Index distribué de k-mers permettant leur comptage et leur localisation

`libGkArrays-MPI/gkampi`

`https://gite.lirmm.fr/doccy/libGkArrays-MPI`

Clément AGRET `<clement.agret@lirmm.fr>`

Annie CHATEAU `<annie.chateau@lirmm.fr>`

Alban MANCHERON `<alban.mancheron@lirmm.fr>`

Ceci est un poster dont vous êtes le héros. Parcourez les chemins au gré de vos envies, notez quelque part votre itinéraire, et découvrez jusqu'où il vous aura mené…

1 Départ

De nombreux outils d'analyse ou de comparaison de génomes / de transcriptomes utilisent des k-mers afin de traiter les volumes de données sans cesses croissants.

Un k-mer kézako ?

2 k-mer

séquence de k nucléotides

Ca me rappelle une histoire de graines ?

3 k-mer espacé

séquence de k nucléotides avec des « trous » à certains endroits prédéfinis.

[Noe *et al.*, 2005 ;Brinda *et al.*, 2015]

Pourquoi les compter ?

4 Comptage

Compter les k-mers sert par exemple à déterminer des marqueurs thérapeutiques, ou bien à distinguer des erreurs de séquençage parmi des potentielles variations biologiques.

Mais ils sont où ?

5 Indexation

L'indexation permet de retrouver l'origine d'un k-mer.

La plupart des outils ne le permettent pas.

La `libGkArrays-MPI` le permet ⇒ `gkampi` le fait!

Et si j'essayais avec un arbre ?

(c'est pas drôle en anglais)

Et si j'allais n'importe où ?

6 Dictionnaire

Les dictionnaires `python` sont des tables de hachage. Rendez-vous en 7.

Trois lignes de `python` et le tour est joué

Les k-mers, c'est cool

7 Table de hachage

Une table de hachage « préfabriquée » stocke généralement les clés (donc ici les k-mers) dans une liste pour gérer les collisions. cela implique des structures gourmandes en mémoire et potentiellement coûteuses en calcul dans le cas de très gros volumes de données.

Si c'est pour compter quelques k-mers, cela fait l'affaire, mais ce n'est pas adapté à notre situation.

Une table de hachage permettant d'associer un compteur à chaque k-mer, voilà l'idée

Je re-lis le panneau de départ

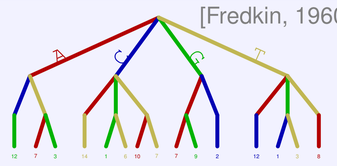
passage secret

Je ne suis pas convaincu(e)

Une Démo!

8 *Trie* des k-mers

[Fredkin, 1960]



Il est possible d'associer les localisations aux feuilles au lieu des simples comptages, mais cela à un coût.

Construire l'arbre explicite n'est pas une solution envisageable (sinon, il suffit d'aller à la case 6)

Quid des symboles dégénérés ?

Représentons l'arbre par des tableaux

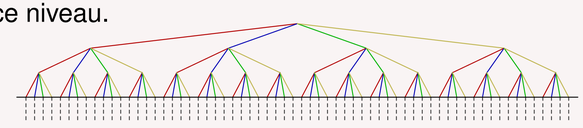
C'est quoi cette histoire de préfixe ?

Si je peux approcher $\mathcal{N}$ ⇒ je peux estimer ℓ

Quelle propriété exploiter ?

9 Topologie d'un *Trie*

Le *fill-up* d'un arbre n-aire désigne le plus bas niveau tel que tous les nœuds ont exactement n fils jusqu'à ce niveau.



Dans le cas de gros volumes de séquences composées des 4 nucléotides (sans biais trop important d'un des nucléotides), le *fillup* du *Trie* des k-mers est théoriquement de :

$$\ell = \frac{\log_2 \mathcal{N} - \log_2 \log_2 \mathcal{N} + \mathcal{O}(1)}{2}$$

(où $\mathcal{N}$ est le nombre de k-mers). [Park *et al.*, 2009]

Au secours, y a des logarithmes!

Je n'ai plus peur des logarithmes

Quel lien entre l'ADN et le binaire ?

10 Codage binaire

Avec deux bits (entités pouvant prendre deux valeurs distinctes : 0 ou 1), nous pouvons représenter 2×2 valeurs distinctes. Il suffit donc de définir une bijection entre l'ADN et ces 4 valeurs.

Dans la `libGkArrays-MPI`, nous avons choisi de préserver le lien entre l'ordre lexicographique des lettres et l'ordre numérique donné par les combinaisons.

Ainsi 00 code pour `A`, 01 code pour `C`, 10 code pour `G` et 11 code pour `T`,

Quel lien entre binaire et logarithme ?

4 au de 2 ?

11 Logarithme

Le logarithme d'une valeur n dans la base b est le nombre minimal d'entités pouvant prendre chacune b valeurs distinctes qui sont nécessaires pour représenter n combinaisons distinctes.

Par exemple, pour fabriquer un cadenas qui aurait 1000 combinaisons avec des roulettes numérotées de 0 à 9 (donc pouvant prendre 10 valeurs), il faut $\log_{10} 1000 = 3$ roulettes…

Et si j'essayais avec un arbre ?

12 Alphabet dégénéré

La gestion de l'ambiguïté rajoute un temps de traitement important, là où les approches par k-mers sont faites pour brasser d'importants volumes de données.

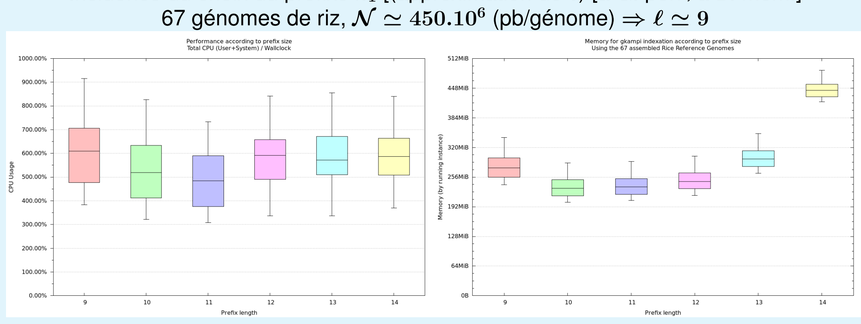
En outre, la norme IUPAC permet de représenter avec une seule lettre toutes les combinaisons qu'il est possible de réaliser avec tout ou partie des 4 nucléotides. Cela représente 16 combinaisons, soient au minimum 4 bits par lettre au lieu de 2 (donc $2\times$ plus).

⇒ ignorons les k-mers qui contiennent des nucléotides dégénérés.

13 Le choix de la taille préfixe

Incidence du choix du préfixe k_1 [(approximation de ℓ) [à G. perf ; à D. mem.]

67 génomes de riz, $\mathcal{N} \simeq 450.10^6$ (pb/génome) ⇒ $\ell \simeq 9$

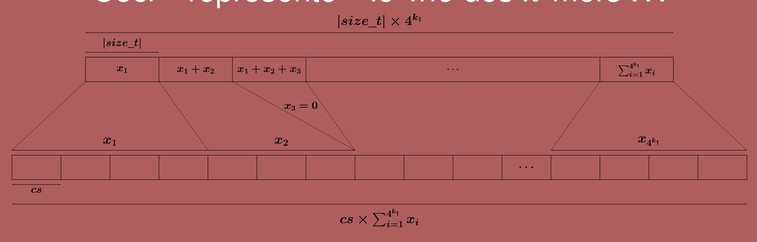


k_1 ?

Vais-je pouvoir intégrer cette librairie ?

14 La structure de données

Ceci « représente » le *Trie* des k-mers …



Cette structure se découpe et distribue facilement.

⇒ Lecture en deux fois des k-mers à partir des fichiers.

⇒ Formats acceptés : **`Fasta`**/**`Fastq`** (compression **`gzip`** possible).

Le format Fasta ?

Le format Fastq ?

16 Format Fasta

Entête 1 seule ligne commençant par `>` ou `;`

```
>␣description␣sequence
ACGTA␣␣GTCAGCA␣;␣commentaire
␣␣AYNG␣;␣AATTT␣(commentaire)

KWSM
␣;␣Ceci␣peut␣etre␣considere␣comme
;␣un␣commentaire␣sur␣cinq␣lignes,
;␣mais␣en␣realite,␣cela
;␣represente␣un␣commentaire
; et␣quatre␣sequences␣vides...
```

[Pearson & Lipman, 1988]

Séquence 1 ou + ligne(s)

⇒ codage IUPAC

⇒ espaces et lignes vides autorisés

⇒ commentaires possible

17 Format Fastq

Entête 1 seule ligne commençant par `@`

Séparateur 1 seule ligne commençant par `+` (suivie de l'entête ou vide)

```
@␣description␣sequence
ACGTA␣␣GTCAGCA␣
␣␣AYNG␣

KWSM
+␣description␣sequence
%*A0_B~ab␣␣:

␣␣@~@!␣
AG
+
1=
```

[Li *et al.*, 2008] [Cock *et al.*, 2010]

Séquence 1 ou + ligne(s)

⇒ codage IUPAC

⇒ espaces et lignes vides autorisés

score qualité associé à chq posit° de la séquence

⇒ une ou plusieurs lignes

⇒ codage ASCII du code 33 (`!`) au code 126 (`~`)

⇒ espaces autorisés (ou pas ?)

⇒ lignes vides autorisées

15 Implémentation

`libGkArrays-MPI` : CeCILL-C **`gkampi`** : CeCILL

C++ doxygen git (valgrind) (gdb) OpenMP (Open-MPI) (autotools)

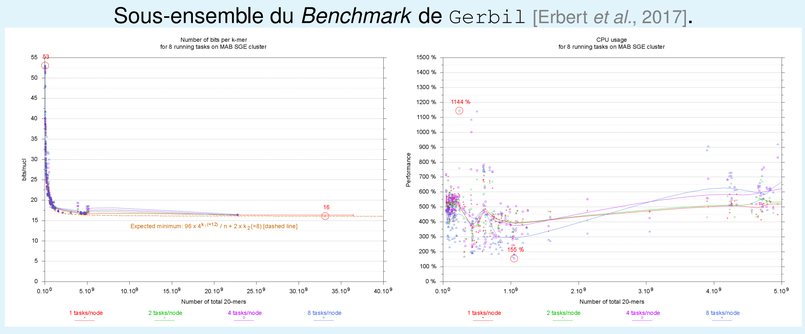
`https://gite.lirmm.fr/doccy/libGkArrays-MPI`

Ohloh Line Count Summary

Language	Files	Code	Comment	Comment %	Blank	Total
autoconf	27	86859	418	0.5%	12442	99719
shell	33	38843	6691	14.7%	4577	50111
make	15	10299	1436	12.2%	1116	12851
cpp	44	10144	6860	40.3%	1630	18634
tex	1	5619	2941	34.4%	731	9291
automake	15	650	1094	62.7%	161	1905
c	2	81	169	67.6%	83	333
perl	1	35	15	30.0%	7	57
Total	138	152530	19624	11.4%	20747	192901

18 Quelques résultats

Sous-ensemble du *Benchmark* de `Gerbil` [Erbert *et al.*, 2017].



Une Démo!

19 Arrivée

Vous avez envie d'en savoir plus ? De voir des cas concrets d'utilisation de **`gkampi`** ? De découvrir l'API de la **`libGkArrays-MPI`** ?

Venez assister à la démonstration qui aura lieu jeudi à 11h35 en salle 14.

Vous pourrez y poser toutes vos questions.

Index distribué de k-mers permettant leur comptage et leur localisation

`libGkArrays-MPI/gkampi`

Clément AGRET
<clement.agret@lirmm.fr>

Annie CHATEAU
<annie.chateau@lirmm.fr>

Alban MANCHERON
<alban.mancheron@lirmm.fr>

LIRMM, CNRS Université Montpellier 2 - CC 477
161, rue ADA 34095 Montpellier CEDEX 5

https://gite.lirmm.fr/doccy/libGkArrays-MPI

Le comptage de k-mers

Les progrès des 15 dernières années en matière de séquençage d'ADN et d'ARN, associés à la baisse de leurs coûts ont eu comme effet direct une production massive de données à analyser. Ce changement d'échelle du volume de données a induit l'apparition de nouvelles méthodologies et notamment celles basées sur le comptage des k-mers– fragments de longueur k – présents dans les séquences. Ces comptages peuvent être utilisés de différentes manières. Par exemple, rechercher des marqueurs spécifiques de certaines populations ou bien pour discriminer les erreurs de séquençage des variations biologiques...

Bien que le comptage de k-mers consiste à associer à une séquence de k nucléotides une valeur entière, il existe de multiples manières de structurer cette information et de l'interroger [1] et les choix algorithmiques et méthodologiques ont une incidence forte sur les performances et la fiabilité des méthodes. De nombreux outils ont été développées pour effectuer ces comptages, tels que `Jellyfish` [2], `DSK` [3], `KMC3` [4], ... Cependant, l'inconvénient commun à la plupart des méthodes existantes actuellement est qu'elles sont limitées par les capacités matérielles de la machine sur laquelle elles sont exécutées. Aussi pour compter les k-mers sur de très gros volumes de données, est-il nécessaire de disposer de machines surpuissantes ou d'adapter les méthodes existantes afin de distribuer les calculs (stratégies *MapReduce* [5]). Nous avons développé une méthode originale permettant de distribuer le calcul sur plusieurs machines, repoussant *de facto* les limitations de ces autres outils.

Comptage et indexation massivement parallélisés de k-mers

Nous avons développé une librairie en `C++` (intitulée `libGkArrays-MPI` et distribuée sous la licence libre `CeCILL-C`), exploitant le parallélisme léger (*multithreading*) mais également le calcul distribué, permettant de compter les k-mers des séquences décrites dans un ou plusieurs fichiers (`fasta`, `fastq`, compressés ou non). Outre le simple comptage, cette librairie permet également de les indexer (donc de pouvoir retrouver leurs séquences d'origine). Sur la base de cette librairie, nous avons également développé un outil (intitulé `gkampi` et distribué sous licence libre `CeCILL`) pouvant s'exécuter sur une simple machine comme sur un *cluster* de calcul.

L'outil `gkampi` et la librairie `libGkArrays-MPI` permettent également de compter/indexer des k-mers espacés [6], proposent les même fonctionnalités que les outils standards (`Jellyfish`, `KMC`, ...) et sont documentés. Leur installation est conforme aux standards des `GNU autotools` et le code respecte strictement la norme ISO 2011 du `C++`.

Remerciements

Ce travail a été en partie financé par l'Institut de Biologie Computationnelle de Montpellier, le projet *GenomeHarvest* (*Agropolis foundation*) et le projet *Coalab* (région Languedoc-Roussillon). Nous tenons également à remercier les rapporteurs pour leurs commentaires.

Bibliographie

[1] Hilde VINJE, Kristian Hovde LILAND, Trygve ALMØY et Lars SNIPEN :
Comparing k-mer based methods for improved classification of 16S sequences.
BMC *Bioinformatics*, 16:205, 2015.

[2] Guillaume MARÇAIS et Carl KINGSFORD :
A fast, lock-free approach for efficient parallel counting of occurrences of k-mers.
Bioinformatics, 27(6):764–770, mars 2011.

[3] Guillaume RIZK, Dominique LAVENIER et Rayan CHIKHI :
DSK : k-mer counting with very low memory usage.
Bioinformatics, 29(5):652–653, 2013.

[4] Marek KOKOT, Maciej DŁUGOSZ et Sebastian DEOROWICZ :
KMC 3 : counting and manipulating k-mer statistics.
Bioinformatics, 33(17):2759–2761, 2017.

[5] Tao GAO, Yanfei GUO, Yanjie WEI, Bingqiang WANG, Yutong LU, Pietro CICOTTI, Pavan BALAJI et Michela TAUFER :
Bloomfish : A Highly Scalable Distributed k-mer Counting Framework.
*In 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (*ICPADS*)*, pages 170–179, décembre 2017.

[6] Karel BŘINDA, Maciej SYKULSKI et Gregory KUCHEROV :
Spaced seeds improve k-mer-based metagenomic classification.
Bioinformatics, 31(22):3584–3592, novembre 2015.