Correction et assemblage de lectures issues de technologies de séquençage de 3e génération

Jean-François Gibrat

AG département MIA

22 mai 2019

Mathématiques et Informatique Appliquées
MaIAGE
du Génome à l'Environnement

Caractéristiques des technologies de 2e et 3e générations

- Deuxième génération
  - Très grand nombre de lectures > 1 milliard
  - Lectures courtes 200-400 pb
  - Faible taux d'erreur < 0.1% surtout des substitutions
- Troisième génération
  - Peu de lectures ~100 000
  - Distribution de longueurs (min 500 pb, max 100 kbp, médiane 15 kbp)
  - Taux d'erreur important ~15% dont 7% insertions et 4% délétions

Caractéristiques des technologies de 2e et 3e générations

- Deuxième génération
  - Très grand nombre de lectures > 1 milliard
  - Lectures courtes 200-400 pb ← répétitions
  - Faible taux d'erreur < 0.1% surtout des substitutions
- Troisième génération
  - Peu de lectures ~100 000
  - Distribution de longueurs (min 500 pb, max 50 kbp, médiane 12 kbp)
  - Taux d'erreur important ~15% dont 7 insertions et 3 délétions

Caractéristiques des technologies de 2e et 3e générations

- Deuxième génération
  - Très grand nombre de lectures > 1 milliard
  - Lectures courtes 200-400 pb
  - Faible taux d'erreur < 1% surtout des substitutions
- Troisième génération
  - Peu de lectures ~100 000
  - Distribution de longueurs (min 500 pb, max 50 kbp, médiane 12 kbp)
  - Taux d'erreur important ~15% dont 7 insertions et 3 délétions ← en particulier indels

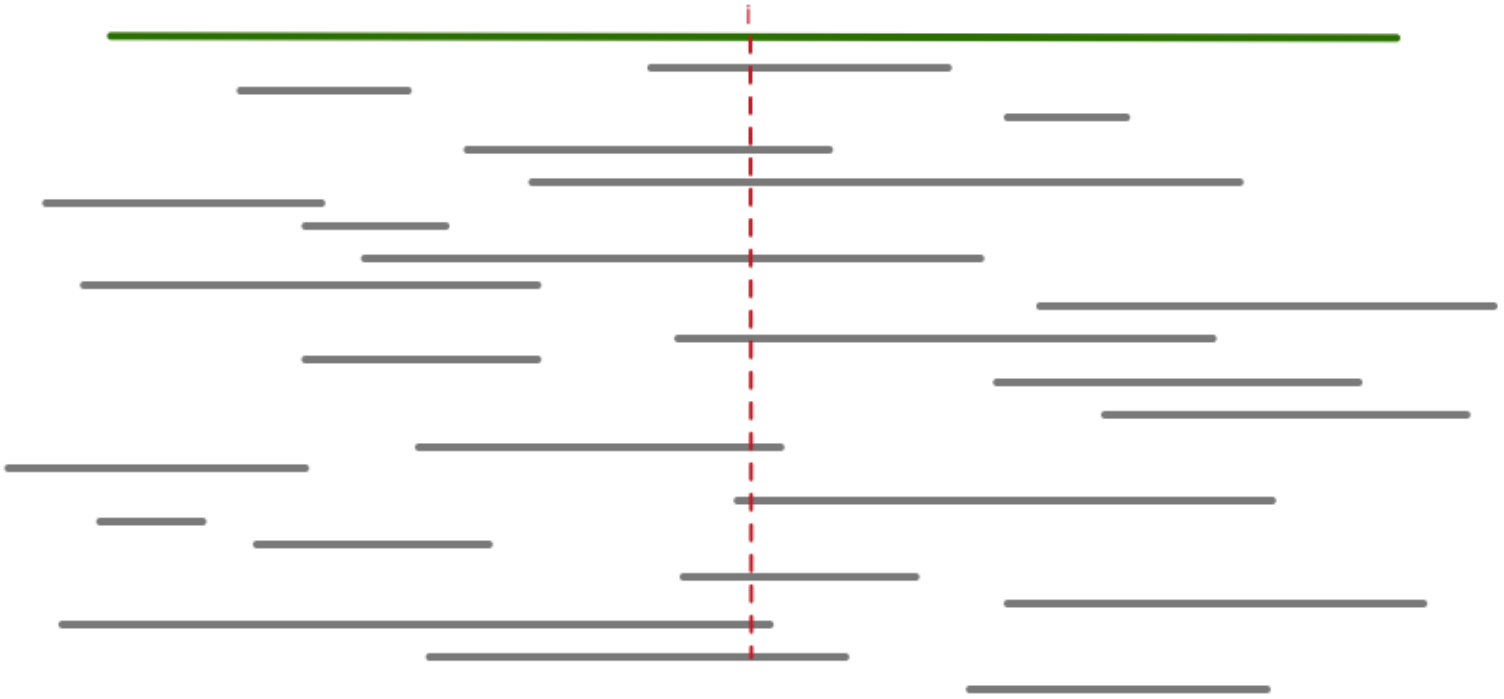
L'algorithme de programmation dynamique

- Principal problème des lectures 3e G : indels
- *Une seule méthode exacte pour aligner des séquences avec des indels : la programmation dynamique*
- *Algorithme est $O(L_1L_2)$ en temps et mémoire*
- *Peu efficace dès que les séquences sont longues*
- *Hors de question de faire un alignement de lectures sur le génome humain avec prog. dyn*.

L'algorithme de programmation dynamique

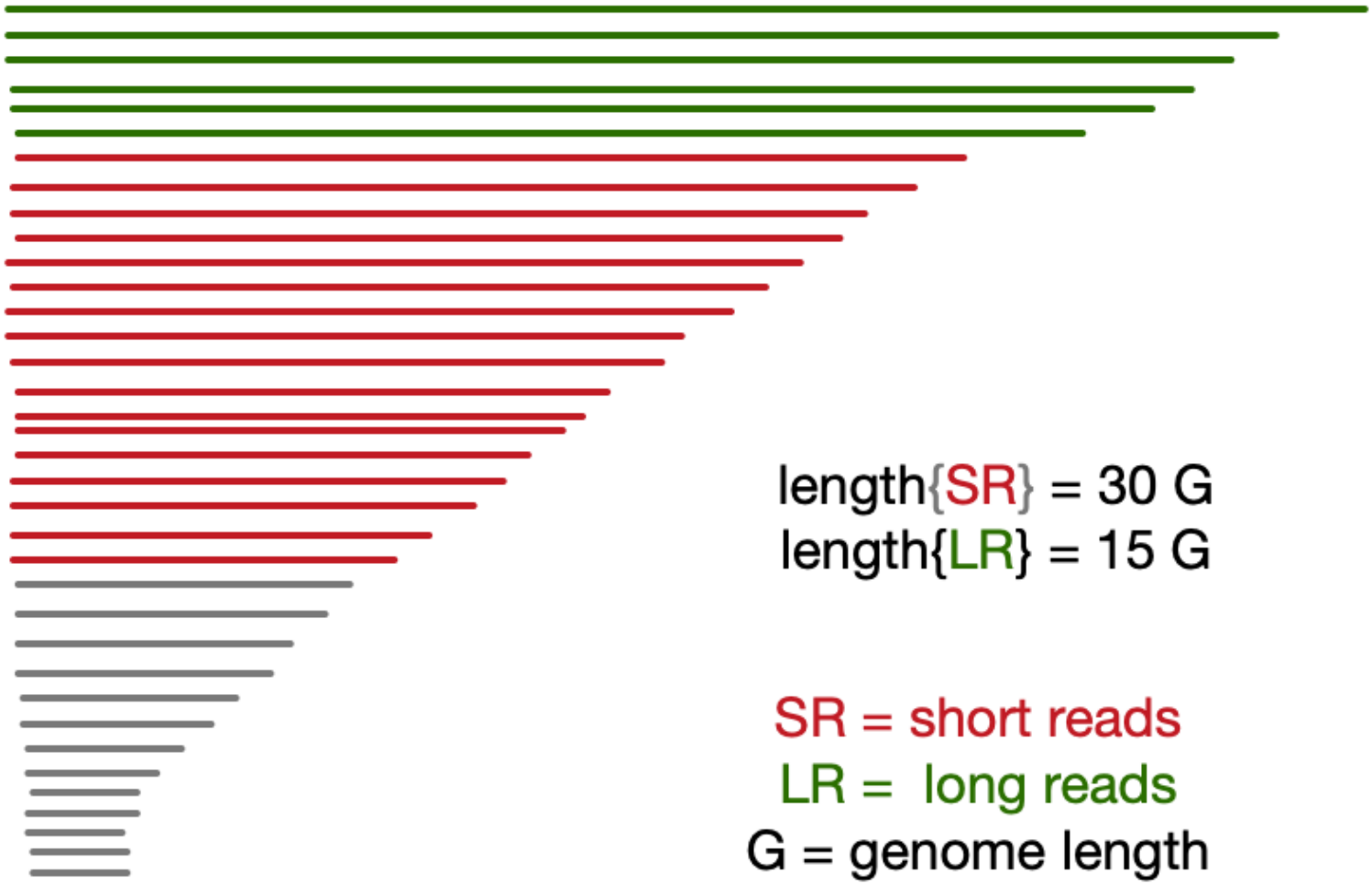
- Principal problème des lectures 3e G : indels
- Une seule méthode exacte pour aligner des séquences avec des indels : la programmation dynamique
- Algorithme est $O(L_1 L_2)$ en temps et mémoire
- Peu efficace dès que les séquences sont longues
- Hors de question de faire un alignement de lectures sur le génome humain avec prog. dyn.

Principe général de la méthode



En moyenne, chaque nucléotide, *i*, est couvert par K SR

Alembic



length{SR} = 30 G
length{LR} = 15 G

SR = short reads
LR = long reads
G = genome length

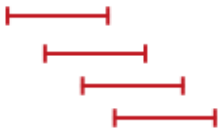
Implémentation de la méthode

- 1re étape : alignement des SR sur les LR
- 2e étape : alignement des LR entre eux
- 3e étape :
  - Recueil des données nécessaires pour l'assemblage
  - Vérification de la cohérence des alignements LR et correction des erreurs restantes éventuellement
- 4e étape : assemblage des LR

Les 3 premières étapes sont destinées à éliminer les erreurs

1re étape: alignement des SR sur un LR

- AGCTGTCAGGGACTAAGGTCCATCTCTCAGTTTACG LR n° 10



HASH[AGCT] = position 1 LR n° 10, position x LR n° Y, etc

HASH[GCTG] = position 2 LR n° 10, etc.

HASH[CTGT] = position 3 LR n° 10, etc.

- GACTAAGAGCTGTTTAGGTC SR n° z

Alignement des k-mers dans les 2 sens !

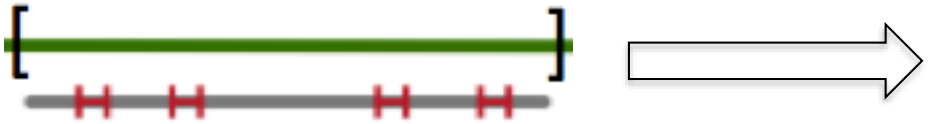
- 

Détermination des SR s'alignant sur LR

- Si *N* k-mers s'alignent aux *bonnes positions,* on sélectionne le SR correspondant.
  - $N \geq E[N(w)] - 3\,\sigma_{N(w)}$ ⇐ détermination empirique ou exacte

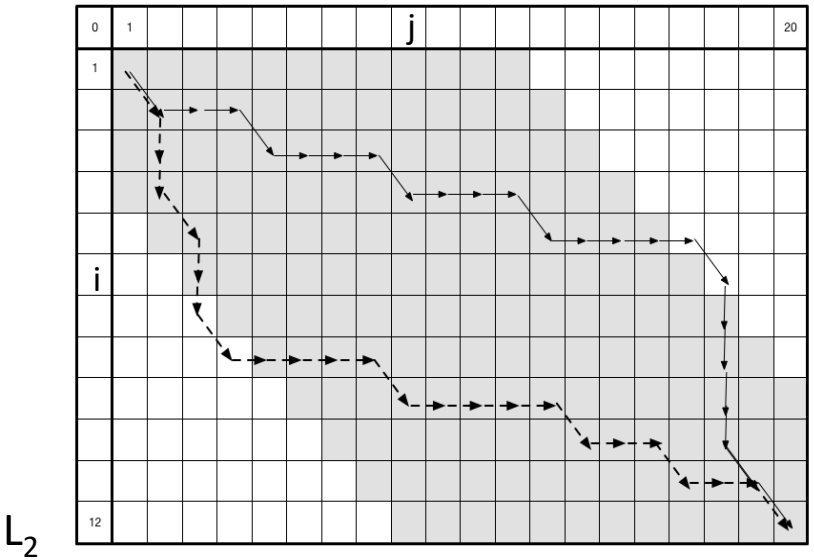
- Calcul récursif du plus long chemin entre k-mers localisés aux *bonnes positions :*
  1. les k-mers sont en ordre croissant
  2. les k-mers s'alignent dans le même sens
  3. les distances entre les k-mers sont compatibles
     - $E[\Delta L] = 0 \qquad V[\Delta L] = 2L[p_u + p_s + 4p_i - (p_u + p_s + 2p_i)^2]$

- On extrait la partie du LR qui s'aligne sur le SR et on aligne la paire LR-SR *exactement* avec l'algorithme de programmation dynamique *dans une bande*.



⇒ Algo. prog. dyn. *dans une bande*

Algorithme de programmation dynamique dans une bande



$$M[i,j] = \max \begin{cases} M[i-1,j] + gap \\ M[i-1,j-1] + c(s[i],t[i]) \\ M[i,j-1] + gap \end{cases}$$

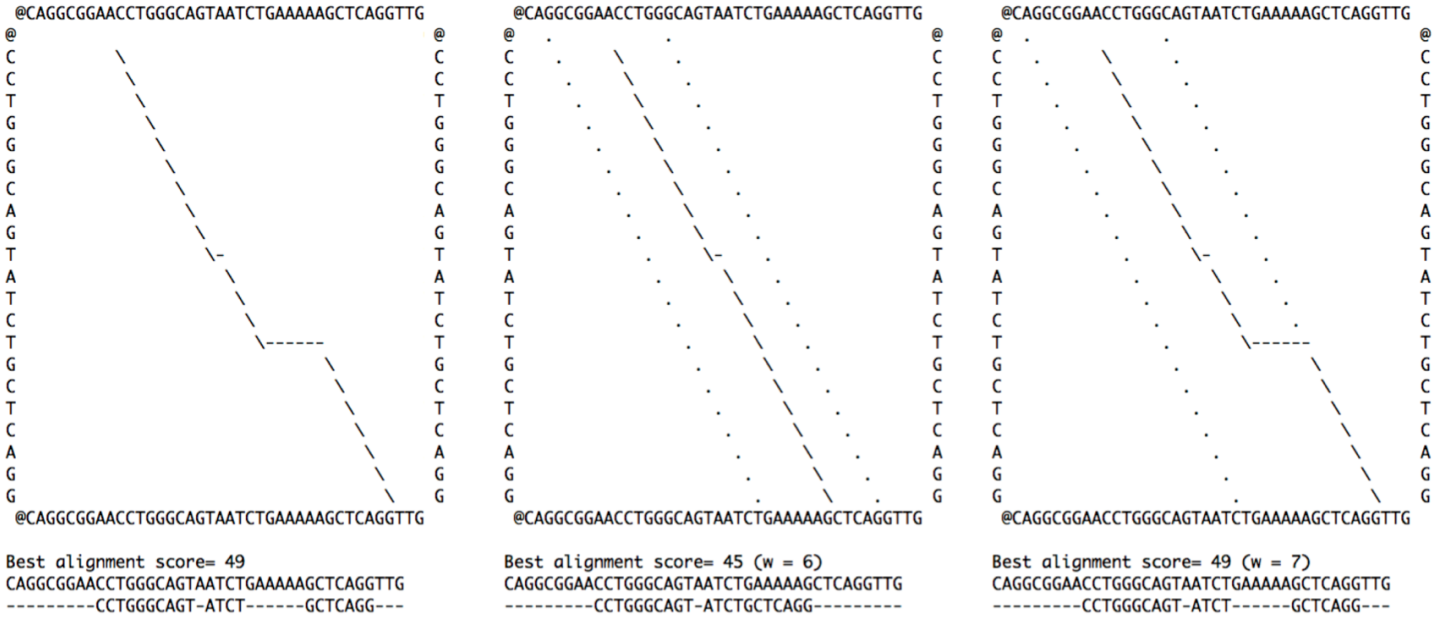
```
AT-AGTA      ATAGTA
ATT-GTA      ATTGTA
```

- Question : Etant donné les taux d'erreur, quelle est la probabilité que le chemin quitte la bande quand on aligne des lectures de longueur L dû à une accumulation aléatoire de gaps dans une direction particulière ?
- On peut modéliser ce problème comme une marche aléatoire 1D

- $E(\Delta d) = 0$ où Δd est la différence entre les distances parcourues dans les 2 directions
- $V(\Delta d) = 2Lp$ où $p = 2 \times (p_i + p_d - p_d \times p_d - p_i \times p_i - O(p_i, p_d))$

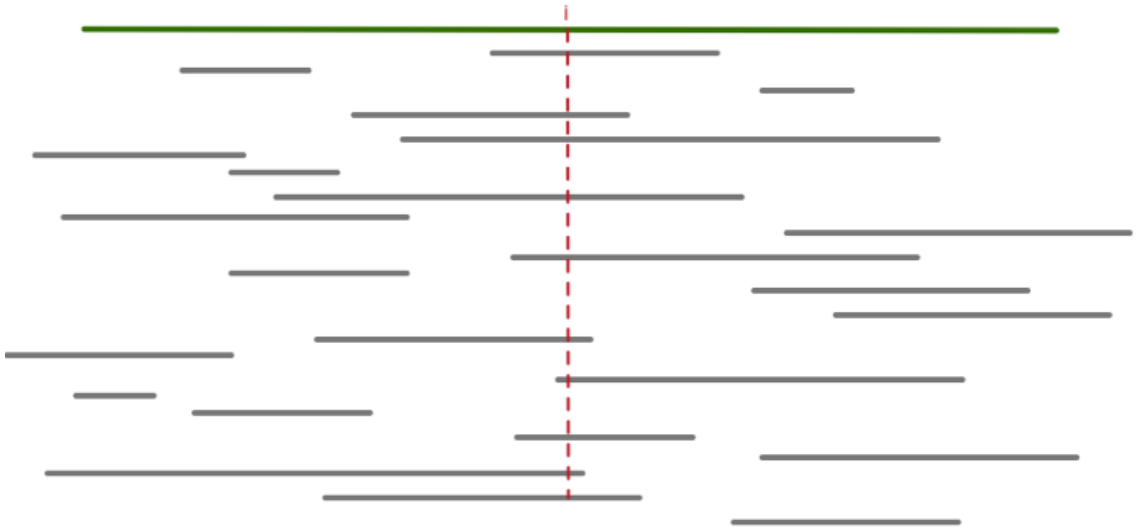
Algorithme de programmation dynamique dans une bande

- $\sigma_{\Delta d} = c\sqrt{L}$
- La largeur de la bande est *2w+1*



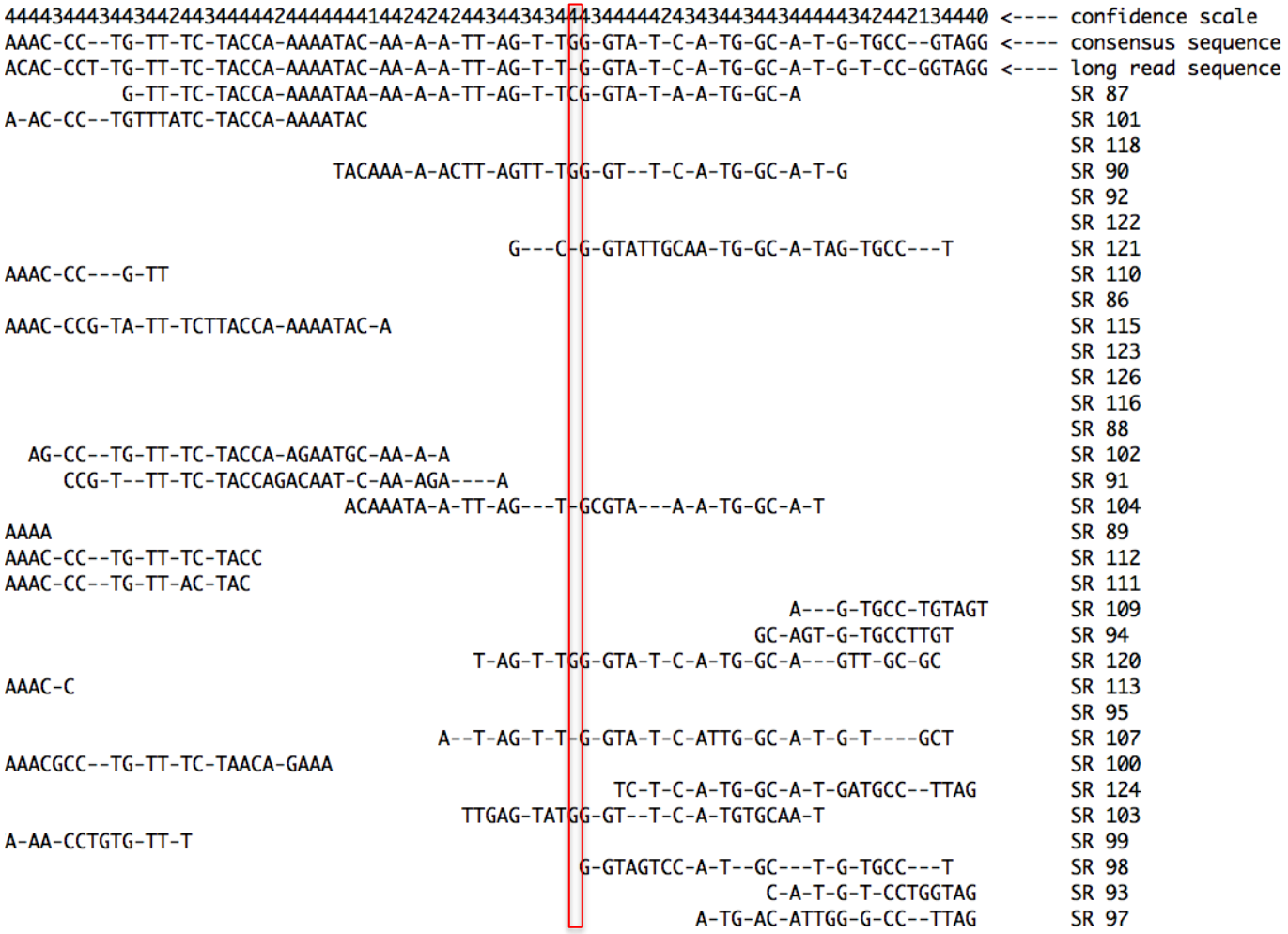
- Si on choisit $w = 3\sigma_{\Delta d}$, il y a <0.3% de chance que l'alignement quitte la bande
- On obtient donc un algorithme qui est $O(L^{3/2})$ en temps et mémoire.

Alignement multiple de séquences *pivot*



- On crée un alignement multiple de séquence *pivot* en alignant graduellement par PD dans une bande tous les SR connectés.
- Une fois qu'un SR est aligné, il devient solidaire du LR.

Alignement multiple de séquences pivot



Déterminer la séquence consensus

- On analyse colonne par colonne le pseudo alignement multiple pour essayer de déterminer une séquence consensus pour le LR.
- L'analyse suppose qu'il n'y a pas d'erreur dans l'alignement !

- $P(S\ |\mathrm{S}_0 col) = P(\mathrm{S}_0 col|\ S)\frac{P(S)}{P(\mathrm{S}_0 col)}$ $\quad \mathrm{S}_0, S \in$ {A, G, C, T, -}

Probabilités a priori : $\boldsymbol{P(S)}$

- Pour créer un gap, il suffit d'une insertion dans les SR ou une délétion dans LR
- $N_{gaps} = L\,[1 - (1 - p_i)^N + p_d]$ $\quad L$ longueur LR
- $N_{symb} = L + N_{gaps}$
- $P(-) = N_{gaps} / N_{symb}$ quand N est grand, $P(-)$ tend vers 0.5^+ (dépend de p_d)

- $P(l) = \big(1 - P(-)\big)P_o\ (l)$ où P_o est la fréquence observée du nucléotide l dans les lectures

Déterminer la séquence consensus

Vraisemblance de la colonne et de S_0 : $\boldsymbol{P(S_0 col \mid S)}$

1. Un nucléotide est observé dans LR
   - La lettre S_o observée dans LR n'a pas été modifiée: $p_1 = p_u + p_i$
     - $P(S_o col \mid S) = P(S_o \mid S) \times P(col \mid SS_o) = p_1 \times \mathcal{M}(\boldsymbol{counts}, \boldsymbol{P})$
   - La lettre S_o observée dans LR est le résultat d'une substitution de l_i : p_s
     - $P(S_o col \mid S) = P(S_o \mid S) \times P(col \mid SS_o) = p_s \times \mathcal{M}(\boldsymbol{counts}, \boldsymbol{P})$
   - La lettre S_o observée dans LR est le résultat d'une insertion: p_i
     - $P(S_o col \mid -) = P(S_o \mid S) \times P(col \mid SS_o) = p_i \times \mathcal{M}(\boldsymbol{counts}, \boldsymbol{P})$

2. Un gap est observé dans LR
   - Le gap résulte de l'insertion d'au moins un nucléotide dans la colonne
     - $P(-col \mid S) = \mathcal{M}(\boldsymbol{counts}, \boldsymbol{P})$
   - Le gap résulte d'une délétion dans LR
     - $P(-col \mid S) = p_d \times \mathcal{M}(\boldsymbol{counts}, \boldsymbol{P})$

Résultats 1re étape

JEU DE TEST

- Fragment d'ADN 10 000 nucléotides, couverture 15 pour LR, 30 pour SR.
- On génère 249 LR (155K nt, taille 743-528) et 1196 SR (309K nt, taille 352-209) avec taux d'erreur 15% (M=3%, I=7.5%, D=4.5%) sur les 2 brins d'ADN

RESULTATS BRUTS

- Erreur moyenne 0.83% ou 8.30 ‰
- Problème avec les délétions dans LR

CORRECTION

- Ajout d'une étape pour corriger l'AMS pivot obtenu précédemment
  - Alignement multiple de séquence *progressif* par morceaux
  - Recalcul de la séquence consensus

RESULTATS CORRECTION

- Erreur moyenne 2.75‰

2e étape : alignement des LR entre eux

Même principe que pour la 1re étape en remplaçant les SR par les LR eux-mêmes

1. On indexe les LR_i i = 1…. N_{LR}
2. On aligne les LR_j sur l'un d'entre eux LR_i ($i \neq j$)
3. On excise les zones correspondantes
4. On aligne par PD (AMS pivot) ces zones (fragments)
5. Eventuellement, on corrige l'AMS pivot résultant

Résultats : erreur moyenne 0.03 ‰. Cela représente 2 délétions dans 2 LR

3e étape : vérification cohérence des AMS entre LR

Deux objectifs:

1. Obtenir des données pour l'assemblage des LR
2. En profiter pour corriger les dernières erreurs des LR

Correction des erreurs

1. On réaligne une fois de plus les LR corrigés entre eux (*mais sans prog. dyn.)*
2. On vérifie que les alignements sont cohérents (pour chaque paire {LR_i , LR_j} on vérifie que le chevauchement est le même quand LR_i est le pivot et quand LR_j est le pivot).
3. Si nécessaire, on corrige les LR pour lesquels les alignements ne sont pas cohérents.
4. Résultats : il n'y a plus d'erreur dans les LR

Obtention de données pour assemblage

- Pour chaque LR on a une liste des autres LR chevauchants et la valeur du chevauchement en nombre de nucléotides.

4e étape : assemblage des LR corrigés

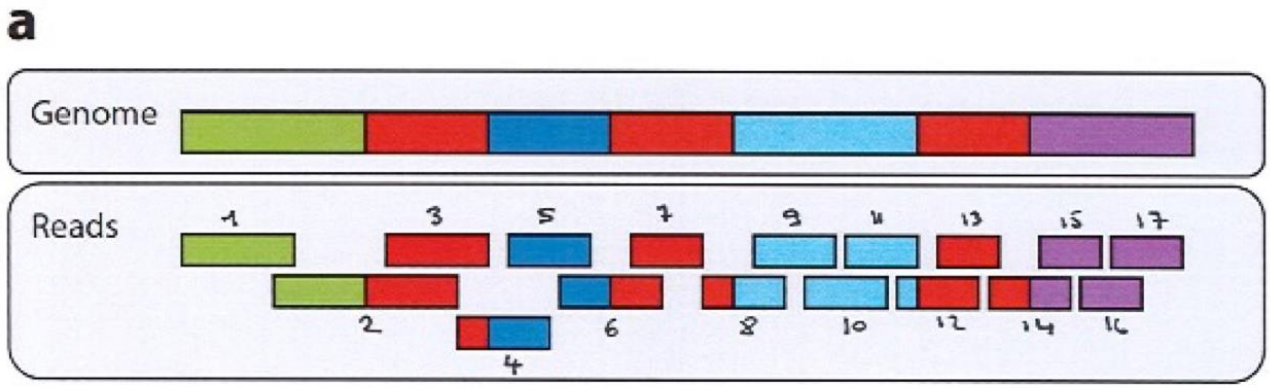
1. On part de toutes les paires de LR qui se chevauchent
2. On construit un graphe :
   - sommets = LR, arêtes si les LR se recouvrent
3. On calcule les composantes connexes (1 CC = 1 contig) O(V+E)

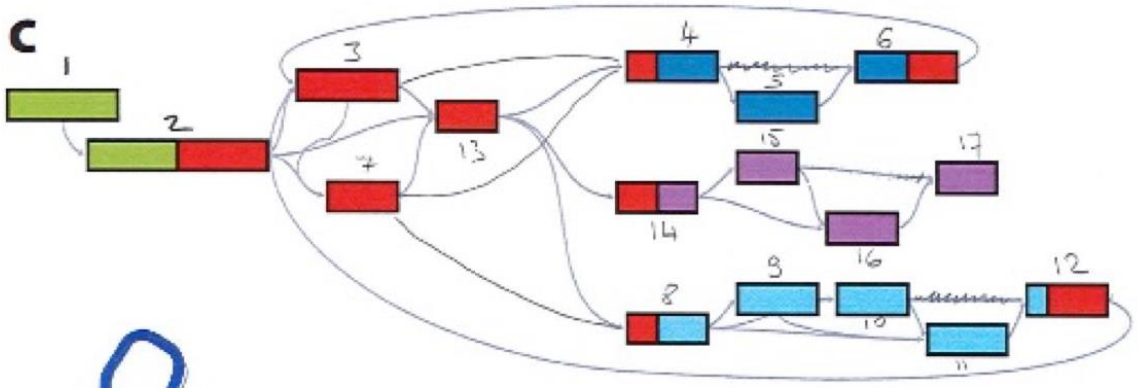
4. Pour chaque composante connexe on construit un arbre UPGMA O(V)

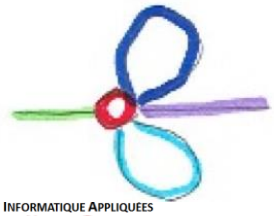
ou

4. On assemble les séquences le long de l'arbre de recouvrement minimal au fur et à mesure du calcul

Répétitions







Simpson & Pop, Ann. Rev. Genomics Hum. Genet., 2015

MATHEMATIQUES ET INFORMATIQUE APPLIQUEES
MaIAGE
DU GÉNOME À L'ENVIRONNEMENT

MERCI POUR VOTRE ATTENTION