Mapping sur graphe de variation : Tutoriel rapide

Construction de graphe

```
vg construct -M {MSA}.fasta -m 1000 > {graphe}.vg
```

{MSA} : fichier fasta d'alignement multiple /!\ Les nucléotides sont exclusivement ACTG
-M: utilisé pour construire à partir d'un alignement multiple (MSA)
-m: taille d'un noeud du graphe maximal, 1000 utilisé ici pour ne fixer "aucune" limite.

graph.vg : graphe au format vg

Indexation du graphe n°1

```
vg index {graph}.vg -x {name}.xg
```

Utilisé pour produire le fichier d'indexation .xg nécessaire à la simulation. Obtention possibles d'autres formats.

graph.vg : nom du graphe d'extension .vg utilisé
-x : instruction d'obtention du .xg
{name}.xg : Nom souhaité à entrer pour le fichier de sortie

Simulation des reads

```
vg sim -n {number} -l {length} -e {SNP} -x {index} -i {indel} -A -J > \
{reads}.json # => format pour GraphAligner

vg sim -n {number} -l {length} -e {SNP} -x {index} -i {indel} -A -a > \
{reads}.gam # => format pour Giraffe
```

Caractéristiques des reads (optionnelles mis à par -n et -l)
-n : nombre de reads souhaités
-l : longueur des reads
-e : taux de substitution souhaité (0.1...)
-x : fichier.xg produit précédemment

Principales options des paths sélectionnées:
-A : Simule à partir de paths aléaoires
-P : {name} : Simule à partir de la path "name"

Format des fichier de sortie :
-J : json
-a : gam

Mapping par GraphAligner

```
    GraphAligner -g {graph}.vg -f {read}.fasta -a {mapping}.gaf -x vg
```

/!\ Requiert un fasta => nécessite un script / outil pour convertir de json à fasta

```
# -g : entrée du graphe .gfa / .vg
# -f : entrée des reads
# -a : Nom du mapping, détecte l'extension souhaitée ( gaf / gam / json )
# -x : optimisation du mapping pour l'extension de graphe choisie. (A défaut d'entrer des paramètres plus élaborés)
```

Convertion VG -> GFA pour auto indexation

```
    vg view -V {graphe}.vg -g > {name}.gfa

# -V : entrée du graphe en .vg
# -g : demande d'une sortie en gfa
```

Auto indexation du graphe pour mapping giraffe

```
    vg autoindex --workflow giraffe -g {graphe}.gfa -p {name}

## Construit l'index de distance / minimizer / graphe gbz pour le mapping giraffe
# --workflow : désigne le type de mapping pour lequel seront utilisés les index (ici giraffe)
# -g : graphe entré en gfa
# -p : préfixe pour les fichier en sortie (ici donne name.gbz name.min name.dist)
```

Mapping giraffe

```
    vg giraffe -Z {graph}.gbz -d {index}.dist -m {minimize}.min -G {reads}.gam -o {format} > {mapping}

# -Z : entrée du graph gbz
# -d : index distance
# -m : minimizer
# -G : read entré ici en gam
# -o : précise le format de sortie souhaité, détecte l'extension précisée (name.gaf, name.json...)
```

Liste résumée :

```
vg construct -M MSA.fasta -m 1000 > graphe.vg

vg index graphe.vg -x name.xg

vg sim -n 10 -l 50 -e 0.1 -x  name.xg -i 0.1 -A -J > reads.json # à convertir en fasta
vg sim -n 10 -l 50 -e 0.1 -x name.xg -i 0.1 -A -a > reads.gam

GraphAligner -g graph.vg -f read.fasta -a mapping.gaf -x vg
```

```
vg view -V graphe.vg -g > graphe.gfa

vg autoindex --workflow giraffe -g graphe.gfa -p name

vg giraffe -Z name.gbz -d name.dist -m name.min -G reads.gam -o gaf >
mapping.gaf
```