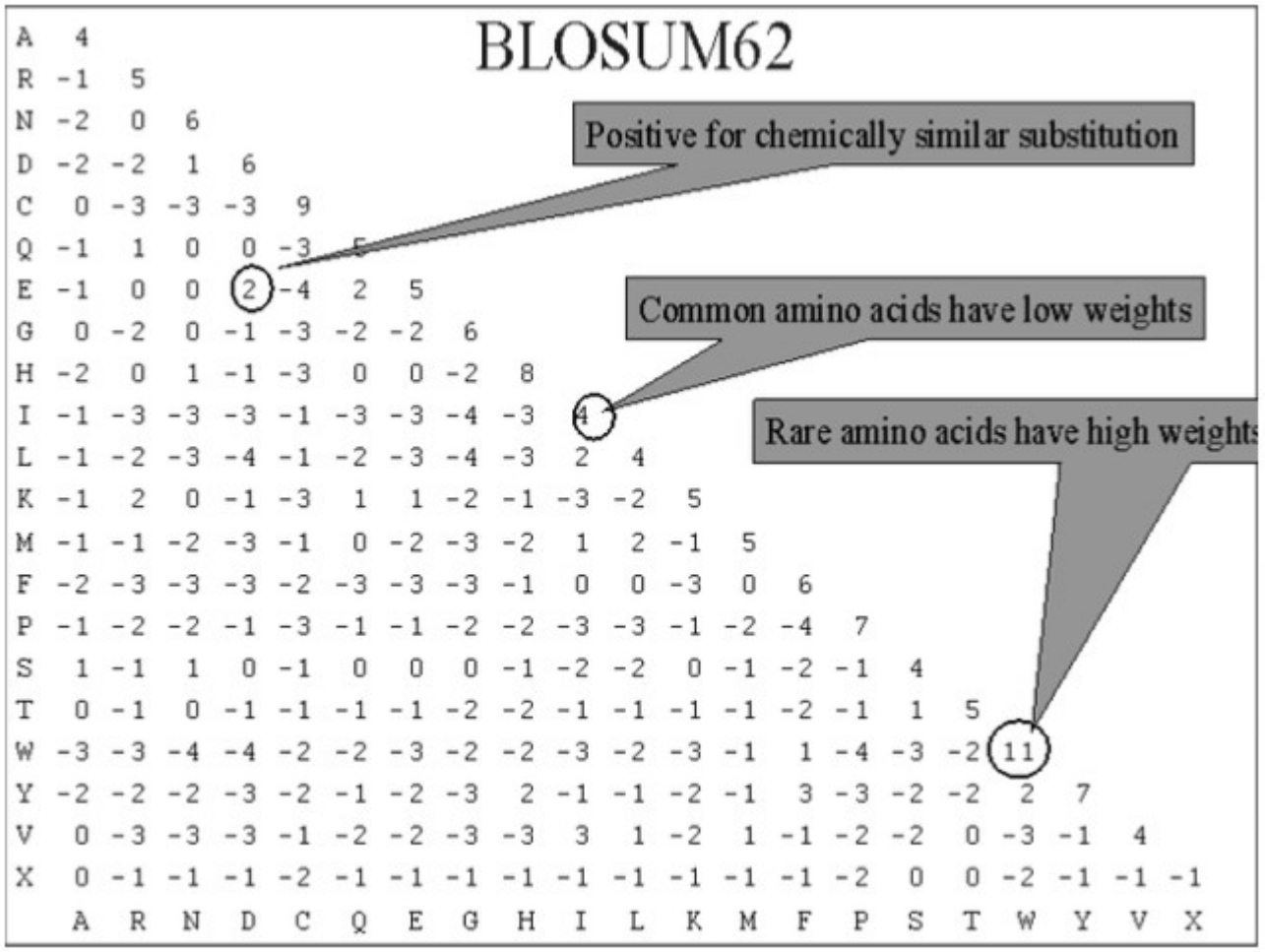
Les acides aminés rares ont un score de conservation haut (W=17,C=12)

Les matrices BLOSUM

- BLOSUM :BLOcks of Amino Acid Substitution Matrix
- Le but : détecter des relations entre protéines plus éloignées
- Avec les matrices PAM, les valeurs pour des protéines éloignées sont extrapolées, avec BLOSUM, ces valeurs sont obtenues en comparant des blocs facilement «alignables» (alignement multiple local sans gaps) dans des familles de protéines très éloignées
- Convient bien pour la recherche de similarités locales
- BLOSUM-N : seuil de similarité, N = % de similarité

Les matrices BLOSUM

- Fréquence de changements entre deux acides aminés avec conservation de structure



BLOSUM-62
(matrice par défaut de BLAST)

Quelle matrice utiliser ?

- Afin de choisir notre matrice, il faut tenir compte du taux de différence (divergence) entre nos deux séquences à aligner

  - Séquences proches et courtes : BLOSUM élevé ou PAM faible
  - Séquences divergentes et longues : BLOSUM faible ou PAM élevé

BLOSUM-80
PAM-1
faible divergence

BLOSUM-62
PAM-120

BLOSUM-45
PAM-250
forte divergence

Alignement deux à deux : Exercices

Exercices 3 à 7

Comment évaluer la qualité d'un alignement ?

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

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

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

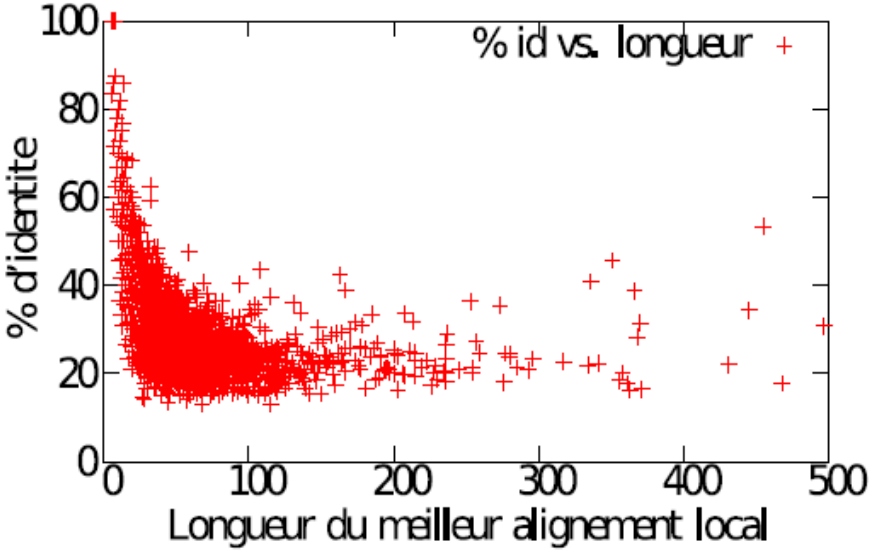
```
- - C T A C C T C G A C T - C A G C
    | |         | |         |
A C C T G A - - C G T A A G C - - -
```

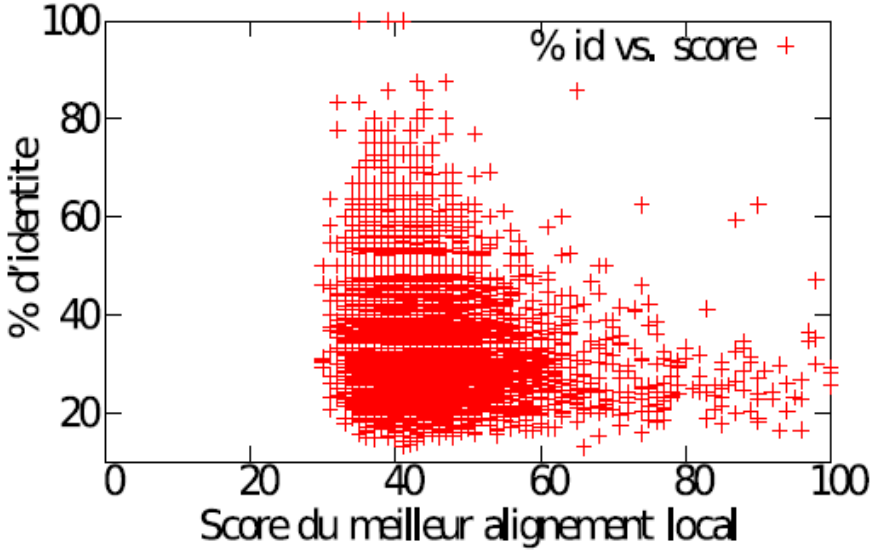
Quelques principes informels

- Robustesse aux paramètres choisis pour le calcul du score
  - on peut douter d'un alignement si de faibles changements (environ 10%) dans l'établissement des pénalités d'insertion-délétion modifient sensiblement cet alignement
- Fréquence des gaps
  - on peut douter d'un alignement s'il nécessite plus d'une insertion en moyenne pour 20 acides aminés
- Deux séquences nucléiques d'au moins 100 bases et identiques à 50% n'ont pas forcément de relation biologique
- Des séquences protéiques de 100 résidus ou plus, possédant au moins 25% d'identité entre elles ont certainement un ancêtre commun (Doolittle, 1990 - PDB).

Le pourcentage d'identité

- Pourcentage de résidus identiques
- Dépend de la composition en bases ou acides aminés
- Dépend de la longueur des séquences





Une fausse bonne idée

⇒ Prendre en compte d'autres métriques

Approche empirique

- Test de la robustesse du score
- S : score de l'alignement entre U et V
- Méthode :
  1. Génération de 100 (200, 1000, . . .) permutations de V (même longueur, même composition)
  2. Alignements avec U
  3. Distribution des scores d'alignement selon une distribution de valeurs extrêmes
- Où se situe S dans cette distribution ?

E-value

- E-value = Nombre de fois attendu de trouver un alignement de score supérieur à S par hasard quand on aligne une séquence de longueur n avec une séquence de longueur m

- **Plus la E-value est proche de 0, plus la similarité est significative**

Exemples

Human alpha haemoglobin (141 aa) vs. Human myoglobin (153 aa)

```
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLS-----HGSAQVKGHGKKVADALTNAVAHVDDMPNALSAL
 :: ..   : ..::::.:.  ..:.:.: :.: . :.: . : .: .:.     ..:...: :: .: .::.. . . ..  .....:
GLSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKASEDLKKHGATVLTALGGILKKKGHHEAEIKPL

SDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR------
.. :: : ..    ....:.:.. .:... :..:.........: :.  .. ..:.:.
AQSHATKHKIPVKYLEFISECIIQVLQSKHPGDFGADAQGAMNKALELFRKDMASNYKELGFQG
```

Chicken lysozyme (129 aa) vs. Bovine ribonuclease (124 aa)

```
               10        20        30        40        50        60
chicke KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINS

bovine                                                 KESAAAKFE


               70        80        90       100       110
chicke RWWCNDGRTPGSR-NLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTD
          ..:..:.:  : ::.
bovine RQHMDSGNSPSSSSNYCNLMMXXRKMTQGKCKPVNTFVHESLADVKAVCSQKKVTCKNGQ
      10        20        30        40        50        60

     120
chicke VQAWIRGCRL

bovine TNCYQSKSTMRITDCRETGSSKYPNCAYKTTQVEKHIIVACGGKPSVPVHFDASV
      70        80        90       100       110       120
```

Exemples : PRSS sur les globines humaines

ExPASy Bioinformatics Resource Portal

[EMBnet-Server] Date: Thu Jan 25 10:35:00 2018

```
SSEARCH performs a Smith-Waterman search
 version 36.3.5e Nov, 2012(preload8)
Please cite:
 T. F. Smith and M. S. Waterman, (1981) J. Mol. Biol. 147:195-197;
 W.R. Pearson (1991) Genomics 11:635-650

  1>>>alpha haemoglobin 141 bp - 141 aa
      153 residues in     1 sequences

Statistics: (shuffled [500]) MLE statistics: Lambda= 0.2958;  K=0.08446
 statistics sampled from 1 (1) to 500 sequences
Algorithm: Smith-Waterman (SSE2, Michael Farrar 2006) (7.2 Nov 2010)
Parameters: blosum62.mat matrix (11:-4), open/ext: -12/-2 Scan time:  0.000

The best scores are:                                      s-w bits E(1)
myoglobin 153 bp                                  ( 153)  114 52.2 4.1e-12

>>myoglobin 153 bp                                        (153 aa)
 s-w opt: 114  Z-score: 249.9  bits: 52.2 E(1): 4.1e-12
Smith-Waterman score: 114; 26.0% identity (52.7% similar) in 146 aa overlap (2-141:2-147)

               10        20        30        40              50
alpha  VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF------DLSHGSAQ
        :: ..   :  .:::: :    .: :.: :.: . : :   :  :      :   .: .
myoglo GLSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKASED
               10        20        30        40        50        60

           60        70        80        90       100       110
alpha  VKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLP
       .: ::  :  :: . .          .. :.. :: : ..    ....:.:.. .: .  :
myoglo LKKHGATVLTALGGILKKKGHHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVLQSKHP
               70        80        90       100       110       120

          120       130       140
alpha  AEFTPAVHASLDKFLASVSTVLTSKYR
       ..:   .....: :      ..:.:.
myoglo GDFGADAQGAMNKALELFRKDMASNYKELGFQG
               130       140       150



141 residues in 1 query   sequences
153 residues in 1 library sequences
 Tcomplib [36.3.5e Nov, 2012(preload8)] (1 proc in memory [0G])
 start: Thu Jan 25 10:35:01 2018 done: Thu Jan 25 10:35:01 2018
 Total Scan time:  0.000 Total Display time:  0.000

Function used was SSEARCH [36.3.5e Nov, 2012(preload8)]
```

Exemples : PRSS poulet/bovin

ExPASy Bioinformatics Resource Portal

[EMBnet-Server] Date: Thu Jan 25 10:39:40 2018

```
SSEARCH performs a Smith-Waterman search
 version 36.3.5e Nov, 2012(preload8)
Please cite:
 T. F. Smith and M. S. Waterman, (1981) J. Mol. Biol. 147:195-197;
 W.R. Pearson (1991) Genomics 11:635-650

  1>>>chicken lysosyme 129 bp - 129 aa
      124 residues in     1 sequences

Statistics: (shuffled [500]) MLE statistics: Lambda= 0.3086;  K=0.1117
 statistics sampled from 1 (1) to 500 sequences
Algorithm: Smith-Waterman (SSE2, Michael Farrar 2006) (7.2 Nov 2010)
Parameters: blosum62.mat matrix (11:-4), open/ext: -12/-2 Scan time:  0.000

The best scores are:                                      s-w bits E(1)
bovine ribonuclease 124 bp                          ( 124)   26 14.7    0.44

>>bovine ribonuclease 124 bp                                (124 aa)
 s-w opt:  26  Z-score: 49.7  bits: 14.7 E(1): 0.44
Smith-Waterman score: 26; 40.0% identity (80.0% similar) in 15 aa overlap (65-78:14-28)

               10        20        30        40        50        60
chicke KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINS

bovine                                                 KESAAAKFE


               70        80        90       100       110
chicke RWWCNDGRTPGSR-NLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTD
          ..:.:.:  : ::.
bovine RQHMDSGNSPSSSSNYCNLMMXXRKMTQGKCKPVNTFVHESLADVKAVCSQKKVTCKNGQ
      10        20        30        40        50        60

     120
chicke VQAWIRGCRL

bovine TNCYQSKSTMRITDCRETGSSKYPNCAYKTTQVEKHIIVACGGKPSVPVHFDASV
      70        80        90       100       110       120
```

Significativité des scores : Exercices

Exercice 8

Recherche d'une séquence dans les banques de données

Recherche dans les banques de données

- On se donne :
  - une séquence requête q
  - une banque de séquences T = {t1, ..., tn}

- On veut :
  - trouver des alignements **significatifs** entre q et les t_i

- Les algorithmes classiques ne fonctionnent pas : trop long, il est nécessaire de trouver des parades

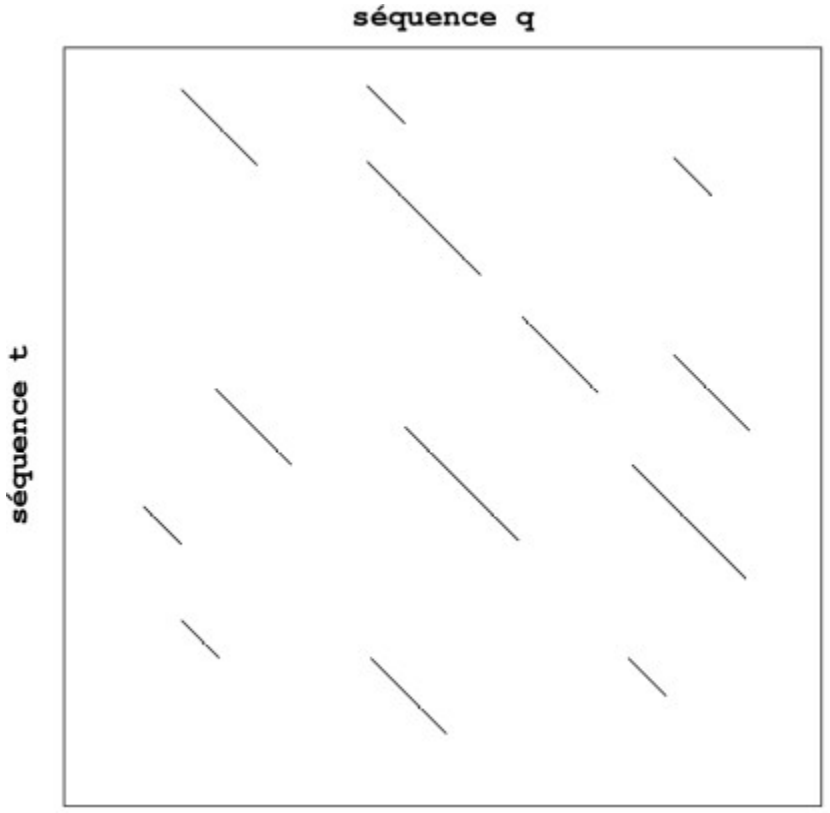
  → méthodes de recherche heuristique

FASTA (Pearson et Lipman, 1988)

- Programme d'alignement de séquences d'ADN et de protéine
- Alignement global avec gaps
- Traite les séquences de la banque les unes après les autres

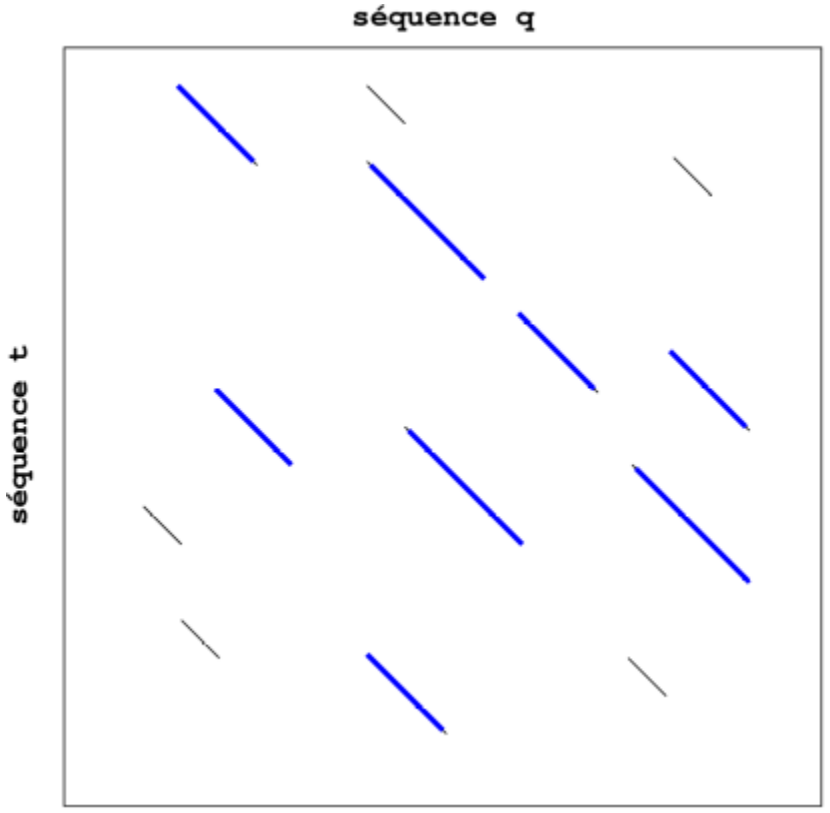
FASTA (Pearson et Lipman, 1988)

1- Trouve tous les mots exacts de longueur ≥ l communs à q et t_i



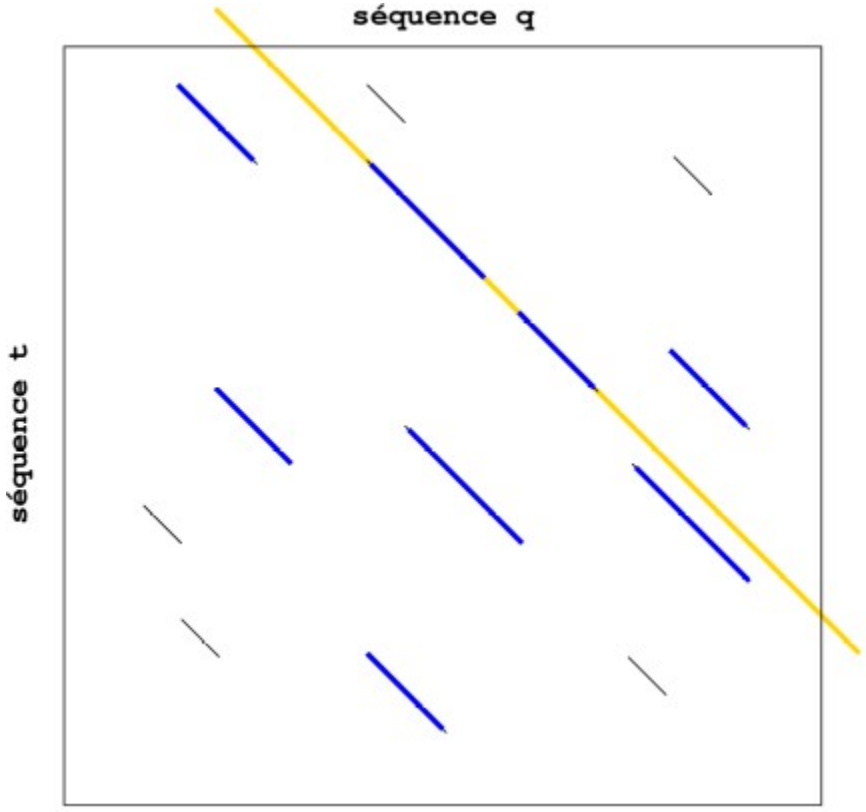
FASTA (Pearson et Lipman, 1988)

2- Sélectionne ceux de score suffisamment élevés



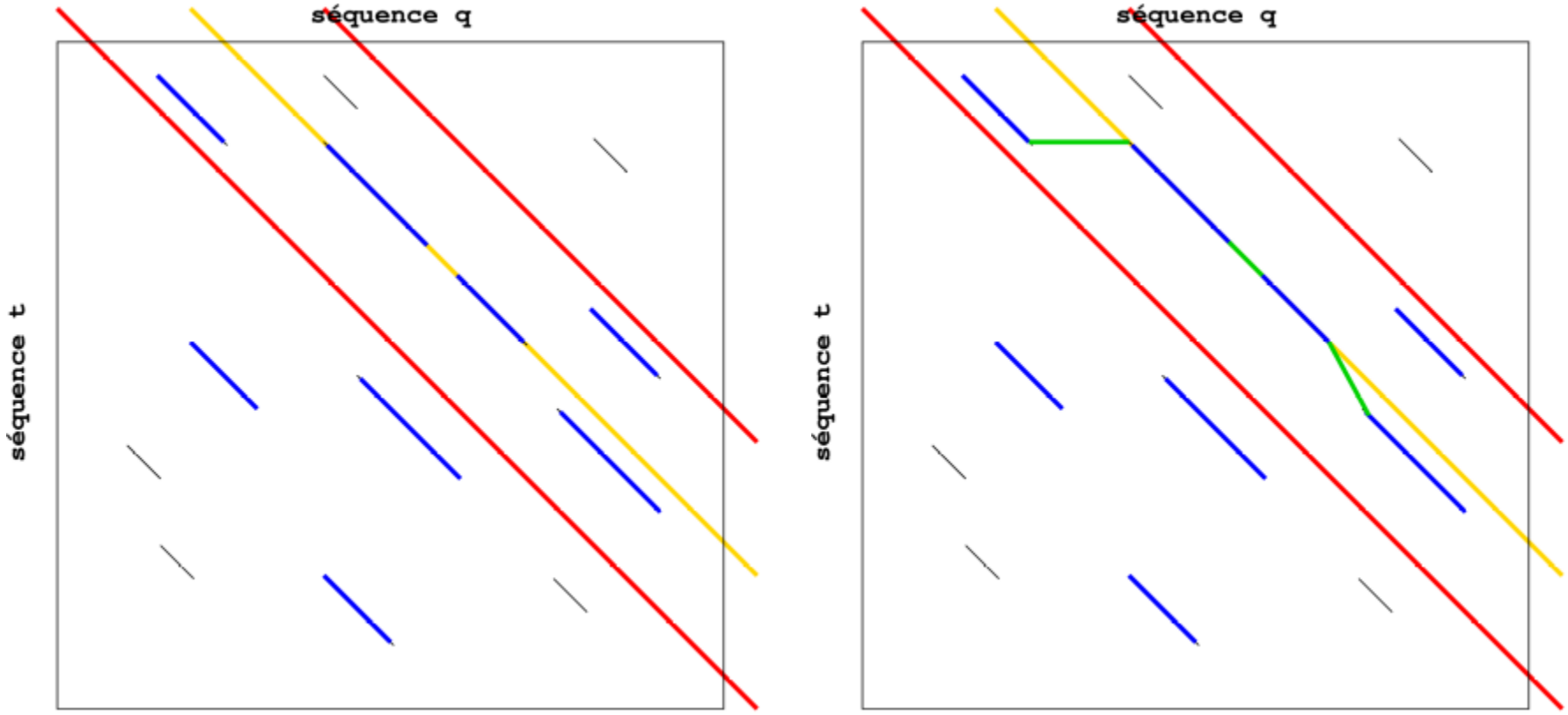
FASTA (Pearson et Lipman, 1988)

3- Sélectionne une diagonale d (du dotplot) contenant le maximum de mots exacts de longueur ≥ l



FASTA (Pearson et Lipman, 1988)

4- Procède à un alignement global ”classique” dans une bande de largeur 2k autour de la diagonale d



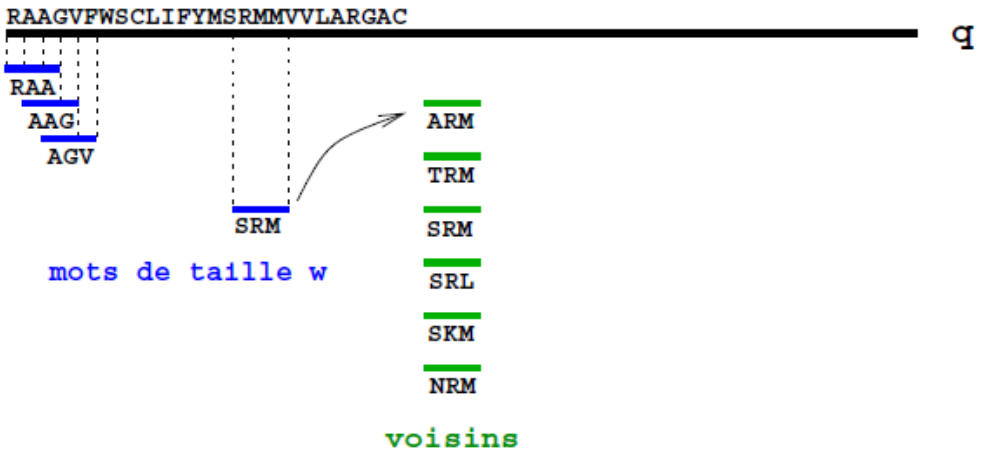
⇒ deux paramètres : k et l , l généralement de longueur 6 pour l’ADN et 2 pour les protéines

BLAST (Basic Local Alignment Search Tool)

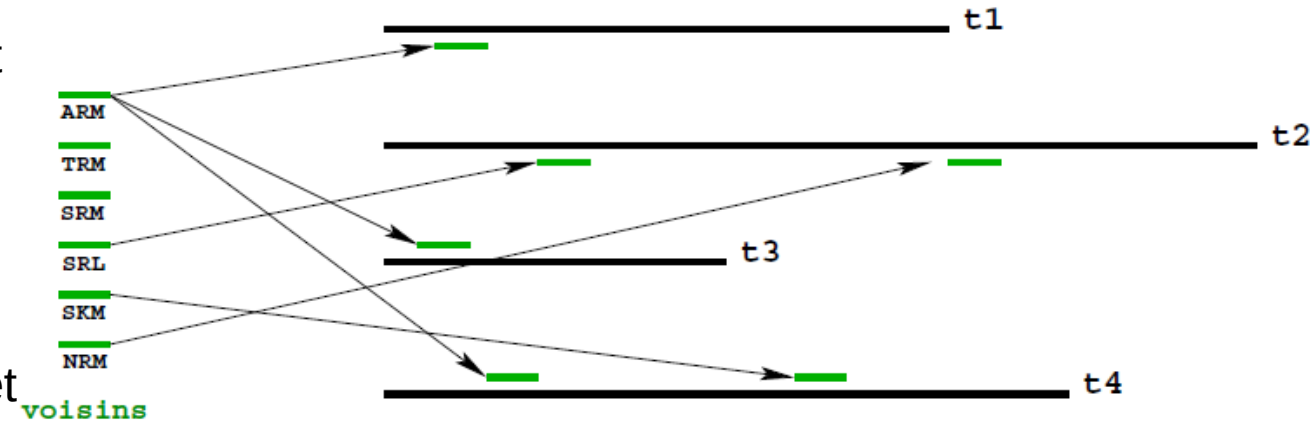
- Né en 1990
- Permet de retrouver rapidement dans des bases de données les séquences répertoriés ayant des zones de similitudes avec la séquence recherchée
- Évolution:
  - BLAST1 : trouve des matchs significatifs sans gaps
  - BLAST2 : inclusion des gaps
  - Évolution vers des versions avec raffinement des résultats

BLAST 1

1) Recherche des mots de taille w (ADN=11, protéine=3) et de score ≥ au seuil T pour chaque position de q



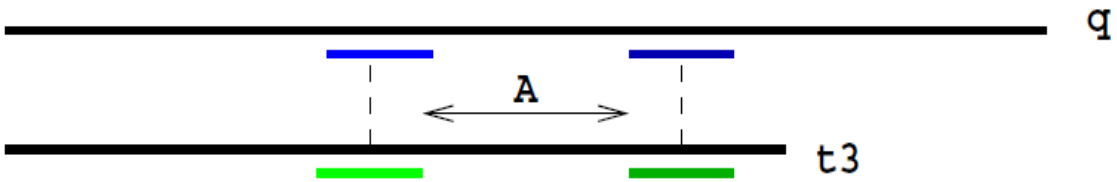
2) Chaque couple de mot entre q et t_i forme un hit



3) Extensions à droite et à gauche du hit
L'extension est stoppée lorsque le score du hit décroît de X (X-drop)

BLAST 2

- Incorpore des gaps
- Mise en œuvre : se base sur 2 hits distants au maximum de A



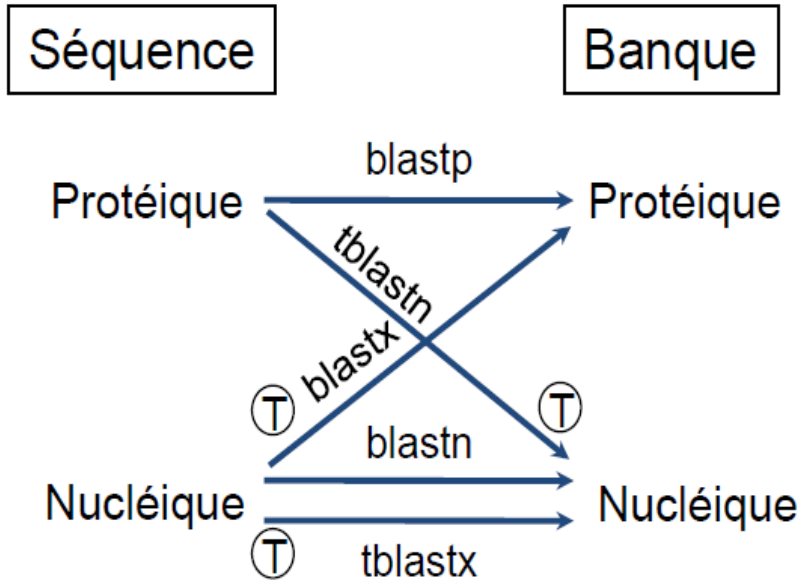
- Extension des hits comme dans BLAST 1 (avec limitation de score) mais en autorisant les gaps
- Plus rapide que la précédente version

MEGABLAST pour l'ADN

- Idée : un BLAST plus rapide lorsqu'on recherche une grande similarité
- Mise en œuvre : utiliser des mots de taille plus grande (28 contre 11)
- A réserver à des requêtes du type : trouver la séquence dans la banque
- Évolution : Discontiguous MegaBLAST
  - Principe : utiliser un pattern plutôt qu'un mot exact pour les ancres
  - Exemple : un pattern de longueur 21 : 100101100101100101101
    → Les 1 représentent les matches et les 0 les mismatches
  - Peut se révéler meilleur que BLAST

Les différentes fonctionnalités de BLAST

- blastp : protéine *vs.* protéine.
- blastn : utile pour le non-codant.
- blastx : séquences co-dantes non identifiées.
- tblastn : homologues dans un génome non complètement annoté.



©Guy Perrière

Détection de répétitions

Les répétitions

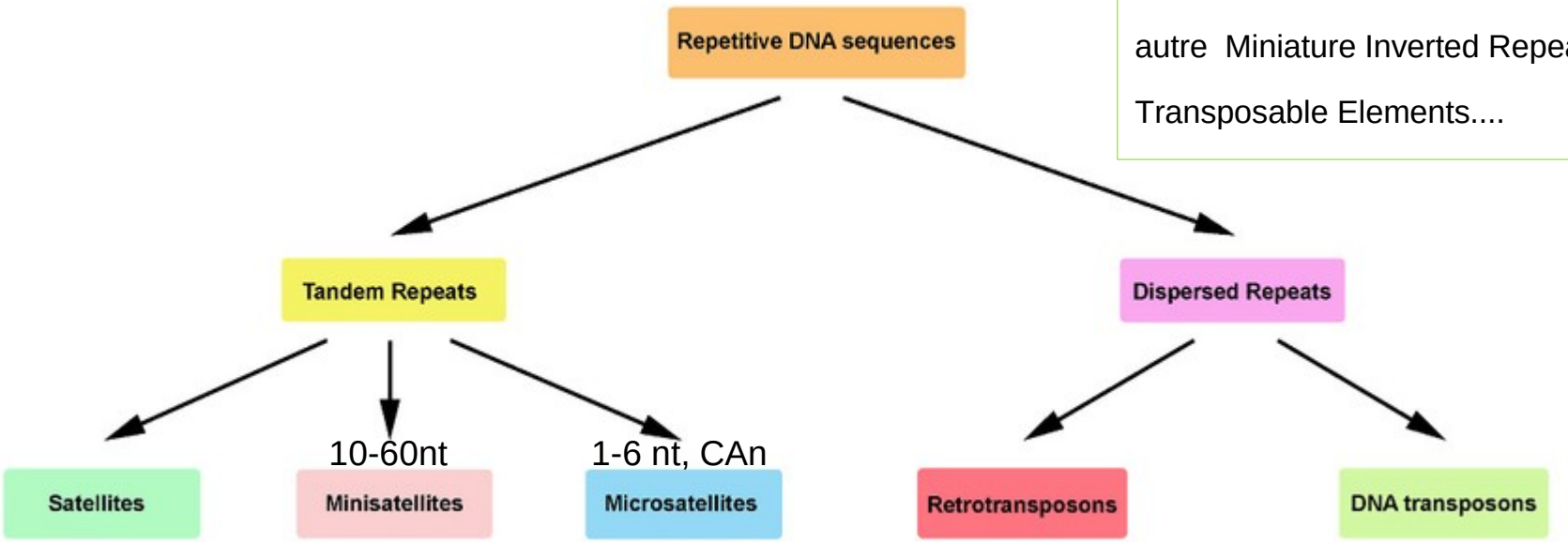
- Séquences qui sont retrouvées plus d'une fois dans un génome
- Définition très large :
  - Séquences répétées simples (exactes ou approximatives)
  - Séquences très longues capables de se répliquer (retro-virus)
- La proportion des répétitions varie fortement d'un génome à l'autre :
  - 3 % chez la levure *Saccharomyces cerevisiae*
  - > 80 % chez le maïs
  - Environ 45 % dans le génome humain

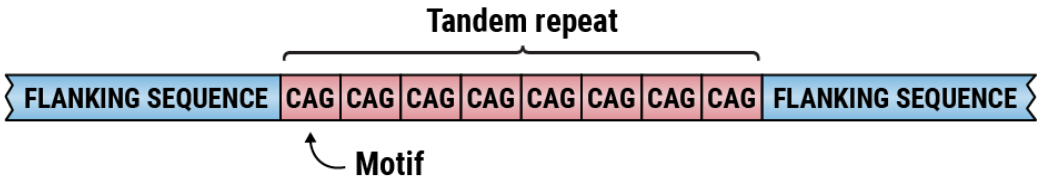
Les répétitions

- Permet d'évaluer la complexité d'un génome
- Responsables du réarrangement du génome, impliquées dans la régulation de gènes...
- Les éléments répétés sont très utiles pour l'inférence phylogénétique, surtout pour des espèces très proches
  - Par exemple, les SINEs (Short Interspersed Nuclear Elements) sont très utiles pour la reconstruction phylogénétique des mammifères [Bannikova, 2004]
- Les répétitions ont toujours besoin d'être masquées avant toute analyse bio-informatique (en dehors de l'étude des répétitions elle-mêmes)
  - Information nécessaire pour les étapes d'assemblage
  - Biais dans l'annotation, la recherche dans les banques....

Les répétitions

Couper-Coller => ADN qui bouge directement d'une position à une autre Miniature Inverted Repeat Transposable Elements....





Copier-Coller => transcription de l'ADN en ARN, reverse transcriptase pour avoir une copie d'ADN insérée à une nouvelle position) Long Terminal Repeats, Long Interspersed Nuclear elements, Short Interspersed nuclear elements...

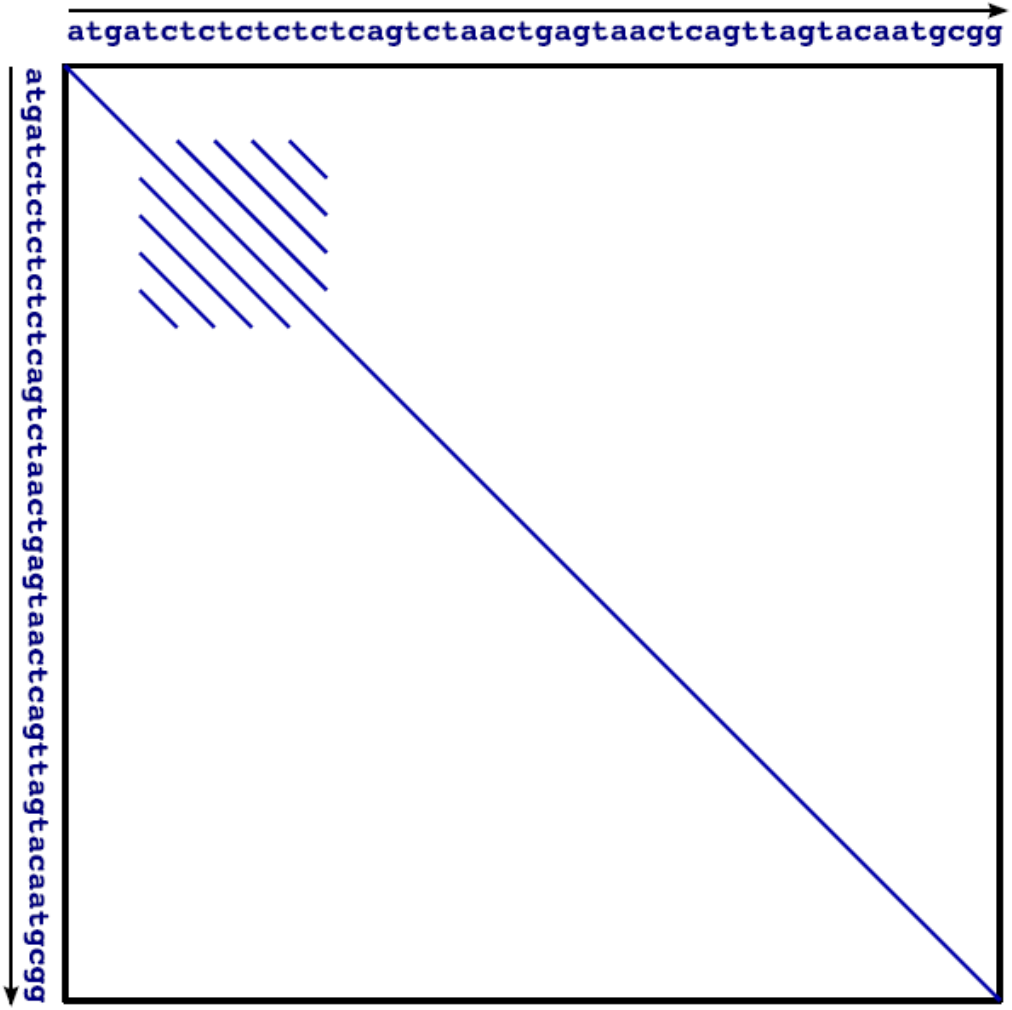
Les répétitions

- La détection de répétition est un problème complexe en bio-informatique :
  - Difficile de déterminer les limites des répétitions
  - Séquence non identiques (mutations, indels, réarrangements...)

    => copies divergentes et fragmentées
  - Nombre de copies variables en fonction de la classe : de 2 à plusieurs millions de copies dans le génome (SINEs dans le génome humain) besoin de temps et de mémoire considérable en fonction des organismes
  - Une fois identifiés, les éléments répétés peuvent être difficiles à classer

Les répétitions

- Les dotplots permettent de visualiser rapidement des régions répétées



Identification des répétitions

Heredity (2010) 104, 520–533
© 2010 Macmillan Publishers Limited All rights reserved 0018-067X/10 $32.00
www.nature.com/hdy

REVIEW

Identifying repeats and transposable elements in sequenced genomes: how to find your way through the dense forest of programs

E Lerat
Université de Lyon, F-6900 Lyon; Université Lyon 1, Lyon, France; CNRS, UMR5558, Laboratoire de Biométrie et Biologie Evolutive, F-69622 Villeurbanne, France

Identification des répétitions

- 3 grands types de méthodes dépendantes de la connaissance des répétitions qui est prise en compte :
  - Recherche d'un élément spécifique
  - Recherche d'éléments avec des caractéristiques structurales particulières
  - Recherche sans *a priori* de répétions (juste basée sur la nature répétitive)

Approches basées sur les librairies

- Les répétitions sont recherchées par similarité en comparant l'entrée à un ensemble de séquences de référence (BLAST-like) contenues dans une librairies
  - Librairie donnée par l'utilisateur
  - Librairie générale (exemple REPBASE curated consensus sequences of repeats from various eukaryotic organims)

RepeatMasker	http://www.repeatmasker.org	Search by homology	Smit *et al.* (1996–2004)	Online
Censor	http://www.girinst.org/censor/download.php	Website; Search by homology	Jurka *et al.* (1996)	Online
PLOTREP	http://repeats.abc.hu/cgi-bin/plotrep.pl	Website; Visualization tool; Uses Censor	Tóth *et al.* (2006)	Online
MaskerAid	No web site	Search by homology	Bedell *et al.* (2000)	?
Greedier	No web site	Search by homology	Li *et al.* (2008)	?

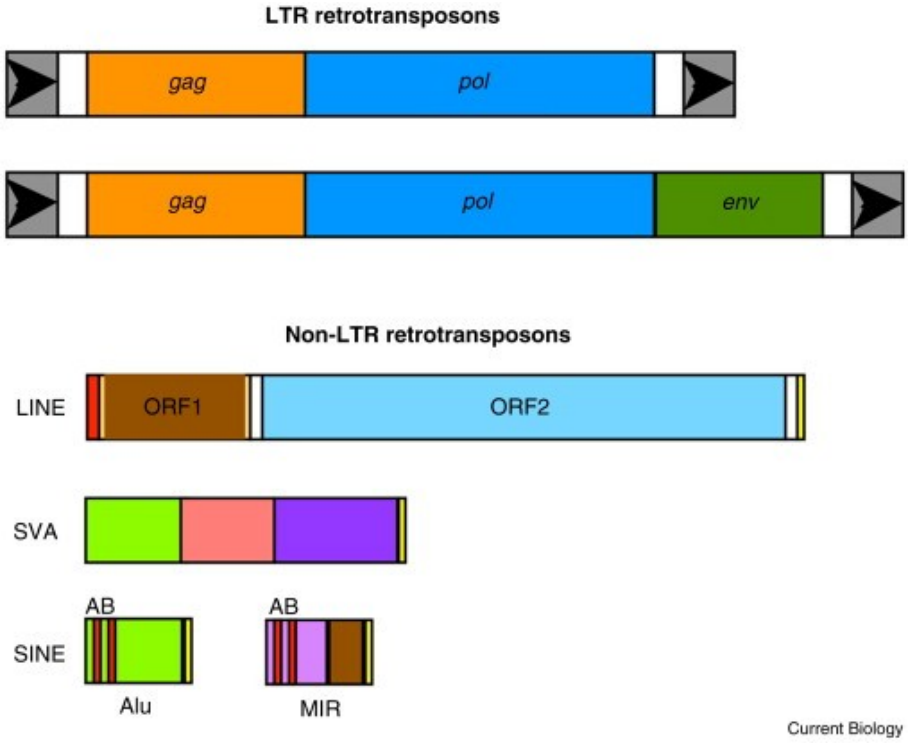
- Programme le plus utilisé RepeatMasker

RepeatMasker

- Recherche par homologie les répétitions dans une banque
- En sortie :
  - Annotation complète des répétitions identifiées
  - La séquence d'entrée masquée (les répétitions sont remplacées par des N)
- Outil facile à utiliser, disponible en ligne
- Peut être complémentaire à une méthode *de novo*
  - Recherche des séquences répétées identifiées par une autre méthode
- Avantage: capable de détecter des répétitions en nombre faible de copies

Approches basées sur les signatures

- Recherche de structure et motifs spécifiques d'un élément répété donné
  - Identification de nouveaux éléments mais pas de nouvelles classes de répétition
  - Très dépendant des connaissances et de l'existence d'une structure caractéristique conservée



Approches basées sur les signatures

- Chaque programme est spécialisé dans l'identification d'un élément spécifique ou d'une classe de répétitions

LTR_STRUC	http://www.mcdonaldlab.biology.gatech.edu/finalLTR.htm	Search for LTR retrotransposons structural features	McCarthy and McDonald (2003)	Yes
RetroTector	http://www.kvir.uu.se/RetroTector/RetroTectorProject.html	Search for LTR retrotransposons structural features	Sperber *et al.* (2007)	Online
LTR_FINDER	http://tlife.fudan.edu.cn/ltr_finder/	Website; Search for LTR retrotransposons structural features	Xu and Wang (2007)	Online
LTRharvest	http://www.zbh.uni-hamburg.de/LTRharvest/index.php	Search for LTR retrotransposons structural features	Ellinghaus *et al.* (2008)	Yes
LTR_par	http://www.eecs.wsu.edu/~ananth/software.htm	Search for LTR retrotransposons structural features	Kalyanaraman and Aluru (2006)	Yes
find_ltr	http://darwin.informatics.indiana.edu/cgi-bin/evolution/ltr.pl	Search for LTR retrotransposons structural features	Rho *et al.* (2007)	Yes
RTAnalyzer	http://www.riboclub.org/cgi-bin/RTAnalyzer/index.pl	Website ; Detects LINE, Alu or retroposed genes Uses Blast as initial step	Lucier *et al.* (2007)	Online
TSDfinder	http://www.ncbi.nlm.nih.gov/CBBresearch/Landsman/TSDfinder/	Detects LINEs Uses RepeatMasker output as initial step and Blast2seq for TSD identification	Szak *et al.* (2002)	No
SINEDR	No web site	Detects known SINEs.	Tu *et al.* (2004)	No
FINDMITE	http://jaketu.biochem.vt.edu/dl_software.htm	Search for MITE features	Tu (2001)	Online
MAK (MITE Analysis Kit)	Indicated URL not valid	3 independent programs that use Blast for the initial step; Identification given a known element	Yang and Hall (2003)	?
MUST—MITE Uncovering SysTem	http://csbl1.bmb.uga.edu/ffzhou/MUST/	Website; Search for TIR structure in genomes.	Chen *et al.* (2009)	No
TRANSPO	http://alggen.lsi.upc.es/recerca/search/transpo/transpo.html	Website; Search for a given TIR	Santiago *et al.* (2002)	Online
HelitronFinder	http://limei.montclair.edu/HF.html	Website; Dedicated to the prediction of HelA in maize	Du *et al.* (2008)	Online

Les approches *de novo*

- Basées sur la nature répétitive des éléments
- But : découvrir de nouveaux éléments répétés
  - Production de la liste exhaustive des répétitions d'un génome
  - Définir des familles de répétitions, extraction d'une séquence consensus pour une famille (qui peut être ensuite utilisées par exemple par RepeatMasker) pour rechercher les positions exactes de ces nouveaux éléments
- 2 approches principales :
  - Comparaison de la séquence sur elle-même
  - Recherche des occurrences répétées de petits mots (k-mers)

Les approches *de novo* : self-comparison

- BLAST pour faire la comparaison de la séquence sur elle-même
- Clustering des répétitions pour former les familles

Initial identification of repetitive sequences				
Self-comparison approaches				
Repeat Pattern Toolkit	No web site	Scoring system based on sequence similarity. Identifies family by graph representation.	Agarwal and States (1994)	?
RECON	http://selab.janelia.org/recon.html	Based on multiple alignments. Identifies family by graphical representation.	Bao and Eddy (2002)	Yes
PILER	http://www.drive5.com/piler/	Uses PALS to generate alignments; Identifies repeat families by clustering that distinguishes between the different types of repeats.	Edgar and Myers (2005)	Yes
BLASTER suite	http://urgi.versailles.inra.fr/development/blaster/	Contains three programs (Blaster, Matcher and Grouper). Uses Blast to obtain self-alignment. Identifies repeat families by clustering	Quesneville (unpublished)	Yes

Les approches *de novo* : k-mers et graines

- Une répétition = sous-chaîne de caractère de taille k qui existe plus d'une fois dans la séquence
  - Les matches doivent être identiques
- Graines espacées = extension des k-mers qui autorise des variations dans la séquences (substitutions, longueur...)

Les approches de novo : k-mers et graines

K-mer and spaced seed approaches

ReAS	ftp://ftp.genomics.org.cn/pub/ReAS/software/	Produces a library of consensus. Identifies families by string extension.	Li *et al.* (2005)	Yes
RepeatScout	http://repeatscout.bioprojects.org/	Identifies families by string extension. Applicable to all kind of repeats.	Price *et al.* (2005)	Yes
RAP	http://genomics.cribi.unipd.it/index.php/Rap_Repeat_Filter	Identifies exact and inexact words.	Campagna *et al.* (2005)	? (Program not provided by the authors)
REPuter	http://www.genomes.de/	Used by RepeatFinder and RepeatGluer; Identifies families by string extension.	Kurtz and Schleiermacher (1999)	Yes
Repeat-match	http://mummer.sourceforge.net/	Part of the MUMmer package. Not directly created to identify repeat families	Delcher *et al.* (1999)	Online
RepSeek	http://wwwabi.snv.jussieu.fr/public/RepSeek/	Any kind of repeats. No construction of repeat families.	Achaz *et al.* (2007)	Yes
Tallymer	http://www.zbh.uni-hamburg.de/Tallymer/	Not directly created to identify repeat families. Helps genome annotation.	Kurtz *et al.* (2008)	Yes
Vmatch	http://www.vmatch.de/	Generalist software (Subsumes REPuter)	Kurtz (unpublished)	Yes
mer-engine	http://roma.cshl.org/mer-home.php	Not directly created to identify repeat families. Helps genome annotation.	Healy *et al.* (2003)	Online
FORRepeats	http://al.jalix.org/FORRepeats/	Based on the factor oracle data structure.	Lefebvre *et al.* (2003)	No
P-Clouds	http://www.evolutionarygenomics.com/PClouds.html	Uses oligo frequencies and create clusters of similar repeated oligos.	Gu *et al.* (2008)	Yes

Autres approches

- Chaînes de Markov cachées
- transformée de Fourrier
- ...

Periodicity approaches				
Spectral repeat finder	http://www.imtech.res.in/raghava/srf/	Website. Any kind of repeats, but size limitation.	Sharma *et al.* (2004)	Online
Only to define repeat families				
Clustering				
RepeatFinder	http://cbcb.umd.edu/software/RepeatFinder/	Uses REPuter or Repeatmatch to define exact repeats.	Volfovsky *et al.* (2001)	Yes
Graph representation with heuristics				
REPEATGLUER	http://nbcr.sdsc.edu/euler/intro_tmp.htm	Is intended to determine the boundaries of repeats (but not to characterize the family). Uses the de Bruijn graph representation.	Pevzner *et al.* (2004)	No

Les pipelines

- Incluent plusieurs programmes de détections de répétitions en combinant leurs avantages/inconvénients
- Généralement, leur but est de répondre à une question spécifique
  - EX : REPEAT-MODELER inclus RECON, RepeatScout, RepeatMasker et TRF (RECOn et RepeatScout pour construire affiner et classifier les consensus des répétitions des interspersed repeats)

DAWG-PAWS	http://dawgpaws.sourceforge.net/	Pipeline to annotate genes and TEs (LTR_struc, LTR_Finder, LTR_par, Find_LTR, FINDMITE, TRF, Repseek, RepeatMasker, TENest)	Estill and Bennetzen (2009)	Yes
RepeatModeler	http://www.repeatmasker.org/RepeatModeler.html	Pipeline to annotate repeats (RepeatMasker, RECON and Repeatscout, TRF)	Smit (unpublished)	Yes
RepeatRunner	http://www.yandell-lab.org/software/repeatrunner.html	Pipeline (PILER, RepeatMasker and Blastx)	Smith *et al.* (2007)	Yes (installation)
REannotate	http://www.bioinformatics.org/reannotate/index.html	Uses the output of RepeatMasker	Pereira (2008)	Yes
ReRep	http://bioinfo.pdtis.fiocruz.br/ReRep/	Pipeline to identify repeats before the genome assembly step (Blast or MUMmer)	Otto *et al.* (2008)	Yes (installation)
RetroPred	http://www.juit.ac.in/assets/RetroPred/home.html	Pipeline to identify non-LTR retrotransposons (PALS, PILER, and	Naik *et al.* (2008)	Yes (installation)

Les répétitions : Exercices

Exercice 9, 10, 11 partie 1

Sources

- D'après les cours de :
  - l’équipe Bonsai, CRIStAL UMR 9189
  - Géraldine PASCAL, Genotoul bioinfo
  - Sophie Pasek, Jussieu
- Identifying repeats and transposable elements in sequenced genomes: how to find your way through the dense forest of programs, Lerat E. Heredity (Edinb). 2010
- http://www.info.univ-angers.fr/pub/richer/recbioal3.php