

Stage de Maîtrise IGIS ITA

K-mers et k-mers espacés

Gabriel Toublanc

26 janvier 2017

Université de Rouen, U.F.R des Sciences et Techniques de Saint-Etienne-du-Rouvray,
Equipe LITIS TIBS

Introduction

Séquençage ADN :

- Lectures courtes (2005)

Séquençage ADN :

- Lectures courtes (2005)
- Lectures longues (2010)

Comptage des k-mers

Les k-mers sont des facteurs de séquences ADN.

Les k -mers sont des facteurs de séquences ADN.

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

Les k -mers sont des facteurs de séquences ADN.

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

$k_1 :$

A

A

C

C

G

G

T

T

Les k-mers sont des facteurs de séquences ADN.

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

k_1 : A A C C G G T T

k_2 : A A C C G G T T

Les k -mers sont des facteurs de séquences ADN.

Ex : Avec la séquence $x = AACCGGTT$, on obtient les k -mers de taille 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

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

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

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

K-mers espacés à délétion

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

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

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

K-mers espacés à délétion

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

Ex : Avec la séquence $x = AACCGGTT$ 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-mers espacés à délétion

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

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

k_1 : A A C ~~✗~~ G G T T

k_2 : A A C C ~~✗~~ G T T

k_3 : A A C C G ~~✗~~ T T

K-mers espacés à insertion

Utilisé afin de simuler les erreurs de délétion sur les lectures longues.

K-mers espacés à insertion

Utilisé afin de simuler les erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k-mers espacés à insertion possibles.

K-mers espacés à insertion

Utilisé afin de simuler les erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k-mers espacés à insertion possibles.

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

K-mers espacés à insertion

Utilisé afin de simuler les erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k-mers espacés à insertion possibles.

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

k_1 :

A	A	C	$\overbrace{A, C, G, T}$	C	G
---	---	---	--------------------------	---	---

 G T T

K-mers espacés à insertion

Utilisé afin de simuler les erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k-mers espacés à insertion possibles.

Ex : Avec la séquence **x = 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 C $\underbrace{A, C, G, T}$ G G T T

K-mers espacés à insertion

Utilisé afin de simuler les erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k-mers espacés à insertion possibles.

Ex : Avec la séquence **x = 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 C $\underbrace{A, C, G, T}$ G G T T

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

K-mers espacés à insertion

Utilisé afin de simuler les erreurs de délétion sur les lectures longues.

$(L - k + 1) * 4^t$ k-mers espacés à insertion possibles.

Ex : Avec la séquence **x = 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 $\underbrace{C, A, G, T}_{\text{insertion}} G G T T$

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

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

KmersDel

Entrées : `table_hachage` *table*, chaînes *lectures*, chaîne *motif*, entier *k*

Entrées : `table_hachage` *table*, chaînes *lectures*, chaîne *motif*, entier *k*
pour chaque *lecture de lectures* faire

Entrées : *table_hachage* *table*, chaînes *lectures*, chaîne *motif*, entier *k*
pour chaque *lecture* de *lectures* faire

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

kmerEntier = 0;

kmer = "" ;

Entrées : *table_hachage table*, chaînes *lectures*, chaîne *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

Entrées : *table_hachage table*, chaînes *lectures*, chaîne *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

Entrées : *table_hachage table*, chaînes *lectures*, chaîne *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

KmersExpand

kmerExpand

kmerExpand

Entrées : chaînes *lectures*, chaîne *motif*, entier k

kmerExpand

Entrées : chaînes *lectures*, chaîne *motif*, entier k
pour chaque *lecture de lectures* faire

|

kmerExpand

Entrées : chaînes *lectures*, chaîne *motif*, entier k

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

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

$kmer = lecture[i : i+k]$ $kmerExpandRec(kmer, motif, 0)$

fin

fin

KmersExpand

kmerExpand

Entrées : chaînes *lectures*, chaîne *motif*, entier *k*

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

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

$kmer = lecture[i : k]$ $kmerExpandRec(kmer, motif, 0)$

fin

fin

kmerExpandRec

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

kmerExpand

Entrées : chaînes *lectures*, chaîne *motif*, entier *k*

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

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

$kmer = lecture[i : k]$ *kmerExpandRec*(*kmer*, *motif*, 0)

fin

fin

kmerExpandRec

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

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

kmerExpand

Entrées : chaînes *lectures*, chaîne *motif*, entier *k*

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

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

$kmer = lecture[i : k]$ $kmerExpandRec(kmer, motif, 0)$

fin

fin

kmerExpandRec

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

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

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

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

kmerExpand

Entrées : chaînes *lectures*, chaîne *motif*, entier *k*

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

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

kmer = *lecture*[*i* : *k*] *kmerExpandRec*(*kmer*, *motif*, 0)

fin

fin

kmerExpandRec

Entrées : chaîne *kmer*, chaîne *nvKmer*, chaîne *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

Mots Minimaux absents

Plus long sous-mot commun

Plus long sous-mot commun

Résultats obtenus

Résultats : KmersDel et kmersExpand

Résultats : Plus long sous-mot commun

Conclusion