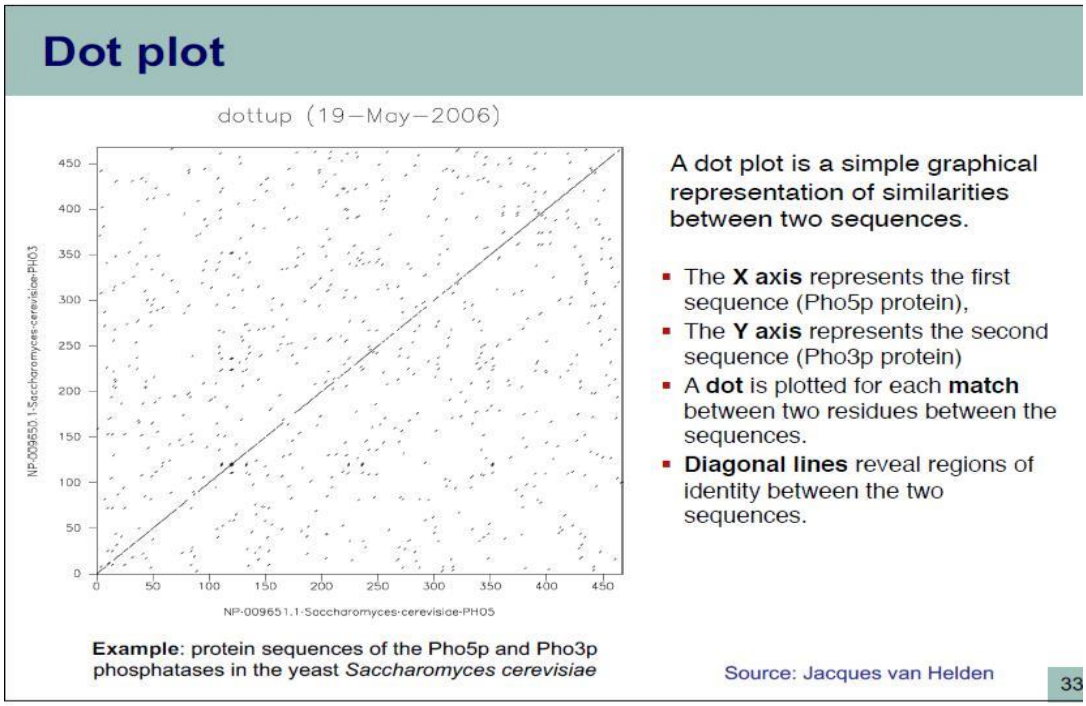
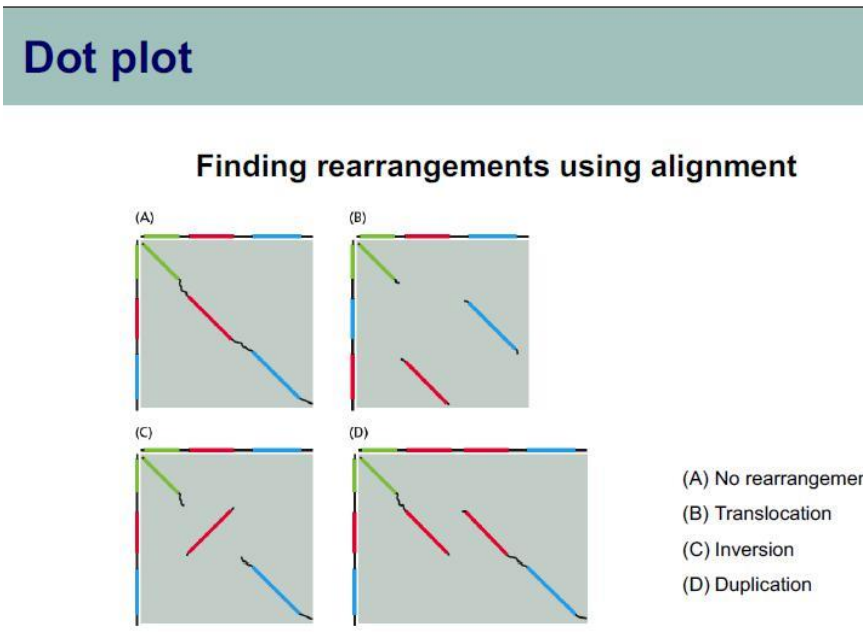
Le dot Plot

Dans un dot plot, a dot est plotté pour chaque match entre deux résidus ou deux séquences.



On peut jouer avec le paramètre word-size (= k-tuples similaires). Si trop petit, on va avoir des points partout (graphe tout noir), si il est grand (10 par ex), on ne va apercevoir que la diagonale de similarité. Une ligne diagonale révèle un alignement parfait ou partiel entre les deux séquences. Les répétitions de séquence sont facilement détectées dans un dot plot quand la séquence est comparée à elle-même. Elles apparaissent comme un segment de lignes parallèle à la diagonale.



Softwares : Dotlet, Dotter, Dot-matcher,..

Matrices de substitution

Matrices basées par le fait que les substitutions puissent valoir un score négatif. Exemple : A-T = -1 (si ça arrive souvent), les autres = -2. Les matchs eux ont des scores positifs.

	A	C	G	T
A	2			
C	-2	2		
G	-2	-2	2	
T	-1	-2	-2	2

	A	A	T	C	T	T	C	A	G	C	G	T	A	T	T	G	C	T
A	2	2	-1	-2	-1	-1	-2	2	-2	-2	-2	-1	2	-1	-1	-2	-2	-1
T	-1	-1	2	-2	2	2	-2	-1	-2	-2	-2	2	-1	2	2	-2	-2	2
C	-2	-2	-2	2	-2	-2	2	-2	-2	2	-2	-2	-2	-2	-2	-2	2	-2
T	-1	-1	2	-2	2	2	-2	-1	-2	-2	-2	2	-1	2	2	-2	-2	2
T	-1	-1	2	-2	2	2	-2	-1	-2	-2	-2	2	-1	2	2	-2	-2	2
A	2	2	-1	-2	-1	-1	-2	2	-2	-2	-2	-1	2	-1	-1	-2	-2	-1
G	-2	-2	-2	-2	-2	-2	-2	-2	2	-2	2	-2	-2	-2	-2	2	-2	-2
C	-2	-2	-2	2	-2	-2	2	-2	-2	2	-2	-2	-2	-2	-2	-2	2	-2
C	-2	-2	-2	2	-2	-2	2	-2	-2	2	-2	-2	-2	-2	-2	-2	2	-2
G	-2	-2	-2	-2	-2	-2	-2	-2	2	-2	2	-2	-2	-2	-2	2	-2	-2
G	-2	-2	-2	-2	-2	-2	-2	-2	2	-2	2	-2	-2	-2	-2	2	-2	-2
A	2	2	-1	-2	-1	-1	-2	2	-2	-2	-2	-1	2	-1	-1	-2	-2	-1
G	-2	-2	-2	-2	-2	-2	-2	-2	2	-2	2	-2	-2	-2	-2	2	-2	-2
G	-2	-2	-2	-2	-2	-2	-2	-2	2	-2	2	-2	-2	-2	-2	2	-2	-2
T	-1	-1	2	-2	2	2	-2	-1	-2	-2	-2	2	-1	2	2	-2	-2	2
A	2	2	-1	-2	-1	-1	-2	2	-2	-2	-2	-1	2	-1	-1	-2	-2	-1
T	-1	-1	2	-2	2	2	-2	-1	-2	-2	-2	2	-1	2	2	-2	-2	2
T	-1	-1	2	-2	2	2	-2	-1	-2	-2	-2	2	-1	2	2	-2	-2	2

Voici un exemple pour une séquence plus longue :

D'autres scores sont bien-sûr utilisés, considérant l'importance d'une transition ou transversion (5/0/-1), ou le tout ou rien (+5/-4)

Les matrices PAM (pourcentage de mutations acceptées)

Se basent sur le taux de divergence entre les séquences (evolutionary distances), selon un taux spécifique : 1, 50, 250,… Ce chiffre signifie le pourcentage de mutation par résidus.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Q	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
G	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
H	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
I	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
L	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
K	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
M	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
F	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
P	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
S	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
T	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
W	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Y	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
V	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

For clarity, the values have been multiplied by 10000

La diagonale représente la probabilité de toujours observer le même résidu (ici, après 1PAM, ce qui représente une période de temps évolutive donnant 1 mutation pour 100 acides aminés). En quelque sorte, ce sont les 99% du cas de non-mutation. C'est l'inverse pour le reste. Des pourcentages plus faibles sont corrélés avec les propriétés physicochimiques des acides aminés (une Glycine ne sera jamais substituée en proline, trop de bouleversements structurels)
Autre exemple avec PAM250 (chiffres multipliés par 100) :

PAM250 derived by Dayhoff for the 20 amino acids

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
V	7	4	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	72	4	17

Par exemple, A vers A après PAM250 est de 13%

De cette matrice de probabilité, il faut passer à une matrice de scores, qui reflèterait à la significativé d'un alignement survenant comme un résultat d'un processus évolutif par rapport à ce que nous pourrions attendre par hasard. On utilise souvent le log-odd score, où les valeurs positives signifient une mutation acceptée (Glycine/Alanine par exemple).

In practice, we often use the log-odd scores defined by

$$s_n(i,j) = \log \frac{M_{ji}^n}{f_j} = \log \frac{P_{ji,n}}{f_i f_j}$$

This definition has convenient practical consequences:

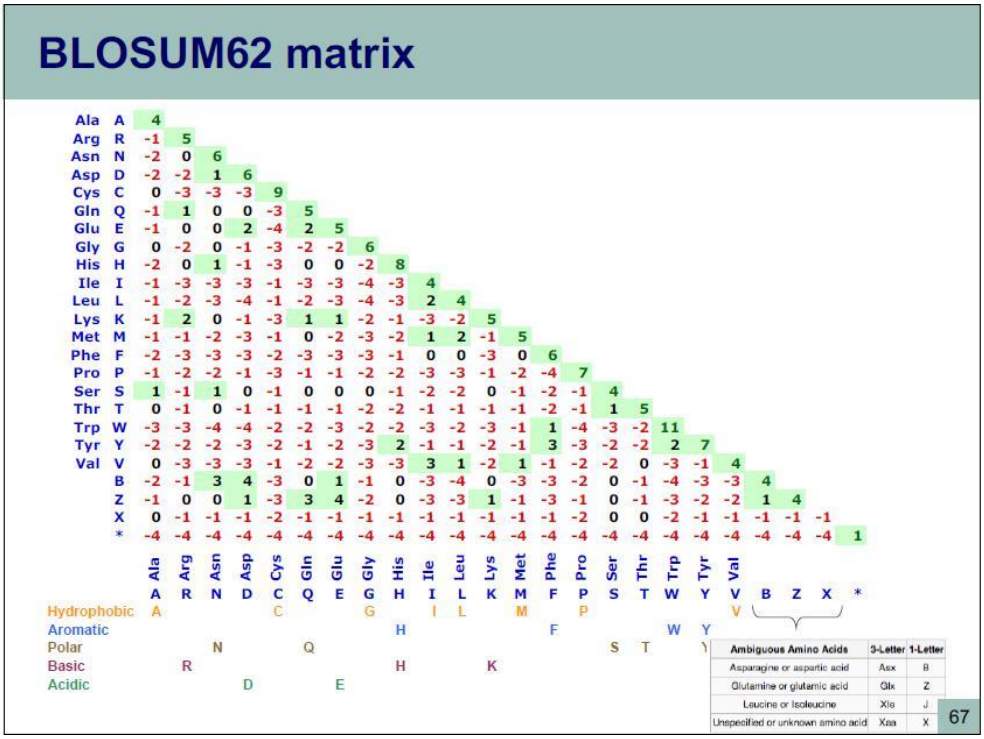
A **positive score** ($s_n > 0$) characterizes the accepted mutations
A **negative score** ($s_n < 0$) characterizes the unfavourable mutations

Au plus le pourcentage d'identité est faible, au plus il faut monter dans les PAM, car les séquences sont plus distantes.

Les matrices BLOSUM

Se basent sur la conservation de domaines protéiques.

Ils ont scanné la base de données BLOCKS pour des régions très conservées des familles de protéines (qui n'ont pas de gaps dans l'alignement des séquences), puis ont compté les fréquences relatives des acides aminés et leurs probabilités de substitution. Ensuite, ils ont calculé un score log-odds pour chacune des 210 paires de substitution possibles des 20 acides aminés standards.



Des index sont disponibles : 62, 80, …
Ils dépendent du taux de similarité entre les deux séquences. Plus l'index est haut, plus elles sont conservées (contraire de PAM où elles divergent suivant l'index montant)

Bref rappel :

- Avec les matrices PAM : le score indique le pourcentage de substitution par position. Un index élevé est plus approprié pour des protéines distantes.
- Avec les matrices BLOSUM : le score indique le pourcentage de conservation. Un index élevé est plus approprié pour des protéines conservées.

Needleman-Wunsch

Cet algorithme fait partie de la **programmation dynamique**, qui consiste à trouver le meilleur alignement, sans avoir besoin à passer par tous les alignements possibles. Gain de temps énorme : au lieu que le temps augmente de manière exponentielle suivant la longueur de la séquence, celui-ci n'augmente que quadratiquement.

Il garantit de retourner le plus grand score entre deux séquences.

Comment est construit cet algorithme ?

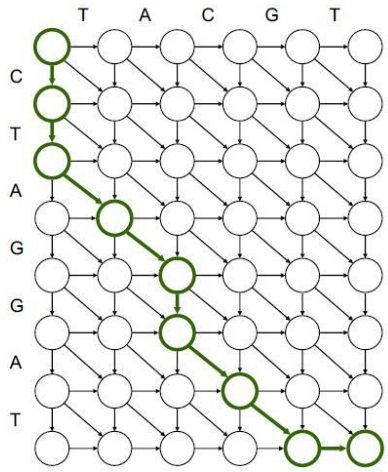
Pour chaque position dans l'alignement, on peut retrouver soit un gap en séquence 1, soit un gap en séquence 2, soit un match entre elles.

Il suffit là de construire la réponse progressivement, en rejetant directement les alignements avec un mauvais score.

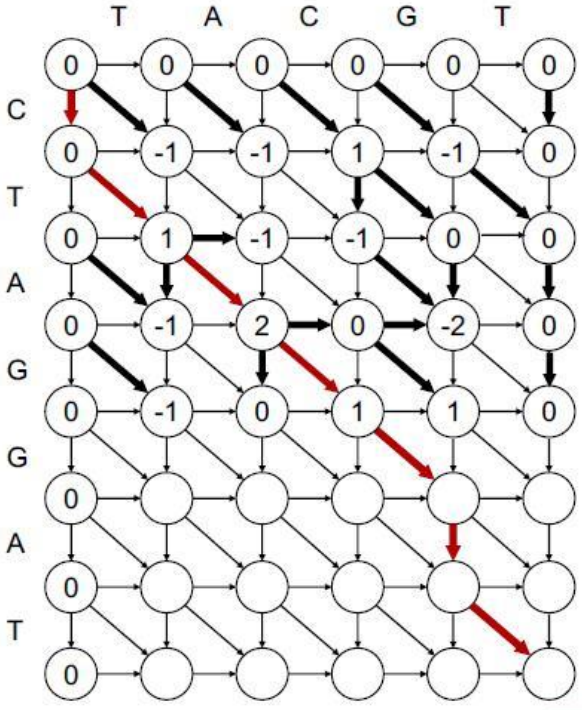
- Any possible alignment between the two sequences can be represented as a path from the left top corner to the right bottom corner.
- For example, the path highlighted in green corresponds to the alignment below.

```
- - T A - C G T
g g s s g s s g
C T A G G A T -
```

- This alignment is obviously not optimal : it does not contain a single match !



La première ligne et colonne commencent par 0. On va dans les 3 directions et on calcule la cellule suivante, en sachant qu'un match vaut 1, une substitution vaut -1 et qu'un gap vaut -2. On reste à 0 dans la première ligne et colonne même si on retrouve un gap car il n'y a pas de pénalité en premier ou dernier résidu de chaine. On procède au score ligne par ligne jusqu'au moment où l'on arrive au coin droit. L'alignement final est obtenu par backtracking où l'on retourne en arrière pour trouver le chemin correspondant au meilleur score. Dans cet exemple, le meilleur score est :




```
-TACG-T
CTAGGAT
```

Ce score est donné par les scores de matrice de substitution, PAM et BLOSUM.
Pour implémenter ceci en python, il faudrait :

- Une fonction qui retourne les valeurs en cas de Match ou Mismatch

```
def Diagonal(n1,n2,pt):
    if(n1 == n2):
        return pt['MATCH']
    else:
        return pt['MISMATCH']
```

- Une fonction qui obtient les éléments de la matrice et qui retourne les éléments des pointeurs de la matrice.

```
pointer = max(di,ho,ve) #based on python default maximum(return the first element).

if(di == pointer):
    return 'D'
elif(ho == pointer):
    return 'H'
else:
     return 'V'
```

- Une fonction qui crée l'alignement et les pointeurs de matrices.

```
f NW(s1,s2,match = 1,mismatch = -1, gap = -2):
  penalty = {'MATCH': match, 'MISMATCH': mismatch, 'GAP': gap} #A dictionary for all the
  n = len(s1) + 1 #The dimension of the matrix columns.
  m = len(s2) + 1 #The dimension of the matrix rows.
  al_mat = np.zeros((m,n),dtype = int) #Initializes the alighment matrix with zeros.
  p_mat = np.zeros((m,n),dtype = str) #Initializes the alighment matrix with zeros.
  #Scans all the first rows element in the matrix and fill it with "gap penalty"
  for i in range(m):
      al_mat[i][0] = penalty['GAP'] * i
      p_mat[i][0] = 'V'
  #Scans all the first columns element in the matrix and fill it with "gap penalty"
  for j in range (n):
      al_mat[0][j] = penalty['GAP'] * j
      p_mat [0][j] = 'H'
  #Fill the matrix with the correct values.

  p_mat [0][0] = 0 #Return the first element of the pointer matrix back to 0.
  for i in range(1,m):
      for j in range(1,n):
          di = al_mat[i-1][j-1] + Diagonal(s1[j-1],s2[i-1],penalty) #The value for match/
          ho = al_mat[i][j-1] + penalty['GAP'] #The value for gap - horizontal.(from the
          ve = al_mat[i-1][j] + penalty['GAP'] #The value for gap - vertical.(from the up
          al_mat[i][j] = max(di,ho,ve) #Fill the matrix with the maximal value.(based on
          p_mat[i][j] = Pointers(di,ho,ve)
  print np.matrix(al_mat)
  print np.matrix(p_mat)
```

Avec PAM60 (60% de mutation par résidus), les substitutions sont plus pénalisées et les gaps sont favorisés, alors que c'est le contraire pour PAM250(250% de mutation par résidus)

En conclusion, cet algorithme est approprié quand les séquences sont similaires sur toute leur longueur, mais des similarités locales pourraient être manquées.

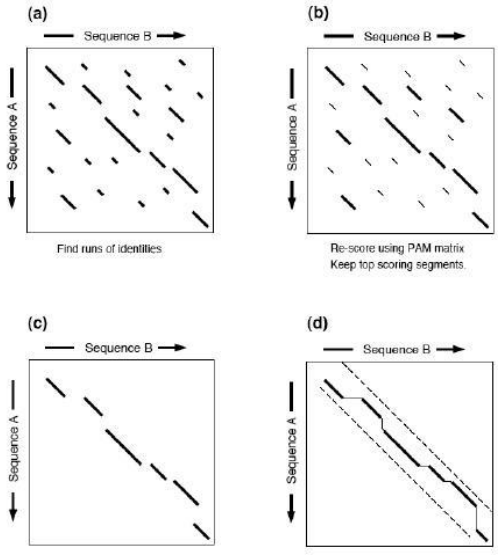
C'est pourquoi on a inventé l'algorithme de Smith-Waterman, qui détecte ces similarités locales (domaines). Semblable au premier, sauf que les scores négatifs dans la matrice sont remplacés par 0, et que l'alignement se stoppe après un score maximum local. Bien savoir choisir entre les deux : l'alignement global est approprié pour l'alignement de séquences qui sont conservées dans toute leur longueur (divergence récente) et l'alignement local va détecter des domaines qui couvrent une fraction de la séquence, et retourne l'alignement total quand la conservation couvre toute la séquence

Différences entre FASTA et BLAST ?

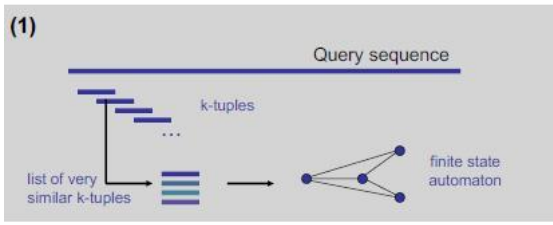
Tout d'abord, les deux sont bien plus rapides que Smith-Waterman. Ils sont utilisés pour aligner une séquence avec une base de données. Leurs réponses sont assez proches de l'optimum.

FASTA en 4 étapes :

1) Comparaison des positions des k-tuples entre la séquence et la base de données
2) Les segments avec un grand score sont identifiés, ce dernier étant spécifié par un seuil. Au-dessus du seuil, le segment est gras. En dessous, le segment est plus fin.
3) Les régions avec un faible score et les segments fins sont filtrés.
4) Les régions restantes sont assemblées avec Smith-Waterman

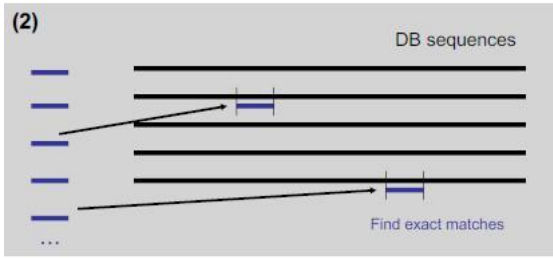


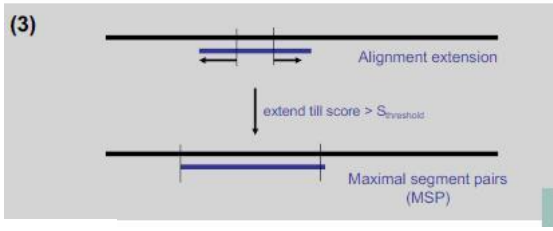
Le score se base par défaut sur PAM250 pour les protéines, et de match/mismatch +5/-4 pour les nucléotides. Utilise aussi le word-size

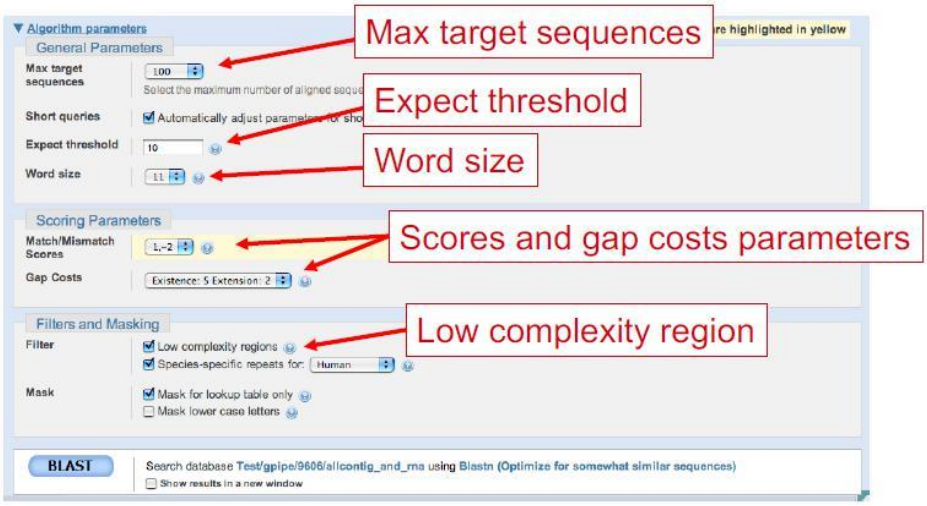


BLAST en 3 étapes :

1) Il compile une liste de k-tuples
2) Il identifie les k-tuples similaires entre la séquence et la base de données
3) Pour chaque hit, il étend l'alignement dans les deux directions et ne retient que les alignements avec le plus grand score, encore une fois suivant un seuil. Le score dépend encore de la matrice de substitution choisie.







Blast fait donc un alignement local qui s'étend ensuite sur tout le génome.

D'autres versions existent, comme :

- Gapped : utilise des plus petits k-mers et étend quand il y'a 2 hits sur la même diagonale
- PSI-Blast : trouve similarités de manière itérative avec matrice de score
- PHI-Blast : Identifie homologies via motifs connus
- MEGABLAST : Optimisation locale successive
- BLASTZ : alignement entier de génome humain/souris
- BLASTX : compare la séquence d'ADN aux protéines du PDB

Les différences pour la première étape entre les deux :

- FASTA regarde seulement les k-tuples identiques dans la séquence et la base de données, alors que BLAST cherche des k-tuples avec un score au-dessus d'un seuil donné.

- FASTA utilise la hashing/chaining méthode, alors que BLAST construit un dictionnaire de k-tuples dans la séquence en utilisant la méthode finite state automaton. Les deux méthodes ont pour but d'identifier les k-tuples communs.

Les différents scores

- P-value : probabilité d'observer un score S par chance
- Bit-score : transformation de la p-value en logarithmes népériens.

$$S' = \frac{\lambda S - \ln(K)}{\ln(2)}$$

- E-value : nombre de matchs avec un score S qui serait obtenu par chance. Il équivaut à $N/2^s$ où N est le search space (size of query x size of database)

Alignement multiple

On l'utilise pour détecter des similarités entre séquences, de motifs structurels, construire des arbres phylogénétiques, etc.
Il arrange les séquences en un tableau rectangulaire avec pour but que les résidus soient homologues (dérivés d'un ancêtre commun), superposables (dans un alignement 3D) ou jouant un rôle fonctionnel commun (site catalytique, etc.)
Pour scorer, un alignement multiple, il y'a deux méthode courantes :

The usual scoring method:

- assumes independance between the columns

$$S(m) = \sum_i S(m_i)$$

S(m) = score of the whole alignment m
$S(m_i)$ = score of column i in this alignment

- scores each column according a "sum-of-pairs" (SP) function using a substitution scoring matrix.

$$S(m_i) = \sum_{k<l} s\left(m_i^k, m_i^l\right)$$

m_i^k = residue in sequence k in column i
S(a,b) = score from a substitution matrix (PAM or BLOSUM for example)

Voici les différentes approches avec les différents programmes :

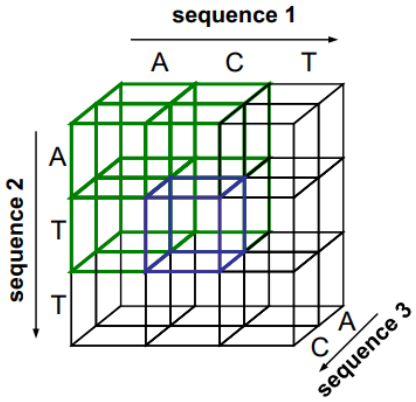
	Global approaches	Local approaches
Simultaneous	DP (Needleman-Wunsch) MSA (Carillo-Lipman)	MAFFT
Progressive	CLUSTALW (NJ) MULTALIGN (UPGMA) PILEUP (UPGMA)	PIMA
Iterative	HMMT (HMM model) SAGA (Genetic algo) MSA-GA (Genetic algo) MSASA (sim. annealing) MUSCLE	DIALIGN MATCH-BOX

Approches globales simultanées

La méthode de score normale est la somme de la colonne, basée sur les matrices de substitution (Indépendance entre les colonnes). Ensuite on fait la somme du score obtenu pour chaque colonne.

- **La programmation dynamique (N-W)** dans ce cas peut s'étendre jusqu'à 3 séquences, où l'on construit une matrice 3 dimensions, avec calcul du meilleur score dans chaque cellule, sur base des cellules précédentes, et un système de notation (matrice de substitution + pénalité de gap)

  Elle peut s'étendre aussi sur n séquences, en utilisant un hyper-cube n-dimensions, mais le temps augmente exponentiellement : GROS PROBLEME.

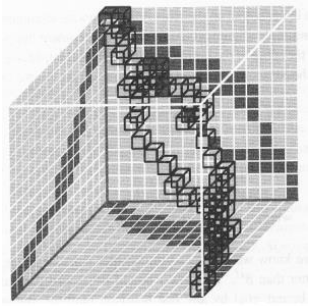


Number of "cubes": L^N

Number of neighbours/cube
=number of scores/cube: 2^N-1

=> computational complexity: $O(2^N L^N)$

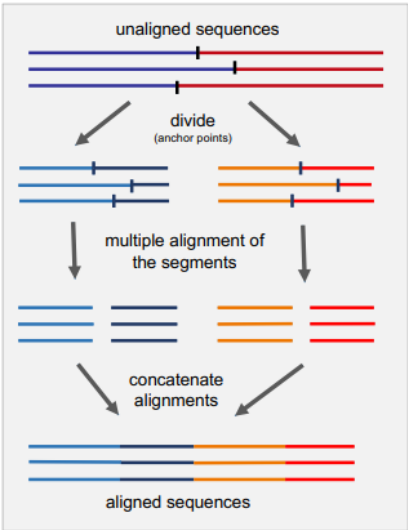
- **L'algorithme Carrillo-Lipman** (programmation dynamique optimisée), qui restreint la recherche en un sous-espace de la matrice n-dimension. Le sous-espace est déterminé en utilisant un chemin aléatoire à travers le cube et tous les alignements par paire entre chaque paire de séquences. Le programme garantit de retourner l'alignement optimal, mais le temps computationnel dépend du chemin aléatoire initial. Au plus celui-ci est proche de l'optimum, au plus c'est rapide. Toutefois, cet algorithme est limité à 8 séquences.

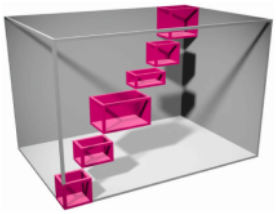


<u>Approches locales simultanées</u> :

L'approche locale se focus sur l'alignement de région par région, dans le but ultime de pouvoir aligner une séquence complète, tout en trouvant des domaines conservés (locaux) dans ces séquences. On divise donc la séquence en sous-séquences, on calcule l'alignement optimal, puis on concatène les sous-alignements. Idéalement, le programme devrait voir automatiquement des régions similaires de séquence, et puis étendre l'alignement reste de la séquence.

"Divide-and-conquer" principle





Idea:

- Divide sequences into small sub-sequences
- Use MSA (or other methods) to calculate optimal alignment for sub-sequences
- Concatenate sub-alignments

Challenge:

Can we ask the program to determine the "sub-sequences"?

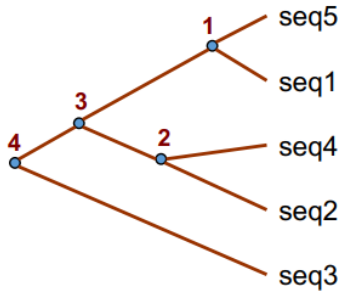
- **MAFFT** : utilise la transformée de Fourier
  1) Les régions homologues sont rapidement identifiées par la transformée rapide de Fourier, dans lequel une séquence d'acides aminés est convertie en une séquence composée des valeurs de volume et de polarité pour chaque acide aminé.
  2) Un système de score simplifié qui va réduire le coût CPU et améliorer la précision de l'alignement, même pour les séquences qui ont de larges insertions ou des extensions aussi bien que pour les séquences distantes liées de même longueur.

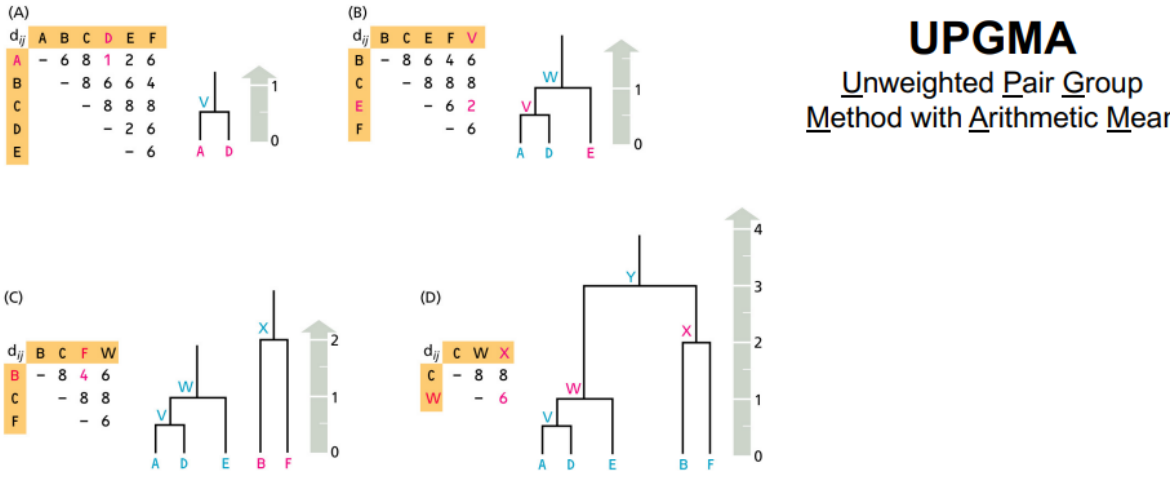
Approches globales progressives

1) On calcule une matrice de distance où l'on représente la distance entre chaque pair de séquences.

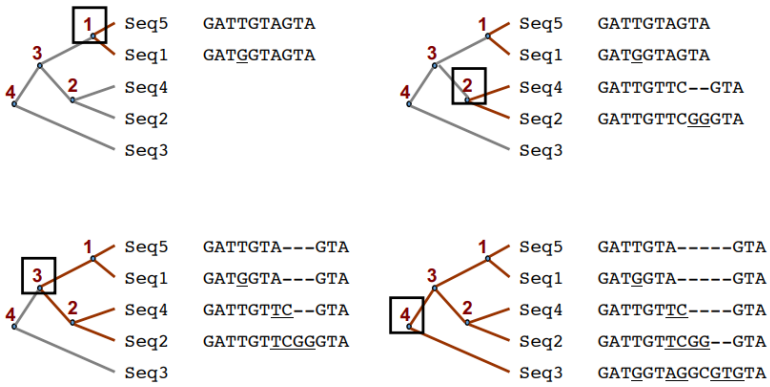
	Seq 1	Seq 2	...	Seq n
Seq 1	$d_{1,1}$	$d_{1,2}$	...	$d_{1,n}$
Seq 2	$d_{2,1}$	$d_{2,2}$	...	$d_{2,n}$
...	...	...	...	...
Seq n	$d_{n,1}$	$d_{n,2}$	...	$d_{n,n}$

2) Ensuite, on construit un arbre phylogénétique (via plusieurs méthodes comme UPGMA)





3) On l'utilise comme guide pour progressivement aligner les séquences.



Cependant, il ne garantit pas de retourner la meilleure solution. Les paramètres clés sont le choix des séquences (ajouter/supprimer une séquence pourrait affecter tout l'alignement, le choix du tree algorithm, et des paramètres de score comme les matrices de substitution, les pénalités de gap, et le poids des séquences (optionnel).

Les distances sont calculées via différents algorithmes.

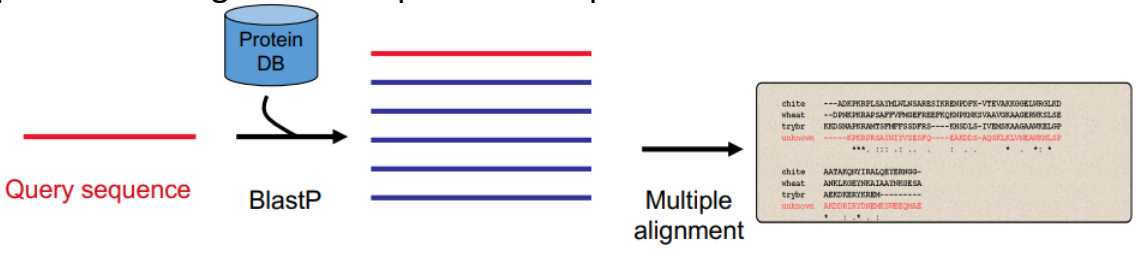
- **ClustalW** : calcul le meilleur match entre les séquences et les met en avant pour que l'on puisse voir les similarités, identités, etc. (mais ne garantit pas de retourner l'alignement optimal)

  In this software, the distance is calculated as (Kimur

$$D_{ij} = -\ln\left(1 - d_{ij} - \frac{d_{ij}^2}{5}\right)$$

  where d_{ij} is the number of mismatches in the optimal alignment of sequences x_i and x_j divided by the alignment length.

- **DBClustal** : utilisé pour les protéines. Aligne les top séquences via une base de données BlastP à partir d'une séquence donnée. L'algorithme est basé sur ClustalW modifié, et qui incorpore les données d'un alignement local sous la forme de points d'ancrage entre les paires de séquences.



Approches locales progressives

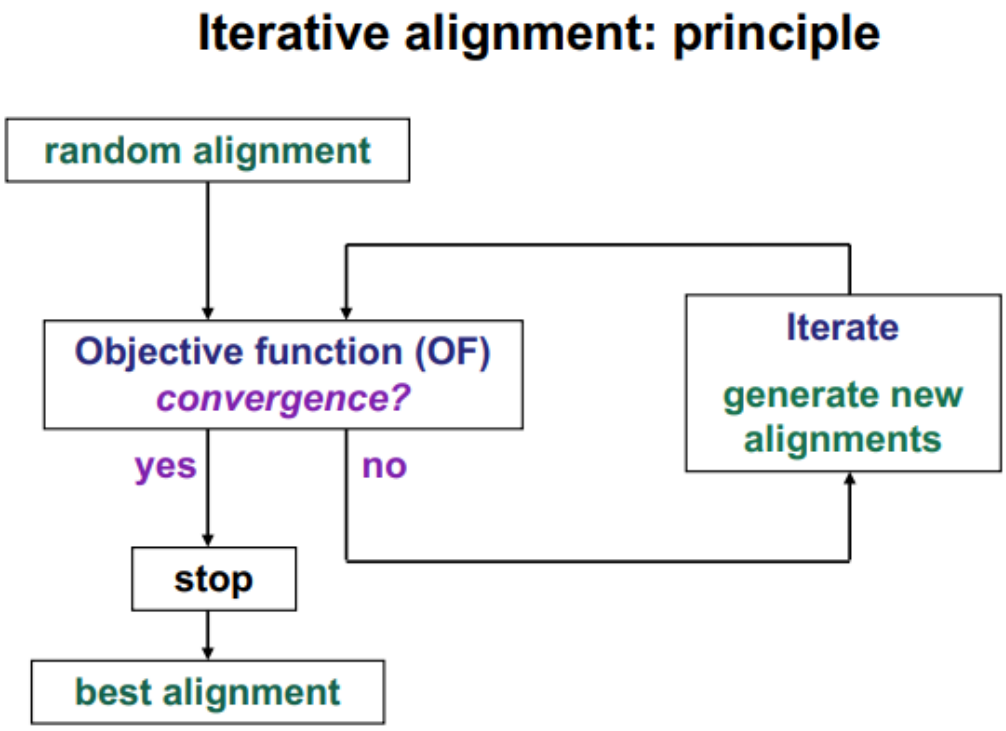
- **PIMA** : utilise un algorithme employant des pénalités de gap dépendant de la structure secondaire pour la modélisation comparative de protéines

Approches itératives globales

Même philosophie que pour le progressif, mais avec une itération.

L'itération se base comme suit :

- Obtention d'un premier alignement, de basse qualité
- Amélioration de cet alignement par diverses modifications
- L'alignement ne peut plus être amélioré : fin.



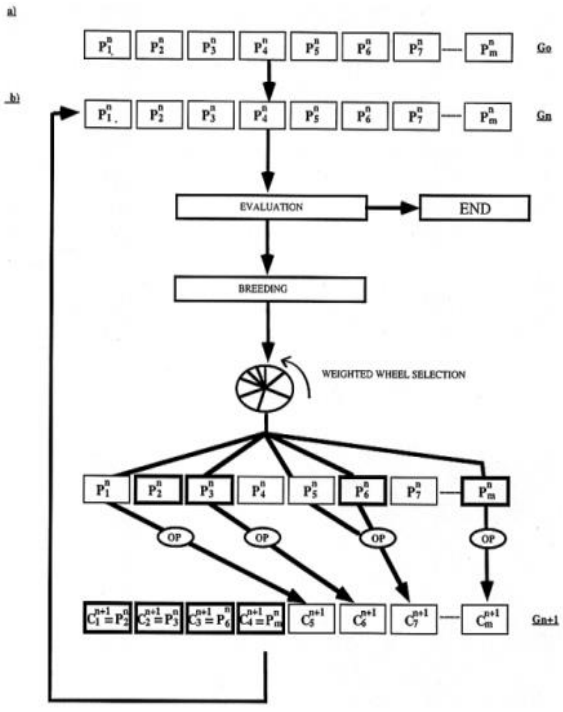
Il faut définir une objective function, dans le but de générer un alignement avec le meilleur score. La philosophie de ces fonctions est basée sur l'algorithme génétique (suivant la sélection naturelle). De nouveaux

alignements sont créés à partir d'alignements précédents. Les alignements avec le meilleur score ont une plus grande chance de donner de nouveaux alignements (nouvelle génération), avec ajouts de quelques biais pour favoriser de meilleurs alignements. Voici les différents programmes :

- **SAGA** (Sequence Alignment by Genetic Algorithm) :
  1) On crée une population random, constituée d'un set d'alignement qui ne contient pas de gaps. (Taille de la population = 100)

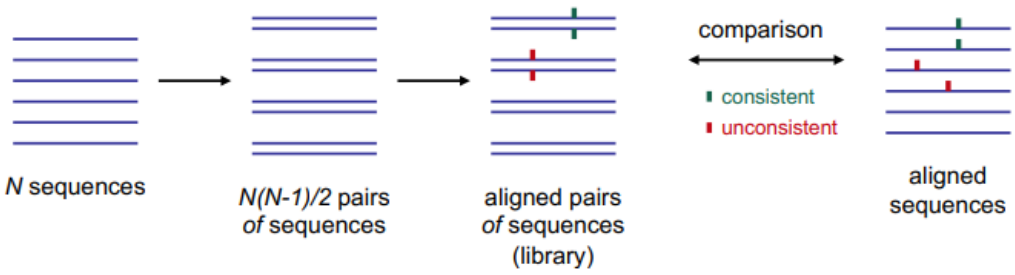
  2) On évalue le score (via objective function). Les meilleurs alignements auront plus de chance de produire une descendance. 50% des parents sont remplacés à chaque génération.

  3) S'en suit des étapes de « reproduction » où 50% de la meilleure population précédente est gardée (avec les meilleurs scores), et 50% restants où les parents subissent de petites modifications via un opérateur, qui est un petit programme modifiant l'alignement (p.e. le merge entre deux alignement, insertion de gap, etc). Chaque opérateur a une probabilité spécifique d'être utilisé. Le choix de l'opérateur (22) en lien avec sa probabilité dépend de son efficacité pour les 10 générations précédentes. Pour créer une nouvelle génération, un opérateur est choisi selon cette probabilité. Le choix des sites de mutations se fait le plus souvent dans une zone avec plus de gaps



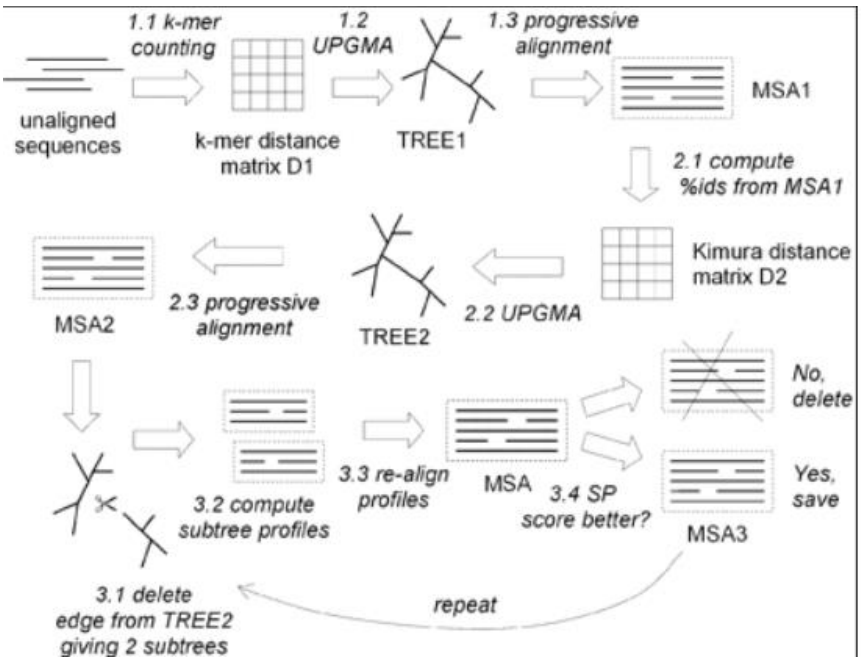
Il n'y a pas de preuve valide à ce que l'alignement final ait atteint l'optimum, même après un nombre infini d'itérations. La décision de stopper la recherche doit alors être un critère arbitraire. Pour SAGA, il utilise le critère de stabilisation : quand le programme n'a pas su améliorer la nouvelle génération pendant 100x, il se stoppe et on obtient l'alignement final.

- **COFFEE** (Consistency based Objective Function For alignmEnt Evaluation) : optimise la consistance de l'alignement en comparant de multiples alignements avec tous les alignements par pairs.



$$\text{score} = \frac{\text{number of consistent residue pairs}}{\text{total number of residue pairs}}$$

- **MUSCLE** (Multiple Sequence Comparison by Log-Expectation : optimise les alignements par prévision logarithmique



MUSCLE is based on a combination of progressive and MSA alignments and optimize the alignment in an iterative way.

Approches locales itératives

- **DIALIGN** : l'algorithme se base sur la collection de diagonales non croisées (combiner des plus petites pour en créer de plus grandes).

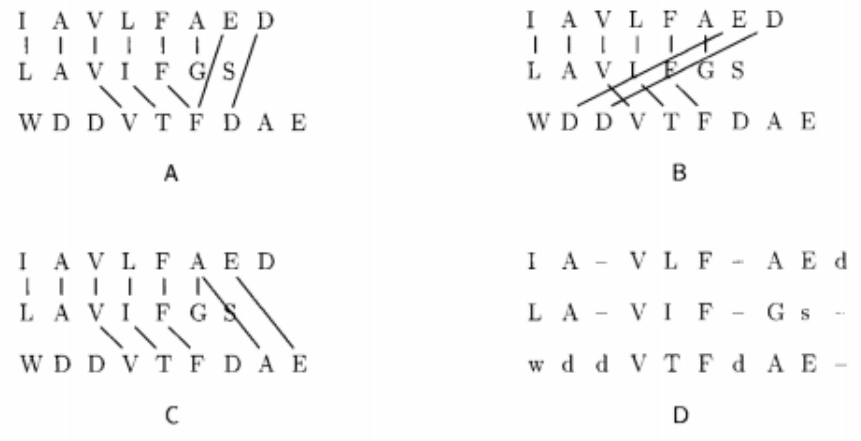
    1) Toutes les paires d'alignement possibles sont faites sur le set de séquences, et les diagonales sont scorées. Les scores incluent un score brut pour la diagonale complète, ainsi qu'un poids d'overlap (pénalité) qui reflète l'extension de quelle diagonale overlappe l'autre.

    2) On combine les plus petites diagonales pour en créer de plus grandes.

    3) On cherche pour de nouvelles diagonales qui n'ont pas encore été découvertes*

    4) . S'il n'y en a pas/plus, l'alignement est considéré comme achevé. DIALIGN requiert seulement 3 étapes d'itération

L'avantage est qu'il y'a peu de paramètres à considérer (pas de pénalité de gaps, …). L'utilisateur peut aussi spécifier un seuil T pour sélectionner les segments, et des points d'ancrage, qui consistent en des paires de segments de longueur identiques qui doivent être alignées par le programme.



Diagonals are drawn between identical or similar residues.

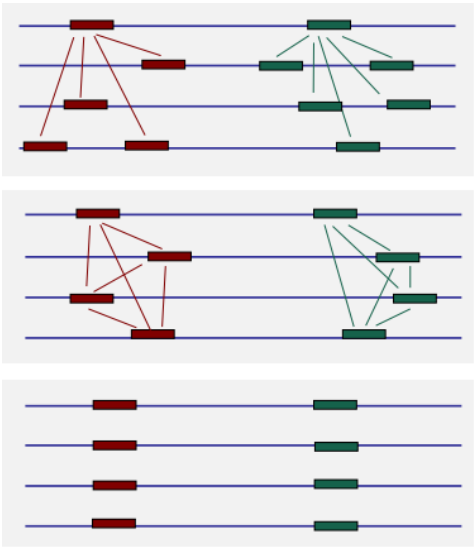
(A) non-consistent collections of diagonals (the third sequence is assigned simultaneously to two different residues of the first sequence)

(B) non-consistent collections of diagonals (there is a cross-over)

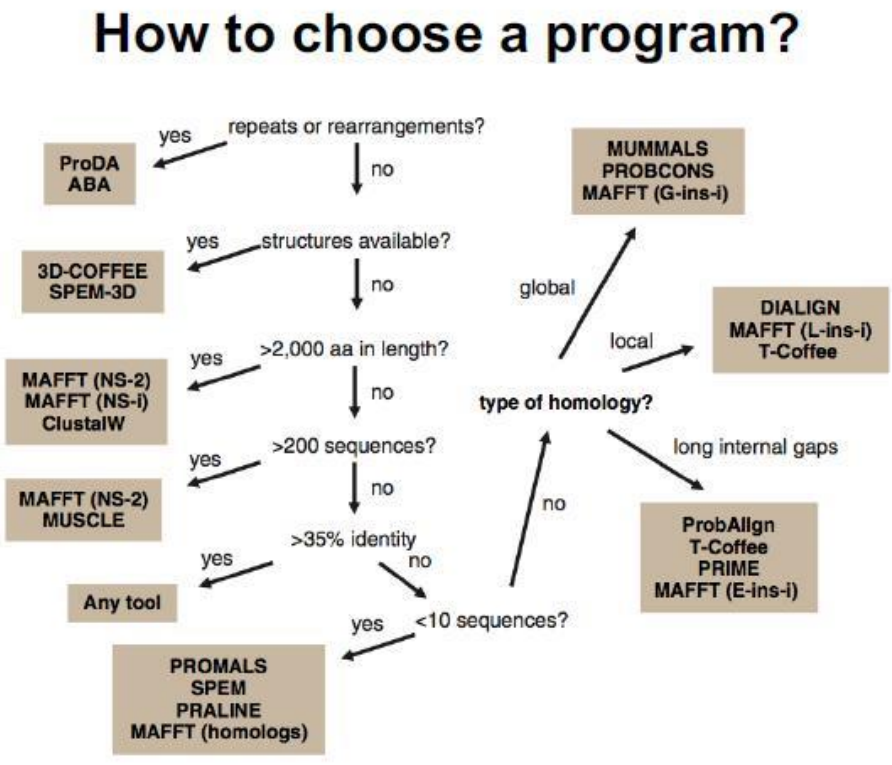
(C) a consistent collection of diagonals. It is possible to introduce gaps into the sequences such that residues connected by diagonals are in the same column of the resulting alignment.

(D) Residues not involved in any of the three diagonals are printed in lower-case letters. They are not considered to be aligned.

- **MATCH-BOX** :
  1) Chaque sous-séquence est caractérisée par un profil physicochimique multivarié, nommé box
  2) L'alignement est exécuté en considérant toutes les séquences simultanément, et l'algorithme détecte ensuite les régions (boxes) qui forment un set de profils similaires. Peut être utilisé pour prédire des régions conservées.



Multiple alignment: choosing a program



Trouver un motif dans une séquence ?

Un motif est une séquence courte de l'**ADN** (TATA box, ECO R1,…), **ARN** (boucles, hairpins qui permettent des interactions pour maintenir le repliement compact de l'ARN), **protéine**(Peptide signal ,…), susceptible d'avoir une fonction biologique : cela peut être un site où se joignent des facteurs de transcription, une séquence qui catalyse une réaction biochimique, etc.

Très important pour savoir comment un gène est régulé, comment il fonctionne, et comment il est structuré. Les motifs ne sont pas à confondre avec les domaines. En effet, ces derniers sont des séquences conservées qui peuvent encore se replier si elles sont extraites de la protéine (Domaine globine par exemple). Dans le cas des motifs, ils ne peuvent pas se replier une fois extraits.

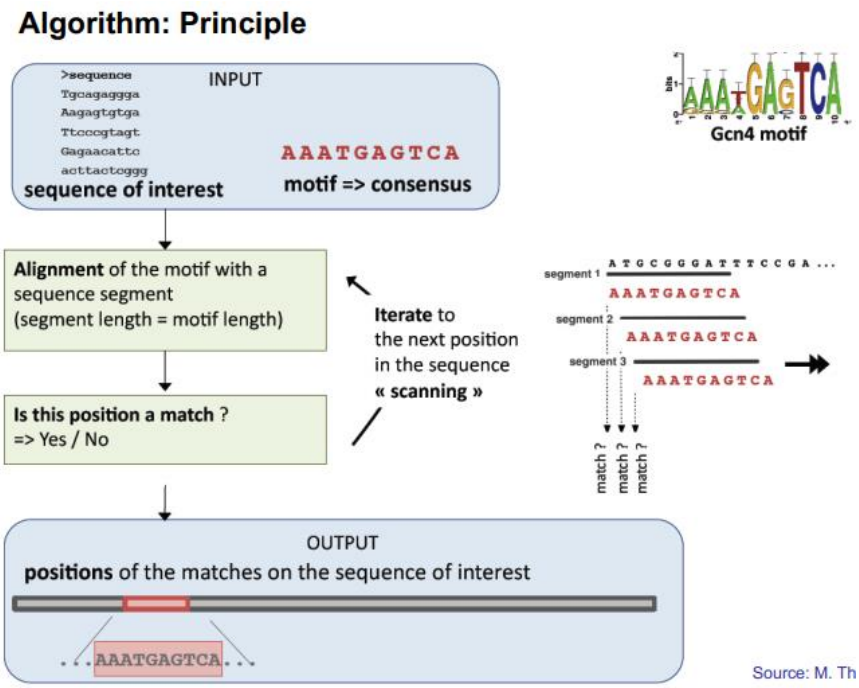
Challenges pour la prédiction des TFBS (Transcription Factor Binding Site) :

1) Très courtes séquences (5-20 pb)
2) Peuvent être cherchés dans de très larges séquences génomiques
3) Peuvent être très éloignées de leurs gènes cibles
4) Sont dégénérés : un facteur de transcription peut lier plusieurs TFBS avec différentes séquences
5) Ont des séquences égales dans des sites aléatoire sans fonction dans le génome
6) …

Les 3 représentations communes de motifs sont la séquence consensus, les profils matriciels (PWM/PSSM) et les modèles de Markov cachés

1) La séquence consensus/expression régulière

Chaine de caractères qui résume le mieux le motif. Chaque lettre correspond à la plus grande probabilité de la trouver à cette position. Un consensus est habituellement construit par un alignement multiple des motifs séquentiels individuels. Elle utilise le code IUPAC pour décrire cette séquence.



DNA motif (alignment)

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

Consensus sequence

```
A C A C - - A T C
```

Une méthode plus flexible et plus générale existe : l'expression régulière, qui représente les différentes possibilités. Voici un exemple :

DNA motif (alignment)

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

Regular expression

```
[AT]C[AC][ACGT]*A[TG][GC]
```

Either A or T

Either A,C, G or T any number of times

Cette méthode donne juste les lettres qui sont permissibles à chaque position du motif. Elle ne prend pas en compte les statistiques de variation dans les motifs séquentiels, elle dit juste quelles lettres sont permissibles à chaque position du motif.

2) Les profils matriciels

Profils (PSSM) qui capturent la fréquence de chaque lettre à chaque position du motif. En partant d'un alignement multiple, on construit une matrice qui reflète le résidu majoritaire pour chaque position :

- Chaque colonne représente une position
- Chaque ligne représente un résidu
- Les cellules indiquent la fréquence de chaque résidu pour chaque position de l'alignement multiple.

Site (1)	A	G	A	T	C	C	A	T
Site (2)	T	G	A	C	T	G	A	T
Site (3)	T	C	A	T	C	G	T	T
Site (4)	A	G	A	T	T	G	A	T
Site (5)	T	C	A	A	G	G	A	T
Site (6)	T	G	A	T	C	G	A	C
Site (7)	A	A	A	T	C	G	A	T
consensus:	T	G	A	T	C	G	A	T
IUPAC consensus:	W	V	A	H	B	S	W	Y

Multiple alignment

	1	2	3	4	5	6	7	8
A	3	1	7	1	0	0	6	0
C	0	2	0	1	4	1	0	1
G	0	4	0	0	1	6	0	0
T	4	0	0	5	2	0	1	6

Position-specific frequency matrix (counts)

	1	2	3	4	5	6	7	8
A	3	1	7	1	0	0	6	0
C	0	2	0	1	4	1	0	1
G	0	4	0	0	1	6	0	0
T	4	0	0	5	2	0	1	6

:7

	1	2	3	4	5	6	7	8
A	0.43	0.14	1	0.14	0	0	0.86	0
C	0	0.29	0	0.14	0.71	0.14	0	0.14
G	0	0.57	0	0	0.14	0.86	0	0
T	0.57	0	0	0.71	0.29	0	0.14	0.86

Ensuite, on calcule la probabilité totale pour chaque résidu (0.57 x 0.57 x 1 x ….), on divise cette proba avec la proba d'obtenir cette séquence par chance (imaginons 0.25 pour chaque lettre)

On obtient donc un score qui détermine la significativité ou non.

Un score de 20 signifie qu'on a 20x plus de chance que la séquence cible corresponde à un motif qu'une séquence aléatoire. (Ici, PFM)

	1	2	3	4	5	6	7	8
A	0.43	0.14	1	0.14	0	0	0.86	0
C	0	0.29	0	0.14	0.71	0.14	0	0.14
G	0	0.57	0	0	0.14	0.86	0	0
T	0.57	0	0	0.71	0.29	0	0.14	0.86

?

...	A	C	A	C	A	T	T	C	A	C	T	A	C	...

			0.43	0.29	1	0.71	0.29	0.14	0.86	0.14				

Thus:

$$\frac{P(\text{ACATTCAC} \mid \text{PFM model})}{P(\text{ACATTCAC} \mid \text{background model})} = \frac{4.33\ 10^{-4}}{1.53\ 10^{-5}} = 28.36$$

	1	2	3	4	5	6	7	8
A	0.54	-0.58	1.39	-0.58	?	?	1.23	?
C	?	0.15	?	-0.58	1.04	-0.58	?	-0.58
G	?	0.82	?	?	-0.58	1.23	?	?
T	0.82	?	?	1.04	0.15	?	-0.58	1.23

Position-specific scoring matrix (PSSM)
= Position-weight matrix (PWM)

$$W_{i,j} = \ln\left(\frac{f_{i,j}}{p_i}\right)$$

p_i = prior probability
(here: $p_A = p_C = p_G = p_T = 0.25$)

On peut aussi passer d'une matrice de probabilité à une matrice de score, via les prior probabilities. (Ici, PSSM)

	1	2	3	4	5	6	7	8
A	0.54	-0.58	1.39	-0.58	-4.27	-4.27	1.23	-4.27
C	-4.27	0.15	-4.27	-0.58	1.04	-0.58	-4.27	-0.58
G	-4.27	0.82	-4.27	-4.27	-0.58	1.23	-4.27	-4.27
T	0.82	-4.27	-4.27	1.04	0.15	-4.27	-0.58	1.23

The pseudo-counts have been introduced by Hertz & Stormo (1999) to account for the small number of sequences used to build the PSSM matrix.

Position-specific scoring matrix (PSSM) with pseudo-counts

$$W_{i,j} = \ln\left(\frac{f'_{i,j}}{p_i}\right)$$

with $$f'_{i,j} = \frac{n_{i,j} + p_i k}{\sum_{r=1}^{A} n_{r,j} + k}$$

k = pseudo-count
(here: k=0.1)

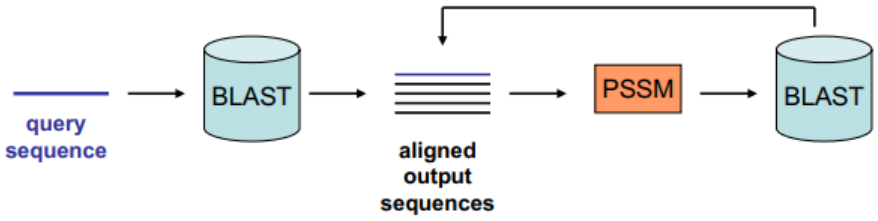
Le pseudo count signifie qu'une fréquence de 0 pour une lettre peut être le fruit d'une erreur. On inclut donc une légère probabilité qu'elle puisse toujours être incluse dans le motif. Quant aux prior, cela dépend du génome en question (l'Homme 41% de GC). On retrouve aussi des variations locales dans les résidus comme les ilots CpG, etc.

A partir de cette matrice de score, on multiplie (idéalement le plus grand chiffre) de chaque colonne pour ainsi obtenir un score final. Une valeur positive indique que la séquence a plus de chance d'être le modèle PSSM qu'un modèle aléatoire. Bien entendu, on peut mettre un seuil sur ce score, pour éviter d'avoir des motifs faux positifs (attention à ne pas le mettre trop haut non plus)

Un programme qui utilise ces PPSM est PSI-BLAST :

PSI-BLAST : Position-Specific Iterated BLAST (Altschul et al, 1997)

- BLAST runs a first time in normal mode.
- Resulting sequences are aligned together (Multiple sequence alignment) and a PSSM is calculated.
- This PSSM is used to scan the database for new matches.
- Steps 2-3 can be iterated several times. This procedure typically converges after a few cycles.



La limitation des profils matriciels :

1) Il est difficile de reconnaitre un motif qui contient des insertions ou délétions.
2) Il ne peut pas capturer les dépendances positionnelles. Par exemple, si on retrouve à chaque fois RD ou QH en position j et j+1, mais jamais QD ou RH.
3) Il n'est pas super adéquat pour représenter les motifs de longueur variable, ni pour détecter les frontières étroites entre deux régions (sharp boundaries)

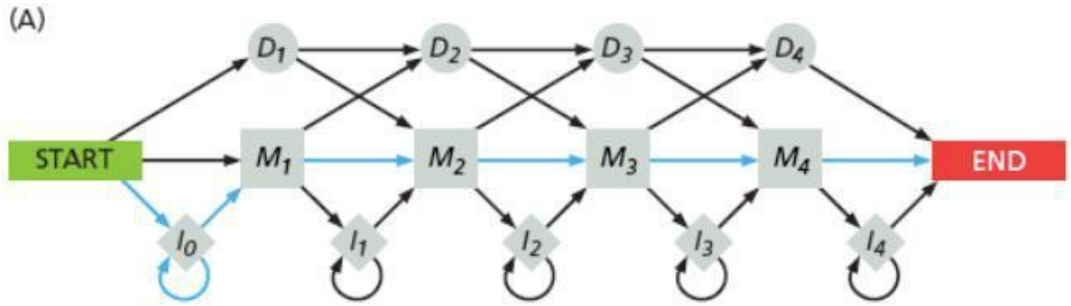
3) **Modèle caché de Markov**

Sorte d'automaton à état fini qui est utilisé pour trouver un motif spécifique dans une séquence.

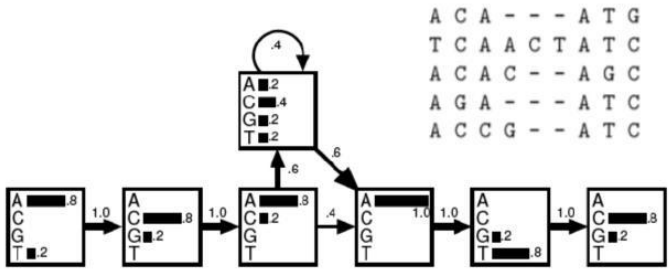
- Un modèle caché de Markov est défini en ayant un set d'états, qui ont tous un nombre limité de transitions vers les autres états et un nombre limité d'émissions à partir d'un état donné.

- Chaque transition entre les états aura une probabilité donnée d'être assignée, la valeur étant indépendante des états précédents. C'est cette propriété qui fait la classe des HMM

- Chaque modèle considéré à un état initial et un état final. Tous les chemins à travers ce modèle à partir du début jusqu'à la fin produira une séquence



Voici le modèle complet, qui comporte un état initial et final, ainsi que 4 état de match. M = match, I = insert et D = delete. Chaque transition parmi ces états est associée à une probabilité. Voici un exemple d'exécution :



$$P(\text{ACACATC}) = 0.8 \times 1 \times 0.8 \times 1 \times 0.8 \times 0.6 \times 0.4 \times 0.6 \times 1 \times 1 \times 0.8 \times 1 \times 0.8 \simeq 4.7 \times 10^{-2}.$$

Cette probabilité finale est ensuite transformée en scores de log-odd :

Log-odd scores

$$\text{log-odds for sequence } S = \log \frac{P(S)}{0.25^L} = \log P(S) - L \log 0.25.$$

probability according the HMM model

prior (background) probability (assumed here equal for each residue).

Les limitations de ce modèle sont :

- Moins faciles à manipuler que PSSM car demande bien plus de ressources informatiques.
- Grand nombre de paramètres à utiliser. Pour le modèle simple, 36 paramètres doivent être pris en compte pour seulement 5 séquences
- Nombre de séquences limité. En effet, la plupart des facteurs de transcription régule un petit nombre de gènes, et donc seulement un petit nombre de sites de liaison peut être utilisé pour calculer toutes les probabilités. C'est pourquoi les pseudo-counts devraient être souvent utilisés pour corriger les observations manquantes.

Quelques databases qui comportent des matrices :

- YEASTRACT and RegulonDB
- TRANSFAC and JASPAR
- TRED

} Transcription factor binding sites

- BLOCKS
- PROSITE
- PFAM

} Protein motifs and domain

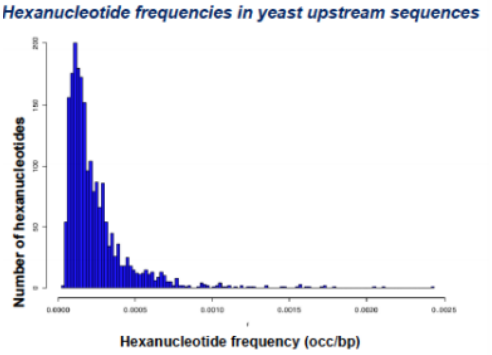
La découverte de motifs requiert :

- Une représentation du pattern : RegExp, PSSM ou HMM
- Une fonction objective :
  1) Seulement matchs parfaits
  2) Autorise un nombre donné de mismatchs
  3) Autorise une densité donnée de mismatchs
- Un algorithme : **Enumération et sur-représentation statistique** (Oligo-analysis, Dyad-analysis, RSA-Tools), **Optimisation déterministe** (MEME, CONSENSUS) et **Optimisation stochastique** (GLAM2)

Description des algorithmes

Enumération et surreprésentation statistique

Elle couvre l'espace de tous les motifs possibles. Elle compte le nombre d'occurrences de chaque n-mers dans la cible et calcule ceux qui sont les plus surreprésentés. Pour prédire les TBFS en commun dans plusieurs familles de gènes, une approche simple consisterait à détecter les oligonucléotides les plus fréquents pour chaque séquence en amont. Mais résultats décevants. Il faut donc détecter les motifs fréquents dans des gènes co-régulés et non de manière aléatoire : nécessité d'un background modèle où les occurrences observées sont comparées à l'attente aléatoire. Possible via modèle de Bernoulli (probabilité spécifique pour chaque nucléotide), Modèle de markov (basé sur les bases de la séquence elle-même) et External background (occurrences pour le même motif dans un dataset de référence)



- **Oligo-analysis** : utilise des statistiques rigoureuses pour la détection de motifs sur-représentés. Pour chaque motif b possible, son nombre actuel d'occurrence est comparé à son nombre attendu d'occurrences E(b). Parce que la distribution des nucléotides dans le génome est loin d'être aléatoire, un nombre spécifique attendu doit donc être utilisé :

$$E(b) = F_{nc}(b) \times 2 \times \sum_{i=1}^{S} (L_i - w + 1) = F_{nc}(b) \times T$$

Frequency of the motif *b* in the non-coding sequences

accounts for the 2 strands

Number of words of length *w* in sequence L_i (*S*=number of sequences)

T = total number of possible matches in the S sequences

Ensuite, on calcule la probabilité d'observer un motif (b) au moins n fois par rapport au nombre attendu d'occurrences E(b) de ce motif. La significativité statistique fait appel à des lois binomiales de distribution. Si la valeur finale est au-dessus de 0, cela veut dire que la surreprésentation d'un motif est significative.

- **Dyad-analysis** : c'est une extension d'oligo-analysis qui prend en compte les motifs composés de deux éléments conservés séparé par un interval moins conservé)

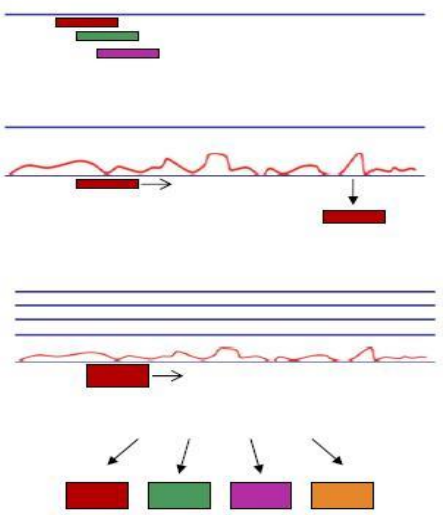
Les deux algorithmes ont été implémentés dans les outils d'analyse de séquences régulatrices (RSA-Tools)

Optimisation déterministe

Repose sur l'optimisation simultanée d'une position-weight-matrix (PWM) pour la description d'un motif.

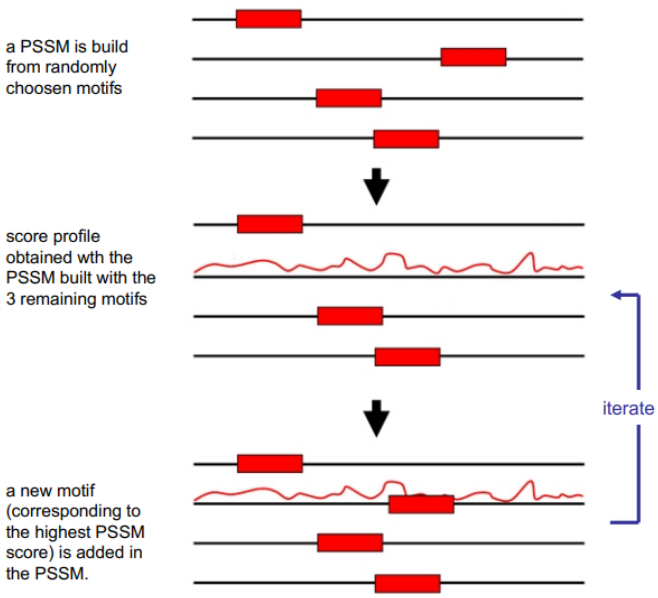
- **MEME** : Utilise un groupe d'ADN ou de protéines donné par l'utilisateur et génère un nombre de motifs. Il utilise ensuite des statistiques pour choisir la meilleure largeur, nombre d'occurrence et description de chaque motif. Tout cela est associé à une p value et E value, et retourne les PSSM avec les meilleures E value.
  1) Un PSSM est initié avec une séquence n-mer seule (+ un petit taux de background nucléotide frequencies)
  2) Pour chaque n-mer dans la séquence input, la probabilité de génération par PSSM et par background distribution est calculée. Si score positif, on adopte ce modèle
  3) Cette procédure est répétée pour chaque n-mer des séquences input
  4) Au final, MEME retourne les PSSM avec le plus grand score

- **CONSENSUS** : les PSSM sont affinées sur base de leur contenu informatif et non sur base de leur probabilité comme dans MEME.
  1) Un PSSM est construit pour chaque k-tuple de la première séquence.
  2) Chaque PSSM est utilisé pour scanner la seconde séquence. Si un grand score de motif est trouvé, il est ajouté dans le PSSM. C'est celui avec le plus grand contenu informatif qui est retenu.
  3) S'en suit une itération avec les sous-séquences, où les PSSM sont complétés et sélectionnés selon leur contenu informatif.
  4) Finalement, les plus grandes IC matrices sont données dans l'output

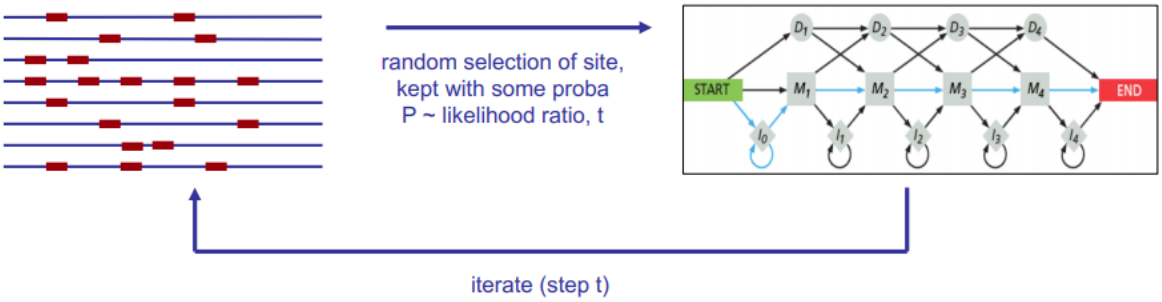


Optimisation stochastique

L'optimisation stochastique peut être vue comme une implémentation stochastique de la maximisation des attentes. Le motif est initialisé avec un ensemble de sites sélectionnés au hasard, et tous les sites dans les séquences cibles sont évalués par rapport au motif initial. À chaque itération, l'algorithme décide d'une manière probable d'ajouter ou non le site et/ou supprimer un ancien site du modèle de motif. Cette procédure est appelée l'échantillonnage de Gibbs. Il est conseillé d'exécuter plusieurs fois Gibbs pour s'assurer de l'efficacité des résultats car c'est une procédure heuristique, elle ne garantit pas de retourner le meilleur résultat en une fois.



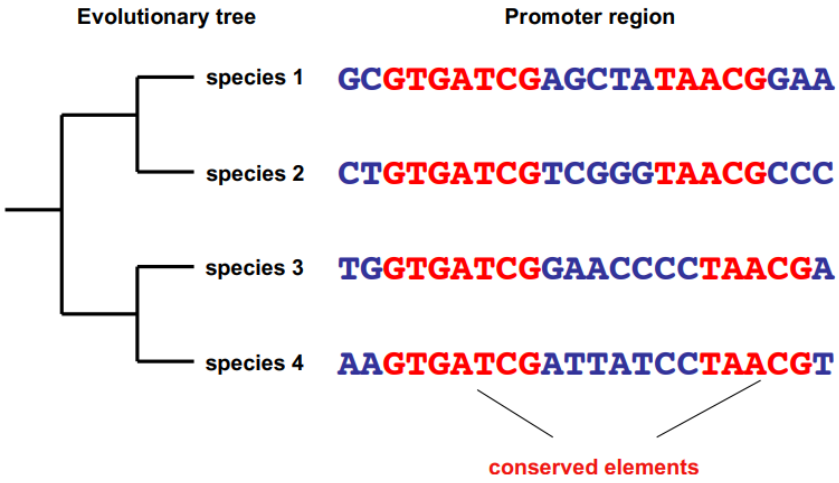
- **GLAM2 (Gapped Local Alignment of Motifs)** existe aussi, et est une généralisation du gapless Gibbs sampling algorithm. Il construit et optimise un modèle caché de Markov, et utilise des probabilités d'insertions et délétions à une position spécifique. A savoir que la plupart des algorithmes courants ne permettent pas des insertions ou délétions dans les motifs.

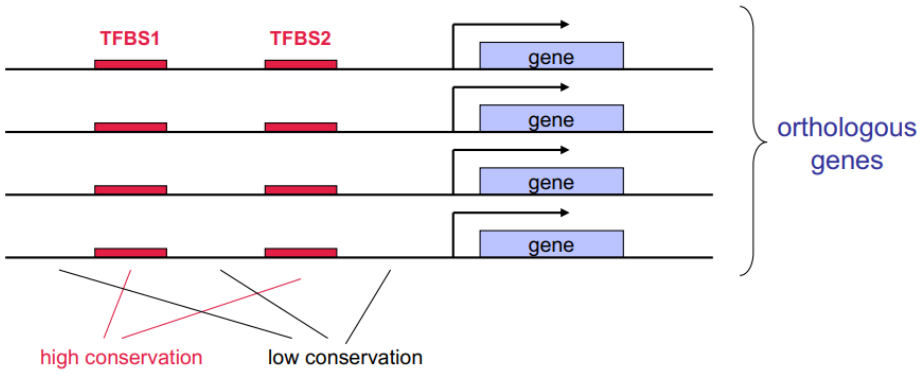


Le footprinting phylogénétique

Méthode pour découvrir les éléments régulateurs dans un ensemble de régions régulatrices homologues, collectées à partir d'espèces multiples. Elle permet de retracer la relation évolutive de ces éléments régulateurs. Ces dernières sont des régions non-codantes hautement conservées. Cette approche est basée sur l'hypothèse que :

1) Dans des séquences non codantes, les éléments régulateurs évoluent plus doucement que les séquences avoisinantes.
2) Dans les séquences non codantes de gènes homologues, les régions conservées peuvent révéler des éléments régulateurs agissant en cis.

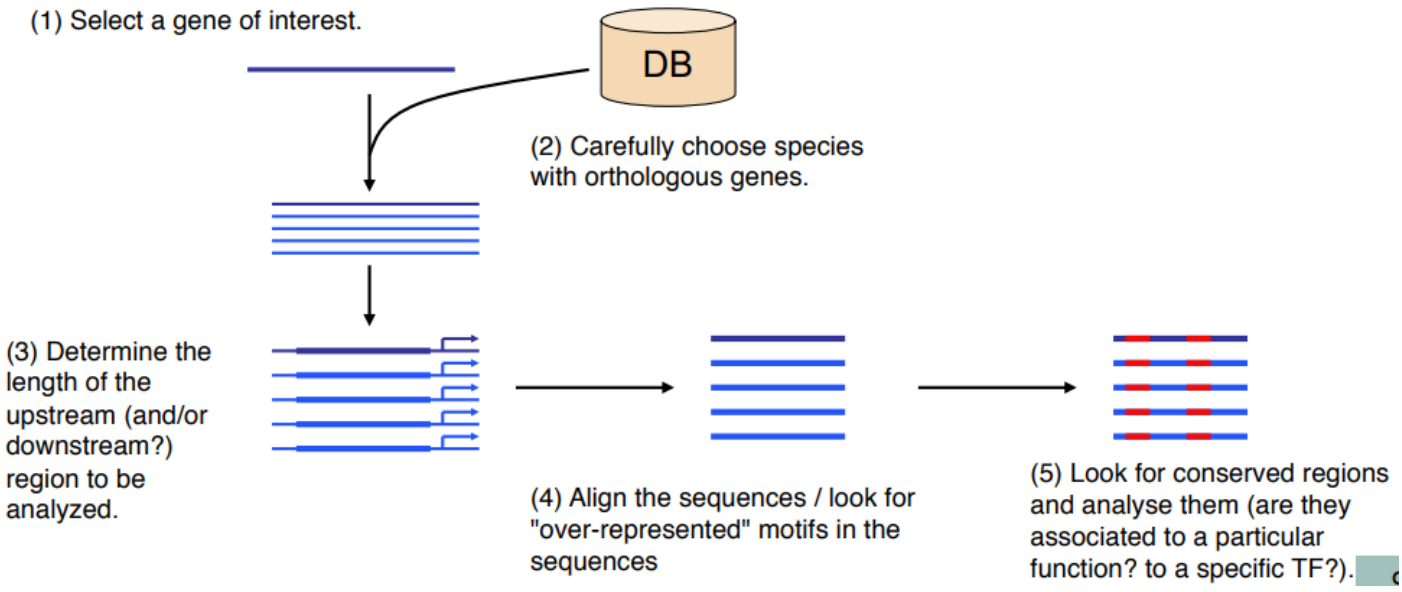




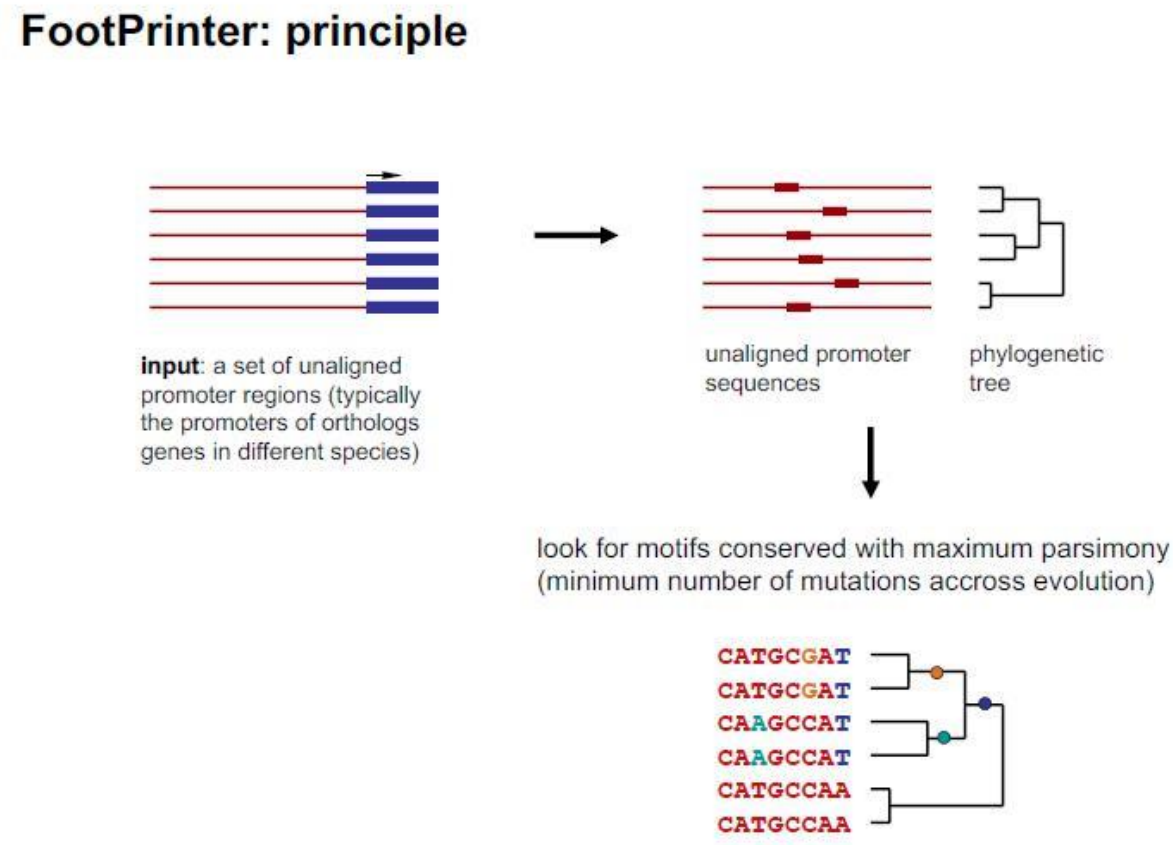
Il faut faire attention à ne pas choisir des espèces trop différentes, sinon on ne pourra pas retrouver les sites de liaison régulateurs. De même pour des espèces trop proches où il sera inutile de trouver les régions conservées.

La stratégie :

1) Choisir un gène d'intérêt
2) Choisir des espèces avec des gènes orthologues (homologues dérivés de spéciation)
3) Déterminer la longueur de la région à analyser (quelques pb)
4) Aligner les séquences choisies
5) Regarder aux régions conservées

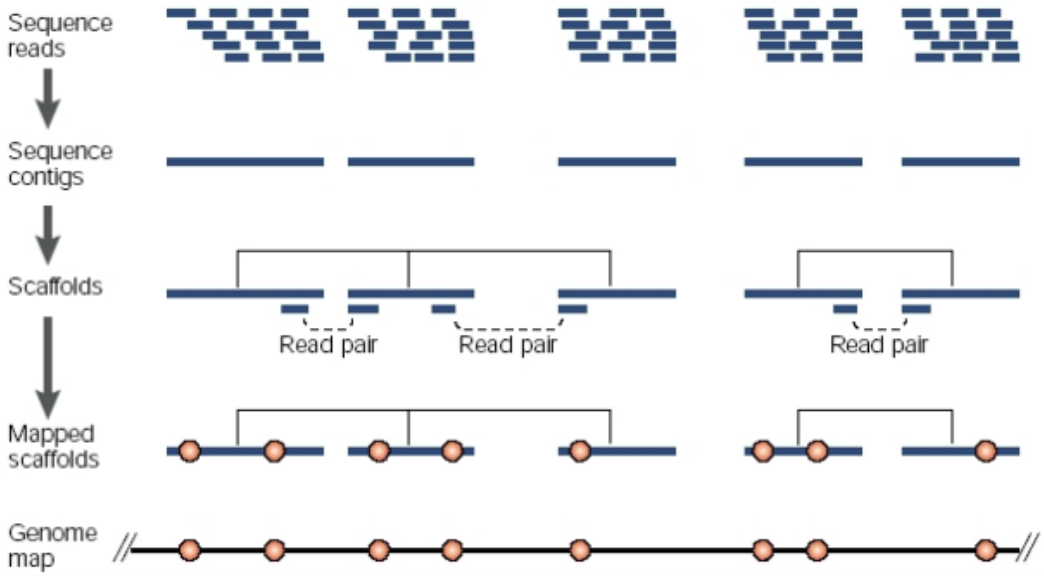


ClustalW (approche progressive globale pour alignements multiples) est approprié car il est basé sur la construction d'arbres phylogénétiques. Mais assez difficile car les séquences cibles sont très courtes par rapport à la quantité de matière à analyser, ainsi que des problèmes d'alignements possibles. Les programmes de découverte de motifs sont plus adéquats, comme **MEME** ou **CONSENSUS**, ainsi que les programmes d'arbres phylogénétiques comme **FootPrinter**.



Prédiction de gènes dans le génome ?

A partir de tous les reads obtenus par séquençage, il faut trouver les chevauchements via des programmes spécifiques, pour obtenir des contigs. Ces derniers vont être assemblés un par un pour les scaffolds, qui donneront la carte génomique. Les erreurs proviennent des régions répétées (overrepresented k-mer) qui doivent être retirées, des erreurs de séquençage ainsi que du polymorphisme qui produit des mismatchs de haute qualité.



Ensuite, on annote le génome de manière :

- Structurelle : régions codantes, introns/exons, promoteurs, régulateurs
- Fonctionnelle : fonction biologique, famille de protéine
- Traitement : interaction de la protéine, régulation des gènes

La prédiction de gène utilise 2 approches :

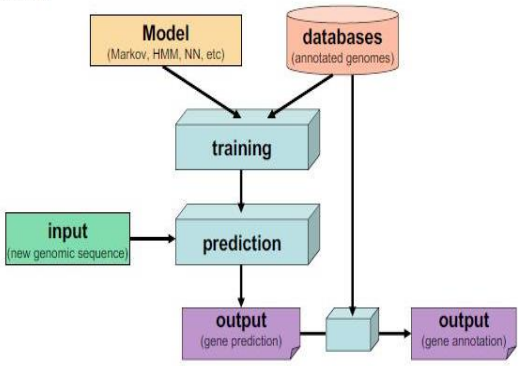
- Ab initio (analyse de séquences) : basé sur les propriétés statistiques de la séquence (et seulement elle)
- Recherche d'homologies via des autres organismes connus.

Les éléments de prédiction :

- Codons start et stop et son utilisation
- Signaux de l'épissage (eucaryotes) : très difficile

Gene prediction

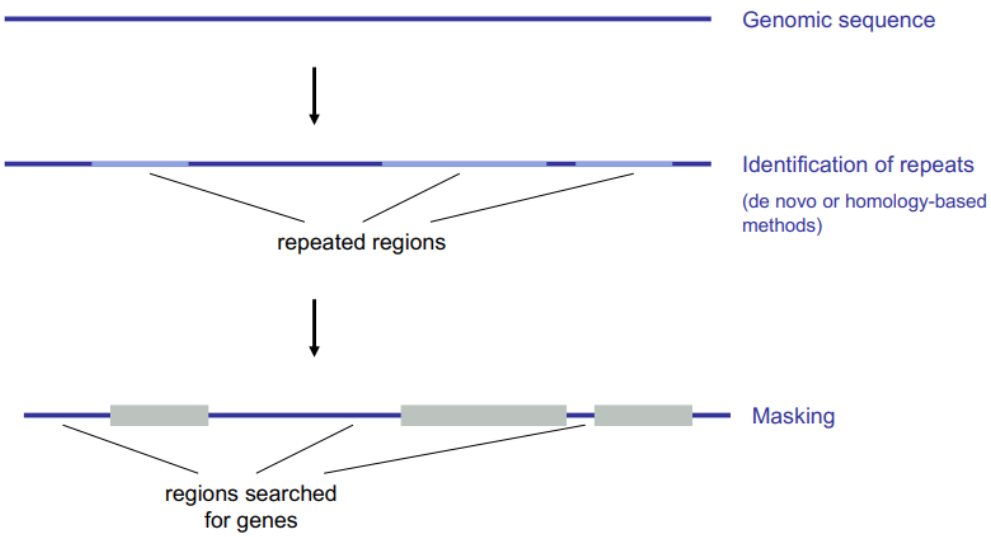
- **Pre-processing:**
  - Detection and filtering of repeats
  - Splitting the genome.
- **Gene-prediction tools:**
  - De Novo tools (based on sequence statistics)
  - Homology-based.



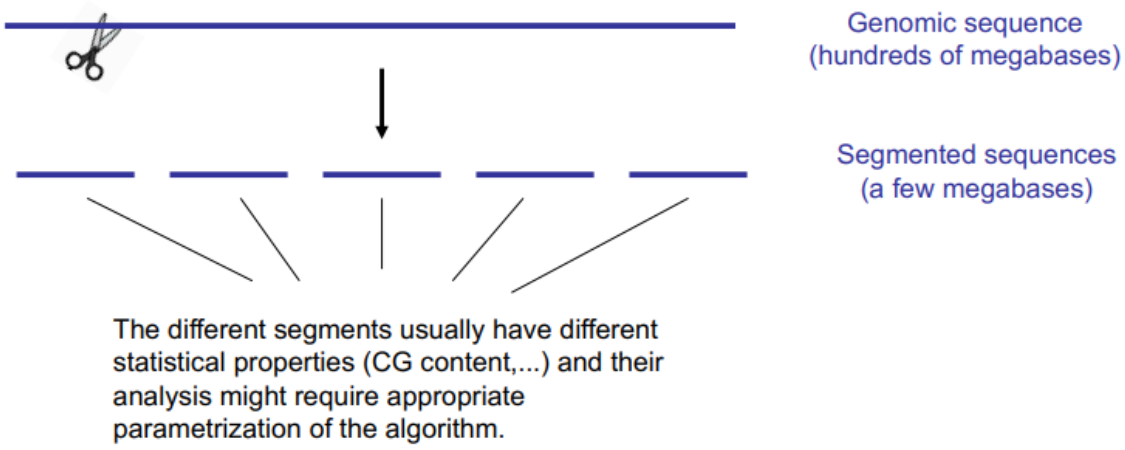
= au système neural de biophysics : un logiciel apprend via une database et divers procédés statistiques. Ainsi, il peut prédire un gène que l'utilisateur rentrera pour analyse.

Cela marche comme suit :

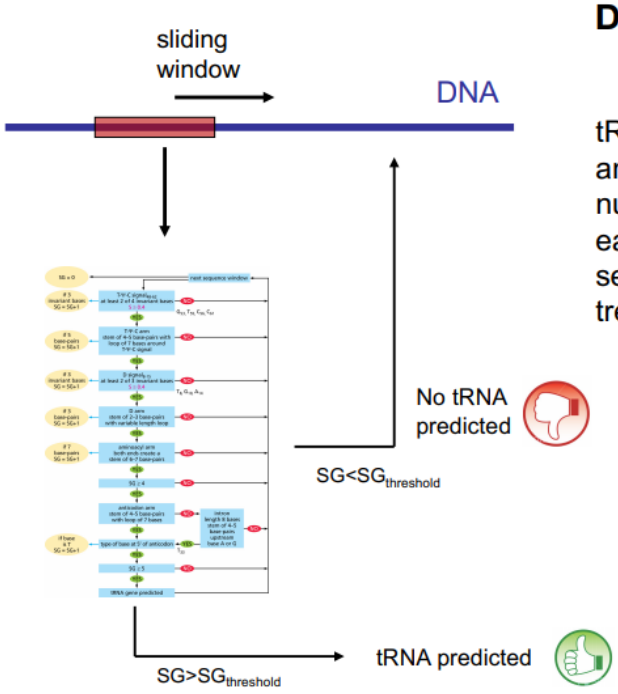
a) Détection des séquences répétées. 2 types de séquences : les non complexes, comme les nucléotides homopolymériques et les éléments transposables (virus, LINEs) et les cours éléments nucléaires intercalés comme les Sines. Les génomes des eucaryotes sont riches en ces éléments répétés (47%) et sont pauvrement conservées (ADN poubelle) Les méthodes utilisées sont l'outil d'homologie et l'outil de novo. Une fois ces séquences répétées détectées, on les masque.



b) On coupe la séquence génomique car la majorité des programmes ne peut pas faire face à un génome complet. Cette découpe est justifiée par la présence de régions avec un taux de GC hyper variable.



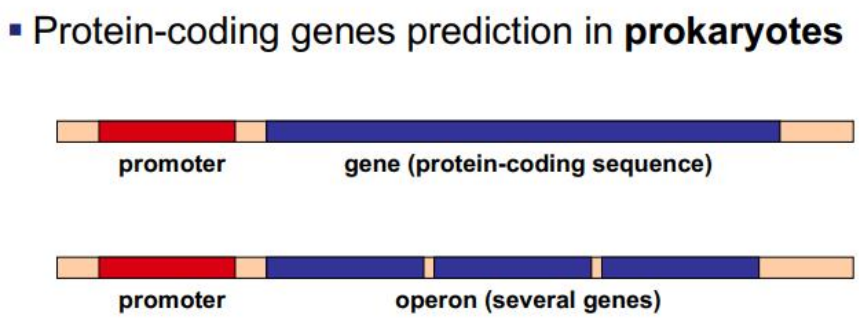
c) Identifier les gènes à ARN non codant(tARN, rARN, snARN, snoARN). Il est plus facile d'identifier cela ainsi que les séquences répétées par rapport à l'identification de gènes codant. On utilise le programme tRNAscan pour les identifier.

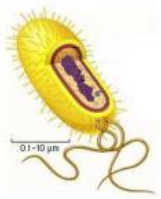


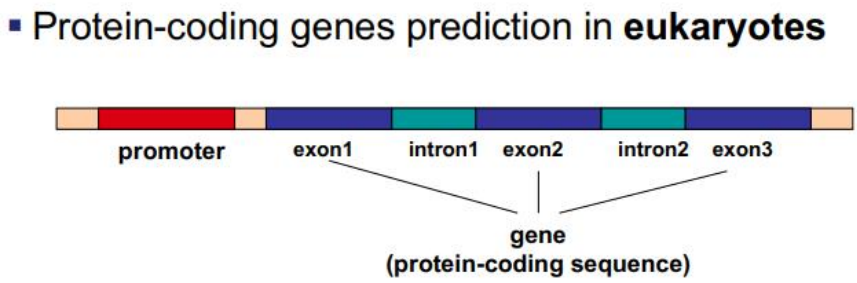
Detection of tRNA genes using a decision tree (tRNAscan)

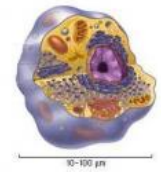
tRNAscan scans DNA sequence by anayzing sub-sequences (of about 75 nucleotides), window by window. At each position of the window the sequence is submitted to the decision tree anaysis.

Différence procaryotes/eucaryote



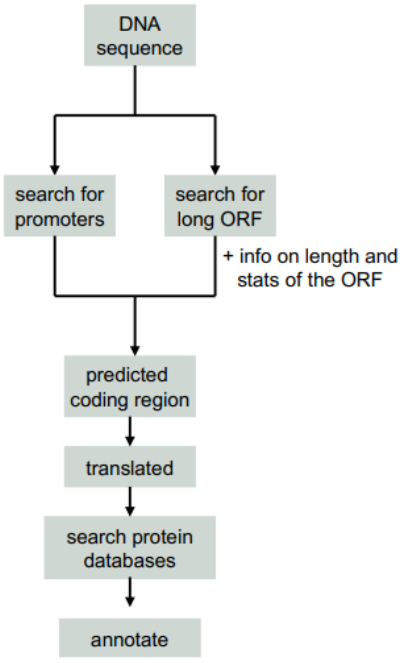






Prédiction chez les procaryotes

Chez les procaryotes, on recherche les régions promotrices (tata etc) ainsi que les longs ORF via les codons start (AUG) et stop (UAA, UAG et UGA). Facile à trouver car pas de présence d'introns.

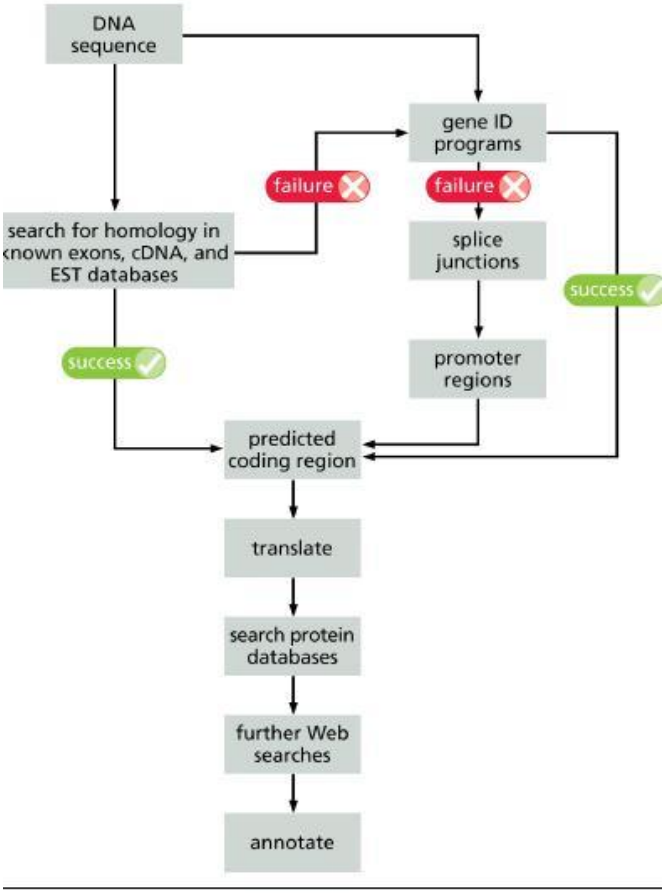


ORF Finder permet de trouver ça.
On regarde aussi aux régions très conservées telles que la boite TATA (-10), la boite Pribnow (-35) dans l'ADN, la séquence Shine-Dalgarno dans l'ARNm (-6/12).

Example

1. ATG CAA TGG GGA AAT GTT ACC AGG TCC GAA CTT ATT GAG GTA AGA CAG ATT TAA
2. A TGC AAT GGG GAA ATG TTA CCA GGT CCG AAC TTA TTG AGG TAA GAC AGA TTT AA
3. AT GCA ATG GGG AAA TGT TAC CAG GTC CGA ACT TAT TGA GGT AAG ACA GAT TTA A

Prédiction chez les eucaryotes

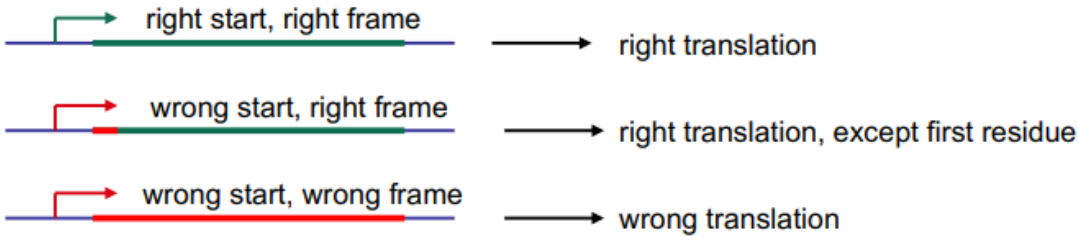


Beaucoup plus compliqué car présence d'introns qui peuvent être très longs (GU…….AG). Les séquences d'épissage sont repérables (AG.GU…….AG.G), mais il faut savoir qu'il existe l'épissage alternatif, ce qui rend la tâche encore plus difficile.

On regarde à la TATA box (ADN), à la séquence kozak (mARN), mais pas assez suffisant. On regarde donc aussi à la fréquence des codons, qui diffèrent selon les autres.

Aussi, analyser les EST : courtes séquences 200-800 nucléotides) qui proviennent de la 3' UTR de l'ARNm, et qui a été convertie en ADNc. Permet d'identifier les gènes inconnus. Une database d'EST existe, appelée dbEST.

Gene predictions must preserve the correct reading frame: when predicting exons, one must remember that they will have to be spliced together to produce a complete protein sequence all in the same reading frame. Because the genetic code involves 3-base codons it is important that the correct frame is retained at the splice site, which may even be in the middle of a codon. If this does not occur, then the following exon wil be translated in the wrong frame.



This type of problems can be circumvented by translating the individual predicted exons and checking the translations against a database search. Coding regions can be translated in 3 different reading frames and also both in the 5' to 3' or 3' to 5' direction. The most suitable protein candidate for database searches should be the segment that contains no or the least number of stop codons in its sequence.

Annotating the gene

Once a gene has been predicted, we can proceed to further steps: its structural and functional annotation. Knowledge of homologous proteins can be transfered.