

# Aide à l'étude

## Introduction à la bioinformatique

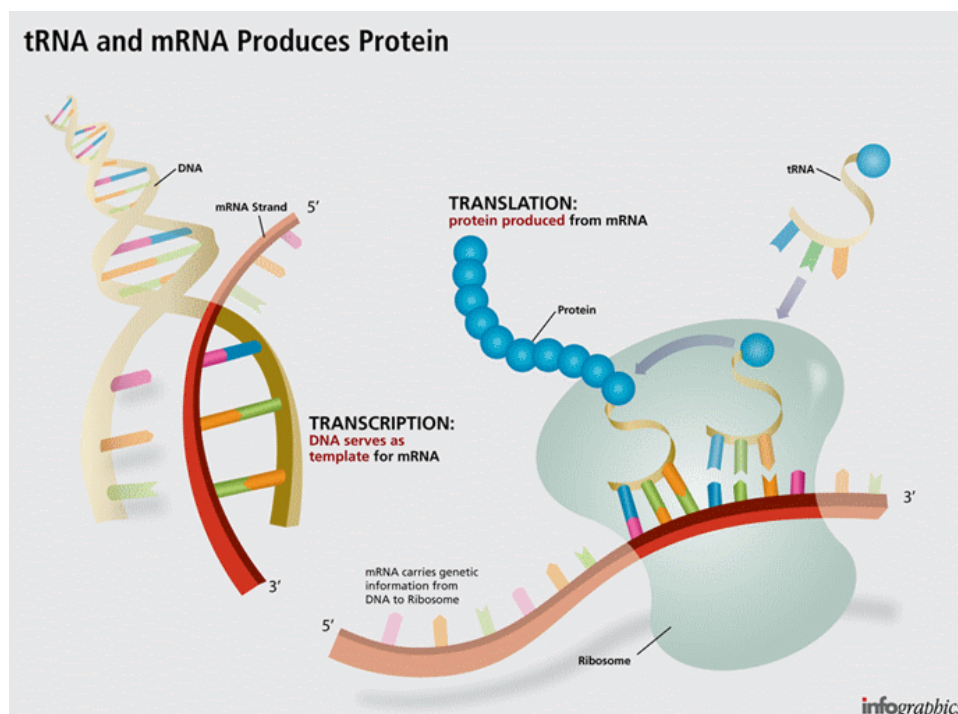
### INFO-F-208

## Définitions

**Transcription :** Des parties de l'ADN sont copiées dans des séquences plus courtes (pour pouvoir sortir du noyau) =  $\text{ARN}_{\text{messenger}} = \text{ARN}_m$

**Traduction :** L' $\text{ARN}_m$  est traduit par l'ARN-polymérase dans un ribosome en une séquence d'acides aminés (une protéine). L' $\text{ARN}_t$  transporte les enzymes correspondante aux codons de l' $\text{ARN}_m$ .

**Théorie fondamentale de la biologie moléculaire :** L'ADN est le support stable et transmissible de l'information génétique qui définit les fonctions biologiques d'un organisme (reproduction, nutrition, excrétion, action sur l'environnement, communication, etc.). Il est transcrit en ARN (un langage similaire) qui n'a qu'une vie temporaire, cet ARN assure soit une fonction structurale (squelette de complexes nucléo-protéiques), soit une fonction enzymatique (synthèse des protéines, exportation des protéines, épissage des ARNm eucaryotes, etc.), soit une fonction de transport de l'information génétique. L'ARN de type messenger ( $\text{ARN}_m$ ) est traduit en protéines par le ribosome (complexe nucléo-protéique). On parle de traduction des ARNm en protéines car le langage change. La théorie fondamentale se résume ainsi : L'ADN dirige sa propre réplication en ADN identique, ainsi que sa transcription en ARN, pouvant ou non être traduit en protéines.



**Brin codant :** ou **sens** ; Est une séquence de bases de l'ADN qui ne décrit pas le gène. Il n'est pas utilisé pour créer l' $\text{ARN}_m$  dans l'étape de la transcription. ex: 5'-GTAACGGTCA-3'. Il est identique à l' $\text{ARN}_m$  excepté **T** devient **U**.

**Brin non codant :** ou **antisens** ; C'est le brin complémentaire au brin codant/sens. L'ordre des deux brins est inversé. Utilisé pour créer l' $\text{ARN}_m$ .

**Random Drift :** Quand il n'y a pas de pression **sélective**, des mutations s'étendront ou

disparaîtront en conséquence d'évènements aléatoires.

**Sélection positive :** Si il y a de la pression sélective et quand la mutation fournit un avantage à l'organisme, la **sélection positive** augmentera la probabilité que cette mutation reste dans le génome.

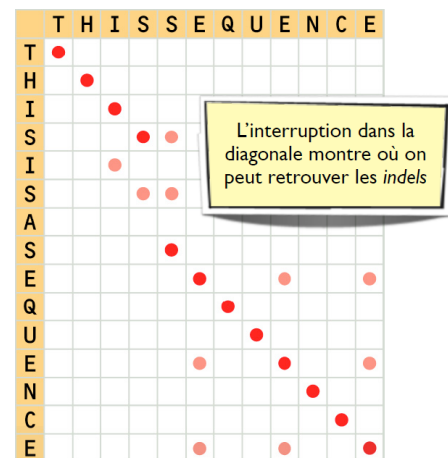
**Sélection négative :** ou **sélection purifiante** ; Quand la mutation fournit un désavantage à l'organisme, il disparaîtra à cause de la la **sélection purifiante ou négative**.

**Kimura's théorie :** ou **évolution neutre** ; Décrit qu'en réalité la plupart des mutations ne rencontrent pas la **sélection** forte (**négative** ou **positive**) et donc les mutations sont conservées en raison du **random drift**.

**Identité :** Le niveau d'identité est la fraction des résidus qui sont identiques dans les deux séquences.

**Similarité :** Deux résidus sont similaires si leur substitution n'a aucun effet sur la fonctionnalité. Le niveau de similarité est la fraction des résidus qui sont similaires.

**Le dot plot :** méthode qui montre la similarité entre deux séquences graphiquement. Surtout utilisée pour identifier des répétitions au sein de la séquence. Il y a souvent un problème de bruit de fonds (trop de similarité montrée) on applique donc un filtre (Une fenêtre superposant dans laquelle il faut avoir un nombre minimal d'associations identiques).



**Matrice de substitution :** Une matrice de substitution permet, pour chaque acide aminé, de connaître sa capacité à être substitué par un autre (y compris lui-même). Nous en avons vu deux types : PAM (PAM1 == 1% de différence de mutation dans les séquences utilisés pour la construction de la matrice) et BLOSUM (BLOSUM 80 == 80% d'identité entre les séquences utilisés pour construire la matrice)

**PAM xx :** Matrices créées en utilisant des groupes de séquences homologues obtenues à partir d'espèces différentes. elles donnent un score de substitution entre les acides aminés.

**BLOSUM xx :** est déterminée en utilisant des blocs d'acides aminés bien conservés dans des protéines. Le **xx** est une indication de l'identité entre les séquences qui étaient utilisées pour la construction de la matrice. On peut remarquer que les substitutions au sein des groupes d'acides aminés sont favorables.

**Chaîne de Transcription d'édition :** chaîne utilisant des caractères de l'alphabet {D,I,R,M} qui décrit la transformation d'une chaîne de caractères vers une autre.

I (insertion) ; D (suppression) ; R (remplacement) ; M (les deux caractères sont les mêmes (match))

**La distance d'édition(DE) :** ou **Levensthein distance** ; est définie comme le nombre minimum d'opérations d'ensemble {D,I,R} nécessaire pour transformer la première chaîne de caractères en la deuxième.

**Transcription optimale :** La transcription d'édition liée à la distance d'édition est la transcription optimale

**Pénalité linéaire :** Même pénalité quand on fait le premier espace que si on rallonge une série d'espaces.

**Pénalité affine :** Contraire de la linéaire, La pénalité affine permet de calculer une pénalité qui tient compte du début d'un espace (gap open) et de l'augmentation d'un espace (gap extend). C'est un modèle plus réaliste que la pénalité linéaire (l'insertion d'un espace à coût constant n'était pas représentatif d'un bon alignement de séquence).

On utilise dès lors le modèle de pénalité affine car :

- Il n'augmente pas la complexité de l'algorithme ;
- Il permet d'avoir des alignements de séquences plus réalistes (Augmenter un espace étant moins coûteux que d'en créer un).

**Structure quaternaire :** La structure ou le niveau quaternaire définit l'interaction d'une protéine avec d'autres protéines. Elle permet d'identifier les sites actifs et les formes.

**Le dogme d'Anfinsen :** *La structure native d'une protéine est déterminée par la séquence des acides aminés.* Cela veut dire qu'avec la séquence, on a assez d'information pour trouver la structure 3D des protéines. On a des algorithmes, qui sont continuellement améliorés, mais ne donnent pas d'aussi bons résultats que les approches expérimentales. Une petite protéine peut prendre des milliards exposant des milliards de formes. Néanmoins, une protéine se plie en quelques millisecondes dans la cellule. Il doit donc y avoir une astuce.

**Gènes orthologues :** Deux gènes avec une fonction commune, provenant de deux espèces différentes qui viennent du même gène héréditaire du dernier ancêtre commun et qui ont évolué par spécialisation et mutation, mais pas par duplication.

**Gènes paralogues :** Deux gènes de la même espèce qui ont été produits par la duplication mais qui ont divergé dans leur fonction. Généralement la duplication est suivie par des mutations indépendantes et des changements dans les séquences, résultant en un ensemble de gènes paralogues.

**Homologie :** Combinaison paralogues et orthologues. Deux séquences sont homologues si elles possèdent un ancêtre commun et la fonction chez chacune des espèces n'est pas forcément la même. L'homologie

implique une relation d'évolution et est à distinguer de la similarité.

**Finite state transducer (Transducteur fini) :** Un transducteur fini est un automate fini qui produit des sorties. C'est une extension des automates finis. Ils opèrent en effet sur les mots sur un alphabet d'entrée et, au lieu de simplement accepter ou refuser le mot, ils le transforment, de manière parfois non déterministe, en un ou plusieurs mots sur un alphabet de sortie. Ceci permet des transformations de langages. Ex : Machine de Moore et Mealey.

**Alignement semi-global :** C'est un alignement dont les pénalités pour les positions  $S(i,0)$  et  $S(0,j)$  sont à 0. On enlève l'obligation que les séquences doivent s'aligner au début et fin de la matrice. Le retour arrière commence à la position dans la dernière ligne avec la plus haute valeur.

**Pseudocounts :** les pseudocounts sont des constantes qu'on ajoute aux valeurs dans le profil. Les pseudocounts représentent la distribution antérieure, qui est la connaissance qu'on a concernant le système avant l'introduction des données. Ils compensent les problèmes posés par l'apparition de 0 lors d'une insuffisance de données.

**Programmation dynamique :**

## Questions / Réponses

### 1. Alignement des paires de séquences

**1.1. Donnez et expliquez les deux caractéristiques qu'un problème doit avoir si on veut utiliser la programmation dynamique ? Utiliser un exemple dans votre explication.**

1. **Sous-structure optimale** (la solution optimale pour le problème contient des solutions optimales pour les sous-problèmes du problème global) ;
2. **Sous-problème superposant** (une solution récursive contient un nombre limité de sous-problèmes distincts répétés beaucoup de fois).

Si un problème a les deux caractéristiques, la programmation dynamique pourrait être utilisée pour la création d'un algorithme.

Exemple: Recherche de la sous-séquence la plus longue entre deux séquences données.

**1.2. Quelles sont les deux caractéristiques qu'on doit observer dans un problème avant qu'on puisse utiliser la programmation dynamique ?**

- ❖ **La sous-structure optimale :** on dit qu'un problème a une sous-structure optimale si la solution optimale pour le problème contient des solutions optimales pour les sous-problèmes du problème global.
  - **Est-ce que la solution pour le problème est basé sur les solutions des sous-problèmes ?**
    - **Exemple :** Si  $z[1..k] = \text{SSL}(x[1..m], y[1..n])$ , chaque préfixe de  $z$  est un SSL d'un préfixe de  $x$  et un préfixe de  $y$
- ❖ **Les sous-problèmes superposant :** une solution récursive contient un nombre limité

de sous-problèmes distincts répétés beaucoup de fois.

- **Est-ce que le nombre de sous-problèmes est plus petit que le nombre d'appels récursive ?**

- **Exemple :** Le nombre de sous-problèmes différents de SSL pour deux séquences de longueur  $m$  et  $n$  et  $mn$ .

Si un problème a les deux caractéristiques, la programmation dynamique pourrait être utilisée pour la création d'un algorithme.

### 1.3. Expliquez pourquoi on a besoin d'une pénalité affine ? Est-ce qu'une pénalité linéaire n'est pas suffisante ? Expliquez les deux principes.

La pénalité affine permet de calculer une pénalité qui tient compte du début d'un espace (gap open) et de l'augmentation d'un espace (gap extend). C'est un modèle plus réaliste que la pénalité linéaire (l'insertion d'un espace à coût constant n'était pas représentatif d'un bon alignement de séquence).

On utilise dès lors le modèle de pénalité affine car :

- ➔ Il n'augmente pas la complexité de l'algorithme ;
- ➔ Il permet d'avoir des alignements de séquences plus réalistes (Augmenter un espace étant moins coûteux que d'en créer un).

### 1.4. Pourquoi est-ce qu'un score d'identité est insuffisant pour un alignement de deux séquences ?

Car le score d'identité ne prend pas en compte les changements acceptés par l'évolution. Par exemple une substitution d'un acide aminé en un autre qui ne change pas la fonction de la protéine.

Il faut aussi un score de similarité car il faut tenir compte des substitutions durant l'évolution. Le score total est la somme du score d'identité avec le score de similarité.

### 1.5. Pourquoi est-ce qu'un alignement qui utilise plusieurs séquences donne un meilleur résultat qu'un alignement entre deux séquences ?

Voir question 1.6.

### 1.6. Pourquoi est-ce qu'un alignement qui utilise plus de deux séquences, donne un meilleur résultat qu'un alignement entre deux séquences ?

Quand on fait des alignements entre plusieurs séquences, un grand nombre, on peut trouver plus d'éléments intéressants. En effet, parfois, des choses qu'on voudrait voir alignées ne le sont pas car le « bruit » autour est aligné à la place. En alignant plein de protéines entre elles, les parties communes ont bien plus de poids que le bruit. Ajouter des séquences permet également de renforcer l'alignement. Parfois, on produit un alignement entre deux protéines qui n'a en fait rien à avoir avec ce que l'évolution a fait. En alignant plein de protéines, on ne garde que ce que l'évolution a fait.

L'alignement multiple fait ressortir les régions conservées sur de multiples séquences, donc les structures et les fonctionnalités conservées, ce qui permet une meilleure validité de l'homologie (les fonctionnalités se conservent chez les descendants).

### 1.7. Pourquoi est-ce qu'on a besoin de trois matrices différentes pour l'implémentation de

**la pénalité affine dans les algorithmes d'alignement ?**

Matrice  $S$  pour les scores, matrice  $V$  pour les trous verticaux (dans la première séquence), et la matrice  $W$  pour les trous horizontaux (dans la deuxième séquence). -> Complexité  $O(mn)$  avec  $m$  et  $n$  la taille des deux séquences. Avec seulement la matrice  $S$  pour les scores, la complexité est de  $O(mn^2)$ . (mémorisation, éviter de recalculer plusieurs fois les mêmes valeurs)

**2. Matrices de substitution****2.1. Expliquez la signification d'un PAM. Qu'est-ce qu'il exprime ? Pourquoi est-il important ?**

PAM ou "Point Accepted Mutation" est une matrice qui montre les probabilités de mutation pour chaque paire d'acides aminés dans un intervalle d'évolution spécifique.

C'est une unité d'évolution/divergence, dans notre cas, 1 PAM = 1%/point of accepted mutation, 1 mutation acceptée par 100 résidu.

**2.2. Expliquez les principes de base qui sont utilisés pour la construction d'une matrice de substitution. Qu'est-ce qu'on essaie de comparer ? Comment ça fonctionne ?**

Voir question 2.3.

**2.3. Expliquez l'idée de base qui est utilisée pour la construction des matrices de substitution. Pourquoi est-ce qu'on a besoin d'un modèle aléatoire et évolutif ?**

Il y a deux mécanismes (mutation aléatoire et sélection évolutive) qui peuvent produire des différences entre des séquences de protéines lors de l'évolution d'un alignement. Un score de similarité entre les acides aminés peut être obtenu selon un modèle aléatoire ou un modèle évolutif. On établit la probabilité d'occurrence de l'alignement des résidus pour chaque modèle : «  $p_a \cdot p_b$  » pour le modèle aléatoire et «  $q_{ab}$  » pour le modèle évolutif. On utilise le concept de taux de log-chance [  $\log(q_{ab}/(p_a \cdot p_b))$  ] qui est la valeur qu'on peut trouver au final dans une matrice de substitution. La probabilité que les deux résidus soient alignés est plus grande dans le modèle aléatoire que dans le modèle évolutif. Une matrice de substitution attribue un score à la mutation d'un AA vers un autre. Ce score sera d'autant plus grand que cette mutation ait été sélectionnée par l'évolution, en comparaison avec la probabilité de voir apparaître cette mutation par hasard.

**→ Modèle aléatoire :**

- ◆ pas de contrainte sur la composition de la séquence ;
- ◆ choix de résidu pour chaque position indépendant des autres positions ;
- ◆ probabilité dépendante de la fréquence de chaque acide aminé dans la population des acides aminés ( $p_a$ ).

**→ Modèle évolutif :**

- ◆ les séquences ont une association évolutive (contrainte) ;
- ◆ probabilité d'occurrence dépend du résidu dans la même position dans l'ancêtre commun ;
- ◆ si deux résidus sont alignés, le modèle donne une probabilité d'occurrence combinée/multipliée (car les résidus sont dépendants) ( $q_{ab}$ ). Cela peut indiquer une homologie entre les séquences

## 2.4. Quelles sont les différences entre les matrices produites par la méthode PAM et la méthode BLOSUM ?

PAM est basé sur un modèle d'évolution explicite (les remplacements sont comptés sur les branches d'un arbre phylogénétique)	tandis que dans BLOSUM les matrices se basent sur un modèle implicite d'évolution.
Les matrices PAM sont basées sur les mutations observées dans un alignement global, comprenant à la fois les hautes conservation et les régions fortement mutables.	Les matrices BLOSUM sont basées uniquement sur des régions hautement conservées à partir d'alignements qui ne contiennent pas de gaps.
La méthode pour compter les remplacements est différente	Contrairement à PAM, la procédure de BLOSUM utilise des groupes de séquences dans lesquelles toutes les mutations ne sont pas comptées de la même manière car les mutations n'ont pas le même poids.
Les grands nombres de PAM signifient une large distance d'évolution,	tandis que les grands chiffres de BLOSUM montrent une très grande séquence de similarité et donc une petite distance d'évolution.

## 3. Chercher des homologues

### 3.1. Expliquez la différence entre une machine de Mealey et une Machine de Moore ? Dessinez un exemple pour chaque machine et nommez les différentes parties.

**Finite state transducer (Transducteur fini) :** Un transducteur fini est un automate fini qui produit des sorties. C'est une extension des automates finis. Ils opèrent en effet sur les mots sur un alphabet d'entrée et, au lieu de simplement accepter ou refuser le mot, ils le transforment, de manière parfois non déterministe, en un ou plusieurs mots sur un alphabet de sortie. Ceci permet des transformations de langages. Ex : Machine de Moore et Mealey.

**Moore :** "Output" sur les états.

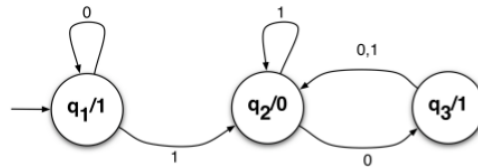
**Mealey :** "Output" à chaque transition.



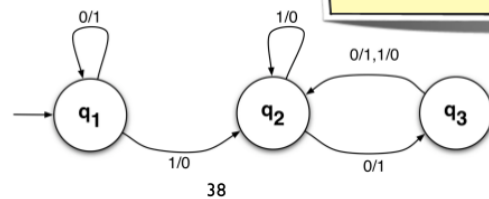
# Finite state transducers

Ce sont des machines qui produisent des symboles de sortie

Les machines de Moore



Les machines de Mealey



Les machines de Mealey sont aussi utilisées en cryptographie

Tous les états sont "ouverts", c-à-d accessibles. Bit 0,1 pour modéliser l'aspect "circuit logique" (Choix binaires).

## 3.2. Quelle machine est utilisée dans le système BLAST et comment est-elle utilisée ?

On emploie une machine de Mealey car on désire avoir un "output" à chaque transition.

.... voir exam 2010

Le système BLAST aligne les séquences en comparant des k-tuples avec un degré de similarité suffisant. Au lieu d'utiliser la technique de hashing comme FASTA, comme les tuples sont plus nombreux il est plus efficace d'utiliser une machine de Mealey. On la construit à partir de la séquence de départ. En lui soumettant les tuples d'une autre séquence, elle va sortir en output les positions des tuples similaires dans la séquence de départ.

## 3.3. Expliquez ce que sont les séquences homologues et pourquoi on les cherche dans des bases de données.

**Gènes orthologues :** Deux gènes de deux espèces différentes qui viennent du même gène héréditaire du dernier ancêtre commun et qui ont évolué par spécialisation et mutation, mais pas par duplication.

**Gènes paralogues :** Deux gènes de la même espèce qui ont été produits par la duplication mais qui ont divergé dans leur fonction. Généralement la duplication est suivie par des mutations indépendantes et des changements dans les séquences, résultant en un ensemble de gènes paralogues.

**Homologie :** Combinaison paralogues et orthologues. Deux séquences sont

homologues si elles possèdent un ancêtre commun et la fonction chez chacune des espèces n'est pas forcément la même. L'homologie implique une relation d'évolution et est à distinguer de la similarité.

Les séquences qui sont trouvées par cette méthode pourraient donner l'information sur la structure et la fonction de la protéine.

Les séquences homologues ont un ancêtre commun. Les fonctionnalités et les structures étant ce que l'évolution conserve le mieux, la comparaison de séquences homologues permettent d'obtenir des informations sur celles-ci.

### 3.4. Comment est-ce qu'on peut déterminer si la séquence trouvée est vraiment une homologue ?

Le système va toujours trouver quelque chose, même s'il n'y a aucune correspondance dans la base de données. Le score de l'alignement local permet de voir si les correspondances trouvées sont réelles, ou si ce sont de mauvais résultats. De plus, rien ne dit que ces homologues ont réellement un lien biologique entre eux.

Faire une estimation du score (score aléatoire ou SA) entre la séquence q et des séquences aléatoires (versions aléatoirement permutées de la séquence d). Si la différence entre le score de similarité (SS), obtenu pour q et d, et le SA est grande, la relation statistique entre q et d est significative.

Pour voir si on a une correspondance aléatoire, on mélange les acides aminés dans une des protéines, et on relance l'algorithme. C'est le logiciel RDF de Pearson et Lipman. Si le score qui en ressort est quasiment le même, c'est que la prédiction est mauvaise, aléatoire. Si le score est très différent, c'est que la correspondance est bonne. Le <z-score>, slide 24, indique l'importance du résultat :  $z \equiv \frac{SS - SA}{\sigma}$ , avec  $SA = \text{moyenne}(SA_i)$ ,  $SA_i$  est le score d'alignement entre la séquence de référence et une permutation de l'autre séquence. Ce score doit être le plus éloigné possible de zéro. Avec des tests, on a découvert qu'à partir de 3, la relation est intéressante. À partir de 6, la relation est probablement significative. À partir de 10, on est sûr que la relation est significative.

### 3.5. Quelles sont les grandes différences entre BLAST et FASTA ?

Dans la première étape, FASTA cherche des k-tuples dans chaque séquence d qui sont identiques. BLAST cherche des des k-tuples qui ont un score (similarité) au dessus un seuil T.

Dans la première étape, FASTA utilise un tableau de consultation. BLAST peut utiliser la même structure de données, mais ils ont découvert qu'un automate fini déterministe était plus efficace.

### 3.6. Quelles sont les 4 étapes de FASTA ? Expliquez brièvement

1. Chercher toutes les régions identiques entre deux séquences (des diagonaux avec une longueur k)
2. Prendre les m meilleurs diagonaux et calculer les scores pour les tuples en utilisant une matrice de substitution et une pénalité pour les espaces entre les régions.
3. Calculer l'alignement optimal en utilisant les régions initiales qui ont les plus hauts scores.

4. Utiliser l'algorithme de Smith-Waterman pour l'alignement entre la séquence q et les séquences d de D qui sont au sommet du classement.

### 3.7. Pourquoi est-ce qu'on a besoin d'une distribution de valeurs extrêmes pour déterminer la signification statistique d'une recherche dans une base de donnée ?

Voir question 3.8.

### 3.8. Pourquoi est-ce qu'un « extrem value distribution » [une distribution de valeurs extrêmes] représente bien les scores assignés aux séquences trouvées par BLAST dans une base de données ?

Quand on recherche des homologues dans des bases de données, on cherche toujours la meilleure séquence, dans chaque alignement, on prend toujours les alignements qui donne le score le plus grand. La distribution des max est une distribution de valeurs extrêmes et non une distribution gaussienne.

La distribution de valeurs extrêmes représente la distribution des scores qu'on peut atteindre quand on fait l'alignement entre deux séquences qui ne sont pas apparentées.

=>> La distribution montre la probabilité d'obtenir un certain score par hasard, or lorsqu'on effectue une recherche d'homologue dans une base de donnée, on aimerait typiquement que la recherche se concrétise en obtenant réellement l'homologue cherché.

Donc seul un petit nombre de séquences vont correspondre à ce critère de recherche ! => La majorité des séquences de la base de données ne sera pas homologue à la séquence considérée, on a une plus grande distribution pour des scores nuls.

## 4. Profils et l'alignement entre plusieurs séquences

### 4.1. Ce que sont des profils ? Quelles approches est-ce qu'on a discuté ? Pourquoi a-t-on besoin de d'eux ?

Un profil est une représentation d'un groupe de séquences qui facilite les alignements entre ces dernières et celles d'un autre groupe (donc utile pour les alignements multiples). Il enregistre les propriétés/caractéristiques d'une collection de séquences. Nous avons abordé ce sujet dans le chapitre 7. La construction d'un profil PSSM (Position-specific scoring matrix) était demandé dans le cadre du 3<sup>ème</sup> mini-projet. Le plus grand problème pour créer des profils est l'insuffisance du nombre de séquences dans un groupe et, par conséquent, l'absence de données de certains acides aminés dans plusieurs colonnes (d'où l'utilisation de pseudo-counts).

On peut ensuite (après avoir construit/téléchargé un profil) aligner une séquence à un profil. Un profil contient des scores et des pénalités et le plus grand problème se situe dans la manière dont les pénalités sont assignées (On peut utiliser les algorithmes Needleman-Wunsch ou Smith-Waterman pour aligner une séquence à un profil).

### 4.2. Pourquoi a-t-on besoin des *pseudocounts* ? Donnez un exemple.

Lorsque il n'y a pas d'information pour un aa dans une colonne d'un profil.

Les pseudocounts sont des constantes qu'on ajoute aux valeurs dans un profil. Lors des

calculs de la valeur d'une case dans cette matrice de profil, on utilise le logarithme. Ce dernier étant égal à -infini pour les valeurs égales à 0, on va instaurer les pseudocounts pour éviter ce cas. Ils donnent également de l'information antérieure sur les acides aminés. (= connaissance qu'on possède concernant le système avant l'introduction des données.

#### 4.3. Expliquez la différence entre des systèmes d'alignement itératifs et progressifs.

Voir question 4.4.

#### 4.4. Expliquez les différences entre les méthodes itératives et progressives utilisées pour l'alignement de plusieurs séquences.

Le système itératif construit une solution de façon aléatoire. Cette dernière est donnée en paramètre à une fonction d'évaluation. S'il y a convergence avec une solution optimale, l'itération s'arrête, on a la meilleure solution. Sinon, on améliore la solution (avec une méthode aléatoire ou stochastique (SAGA) depuis la solution obtenue). On arrive au final (quand il y a convergence) à une approximation de l'alignement optimal.

L'alignement progressif (CLUSTAL) est une approche heuristique pour aligner plusieurs séquences. Il n'y a cependant aucune garantie qu'on trouve l'alignement optimal. L'heuristique est une méthode de calcul qui fournit relativement rapidement une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation. Les erreurs des premiers alignements perdurent dans les alignements suivants. Des algorithmes stochastiques peuvent résoudre ces soucis puisqu'ils peuvent s'échapper de solutions localement optimales.

3 étapes (CLUSTAL):

- Calculez une matrice de distances entre les paires de séquences
- Construisez un arbre phylogénétique en utilisant cette matrice
- Utilisez cet arbre pour aligner chacune des séquences

#### 4.5. Expliquez l'algorithme d'entraînement de Viterbi en utilisant vos propres mots. Pourquoi est-ce qu'on a besoin de cet algorithme ?

Cet algorithme a pour but de trouver la séquence d'états la plus probable ayant produit la séquence mesurée = problème de décodage. Dans le cas de l'entraînement, il faut évaluer les matrices de transitions, d'observations et d'états initiaux ( $A$ ,  $B$ ,  $\pi$ ) à partir d'une série de séquences d'observations quand les séquences d'états correspondantes sont inconnues. L'utilisation de l'algorithme de Viterbi permet de générer les séquences d'états correspondantes et à partir de là calculer  $A$ ,  $B$  et  $\pi$ .

Le fonctionnement de l'algorithme est ressemblant à un algorithme forward. Pour chaque instant  $t$ , pour chaque état, il évalue la transition la plus susceptible d'avoir donné un certain événement et l'enregistre. À la fin, il reconstruit la séquence d'état en "remontant" la liste des maximums retenus.

#### 4.6. Dans le système SAGA, les opérateurs sont utilisés dynamiquement. Expliquez le système et expliquez pourquoi cela a du sens ?

L'algorithme SAGA est un algorithme stochastique et utilise des algorithmes génétiques.[...]

Au début, chaque opérateur (22) a la même chance d'être utilisée. Mais au fil de l'évolution, les opérateurs améliorant le mieux le score sont les plus utilisés. On perfectionne l'algorithme. Quand un opérateur n'améliore plus rien, il est oublié. On arrête l'alignement général quand il ne reste plus d'opérateur, on ne sait plus augmenter la qualité de la

solution. Cela imite les mécanismes de l'évolution.

#### **4.7. Expliquez le problème de décodage dans le contexte des modèles markoviens cachés. Quel algorithme est utilisé pour résoudre ce problème ?**

Les modèles markoviens cachés servent pour la détection des motifs. Ils vont permettre de voir, parmi des dizaines de séquences, quelles sont les choses qui se ressemblent.

Le problème du décodage est de trouver quelle est la séquence d'états la plus probable qui aurait produit une séquence donnée.

Dans la bioinformatique, on a des match, des insertions et des suppressions. En comparant une protéine avec une famille, on aimerait retrouver ses états de match, donc retrouver ce qui correspond entre cette protéine et la famille.

On utilise l'algorithme de Viterbi.

## **5. Arbres phylogénétiques**

### **5.1. Quelle est la différence entre des arbres ultra-métriques et additifs ?**

Sur un arbre ultra métrique le taux de mutation est constant le long d'une branche. L'arbre-ultramétrique prend en compte une échelle supplémentaire : l'horloge moléculaire.

### **5.2. Pourquoi est-ce qu'on préfère utiliser des orthologues pour la construction des arbres d'espèces ?**

Car les séquences orthologues permettent d'étudier les relations évolutives entre les différentes espèces. De cette manière, on peut créer un arbre d'espèce.

Ce sont les gènes orthologues qui désignent une spéciation.

### **5.3. Quels sont les trois types d'arbre phylogénétique ? Expliquez leurs différences.**

**Cladogram :** montre la ligne de la descente du taxa. La taille des lignes n'a aucune signification, on ne sait pas si des spéciations sont apparues plus tôt ou plus tard que d'autres. Seulement la topologie est importante.

Rarement utilisé pour l'analyse des séquences.

**Arbre additif :** montre les lignes de descentes. Les longueurs des branches donnent de l'information évolutive quantitative, qui peuvent être proportionnelle au nombre de mutations entre les noeuds. Mais on ne voit pas si les changements sont survenus rapidement ou pas car il n'y a pas d'information sur le taux de mutations.

La distance évolutive totale entre les taxa est donnée par la somme de toutes les longueurs des branches.

Les arbres additifs sont utilisés le plus souvent

**Arbre ultramétrique :** comme un arbre additif et en même temps le taux de mutation est constant le long de chaque branche de l'arbre. Cette propriété est appelée l'horloge moléculaire. On voit ainsi exactement quand les

mutations se passent.

Les arbres ultramétriques ont toujours une racine et un axe de l'arbre est directement proportionnel du temps.

Note : Les données des séquences ne sont souvent pas conformes à cette horloge moléculaire en raison de différences dans les taux de mutation causée par des différences dans la pression évolutive (on fait comme si le taux de mutation était constant au fil du temps).

#### **5.4. Qu'est-ce qu'une mutation non synonyme ? Sont-elles plus fréquentes que les mutations synonymes ? Expliquez.**

**Mutation synonyme** : Mutation qui ne change pas l'acide aminé encodé.

**Mutation non-synonyme** : Mutation qui change l'acide aminé encodé.

Le plupart des mutations dans la troisième position du codon produit des synonymes.

Des mutations non-synonyme change l'acide aminé et par conséquence ils peuvent changer la structure ou la fonction d'une protéine. Les mutations synonymes ne peuvent pas être effacées par la sélection comme elle sont sans effets.

Les mutations qui introduisent des changements dans des protéines peuvent être maintenues et répandues ou sont rapidement perdues à cause de sélection naturelle.

Par conséquence, le taux de mutation dans la troisième position sera plus élevé que dans les autres positions.

#### **5.5. Expliquez l'algorithme de Fitch-Margoliash pour la construction des arbres phylogénétiques. Pourquoi est-ce que cette méthode donne parfois de mauvais résultats ?**

L'arbre le plus simple est composé de 3 éléments, connectés ensemble par un élément central  $x$ . La distance entre  $A$  et  $x$  est  $b_1$ , entre  $B$  et  $x$  est  $b_2$ , etc. La distance entre  $A$  et  $C$  est donc  $b_1 + b_2$ . Quand on a les matrices avec les distances entre les acides aminés, on essaie simplement de retrouver les points  $x$  centraux. L'algorithme va mettre ensemble les séquences les plus proches, puis va calculer les points centraux et va trouver quelles séquences sont les plus proches de ces points.

On décompose progressivement le groupe des éléments restants en prenant à chaque fois la plus petite distance.

Cet algorithme donne des poids aux branches de l'arbre. Quand les taux d'évolution entre les branchements sont différents, les éléments avec la plus petite distance ne sont pas nécessairement les éléments la plus proche. Cela crée des distances négatives dans l'arbre. En faisant la différence entre les poids finaux trouvés et les distances entre chaque séquence qu'on avait au début, on trouve l'erreur introduite par l'algorithme.

#### **5.6. Pourquoi est-ce que la distance $p$ entre deux séquences ne donne pas un résultat satisfaisant ? Quelles solutions étaient proposées pour mieux estimer les distances entre des séquences ? Donnez 3 exemples.**

**P-distance** (différence d'alignement fractionnaire) : Mesure pour la distance évolutive qui estime le nombre de mutations entre l'ancêtre commun et les séquences.

$P = D / L$  avec  $D$  = Nombre de substitutions dans les positions alignées

$L$  = Nombre de positions alignées entre les deux séquences

- Ne prend pas en compte que plusieurs mutations peuvent se produire à la même position
  - ◆ **Correction de Poisson** :  $d_p = -\ln(1-p)$
- Ne prend pas en compte que le taux de mutation peut dépendre de la position dans la séquence
  - ◆ **Correction de Gama** :  $d_{Gama} = a[(1-p)^{-1/a} - 1]$
- Ne prend pas en compte l'information sur les caractéristiques biochimiques des éléments qui construisent la séquence
  - ◆ **Modèle de Jukes-Cantor** : Toutes les positions sont indépendants et les taux de mutation ( $= \alpha$ ) sont identiques pour chaque base (nucléotides ou acides aminés). Modèle simple, incorrect mais très utile. Utilise une matrice.

	A	C	G	T
A	$-3\alpha$	$\alpha$	$\alpha$	$\alpha$
C	$\alpha$	$-3\alpha$	$\alpha$	$\alpha$
G	$\alpha$	$\alpha$	$-3\alpha$	$\alpha$
T	$\alpha$	$\alpha$	$\alpha$	$-3\alpha$

$$d_{JC} = - (3/4) \ln [1 - (4/3)p]$$

Fonction similaire à la correction de Poisson. Combiné à Gamma pour des taux de mutations différents selon la position :

$$d_{JC+\Gamma} = (3/4)a [(1-(4/3)p)^{-1/a} - 1]$$

- Pas de distinction entre les fréquences relatives des transitions ( $\alpha$ ) et des transversions ( $\beta$ ).

◆ **Modèle de Kimura** :

	A	C	G	T
A	$-2\beta-\alpha$	$\beta$	$\alpha$	$\beta$
C	$\beta$	$-2\beta-\alpha$	$\beta$	$\alpha$
G	$\alpha$	$\beta$	$-2\beta-\alpha$	$\beta$
T	$\beta$	$\alpha$	$\beta$	$-2\beta-\alpha$

Produisant une nouvelle fonction de distance, dans laquelle  $P$  ( $Q$ ) est la fraction observée de mutations de transition (transversion) entre des bases

Observée = extrait des séquences alignées









$$d_{K2P} = - (1/2) \ln [1-2P-Q] - (1/4) \ln [1-2Q]$$

INFO-F-208 - Introduction à la bioinformatique

Résumé modifiable :

<https://docs.google.com/document/d/1IEQbfN7VJ3SQAdC8fXSkkJweZHhVUPC7RFRou9-AA>

Rodrigue Van Driane



Commentaires

Partager

Nyan Cat anonyme

⌵

^ LAWL