

Stage de Master IGIS ITA

Bioinformatique, découverte de motifs entre des ensembles de fragments d'ADN

Gabriel Toublanc

3 juillet 2017

Université de Rouen, U.F.R des Sciences et Techniques de Saint-Étienne-du-Rouvray,
LITIS EquipeTIBS

Encadrants : Thierry Lecroq et Arnaud Lefebvre



1. Introduction
2. K-mers et k-mers espacés
3. Implémentations
4. Résultats obtenus
5. Conclusion

1. Introduction
2. K-mers et k-mers espacés
3. Implémentations
4. Résultats obtenus
5. Conclusion

1. Introduction
2. K-mers et k-mers espacés
3. Implémentations
4. Résultats obtenus
5. Conclusion

1. Introduction
2. K-mers et k-mers espacés
3. Implémentations
4. Résultats obtenus
5. Conclusion

1. Introduction
2. K-mers et k-mers espacés
3. Implémentations
4. Résultats obtenus
5. Conclusion

Introduction

Le séquençage ADN



- Lectures courtes (depuis 2005, 20 à 300 nucléotides, Illumina, Roche, ...)

Le séquençage ADN



- Lectures courtes (depuis 2005, 20 à 300 nucléotides, Illumina, Roche, ...)
- Lectures longues (depuis 2010, 3k à 20k nucléotides, Oxford Nanopore, Pacific Biosciences)

Alignement et assemblage

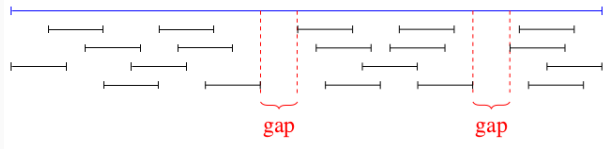


Figure 1 – Alignement sur les lectures courtes

Alignement et assemblage

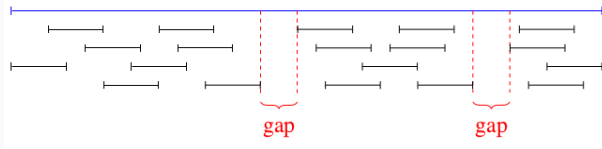


Figure 1 – Alignement sur les lectures courtes

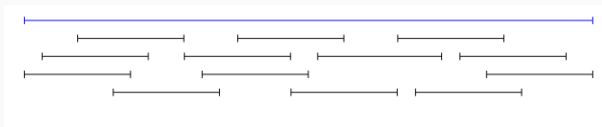


Figure 2 – Alignement sur les lectures longues

Alignement et assemblage

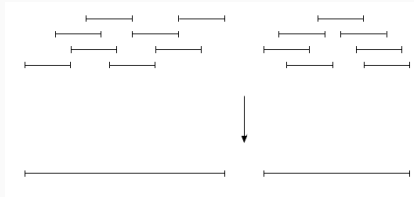


Figure 3 – Assemblage sur les lectures courtes

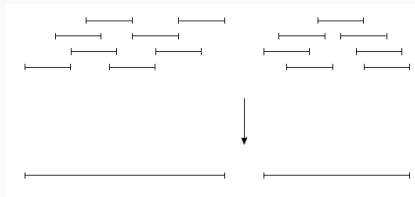


Figure 3 – Assemblage sur les lectures courtes

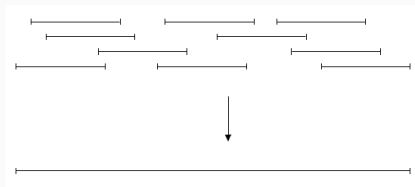


Figure 4 – Assemblage sur les lectures longues

Lectures courtes : problème de couverture du génome

Lectures longues : problème de taux d'erreur de lecture

- Oxford Nanopore : 30% d'erreurs
- Pacific BioSciences : 15% d'erreurs

Une solution : corriger les lectures longues avec les lectures courtes ($< 1\%$ d'erreur, voir **HG-CoLoR**¹).

Notre solution : tenter de corriger les lectures longues directement.

1. Pierre Morisse, Thierry Lecroq and Arnaud Lefebvre[1]

Lectures courtes : problème de couverture du génome

Lectures longues : problème de taux d'erreur de lecture

- Oxford Nanopore : 30% d'erreurs
- Pacific BioSciences : 15% d'erreurs

Une solution : corriger les lectures longues avec les lectures courtes ($< 1\%$ d'erreur, voir **HG-CoLoR**¹).

Notre solution : tenter de corriger les lectures longues directement.

1. Pierre Morisse, Thierry Lecroq and Arnaud Lefebvre[1]

Lectures courtes : problème de couverture du génome

Lectures longues : problème de taux d'erreur de lecture

- Oxford Nanopore : 30% d'erreurs
- Pacific BioSciences : 15% d'erreurs

Une solution : corriger les lectures longues avec les lectures courtes ($< 1\%$ d'erreur, voir **HG-CoLoR**¹).

Notre solution : tenter de corriger les lectures longues directement.

1. Pierre Morisse, Thierry Lecroq and Arnaud Lefebvre[1]

Lectures courtes : problème de couverture du génome

Lectures longues : problème de taux d'erreur de lecture

- Oxford Nanopore : 30% d'erreurs
- Pacific BioSciences : 15% d'erreurs

Une solution : corriger les lectures longues avec les lectures courtes ($< 1\%$ d'erreur, voir **HG-CoLoR**¹).

Notre solution : tenter de corriger les lectures longues directement.

1. Pierre Morisse, Thierry Lecroq and Arnaud Lefebvre[1]

Lectures courtes : problème de couverture du génome

Lectures longues : problème de taux d'erreur de lecture

- Oxford Nanopore : 30% d'erreurs
- Pacific BioSciences : 15% d'erreurs

Une solution : corriger les lectures longues avec les lectures courtes ($< 1\%$ d'erreur, voir **HG-CoLoR**¹).

Notre solution : tenter de corriger les lectures longues directement.

1. Pierre Morisse, Thierry Lecroq and Arnaud Lefebvre[1]

Lectures courtes : problème de couverture du génome

Lectures longues : problème de taux d'erreur de lecture

- Oxford Nanopore : 30% d'erreurs
- Pacific BioSciences : 15% d'erreurs

Une solution : corriger les lectures longues avec les lectures courtes ($< 1\%$ d'erreur, voir **HG-CoLoR**¹).

Notre solution : tenter de corriger les lectures longues directement.

1. Pierre Morisse, Thierry Lecroq and Arnaud Lefebvre[1]

Lectures courtes : problème de couverture du génome

Lectures longues : problème de taux d'erreur de lecture

- Oxford Nanopore : 30% d'erreurs
- Pacific BioSciences : 15% d'erreurs

Une solution : corriger les lectures longues avec les lectures courtes ($< 1\%$ d'erreur, voir **HG-CoLoR**¹).

Notre solution : tenter de corriger les lectures longues directement.

1. Pierre Morisse, Thierry Lecroq and Arnaud Lefebvre[1]

K-mers et *k*-mers espacés

Extraction des k -mers

Les k -mers sont des facteurs de longueur k de séquences d'ADN.

Ex : Avec la séquence $s = AACCGGTT$ de longueur L , on obtient les k -mers de longueur 6 (6-mers) suivants :

k_1 : A A C C G G T T

k_2 : A A C C G G T T

k_3 : A A C C G G T T

Extraction des k -mers

Les k -mers sont des facteurs de longueur k de séquences d'ADN.

Ex : Avec la séquence $s = AACCGGTT$ de longueur L , on obtient les k -mers de longueur 6 (6-mers) suivants :

k_1 : A A C C G G T T

k_2 : A A C C G G T T

k_3 : A A C C G G T T

Extraction des k -mers

Les k -mers sont des facteurs de longueur k de séquences d'ADN.

Ex : Avec la séquence $s = AACCGGTT$ de longueur L , on obtient les k -mers de longueur 6 (6-mers) suivants :

k_1 : A A C C G G T T

k_2 : A A C C G G T T

k_3 : A A C C G G T T

Extraction des k -mers

Les k -mers sont des facteurs de longueur k de séquences d'ADN.

Ex : Avec la séquence $s = AACCGGTT$ de longueur L , on obtient les k -mers de longueur 6 (6-mers) suivants :

k_1 : A A C C G G T T

k_2 : A A C C G G T T

k_3 : A A C C G G T T

Extraction des k -mers

Les k -mers sont des facteurs de longueur k de séquences d'ADN.

Ex : Avec la séquence $s = AACCGGTT$ de longueur L , on obtient les k -mers de longueur 6 (6-mers) suivants :

k_1 : A A C C G G T T

k_2 : A A C C G G T T

k_3 : A A C C G G T T

Les k -mers espacés sont des k -mers discontinus.

On utilise un motif m composé de zéros et de uns pour les représenter, où chaque zéro correspond à une insertion/délétion.

Ex : Avec la séquence $s = AACCGGTT...$

- et le motif $m = 10100111$, on obtient le 5-mers espacé à délétion $ACGTT$
- et le motif $m = 10011$, on obtient les 5-mers espacé à insertion $\{AAAAC, AACACC...\}$

Les *k*-mers espacés sont des *k*-mers discontinus.

On utilise un motif *m* composé de zéros et de uns pour les représenter, où chaque zéro correspond à une insertion/délétion.

Ex : Avec la séquence $s = AACCGGTT\dots$

- et le motif $m = 10100111$, on obtient le 5-mers espacé à délétion $ACGTT$
- et le motif $m = 10011$, on obtient les 5-mers espacé à insertion $\{AAAAC, AACACC\dots\}$

Les *k*-mers espacés sont des *k*-mers discontinus.

On utilise un motif *m* composé de zéros et de uns pour les représenter, où chaque zéro correspond à une insertion/délétion.

Ex : Avec la séquence $s = AACCGGTT...$

- et le motif $m = 10100111$, on obtient le 5-mers espacé à délétion *ACGTT*
- et le motif $m = 10011$, on obtient les 5-mers espacé à insertion {*AAAAC*, *AACACC*...}

Les k -mers espacés sont des k -mers discontinus.

On utilise un motif m composé de zéros et de uns pour les représenter, où chaque zéro correspond à une insertion/délétion.

Ex : Avec la séquence $s = AACCGGTT...$

- et le motif $m = 10100111$, on obtient le 5-mers espacé à délétion *ACGTT*
- et le motif $m = 10011$, on obtient les 5-mers espacé à insertion $\{AAAAC, AACACC...\}$

Les *k*-mers espacés sont des *k*-mers discontinus.

On utilise un motif *m* composé de zéros et de uns pour les représenter, où chaque zéro correspond à une insertion/délétion.

Ex : Avec la séquence $s = AACCGGTT\dots$

- et le motif $m = 10100111$, on obtient le 5-mers espacé à délétion $ACGTT$
- et le motif $m = 10011$, on obtient les 5-mers espacé à insertion $\{AAAAC, AACACC\dots\}$

K-mers espacés à délétion

Utilisé afin de simuler des corrections aux erreurs d'insertions sur les lectures longues.

Ex : Avec la séquence $s = AACCGGTT$ et le motif $m = 111011$, on obtient les 5-mers espacés suivants :



K-mers espacés à délétion

Utilisé afin de simuler des corrections aux erreurs d'insertions sur les lectures longues.

Ex : Avec la séquence $s = AACCGGTT$ et le motif $m = 111011$, on obtient les 5-mers espacés suivants :

k_1 : A A C ~~G~~ G G T T

k_2 : A A C C ~~G~~ G T T

k_3 : A A C C G ~~G~~ T T

K-mers espacés à délétion

Utilisé afin de simuler des corrections aux erreurs d'insertions sur les lectures longues.

Ex : Avec la séquence $s = AACCGGTT$ et le motif $m = 111011$, on obtient les 5-mers espacés suivants :

k_1 : A A C ~~X~~ G G T T

k_2 : A A C C ~~X~~ G T T

k_3 : A A C C G ~~X~~ T T

K-mers espacés à délétion

Utilisé afin de simuler des corrections aux erreurs d'insertions sur les lectures longues.

Ex : Avec la séquence $s = AACCGGTT$ et le motif $m = 111011$, on obtient les 5-mers espacés suivants :

k_1 : A A C ~~G~~ G G T T

k_2 : A A C C ~~G~~ G T T

k_3 : A A C C G ~~G~~ T T

K-mers espacés à délétion

Utilisé afin de simuler des corrections aux erreurs d'insertions sur les lectures longues.

Ex : Avec la séquence $s = AACCGGTT$ et le motif $m = 111011$, on obtient les 5-mers espacés suivants :

k_1 : A A C ~~X~~ G G T T

k_2 : A A C C ~~X~~ G T T

k_3 : A A C C G ~~X~~ T T

K-mers espacés à insertion

Utilisé afin de simuler des corrections aux erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k -mers espacés à insertion possibles, avec t = nombre de 0 dans le motif.

Ex : Avec la séquence **s = AACCGGTT** et le motif **m = 111011**, on obtient les 6-mers espacés suivants :

k_1 : A A C $\underbrace{A, C, G, T}_{} C$ G G T T

k_2 : A A C $\underbrace{C, A, G, T}_{} G$ G T T

k_3 : A A C C $\underbrace{G, A, C, T}_{} G$ T T

k_4 : A A C C G $\underbrace{G, A, C, T}_{} T$ T

K-mers espacés à insertion

Utilisé afin de simuler des corrections aux erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k -mers espacés à insertion possibles, avec t = nombre de 0 dans le motif.

Ex : Avec la séquence $s = \mathbf{AACCGGTT}$ et le motif $m = \mathbf{111011}$, on obtient les 6-mers espacés suivants :

k_1 : A A C $\underbrace{A, C, G, T}_{\text{insertion}} C G G T T$

k_2 : A A C $\underbrace{C A, C, G, T}_{\text{insertion}} G G T T$

k_3 : A A C C $\underbrace{G A, C, G, T}_{\text{insertion}} G T T$

k_4 : A A C C G $\underbrace{G A, C, G, T}_{\text{insertion}} T T$

K-mers espacés à insertion

Utilisé afin de simuler des corrections aux erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k -mers espacés à insertion possibles, avec t = nombre de 0 dans le motif.

Ex : Avec la séquence **s = AACCGGTT** et le motif **m = 111011**, on obtient les 6-mers espacés suivants :

k_1 : A A C $\underbrace{A, C, G, T}_{\text{insertion}} C G G T T$

k_2 : A A C $\underbrace{C A, C, G, T}_{\text{insertion}} G G T T$

k_3 : A A C C $\underbrace{G A, C, G, T}_{\text{insertion}} G T T$

k_4 : A A C C G $\underbrace{G A, C, G, T}_{\text{insertion}} T T$

K-mers espacés à insertion

Utilisé afin de simuler des corrections aux erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k -mers espacés à insertion possibles, avec t = nombre de 0 dans le motif.

Ex : Avec la séquence **s = AACCGGTT** et le motif **m = 111011**, on obtient les 6-mers espacés suivants :

k_1 : A A C $\underbrace{A, C, G, T}$ C G G T T

k_2 : A A C $\underbrace{C, A, C, G, T}$ G G T T

k_3 : A A C C $\underbrace{G, A, C, G, T}$ G T T

k_4 : A A C C G $\underbrace{G, A, C, G, T}$ T T

K-mers espacés à insertion

Utilisé afin de simuler des corrections aux erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k -mers espacés à insertion possibles, avec $t = \text{nombre de 0 dans le motif}$.

Ex : Avec la séquence **s = AACCGGTT** et le motif **m = 111011**, on obtient les 6-mers espacés suivants :

k_1 : A A C $\underbrace{A, C, G, T}_{} C G G T T$

k_2 : A A C $\underbrace{C A, C, G, T}_{} G G T T$

k_3 : A A C C $\underbrace{G A, C, G, T}_{} G T T$

k_4 : A A C C G $\underbrace{G A, C, G, T}_{} T T$

K-mers espacés à insertion

Utilisé afin de simuler des corrections aux erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k -mers espacés à insertion possibles, avec $t = \text{nombre de 0 dans le motif}$.

Ex : Avec la séquence **s = AACCGGTT** et le motif **m = 111011**, on obtient les 6-mers espacés suivants :

k_1 : A A C $\underbrace{A, C, G, T}_{} C G G T T$

k_2 : A A C $\underbrace{C A, C, G, T}_{} G G T T$

k_3 : A A C C $\underbrace{G A, C, G, T}_{} G T T$

k_4 : A A C C G $\underbrace{G A, C, G, T}_{} T T$

K -mers espacés à insertion

Utilisé afin de simuler des corrections aux erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k -mers espacés à insertion possibles, avec $t =$ nombre de 0 dans le motif.

Ex : Avec la séquence **s = AACCGGTT** et le motif **m = 111011**, on obtient les 6-mers espacés suivants :

k_1 : A A C $\underbrace{A, C, G, T}_{} C G G T T$

k_2 : A A C $\underbrace{C A, C, G, T}_{} G G T T$

k_3 : A A C C $\underbrace{G A, C, G, T}_{} G T T$

k_4 : A A C C G $\underbrace{G A, C, G, T}_{} T T$

Implémentations

Jellyfish² est la référence pour l'extraction de k -mers contigus.

Pour les k -mers espacés à délétion, l'outil **GkAmpi**³ est en cours de développement.

Les programmes kmersDel et kmersExpand, développés en C++11 *multi-thread*, traitent les k -mers à délétion et insertion.

Deux versions de kmersDel existent :

- Une renommée kmersCount, favorisant la vitesse en dépit de l'utilisation de la mémoire
- L'autre, kmersDel, est moins efficace mais peut être couplé à Jellyfish sur n'importe quelle longueur de k -mers

2. Guillaume Marcais and Carl Kingsford[2]

3. Alban Mancheron[3]

Jellyfish² est la référence pour l'extraction de k -mers contigus.

Pour les k -mers espacés à délétion, l'outil **GkAmpi**³ est en cours de développement.

Les programmes kmersDel et kmersExpand, développés en C++11 *multi-thread*, traitent les k -mers à délétion et insertion.

Deux versions de kmersDel existent :

- Une renommée kmersCount, favorisant la vitesse en dépit de l'utilisation de la mémoire
- L'autre, kmersDel, est moins efficace mais peut être couplé à **Jellyfish** sur n'importe quelle longueur de k -mers

2. Guillaume Marcais and Carl Kingsford[2]

3. Alban Mancheron[3]

Jellyfish² est la référence pour l'extraction de k -mers contigus.

Pour les k -mers espacés à délétion, l'outil **GkAmpi**³ est en cours de développement.

Les programmes kmersDel et kmersExpand, développés en C++11 *multi-thread*, traitent les k -mers à délétion et insertion.

Deux versions de kmersDel existent :

- Une renommée kmersCount, favorisant la vitesse en dépit de l'utilisation de la mémoire
- L'autre, kmersDel, est moins efficace mais peut être couplé à Jellyfish sur n'importe quelle longueur de k -mers

2. Guillaume Marcais and Carl Kingsford[2]

3. Alban Mancheron[3]

Jellyfish² est la référence pour l'extraction de k -mers contigus.

Pour les k -mers espacés à délétion, l'outil **GkAmpi**³ est en cours de développement.

Les programmes kmersDel et kmersExpand, développés en C++11 *multi-thread*, traitent les k -mers à délétion et insertion.

Deux versions de kmersDel existent :

- Une renommée kmersCount, favorisant la vitesse en dépit de l'utilisation de la mémoire
- L'autre, kmersDel, est moins efficace mais peut être couplé à Jellyfish sur n'importe quelle longueur de k -mers

2. Guillaume Marcais and Carl Kingsford[2]

3. Alban Mancheron[3]

Jellyfish² est la référence pour l'extraction de k -mers contigus.

Pour les k -mers espacés à délétion, l'outil **GkAmpi**³ est en cours de développement.

Les programmes kmersDel et kmersExpand, développés en C++11 *multi-thread*, traitent les k -mers à délétion et insertion.

Deux versions de kmersDel existent :

- Une renommée kmersCount, favorisant la vitesse en dépit de l'utilisation de la mémoire
- L'autre, kmersDel, est moins efficace mais peut être couplé à **Jellyfish** sur n'importe quelle longueur de k -mers

2. Guillaume Marcais and Carl Kingsford[2]

3. Alban Mancheron[3]

Jellyfish² est la référence pour l'extraction de k -mers contigus.

Pour les k -mers espacés à délétion, l'outil **GkAmpi**³ est en cours de développement.

Les programmes kmersDel et kmersExpand, développés en C++11 *multi-thread*, traitent les k -mers à délétion et insertion.

Deux versions de kmersDel existent :

- Une renommée kmersCount, favorisant la vitesse en dépit de l'utilisation de la mémoire
- L'autre, kmersDel, est moins efficace mais peut être couplé à **Jellyfish** sur n'importe quelle longueur de k -mers

2. Guillaume Marcais and Carl Kingsford[2]

3. Alban Mancheron[3]

Implémentation : kmersDel

Entrées : *table_hachage* *table*, mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* faire

pour $i = 0; i + k \leq |lecture|; i++ = 1$ faire

kmerEntier = 0;

kmer = "" ;

pour $j = 0; j < k; j++ = 1$ faire

si *motif*[*j*] $\neq 0$ alors *kmer* = *kmer* + *lecture*[*i* + *j*];

fin

pour chaque *nucleotide* de *kmer* faire

*kmerEntier** = 4;

suivant *valeur* de *nucleotide* faire

cas où *A* faire;

cas où *C* faire *kmerEntier*+ = 1;

cas où *G* faire *kmerEntier*+ = 2;

cas où *T* faire *kmerEntier*+ = 3;

fin

fin

table[*kmerEntier*] + = 1;

fin

fin

Implémentation : kmersDel

Entrées : *table_hachage* *table*, mots *lectures*, mot *motif*, entier *k*
pour chaque *lecture* de *lectures* faire

```
    pour  $i = 0; i + k \leq |lecture|; i++ = 1$  faire
        kmerEntier = 0;
        kmer = "" ;
        pour  $j = 0; j < k; j++ = 1$  faire
            si motif[j]  $\neq 0$  alors kmer = kmer + lecture[i + j];
        fin
        pour chaque nucleotide de kmer faire
            kmerEntier* = 4;
            suivant valeur de nucleotide faire
                cas où A faire;
                cas où C faire kmerEntier+ = 1;
                cas où G faire kmerEntier+ = 2;
                cas où T faire kmerEntier+ = 3;
            fin
        fin
        table[kmerEntier] + = 1;
    fin
fin
```

Implémentation : kmersDel

Entrées : *table_hachage table*, mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i += 1$ **faire**

kmerEntier = 0;

kmer = "" ;

pour $j = 0; j < k; j += 1$ **faire**

 si *motif*[*j*] $\neq 0$ alors *kmer* = *kmer* + *lecture*[*i* + *j*];

fin

pour chaque *nucleotide* de *kmer* **faire**

*kmerEntier** = 4;

suivant *valeur* de *nucleotide* **faire**

 cas où *A* **faire**;

 cas où *C* **faire** *kmerEntier*+ = 1;

 cas où *G* **faire** *kmerEntier*+ = 2;

 cas où *T* **faire** *kmerEntier*+ = 3;

fin

fin

table[*kmerEntier*] + = 1;

fin

fin

Implémentation : kmersDel

Entrées : *table_hachage table*, mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i += 1$ **faire**

kmerEntier = 0;

kmer = "";

pour $j = 0; j < k; j += 1$ **faire**

si *motif*[*j*] $\neq 0$ **alors** *kmer* = *kmer* + *lecture*[*i* + *j*];

fin

pour chaque *nucleotide* de *kmer* **faire**

*kmerEntier** = 4;

suivant *valeur* de *nucleotide* **faire**

cas où *A* **faire**;

cas où *C* **faire** *kmerEntier*+ = 1;

cas où *G* **faire** *kmerEntier*+ = 2;

cas où *T* **faire** *kmerEntier*+ = 3;

fin

fin

table[*kmerEntier*] + = 1;

fin

fin

Implémentation : kmersDel

Entrées : *table_hachage table*, mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i += 1$ **faire**

kmerEntier = 0;

kmer = "" ;

pour $j = 0; j < k; j += 1$ **faire**

si *motif*[*j*] $\neq 0$ **alors** *kmer* = *kmer* + *lecture*[*i* + *j*];

fin

pour chaque *nucleotide* de *kmer* **faire**

*kmerEntier** = 4;

suivant *valeur* de *nucleotide* **faire**

cas où *A* **faire**;

cas où *C* **faire** *kmerEntier*+ = 1;

cas où *G* **faire** *kmerEntier*+ = 2;

cas où *T* **faire** *kmerEntier*+ = 3;

fin

fin

table[*kmerEntier*] + = 1;

fin

fin

Implémentation : kmersDel

Entrées : *table_hachage* *table*, mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i++ = 1$ **faire**

kmerEntier = 0;

kmer = "" ;

pour $j = 0; j < k; j++ = 1$ **faire**

si *motif*[*j*] $\neq 0$ **alors** *kmer* = *kmer* + *lecture*[*i* + *j*];

fin

pour chaque *nucleotide* de *kmer* **faire**

*kmerEntier** = 4;

suivant *valeur* de *nucleotide* **faire**

cas où *A* **faire**;

cas où *C* **faire** *kmerEntier*+ = 1;

cas où *G* **faire** *kmerEntier*+ = 2;

cas où *T* **faire** *kmerEntier*+ = 3;

fin

fin

table[*kmerEntier*] + = 1;

fin

fin

Implémentation : kmersExpand

kmersExpand

Entrées : mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* faire

 pour $i = 0; i + k \leq |lecture|; i++ = 1$ faire

kmer = *lecture*[*i* : *k*];

kmersExpandRec(*kmer*, *motif*, 0);

 fin

fin

kmersExpandRec

Entrées : mot *kmer*, mot *nvKmer*, mot *motif*, entier *posMotif*, entier *posKmer*

si *posSeed* == |*motif*| alors affiche(*nvKmer*);

sinon si *motif*[*posMotif*] ≠ 0 alors

kmersExpandRec(*kmer*, *nvKmer* + *kmer*[*posKmer*], *posMotif* + 1, *posKmer* + 1);

sinon

kmersExpandRec(*kmer*, *nvKmer* + A, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + C, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + G, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + T, *posMotif* + 1, *posKmer*);

fin

Implémentation : kmersExpand

kmersExpand

Entrées : mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* faire

 pour $i = 0; i + k \leq |lecture|; i += 1$ faire

kmer = *lecture*[*i* : *k*];

kmersExpandRec(*kmer*, *motif*, 0);

 fin

fin

kmersExpandRec

Entrées : mot *kmer*, mot *nvKmer*, mot *motif*, entier *posMotif*, entier *posKmer*

si *posSeed* == |*motif*| alors affiche(*nvKmer*);

sinon si *motif*[*posMotif*] ≠ 0 alors

kmersExpandRec(*kmer*, *nvKmer* + *kmer*[*posKmer*], *posMotif* + 1, *posKmer* + 1);

sinon

kmersExpandRec(*kmer*, *nvKmer* + A, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + C, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + G, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + T, *posMotif* + 1, *posKmer*);

fin

Implémentation : kmersExpand

kmersExpand

Entrées : mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i++ = 1$ **faire**

kmer = *lecture*[*i* : *k*];

kmersExpandRec(*kmer*, *motif*, 0);

fin

fin

kmersExpandRec

Entrées : mot *kmer*, mot *nvKmer*, mot *motif*, entier *posMotif*, entier *posKmer*

si *posSeed* == |*motif*| **alors** *affiche*(*nvKmer*);

sinon si *motif*[*posMotif*] ≠ 0 **alors**

kmersExpandRec(*kmer*, *nvKmer* + *kmer*[*posKmer*], *posMotif* + 1, *posKmer* + 1);

sinon

kmersExpandRec(*kmer*, *nvKmer* + A, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + C, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + G, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + T, *posMotif* + 1, *posKmer*);

fin

Implémentation : kmersExpand

kmersExpand

Entrées : mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i += 1$ **faire**

kmer = *lecture*[*i* : *k*];

kmersExpandRec(*kmer*, *motif*, 0);

fin

fin

kmersExpandRec

Entrées : mot *kmer*, mot *nvKmer*, mot *motif*, entier *posMotif*, entier *posKmer*

si *posSeed* == |*motif*| **alors** *affiche*(*nvKmer*);

sinon si *motif*[*posMotif*] ≠ 0 **alors**

kmersExpandRec(*kmer*, *nvKmer* + *kmer*[*posKmer*], *posMotif* + 1, *posKmer* + 1);

sinon

kmersExpandRec(*kmer*, *nvKmer* + A, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + C, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + G, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + T, *posMotif* + 1, *posKmer*);

fin

Implémentation : kmersExpand

kmersExpand

Entrées : mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i += 1$ **faire**

kmer = *lecture*[*i* : *k*];

kmersExpandRec(*kmer*, *motif*, 0);

fin

fin

kmersExpandRec

Entrées : mot *kmer*, mot *nvKmer*, mot *motif*, entier *posMotif*, entier *posKmer*

si *posSeed* == |*motif*| **alors** *affiche*(*nvKmer*);

sinon si *motif*[*posMotif*] ≠ 0 **alors**

kmersExpandRec(*kmer*, *nvKmer* + *kmer*[*posKmer*], *posMotif* + 1, *posKmer* + 1);

sinon

kmersExpandRec(*kmer*, *nvKmer* + A, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + C, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + G, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + T, *posMotif* + 1, *posKmer*);

fin

Implémentation : kmersExpand

kmersExpand

Entrées : mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i += 1$ **faire**

kmer = *lecture*[*i* : *k*];

kmersExpandRec(*kmer*, *motif*, 0);

fin

fin

kmersExpandRec

Entrées : mot *kmer*, mot *nvKmer*, mot *motif*, entier *posMotif*, entier *posKmer*

si *posSeed* == |*motif*| **alors** *affiche*(*nvKmer*);

sinon si *motif*[*posMotif*] ≠ 0 **alors**

kmersExpandRec(*kmer*, *nvKmer* + *kmer*[*posKmer*], *posMotif* + 1, *posKmer* + 1);

sinon

kmersExpandRec(*kmer*, *nvKmer* + A, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + C, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + G, *posMotif* + 1, *posKmer*);

kmersExpandRec(*kmer*, *nvKmer* + T, *posMotif* + 1, *posKmer*);

fin

Implémentation : kmersExpand

kmersExpand

Entrées : mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i += 1$ **faire**

$kmer = lecture[i : k];$

$kmersExpandRec(kmer, motif, 0);$

fin

fin

kmersExpandRec

Entrées : mot *kmer*, mot *nvKmer*, mot *motif*, entier *posMotif*, entier *posKmer*

si $posSeed == |motif|$ **alors** $affiche(nvKmer);$

sinon si $motif[posMotif] \neq 0$ **alors**

$kmersExpandRec(kmer, nvKmer + kmer[posKmer], posMotif + 1, posKmer + 1);$

sinon

$kmersExpandRec(kmer, nvKmer + A, posMotif + 1, posKmer);$

$kmersExpandRec(kmer, nvKmer + C, posMotif + 1, posKmer);$

$kmersExpandRec(kmer, nvKmer + G, posMotif + 1, posKmer);$

$kmersExpandRec(kmer, nvKmer + T, posMotif + 1, posKmer);$

fin

Implémentation : kmersExpand

kmersExpand

Entrées : mots *lectures*, mot *motif*, entier *k*

pour chaque *lecture* de *lectures* **faire**

pour $i = 0; i + k \leq |lecture|; i += 1$ **faire**

$kmer = lecture[i : k];$

$kmersExpandRec(kmer, motif, 0);$

fin

fin

kmersExpandRec

Entrées : mot *kmer*, mot *nvKmer*, mot *motif*, entier *posMotif*, entier *posKmer*

si $posSeed == |motif|$ **alors** $affiche(nvKmer);$

sinon si $motif[posMotif] \neq 0$ **alors**

$kmersExpandRec(kmer, nvKmer + kmer[posKmer], posMotif + 1, posKmer + 1);$

sinon

$kmersExpandRec(kmer, nvKmer + A, posMotif + 1, posKmer);$

$kmersExpandRec(kmer, nvKmer + C, posMotif + 1, posKmer);$

$kmersExpandRec(kmer, nvKmer + G, posMotif + 1, posKmer);$

$kmersExpandRec(kmer, nvKmer + T, posMotif + 1, posKmer);$

fin

Définition

Un mot minimal absent d'une séquence est un mot absent dont les facteurs propres (plus long suffixe propre et plus long préfixe propre) sont tous présents dans la séquence.

Avec la séquence $s = \text{AACACACC}$, on obtient les Mots Minimaux Absents suivants :

$\{\text{AAA}, \text{AACACC}, \text{AACC}, \text{CAA}, \text{CACACA}, \text{CCA}, \text{CCC}\}$

Définition

Un mot minimal absent d'une séquence est un mot absent dont les facteurs propres (plus long suffixe propre et plus long préfixe propre) sont tous présents dans la séquence.

Avec la séquence $s = AACACACC$, on obtient les Mots Minimaux Absents suivants :

$\{AAA, AACACC, AACC, CAA, CACACA, CCA, CCC\}$

Définition

Un mot minimal absent d'une séquence est un mot absent dont les facteurs propres (plus long suffixe propre et plus long préfixe propre) sont tous présents dans la séquence.

Avec la séquence $s = \text{AACACACC}$, on obtient les Mots Minimaux Absents suivants :

$\{\text{AAA}, \text{AACACC}, \text{AACC}, \text{CAA}, \text{CACACA}, \text{CCA}, \text{CCC}\}$

Définition

Un mot minimal absent d'une séquence est un mot absent dont les facteurs propres (plus long suffixe propre et plus long préfixe propre) sont tous présents dans la séquence.

Avec la séquence $s = AACACACC$, on obtient les Mots Minimaux Absents suivants :

$\{AAA, AACACC, AACC, CAA, CACACA, CCA, CCC\}$

Plus long sous-mot commun (PLSC)

Définition

Un mot x est un **sous-mot** d'un mot y s'il existe une factorisation $y = z_0 x_1 z_1 x_2 \cdots x_n z_n$ telle que $x = x_1 x_2 \cdots x_n$.

Par exemple, $x = AAAAC$ est un sous-mot de $y = AACACACC$

Définition

Le plus long sous-mot commun à deux séquences x et y est le mot z tel que z soit le plus long sous-mot à la fois dans x et dans y .

Par exemple, le plus long sous-mot commun aux séquences $x = ACCAAC$ et $y = AACACACC$ est $ACCAC$

Plus long sous-mot commun (PLSC)

Définition

Un mot x est un **sous-mot** d'un mot y s'il existe une factorisation $y = z_0 x_1 z_1 x_2 \cdots x_n z_n$ telle que $x = x_1 x_2 \cdots x_n$.

Par exemple, $x = AAAAC$ est un sous-mot de $y = AACACACC$

Définition

Le plus long sous-mot commun à deux séquences x et y est le mot z tel que z soit le plus long sous-mot à la fois dans x et dans y .

Par exemple, le plus long sous-mot commun aux séquences $x = ACCAAC$ et $y = AACACACC$ est $ACCAC$

Plus long sous-mot commun (PLSC)

Définition

Un mot x est un **sous-mot** d'un mot y s'il existe une factorisation $y = z_0 x_1 z_1 x_2 \cdots x_n z_n$ telle que $x = x_1 x_2 \cdots x_n$.

Par exemple, $x = \text{AAAAC}$ est un sous-mot de $y = \text{AACACACC}$

Définition

Le plus long sous-mot commun à deux séquences x et y est le mot z tel que z soit le plus long sous-mot à la fois dans x et dans y .

Par exemple, le plus long sous-mot commun aux séquences $x = \text{ACCAAC}$ et $y = \text{AACACACC}$ est ACCAC

Plus long sous-mot commun (PLSC)

Définition

Un mot x est un **sous-mot** d'un mot y s'il existe une factorisation $y = z_0 x_1 z_1 x_2 \cdots x_n z_n$ telle que $x = x_1 x_2 \cdots x_n$.

Par exemple, $x = AAAAC$ est un sous-mot de $y = AACACACC$

Définition

Le plus long sous-mot commun à deux séquences x et y est le mot z tel que z soit le plus long sous-mot à la fois dans x et dans y .

Par exemple, le plus long sous-mot commun aux séquences $x = ACCAAC$ et $y = AACACACC$ est $ACCAC$

Plus long sous-mot commun (PLSC)

Définition

Un mot x est un **sous-mot** d'un mot y s'il existe une factorisation $y = z_0 x_1 z_1 x_2 \cdots x_n z_n$ telle que $x = x_1 x_2 \cdots x_n$.

Par exemple, $x = AAAAC$ est un sous-mot de $y = AACACACC$

Définition

Le plus long sous-mot commun à deux séquences x et y est le mot z tel que z soit le plus long sous-mot à la fois dans x et dans y .

Par exemple, le plus long sous-mot commun aux séquences $x = ACCAAC$ et $y = AACACACC$ est $ACCAC$

Plus long sous-mot commun (PLSC)

Définition

Un mot x est un **sous-mot** d'un mot y s'il existe une factorisation $y = z_0 x_1 z_1 x_2 \cdots x_n z_n$ telle que $x = x_1 x_2 \cdots x_n$.

Par exemple, $x = AAAAC$ est un sous-mot de $y = AACACACC$

Définition

Le plus long sous-mot commun à deux séquences x et y est le mot z tel que z soit le plus long sous-mot à la fois dans x et dans y .

Par exemple, le plus long sous-mot commun aux séquences $x = \text{ACCAAC}$ et $y = \text{AACACACC}$ est ACCAC

Plus long sous-mot commun (PLSC)

Définition

Un mot x est un **sous-mot** d'un mot y s'il existe une factorisation $y = z_0 x_1 z_1 x_2 \cdots x_n z_n$ telle que $x = x_1 x_2 \cdots x_n$.

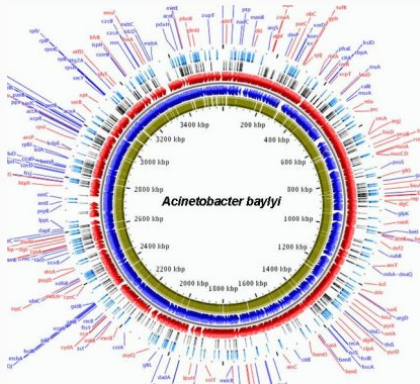
Par exemple, $x = AAAAC$ est un sous-mot de $y = AACACACC$

Définition

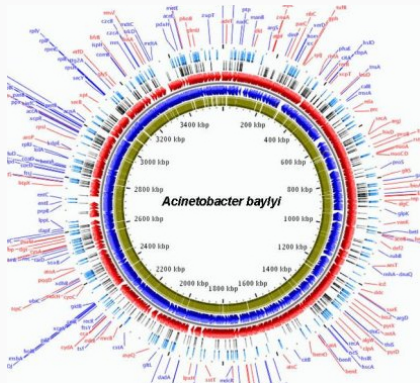
Le plus long sous-mot commun à deux séquences x et y est le mot z tel que z soit le plus long sous-mot à la fois dans x et dans y .

Par exemple, le plus long sous-mot commun aux séquences $x = \text{ACCA_C}$ et $y = \text{_AC_CAC_}$ est $ACCAC$

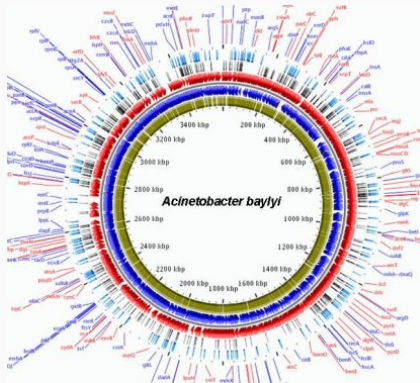
Résultats obtenus



- Espèce étudiée : *Acinetobacter baylyi*
- Taille du génome : $\simeq 3\,600\,000$ nucléotides
- Lectures longues : Oxford Nanopore (30% d'erreur) : 89 011, de longueur moyenne $\simeq 4\,300$ nucléotides



- Espèce étudiée : *Acinetobacter baylyi*
- Taille du génome : $\simeq 3\,600\,000$ nucléotides
- Lectures longues : Oxford Nanopore (30% d'erreur) : 89 011, de longueur moyenne $\simeq 4\,300$ nucléotides



- Espèce étudiée : *Acinetobacter baylyi*
- Taille du génome : $\simeq 3\,600\,000$ nucléotides
- Lectures longues : Oxford Nanopore (30% d'erreur) : 89 011, de longueur moyenne $\simeq 4\,300$ nucléotides

Résultats : kmersDel et kmersExpand

Union des k -mers + k -mers à délétion + k -mers à insertion

Enormément de k -mers inutiles dans les lectures longues (0,23% à 0,44% d'utile)

- 16-mers, $freq = 5$, un trou de longueur 1 \rightarrow 87%
- 16-mers, $freq = 5$, un trou de longueur 1 à 2 \rightarrow 98%

Un grand nombre de k -mers sont trouvés, mais impossible de les filtrer.

Résultats similaires sur 20-mers et 11-mers

Résultats similaires avec les lectures Pacific Biosciences et Oxford Nanopore.

Résultats : kmersDel et kmersExpand

Union des k -mers + k -mers à délétion + k -mers à insertion

Enormément de k -mers inutiles dans les lectures longues (0,23% à 0,44% d'utile)

- 16-mers, $freq = 5$, un trou de longueur 1 \rightarrow 87%
- 16-mers, $freq = 5$, un trou de longueur 1 à 2 \rightarrow 98%

Un grand nombre de k -mers sont trouvés, mais impossible de les filtrer.

Résultats similaires sur 20-mers et 11-mers

Résultats similaires avec les lectures Pacific Biosciences et Oxford Nanopore.

Résultats : kmersDel et kmersExpand

Union des k -mers + k -mers à délétion + k -mers à insertion

Enormément de k -mers inutiles dans les lectures longues (0,23% à 0,44% d'utile)

- 16-mers, $freq = 5$, un trou de longueur 1 \rightarrow 87%
- 16-mers, $freq = 5$, un trou de longueur 1 à 2 \rightarrow 98%

Un grand nombre de k -mers sont trouvés, mais impossible de les filtrer.

Résultats similaires sur 20-mers et 11-mers

Résultats similaires avec les lectures Pacific Biosciences et Oxford Nanopore.

Résultats : kmersDel et kmersExpand

Union des k -mers + k -mers à délétion + k -mers à insertion

Enormément de k -mers inutiles dans les lectures longues (0,23% à 0,44% d'utile)

- 16-mers, $freq = 5$, un trou de longueur 1 \rightarrow 87%
- 16-mers, $freq = 5$, un trou de longueur 1 à 2 \rightarrow 98%

Un grand nombre de k -mers sont trouvés, mais impossible de les filtrer.

Résultats similaires sur 20-mers et 11-mers

Résultats similaires avec les lectures Pacific Biosciences et Oxford Nanopore.

Résultats : kmersDel et kmersExpand

Union des k -mers + k -mers à délétion + k -mers à insertion

Enormément de k -mers inutiles dans les lectures longues (0,23% à 0,44% d'utile)

- 16-mers, $freq = 5$, un trou de longueur 1 \rightarrow 87%
- 16-mers, $freq = 5$, un trou de longueur 1 à 2 \rightarrow 98%

Un grand nombre de k -mers sont trouvés, mais impossible de les filtrer.

Résultats similaires sur 20-mers et 11-mers

Résultats similaires avec les lectures Pacific Biosciences et Oxford Nanopore.

Résultats : kmersDel et kmersExpand

Union des k -mers + k -mers à délétion + k -mers à insertion

Enormément de k -mers inutiles dans les lectures longues (0,23% à 0,44% d'utile)

- 16-mers, $freq = 5$, un trou de longueur 1 \rightarrow 87%
- 16-mers, $freq = 5$, un trou de longueur 1 à 2 \rightarrow 98%

Un grand nombre de k -mers sont trouvés, mais impossible de les filtrer.

Résultats similaires sur 20-mers et 11-mers

Résultats similaires avec les lectures Pacific Biosciences et Oxford Nanopore.

Résultats : kmersDel et kmersExpand

Union des k -mers + k -mers à délétion + k -mers à insertion

Enormément de k -mers inutiles dans les lectures longues (0,23% à 0,44% d'utile)

- 16-mers, $freq = 5$, un trou de longueur 1 \rightarrow 87%
- 16-mers, $freq = 5$, un trou de longueur 1 à 2 \rightarrow 98%

Un grand nombre de k -mers sont trouvés, mais impossible de les filtrer.

Résultats similaires sur 20-mers et 11-mers

Résultats similaires avec les lectures Pacific Biosciences et Oxford Nanopore.

Résultats : kmersDel et kmersExpand

Union des k -mers + k -mers à délétion + k -mers à insertion

Enormément de k -mers inutiles dans les lectures longues (0,23% à 0,44% d'utile)

- 16-mers, $freq = 5$, un trou de longueur 1 \rightarrow 87%
- 16-mers, $freq = 5$, un trou de longueur 1 à 2 \rightarrow 98%

Un grand nombre de k -mers sont trouvés, mais impossible de les filtrer.

Résultats similaires sur 20-mers et 11-mers

Résultats similaires avec les lectures Pacific Biosciences et Oxford Nanopore.

Recherche de Mots Minimaux Absents fréquents :

- Pas concluant, repartition des MAW dans les bons/mauvais k -mers trop homogène

Recherche de Mots Minimaux Absents rares :

- Pas concluant, mêmes résultats que pour les MAW fréquents

Recherche de Mots Minimaux Absents fréquents :

- Pas concluant, repartition des MAW dans les bons/mauvais k -mers trop homogène

Recherche de Mots Minimaux Absents rares :

- Pas concluant, mêmes résultats que pour les MAW fréquents

Recherche de Mots Minimaux Absents fréquents :

- Pas concluant, repartition des MAW dans les bons/mauvais k -mers trop homogène

Recherche de Mots Minimaux Absents rares :

- Pas concluant, mêmes résultats que pour les MAW fréquents

Recherche de Mots Minimaux Absents fréquents :

- Pas concluant, repartition des MAW dans les bons/mauvais k -mers trop homogène

Recherche de Mots Minimaux Absents rares :

- Pas concluant, mêmes résultats que pour les MAW fréquents

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigés correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigés correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigés correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigés correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigés correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigées correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigées correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigés correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

But : Identifier les lectures longues provenant d'une même région du génome de référence.

Procédure de test :

- Sélection d'une lecture longue
- Récupération des lectures longues similaires
- Recherche de la lecture longue corrigée correspondante
- Vérifier qu'elles s'alignent bien sur la même région du génome

Résultat : 20-mers fréquents \Rightarrow 25 lectures longues similaires

Récupération des lectures longues corrigés correspondantes :

- 11 lectures récupérées dont 8 s'alignent sur la même région du génome.

4. Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier and Pierre Peterlongo[4]

Résultats : Plus long sous-mot commun (PLSC)

Extractions de PLSC entre lectures longues similaires :

- Pas d'alignement
- 1 lecture brute et son équivalente corrigée → mauvais alignement.
- 2 lectures corrigées similaires → le PLSC s'aligne très bien

PLSC entre les k -mers (32 et 64) des lectures longues similaires :

- Pas d'alignement

Résultats : Plus long sous-mot commun (PLSC)

Extractions de PLSC entre lectures longues similaires :

- Pas d'alignement
- 1 lecture brute et son équivalente corrigée → mauvais alignement.
- 2 lectures corrigées similaires → le PLSC s'aligne très bien

PLSC entre les k -mers (32 et 64) des lectures longues similaires :

- Pas d'alignement

Résultats : Plus long sous-mot commun (PLSC)

Extractions de PLSC entre lectures longues similaires :

- Pas d'alignement
- 1 lecture brute et son équivalente corrigée → mauvais alignement.
- 2 lectures corrigées similaires → le PLSC s'aligne très bien

PLSC entre les k -mers (32 et 64) des lectures longues similaires :

- Pas d'alignement

Résultats : Plus long sous-mot commun (PLSC)

Extractions de PLSC entre lectures longues similaires :

- Pas d'alignement
- 1 lecture brute et son équivalente corrigée → mauvais alignement.
- 2 lectures corrigées similaires → le PLSC s'aligne très bien

PLSC entre les k -mers (32 et 64) des lectures longues similaires :

- Pas d'alignement

Résultats : Plus long sous-mot commun (PLSC)

Extractions de PLSC entre lectures longues similaires :

- Pas d'alignement
- 1 lecture brute et son équivalente corrigée → mauvais alignement.
- 2 lectures corrigées similaires → le PLSC s'aligne très bien

PLSC entre les k -mers (32 et 64) des lectures longues similaires :

- Pas d'alignement

Résultats : Plus long sous-mot commun (PLSC)

Extractions de PLSC entre lectures longues similaires :

- Pas d'alignement
- 1 lecture brute et son équivalente corrigée → mauvais alignement.
- 2 lectures corrigées similaires → le PLSC s'aligne très bien

PLSC entre les k -mers (32 et 64) des lectures longues similaires :

- Pas d'alignement

Conclusion

- `kmersDel` et `kmersExpand` ont obtenus les meilleurs résultats
- Marquer les k-mers afin de déterminer les zones d'erreurs ou les motifs réguliers

- kmersDel et kmersExpand ont obtenus les meilleurs résultats
- Marquer les k-mers afin de déterminer les zones d'erreurs ou les motifs réguliers



Pierre Morisse, Thierry Lecroq, and Arnaud Lefebvre.

HG-CoLoR : Hybrid-Graph for the error Correction of Long Reads.

In Actes des Journées Ouvertes Biologie Informatique et Mathématiques, 2017.



Guillaume Marcais and Carl Kingsford.

A fast, lock-free approach for efficient parallel counting of occurrences of k-mers.

Bioinformatics, 27(6) :764–770, 2011.



Alban Mancheron.

GkAmpi.

Personal Communication, 2017.



Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier, and Pierre Peterlongo.

Compareads : comparing huge metagenomic experiments.

BMC Bioinformatics 13(Suppl 19), 2012.