Projet ALG 2022 : read organizer

> V1 (26/10/2022)

Notes de versions:

- V1 : version initiale

Il est à effectuer en **binômes**.

Il devra être rendu au plus tard le **14 décembre 23h59**, par mail aux adresses : claire.lemaitre@inria.fr, pierre.peterlongo@inria.fr. L'objet du mail sera "*[UE ALG] Projet NOM1_NOM2*" ou vous remplacerez NOM1_NOM2 par vos noms de famille, dans l'ordre alphabétique.

Ce mail contiendra en pièce jointe un fichier compressé .gz qui contiendra le code et les rapports (**ne pas envoyer de données**).

Dans les grandes lignes

Le but de ce projet est de réorganiser les reads issus de séquençages afin qu'un compresseur générique type gzip compresse mieux ces données. L'idée est que plus les reads similaires sont proches physiquement dans le fichier à compresser, plus la compression est efficace.

Il s'agira donc de proposer et d'implémenter un algorithme permettant, étant donné un fichier de reads pris en entrée, d'écrire ces mêmes reads dans un ordre différent, optimisant la compression en aval de ces reads.

Le sujet est volontairement assez ouvert, à vous de trouver des idées qui soient à la fois raisonnables à mettre en oeuvre d'un point de vue implémentation, raisonnables en termes de temps de calcul et de consommation mémoire et qui permettent une bonne optimisation de la compression.

Exemple basique d'utilisation:

```
python read_organizer.py -i myreads.fa -o myreads_organized.txt
gzip myreads_organized.txt
du -k myreads_organized.txt.gz     # pour obtenir la taille en Ko du fic
```

Précisions

Le format de sortie sera un simple fichier texte avec sur chaque ligne la séquence d'un read. La compression est dite "avec perte". Du fichier brut on perd le header de chaque read, et sa qualité s'il s'agit de données au format fastq. D'autre part l'ordre original des reads est perdu.

L'idée de fond de ce projet est de vous pousser à réfléchir pour proposer diverses techniques pour regrouper les reads similaires. Vous devrez proposer **au minimum 2 techniques**, qui pourront être appelées avec le même programme grâce à un paramètre.

La qualité du groupement obtenu sera mesurée uniquement par la taille du fichier compressé `myreads_organized.txt.gz` en comparaison de la taille du fichier initial (sans les headers) compressé `myreads.txt.gz` : le taux de compression est défini comme la taille du fichier initial compressé (sans les headers) divisé par la taille de votre fichier de sortie compressé. Par exemple :

```
python read_organizer.py -i myreads.fa -o myreads_organized.txt
gzip myreads_organized.txt
grep -v "^>" myreads.fa | gzip > myreads.txt.gz
du -k myreads*.gz
1148    myreads.txt.gz
332     myreads_organized.txt.gz
```

Le taux de compression dans cet exemple est 1148/332 = 3.46. L'approche de ré-organisation des reads permet de diviser par 3.46 la taille du fichier compressé par rapport à gzip seul.

Données

Nous vous fournirons divers jeux de données pour tester votre programme et ses paramètres, qui seront mis à disposition sur Moodle :

- des reads de type Illumina, issus de 100 Kb du génome d'E. coli, à différentes profondeurs de séquençage
- des reads de type Illumina, issus de 1 Mb du chromosome 1 humain, à différentes profondeurs de séquençage

Rapports et documentation

À écrire en français ou en anglais, selon votre préférence.

Court document README

Le fichier README permettra en quelques lignes d'indiquer à quoi sert votre outil, et comment utiliser le programme. Ce document est destiné à des utilisateurs non experts.

Rapport développeur

Ce rapport explicite la structure du programme et précise le fonctionnement des fonctions clefs.

Il doit être court (max 3 pages), au format pdf.

Rapport scientifique

12 pages ((très) grand) max, format pdf. L'idée est de donner le contexte, d'exposer les techniques utilisées pour organiser les reads, et de donner les résultats obtenus en fonction :

- de la technique utilisée et des valeurs de ses paramètres. Quelle technique, avec quels paramètres, permet d'obtenir les meilleurs résultats. Quelles sont les éventuelles contreparties.
- des jeux de données (couverture, type de génome)

Présentez vos résutats à la manière d'un article scientifique: **1.** résultats bruts (taux de compression, temps de calculs, mémoire utilisée) **2.** faire parler ces résultats : discussions et conclusions.

Misc.

Précisions

- Respectez *a minima* le nom du programme, les options et format d'affichage imposés (cf. exemple de ligne de commande plus haut). Vous pouvez ajouter d'autres options si vous le souhaitez, mais votre code sera utilisable avec ces seules options (correction automatique)
- Votre programme ne sera pas interactif. Une fois lancé avec ses arguments, il terminera sans intervention de l'utilisateur.

- Votre code sera largement commenté et les noms de classes, de variables et de fonctions devront avoir un sens.
- Nous fournirons au fil du temps divers jeux de données permettant de tester vos outils. Le sujet et les données seront disponibles sur moodle.
- L'utilisation de bibliothèques extérieures devra se limiter au strict minimum. À voir avec nous si vous avez un doute.
- L'utilisation d'un package type `argparse` pour gérer les options du programme est vivement conseillée.
- En cas d'erreur utilisateur lors de l'appel de votre programme, affichage d'un message détaillé des options possibles.

Notation

La notation de vos projets (rendus à temps et respectant le format du mail de rendu) prendra en compte les aspects suivants :

- Stratégies proposées pour la réorganisation des reads.
- Choix algorithmiques et succès de l'implémentation.
- Organisation, lisibilité et commentaires du code.
- Qualité des rapports et des analyses effectuées.
- Ne sous-estimez pas l'importance des rapports et la forme du code. Ces deux aspects représentent environ 50% de la note finale.

Aide au codage

Des suggestions pour faciliter la mesure des performances de votre programme sont proposées dans le fichier `aide_au_codage.md`.