Annotation des éléments transposables du moustique tigre *Aedes albopictus*

Auteur:
Tommaso
BARBERIS
tommaso.barberis@etu.univ-lyon1.fr

Encadrant:
Matthieu
BOULESTEIX
matthieu.boulesteix@univ-lyon1.fr

Janvier - Juillet 2022

Résumé

Aedes albopictus, mieux sous le nom de moustique tigre, est un des vecteurs de certaines maladies virales, comme la dengue et le chikungunya. Des études récentes on montré que son génome est peuplé de nombreux éléments transposables (ET).
Dans ce stage, nous nous sommes ainsi consacré à leur annotation, à l'aide d'une double approche homologie - *de novo* et, à la construction de la librairie d'ET conséquente. Les séquences contenues dans la librairie ont été polies à l'aide d'un pipeline conçu pour l'occasion et, classifiées pour connaître le paysage d'ET du génome de cette espèce.

This work was performed using the computing facilities of the CC LBBE/PRABI.

Table des matières

Table des figures

Table des tableaux

Abréviations

DRP Discordant Read Pair. 3
ET éléments transposables. ii–iv, vi, 1–9, 11–15, 17–28, 33
LINE Long Interspersed Nuclear Elements. 13–15, 23–26, 28
LR long reads. 4, 5, 12, 20
LTR Long Terminal Repeats. iii, vi, 3–6, 8, 9, 12–15, 17, 22–28
MITE Miniature Inverted-repeat Transposable Elements. 12–15, 19, 23, 28
OCTFTA OneCodeToFindThemAll. 8, 9, 17, 27
ORF Open Reading Frame. 27
PBS Primer Binding Site. iii, vi, 5
PLE Penelope-Like Element. 24
RC Rolling-circle (Helitron). 24
RT Retrotranscriptase. vi
ShR short reads. 4
SINE Short Interspersed Nuclear Elements. 13–15, 23, 24, 26, 28
SR Split-Read. 3
SSR Simple Sequence Repeats. 9, 10, 20, 22–24, 27
TIPs Transposon Insertion Polymorphisms. 3
TIR Terminal Inverted Repeats. iii, vi, 3, 5, 6, 28
TSD Target Site Duplication. iii, vi, vii, 3, 5

Glossaire

Terminal Inverted Repeats (TIR) : il s'agit d'une répétition sur les extrémités d'une séquence, où une extrémité est le complémentaire inverse de l'autre.

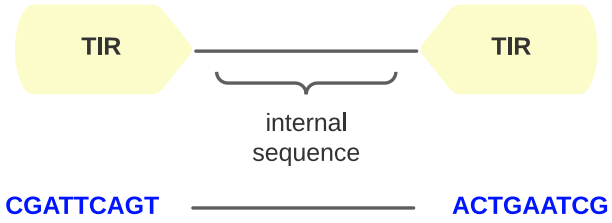

FIGURE G1 – Exemple de TIR.

Long Terminal Repeats (LTR) : c'est une paire de séquences d'ADN identiques, qui peuvent avoir une longueur jusqu'à quelques centaines de nucléotides. Dans les rétrotransposons (Classe I) à LTR, ces deux séquences se trouvent aux extrémités 5' et 3' de la séquence.

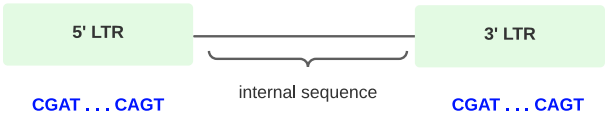

FIGURE G2 – Exemple de LTR.

Primer Binding Site (PBS) : il s'agit de la région nucléotidique où des amorces d'ADN peuvent venir se hybrider pour permettre à la réplication d'avoir lieu. Dans les ET du type LTR, le PBS a une taille qui est comprise entre 20 et 25 nucléotides, il est le complémentaire d'un *tARN* et se trouve en 3', à quelque nucléotide près, du LTR en 5'. Il est donc utilisé comme amorce pour la Retrotranscriptase (RT) [1].

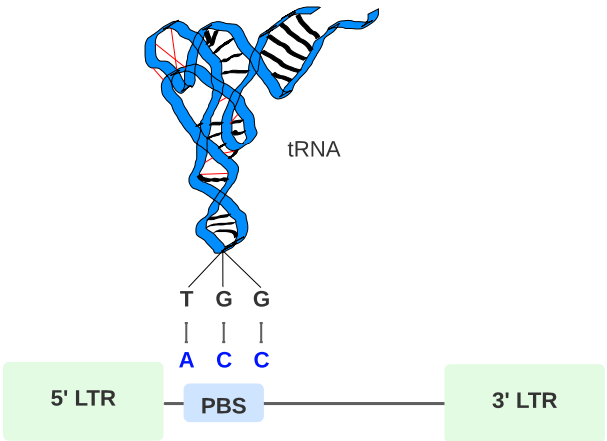

FIGURE G3 – Exemple de PBS.

Target Site Duplication (TSD) : il s'agit généralement d'une courte répétition directe qui est obtenue sur les régions flanquantes d'ET après son insertion.

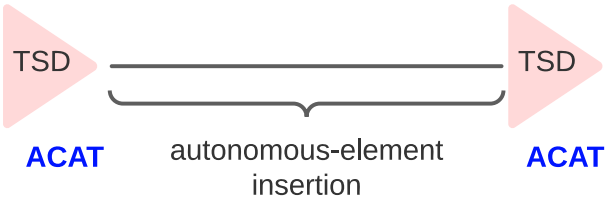

FIGURE G4 – Exemple de TSD.

Logiciels et Bases de données

Logiciels

- RepeatModeler - 2.0.2
  Commentaire de la ligne 140 :
  ```
  # use Devel::Size qw(size total\_size);
  ```
  - RECON - 1.08
  - RepeatScout - 1.0.6
  - Tandem Repeats Finder - 4.09
  - RMBlast - 2.11.0
  - LtrHarvest (genometools) - 1.5.9
    - https://github.com/genometools/genometools/pull/917/files, changements sur les fichiers :
      - src/ltr/ltr_cluster_prepare_seq_visitor.c
      - src/ltr/ltr_cluster_stream.c
      - src/match/eis-bwtseq-context.c)
    - https://github.com/genometools/genometools/pull/857/files, changements sur les fichiers :
      - src/extended/gff3_escaping.c
      - src/match/eis-bwtseq-context.c
    - compilé avec la ligne de code suivante :
      ```
      make threads=yes cairo=no
      ```
  - LTR_retriever - 2.6
    - BLAST+ package - 2.12.0
    - cdhit - 4.8.1
    - HMMER - 3.3.2
    - RepeatMasker - 4.1.2 p1
      - Module Python installé via pip : h5py - 3.6.0
  - MAFFT - 7.490
  - NINJA - 0.98-cluster_only
  - UCSC TwoBit Tools - 1.04.00
  - modules Perl (compilés à partir du code source) :
    - JSON - 4.05
- MITE-Tracker - commit 23ee261 (avec python3)
  - vsearch - 2.7.1
  - modules Python (installés via conda) :
    - biopython - 1.79
    - numpy - 1.22.1
    - pandas - 1.4.0
- singularity - 3.7.2
- conda - 4.12.0
- EDTA - 2.0.0 (avec *perl - 5.26.2*)
- One code to find them all - 1.0
- bedops - 2.4.40
- samtools - 1.15
- bedtools - 2.30.0
- seqtk - 1.3
- ART - 2.5.8
- DeepTE - commit 69a7465
- dnaPipeTE - v.1.3.1_07
- CIAlign - 1.0.15
- TE-Aid - v.0-dev

Bases de données

- RepBase - 25.08
- Dfam - 3.5

1 Introduction

S'intéresser aux mécanismes d'évolution et d'adaptation d'une espèce est le point clé de la biologie évolutive. Celle-ci permet d'étudier et comprendre en détail le comportement ainsi que les environnements où l'espèce habite. Par exemple, elle permet d'étudier les mécanismes évolutifs qui ont permis au moustique tigre *Aedes albopictus* de coloniser le monde entier dans les dernières décennies.

Le moustique tigre (FIGURE 1), est un insecte diptère de la famille *Culicidae* provenant de l'Asie du sud-est qui peut être reconnu facilement grâce à ses rayures blanches et noires qu'il porte sur les pattes. Il est de couleur noir et il porte sur le thorax une ligne blanche. Le genre *Aedes* regroupe environ 263 espèces dont aussi *Aedes aegypti*, lequel se distingue de l'espèce sœur par ses marque blanches en forme de lyre sur le thorax et sa couleur marron.

FIGURE 1 – Image représentative du moustique tigre.

Photo DR

Pourquoi étudier l'évolution de *Ae. albopictus* ? L'étude d'*Ae. albopictus* a pris une importance depuis sa propagation dans le monde entier, en devenant un sujet de recherche de plus en plus ciblé, mais également, du fait de son implication en tant que vecteur principal de certains arbovirus (virus de la dengue et du chikungunya) lors d'épidémies récentes [2].
Dans les 30 dernières années, il a colonisé l'Afrique, l'Europe et l'Amérique à partir de l'Asie de l'Est, principalement grâce au commerce de pneus usagés, lesquels sembleraient être un endroit préférentiel pour la ponte des oeufs [3][4].

Le moustique tigre, comme plusieurs autres espèces du genre *Aedes*, est vecteur de maladies infectieuses d'origine virale, on a déjà cité la dengue et le chikungunya, mais on retrouve aussi beaucoup d'autres virus [5][6] (parmi eux, on suspecte aussi la transmission du virus zika). Aujourd'hui plusieurs de ces maladies n'ont pas un traitement spécifique disponible, pour cette raison, le contrôle des vecteurs reste la meilleure stratégie pour les contrer.
La biologie évolutive permet notamment d'étudier les caractéristiques d'*Aedes albopictus* qui lui ont permis de survivre et de s'adapter à des environnements qui lui étaient défavorables dans le passé et, ainsi potentiellement d'imaginer des nouvelles méthodes de lutte.
Il a pu s'adapter rapidement en Occident malgré le climat tempéré, par opposition au climat tropical et humide de sa région d'origine. Plusieurs hypothèses sur l'acquisition de cette nouvelle capacité peuvent être faites, comme la symbiose avec un nouveau partenaire (viral ou bactérien) ou encore l'influence du changement climatique.
Cependant, ici on s'intéressera uniquement à l'adaptation de l'espèce via des événements évolutifs provoqués par des structures génomiques particulières, appelés éléments transposables (ET) lesquels seront

décrits dans le paragraphe suivant.

Les éléments transposables Les éléments transposables sont des séquences d'ADN qui possèdent la capacité de se déplacer/répliquer au sein du génome sur lequel ils se trouvent, grâce à un mécanisme de duplication « copy-and-paste » [7] ou de déplacement « cut-and-paste ». Cette propriété qui les rend mobiles, peut être à l'origine de modifications génomiques lesquelles peuvent avoir un impact sur l'évolution de l'espèce.

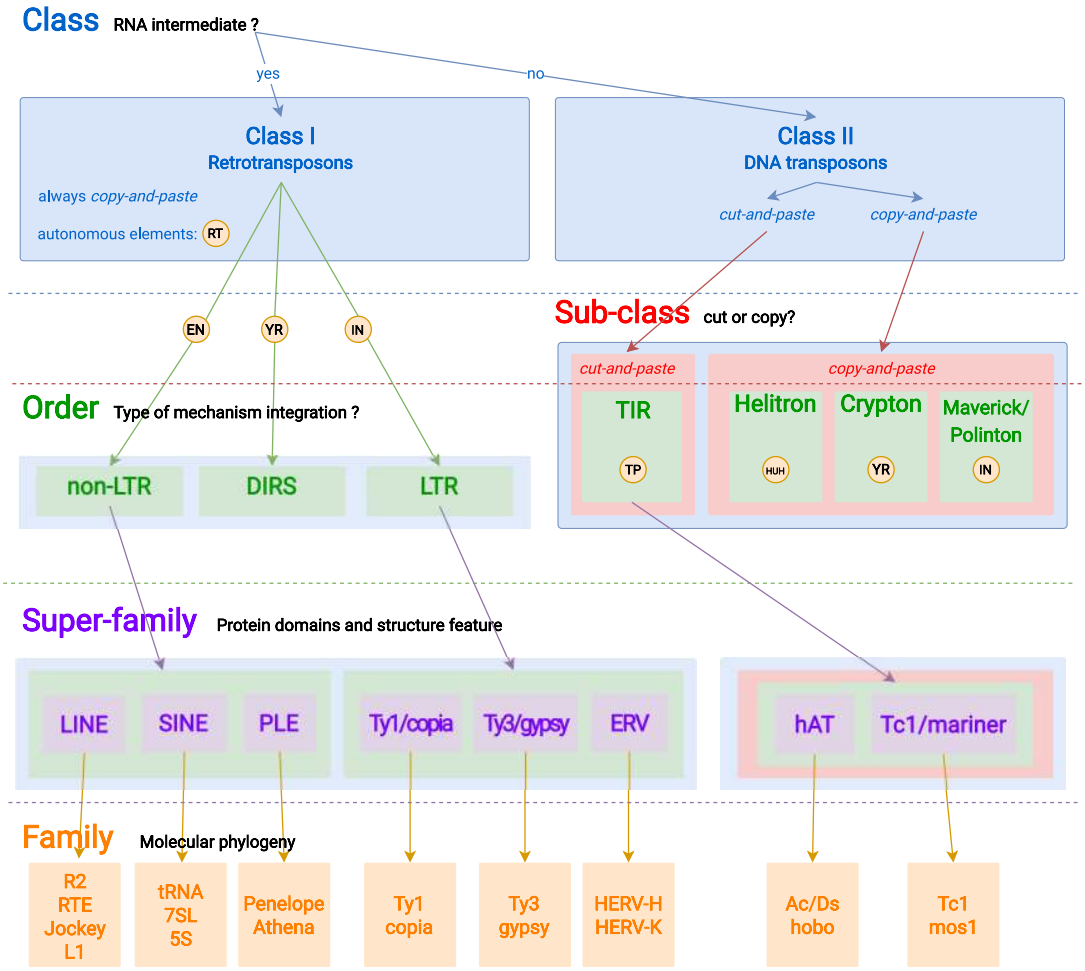

FIGURE 2 – Classification des éléments transposables (ET).

Chaque couleur (bleu, rouge, vert, violet et orange) représente un niveau dans la hiérarchie. Les ronds orange représentent les domaines protéiques (pour les éléments autonomes) associés au niveau considéré.
RT : retrotranscriptase, EN : endonuclease, YR : recombinase à tyrosine, IN : integrase, TP : transposase, HUH : hélicase.
Sources : *Wicker et al.* [1], *Borque et al.* [7].

Les éléments transposables ont été découverts par Barbara **McClintock** dans les années 1940. Dans ses travaux, elle a réussi, en particulier, à comprendre le rôle des ET dans la cassure chromosomique et ainsi dans le changement phénotypique (couleur de la graine) chez le maïs [8]. Depuis, ce qui été considéré avant comme de l'ADN « poubelle », en raison de sa nature hautement répétée et de l'absence de fonctions biologiques évidentes pour l'organisme, est aujourd'hui de plus en plus étudié.
La plupart des eucaryotes possèdent des éléments transposables et, leur proportion au sein du génome peut varier d'une espèce à l'autre [9]. Par exemple, le génome du maïs a un contenu en ET très élévé (85% du génome) [10] alors que le génome humain possède 45% d'ET [11] et le génome de *Drosophila*

melanogaster n'en possède "que" 20% [12].
Les ET peuvent être organisés selon une vraie hiérarchie [1] (FIGURE 2). Ils se divisent en deux classes principales, les éléments de classe I, ou rétrotransposons, qui utilisent un intermédiaire à ARN pour se déplacer et, les éléments de classe II, ou transposons à ADN. Tous les éléments de classe I utilisent un mécanisme de réplication *« copy-and-paste »*, alors que dans les classe II on retrouve aussi des ET utilisant un mécanisme de *« cut-and-paste »*. Cette discrimination est souvent utilisée pour séparer les transposons à ADN en deux sub-classes.
Chaque classe (ou sub-classe) se divise ensuite en plusieurs ordres, en fonction du type du mécanisme d'intégration utilisé et, encore plus en détail, ils peuvent être classifiés selon une super-famille et une famille. Les super-familles se distinguent les unes des autres par les domaines protéiques portés ainsi que les différentes caractéristiques structurales (Long Terminal Repeats (LTR), Terminal Inverted Repeats (TIR) et Target Site Duplication (TSD)) [7].
Les familles se différencient les unes des autres par des phylogénies moléculaires différentes. Historiquement, deux séquences d'ET sont attribuées à la même famille si elles respectent la règle 80-80-80 proposé par *Wicker* [1] ; si elles partagent au moins 80% d'identité sur au moins 80% des leurs régions codantes (généralement « banalisé » sur au moins 80% de la longueur de l'alignement entre les deux séquences) et qu'elles ont au moins 80pb.

1.1 Contexte scientifique

Naissance de la question scientifique. Le sujet de ce stage naît à partir d'un projet intitulé *MosquiTEs* et coordonné par M. Boulesteix, lequel est aussi tuteur de ce stage.
Le projet vise à analyser le rôle des éléments transposables dans l'adaptation d'*Ae. albopictus* dans des climats tempérés.
La stratégie qui sera suivie dans le projet pour déterminer l'empreinte des ET dans l'adaptation est double : d'une part il y a le génotypage des nouvelles insertions d'ET de nombreux individus d'*Ae. albopictus* (projet 1000 génomes) ; et d'autre part la caractérisation d'un point de vue moléculaire et macroscopique des phénotypes acquis grâce à ces insertions.

Dans ce stage, on s'intéresse principalement à l'annotation des éléments transposables sur le génome d'*Ae. albopictus*. La librairie obtenue sera utilisée par la suite pour le génotypage.
Deux stratégies principales sont utilisées pour le génotypage : la méthode dite Split-Read (SR), s'appuyant sur la propriété des lectures provenant d'une nouvelle insertion de chevaucher, d'une part une extrémité d'un élément annoté et de l'autre part le génome de référence et ; la méthode dite Discordant Read Pair (DRP), laquelle se sert des lecture paired-end pour explorer un possible Transposon Insertion Polymorphisms (TIPs). Par exemple, il explore deux lectures d'une paire ayant les extrémités s'alignant à des endroits différents, ou orientées de manière opposée ou encore si une seule des deux lecture s'aligne à la référence.
Il est donc d'importance capitale que les séquences d'ET contenues dans la librairie soient les plus complètes possibles, en particulier sur les extrémités, afin d'améliorer le résultat du génotypage.

1.2 État de l'art

Annotation des ET La construction d'une librairie d'éléments transposables est une étape fondamentale pour l'annotation d'un nouveau assemblage d'un génome. La détection d'ET est généralement réalisée pour masquer les régions répétées du génome et permettre ainsi une meilleure annotation des éléments fonctionnels, comme les gènes.
Que ce soit pour cette dernière raison ou pour annoter réellement les ET, l'annotation des ET chez *Ae. albopictus* a été déjà réalisée par plusieurs auteurs (Tableau 1). Le contenu d'éléments transposables chez *Ae. albopictus* semblerait varier entre 40% et 55% en ayant une quantité plus ou moins équivalente selon l'annotation, de rétrotransposons et transposons à ADN.

	***Palatini et al*, 2020**	***Melo et al*, 2020**	***Goubert et al*, 2015**
Rétroéléments	22.69%	17.94%	17.67%
dont LTR	5.6%	7.84%	4.79%
Transposons à ADN	15.82%	7.4%	15.43%
Non classifiés et autres éléments répétés	17.3%	10.58%	16.63%
Total	**55.81%**	**43.76%**	**49.73%**

Table 1 – Composition en éléments transposables du génome d'*Ae. albopictus* selon trois annotations différentes.

L'annotation réalisé par *Goubert et al,* [13] a été effectuée avec l'outil (`dnaPipeTE`) qu'ils ont proposé dans leur publication en utilisant des données short reads (ShR). L'annotation proposé par *Melo et al,* [14] a été réalisé en utilisant un assemblage obtenu avec des lectures ShR [15] et le package `REPET` pour l'annotation. Enfin, la librairie proposée par *Palatini et al,* [16] a été réalisée avec leur assemblage obtenu à partir de données long reads (LR) (en utilisant `RepeatModeler`), mais n'a pas été rendue publique.

L'annotation d'ET peut être faite en utilisant trois types d'approches : par homologie (en utilisant une base de données d'ET et un assemblage), par *de novo* en utilisant uniquement un assemblage et par *de novo* mais en utilisant un génome séquencé (données brutes short reads (ShR) ou long reads (LR)) [17].
Pour l'approche par homologie, l'outil qui actuellement représente le standard étant le plus utilisé au sein de la communauté, est `RepeatMasker` [18], lequel est majoritairement utilisé avec deux bases de données : `RepBase` [19] et `Dfam` [20]. `RepBase` est réputée pour être la plus exhaustive ; mais il existe aussi d'autres bases de données qui peuvent être spécifiques à un type d'élément (ex. `SINEBase`).

Pour l'approche *de novo* en utilisant un assemblage, il existe une grande variété d'outils qui permettent l'annotation, mais deux se distinguent des autres par leur large utilisation au sein de la communauté : `RepeatModeler2` [21] et `REPET` [22]. Parmi les outils en utilisant des données de séquençage on retrouve `DNApipeTE` [13]. On mentionne aussi `EDTA` [23] qui est de plus en plus utilisé.
Ici de suite, on donne quelques détails concernant les outils d'annotation utilisés dans ce travail :

- `RepeatMasker` est l'outil de référence pour l'annotation par homologie. Son fonctionnement consiste principalement à « masquer » les séquences contenues dans la librairie fournie sur le génome analysé. Les séquences fournies comme base de données (ex. `RepBase` mais il peut également s'agir d'une librairie personnalisée) sont identifiées grâce un algorithme d'alignement local (`blastn` il en est un exemple).
- `RepeatModeler2` est un pipeline pour l'annotation automatique des ET. Il utilise quatre modules : `RECON` [24] et `RepeatScout` [25] lesquels s'appuient sur la répétitivité des ET et, `LTRharvest` [26] et `LTR_retriever` [27] lesquels s'appuient sur la structure caractéristique des rétrotransposons LTR.
  `RepeatScout` est utilisé pour sa rapidité dans la découverte des familles les plus récentes et abondantes sur le génome. `RECON` utilise des matrices de substitution sophistiquées ainsi que des pénalités pour les gaps affinées, en lui permettant de détecter aussi les familles les plus anciennes.

`LTRharvest` est donc utilisé pour la détection des éléments (grâce à la signature caractéristique des LTR) et `LTR_retriever` pour filtrer les faux positifs détecté par `LTRHarvest`.

☛ `EDTA` utilise plusieurs modules pour l'annotation : `LTRharvest`, `LTRfinder` [28], `LTR_retriever`, `GRF` [29], `TIR-Learner` [30], `HelitronScanner` [31], et aussi `RepeatModeler` qui est utilisé comme filtre.
L'idée à la base du pipeline est la suivante : d'abord il appelle les modules spécifiques à des caractéristiques (LTR, TIR, etc) et puis il masque le génome de référence avec les séquences obtenues (c'est-à-dire que les séquences détectées dans les étapes précédentes sont marquées comme telles ; généralement une séquence masquée est remplacée par des « X » sur le génome de référence utilisé) et ensuite il utilise `RepeatModeler` pour détecter les non-LTR et les éléments non classifiés.

Génome de référence Comme vu dans le paragraphe précédent, l'assemblage du génome de l'espèce sur laquelle on souhaite travailler est un choix important, car la qualité de l'annotation des éléments transposables en dépend.
Pour *Ae. albopictus*, 6 assemblages sont actuellement déposés sur le NCBI [32] dont uniquement 3 génomes sont assemblés au niveau des scaffolds : *Chen et al.* [15], *Palatini et al.* [16] et *Boyle et al.* [33].

Les deux derniers, ont été obtenus en utilisant des données de séquençage de type long reads (LR). Parmi les trois, celui de *Boyle et al.* est le plus récent et le moins fragmenté, (du coup d'une certaine manière c'est le meilleur assemblage) mais à la différence de l'assemblage de *Palatini et al.*, il manque une fraction de 200 Mbp.
En contrepartie, le travail de *Palatini et al.*, présente dans son assemblage de nombreux haplotigs [1] , lesquels faussent la taille estimée du génome, étant en désaccord avec l'estimation par cytométrie en flux (∼2.5 Gbp contre 1.190–1.275 Gbp). Un tableau résumant ces trois assemblages est disponible dans le Tableau S1.

Classification La classification est une étape indispensable pour connaître à quel groupe d'éléments transposables une séquence appartient (voir Figure 2).
Comme pour l'annotation, plusieurs outils existent, mais une méthode précise et répandue au sein de la communauté n'est pas définie. Cependant, plusieurs critères sont souvent retrouvés dans la bibliographie et utilisés. En particulier, il est très commun de trouver des approches similaires à ce décrit dans *Wicker et al,* 2007. Dans ce papier, ils proposent de réaliser la classification en plusieurs étapes.
La première étape consiste en une recherche par homologie des séquences putatives (préalablement identifiées avec l'annotation) contre une base de données d'ET d'acides nucléiques (ADN vs ADN) en utilisant un outil du type `blastn` et en filtrant les résultats selon la règle 80-80-80, pour pouvoir associer les séquences aux familles correspondantes (quand cela est possible).
La deuxième étape consiste toujours dans un approche par homologie, mais cette fois-ci contre une base de données contenant les séquences des domaines protéiques des éléments transposables (ex. `Pfam` ou `RepeatPeps.lib`, provenant de `RepeatMasker`) avec un outil comme `blastx` (ADN traduit vs protéine). Ici l'idée est que, une nouvelle famille d'ET (qui donc n'a pas eu une forte similarité contre la base de données nucléiques), devrait quand même avoir des fortes correspondances avec les domaines protéiques de toutes les autres familles faisant partie de la même super-famille, avec lesquelles elle partage les domaines protéiques.
Dans le cas où la recherche contre les domaines protéiques ne donne aucun résultat, cela pourrait signifier que la famille fait partie d'une super-famille (ou même d'une classe) inconnue ou, alternativement il pourrait s'agir d'un élément non-autonome [2].
Du coup, si les deux premières méthodes ne donnent pas de résultats, une recherche de motifs structuraux (comme TIR, PBS, TSD ou encore LTR ; pour plus de détails, consulter le Glossaire) peut être réalisée.
Si les étapes précédentes échouent à donner un résultat satisfaisant pour la classification, la séquence est généralement marquée comme non-classifiée (*Unknown*).

1. Il s'agit de plusieurs contigs qui provenant d'haplotypes différents. En d'autres mots, un haplotig est la duplication d'un contig en raison de la nature hétérozygote du génome analysé (deux copies du même allèle).
2. ET sans séquence codante fonctionnelle mais pouvant être mobilisé par l'intervention de protéines produites par d'autres ET.

2 Matériel et méthodes

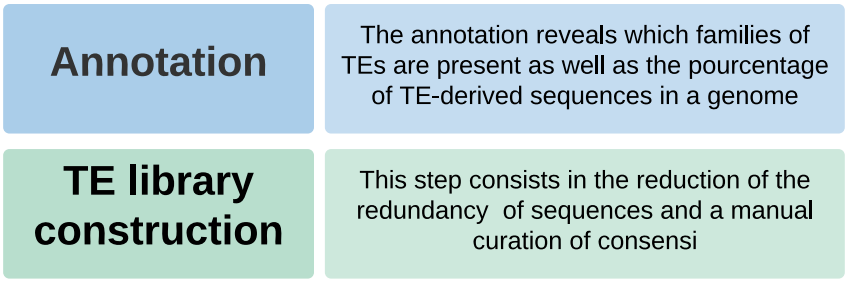

FIGURE 3 – Étapes principales.

En FIGURE 3 on retrouve les étapes qui ont été abordées au cours du projet. La première étape consiste dans la réalisation d'une nouvelle **annotation** des éléments transposables pour pouvoir obtenir des séquences candidates qui seront traitées dans la deuxième étape, la **construction de la librairie**. La librairie produite sera utilisée pour effectuer le **génotypage** de nouvelles insertions (« TE insertion calling ») dans le projet *1000 génomes*.

2.1 Annotation

Pour annoter les familles d'ET présentes dans le génome de référence on utilise un mélange de deux approches différentes et complémentaires : *de novo* et par homologie.

2.1.1 Approche *de novo*

Dans ce type d'approche, les ET sont identifiés grâce à les caractéristiques structurales qui sont spécifiques à chaque ordre (LTR, TIR, etc) ainsi que grâce à leur nature répétée.
On a ainsi utilisé les programmes suivants sur le génome de référence :

- `RepeatModeler2` : les commandes utilisées sont disponible dans le CODE S1.
- `EDTA` [23] : selon les instructions de l'auteur, on a séparés l'analyse en trois exécutions différentes, chacune pour un type d'ET différent (`helitron`, `TIR` et `LTR`). Les commandes utilisées se trouvent ici : CODE S2.
- `MITE-Tracker` [34] : pour cet outil, il a été nécessaire de diviser le génome de référence en 11 fichiers plus petits pour limiter le temps de calcul, chaque fichier a été donc lancé avec l'option `--task candidates`. Un dernier run avec l'option `--task cluster` a permis de réunir les résultats obtenus (CODE S3).

Évaluation des résultats Pour évaluer les librairies obtenues avec les outils *de novo*, on a effectué plusieurs types d'analyses :

- Estimation de la redondance de chacune des librairies séparément (SOUS-SECTION 3.1.1 - ÉVALUATION DE LA REDONDANCE DES DIFFÉRENTES LIBRAIRIES). Pour cela, on a réalisé un clustering de chacune des librairies à l'aide de `cd-hit-est` [35] (`-c 0.8 -G 0 -aS 0.8 -d 0`).

- Exploration la distribution des longueurs des différents consensi (SOUS-SECTION 3.1.1 - ÉVALUATION DES DISTRIBUTIONS DES LONGUEURS). Pour le faire, on a d'abord classifiées les séquences (de manière non définitive) avec `DeepTE` [36] et ensuite on a comparé les longueurs de consensi

issus *de novo* avec ceux contenus en `RepBase`.

- ☛ Réalisation d'une estimation de la couverture génomique (Sous-section 3.1.1 - Évaluation de la couverture génomique). Pour faire cela, on a simulé des lectures *Illumina* (au format `fastq`) à partir du génome de référence en utilisant `ART` [37], `samtools` [38] et `bedtools` [39] (Code S4). Ensuite, on a utilisé le fichier obtenu pour estimer la couverture génomique par des éléments répétés en utilisant `dnaPipeTE` [13] (Code S5).

2.1.2 Approche par homologie

Dans ce cas, les ET sont identifié par similarité avec des séquences d'ET précédemment annotés chez la même espèce ainsi que dans des espèces proches et, qui sont donc contenus dans une base de données. Ici nous avons choisi d'utiliser uniquement les séquences provenant d'*Arthropodes* de la base de données `RepBase v25.08` (*FASTA Edition*) [19]. On a donc utilisé les fichiers suivants (description des fichiers pourvu par les auteurs de `RepBase`) :

- ☛ `angrep.ref` : ET de l'espèce *Anopheles gambiae* ;
- ☛ `drorep.ref` : ET de l'espèce *Drosophila melanogaster* ;
- ☛ `invrep.ref` : ET d'autres espèces invertébrés ;
- ☛ `invsub.ref` : ET de sous-familles d'invertébrés.

2.1.3 Combinaison des deux approches

La combinaison des deux approches permet d'obtenir un certain nombre de familles d'ET fiables grâce à l'annotation par homologie mais également des possibles nouvelles familles grâce à l'approche *de novo*.
On a donc concaténé les librairies générés par les outils *de novo* avec le sous ensemble de `RepBase` pour affiner et construire une librairie (voir Sous-section 2.2) propre d'éléments transposables.

2.2 Construction de la librairie d'ET

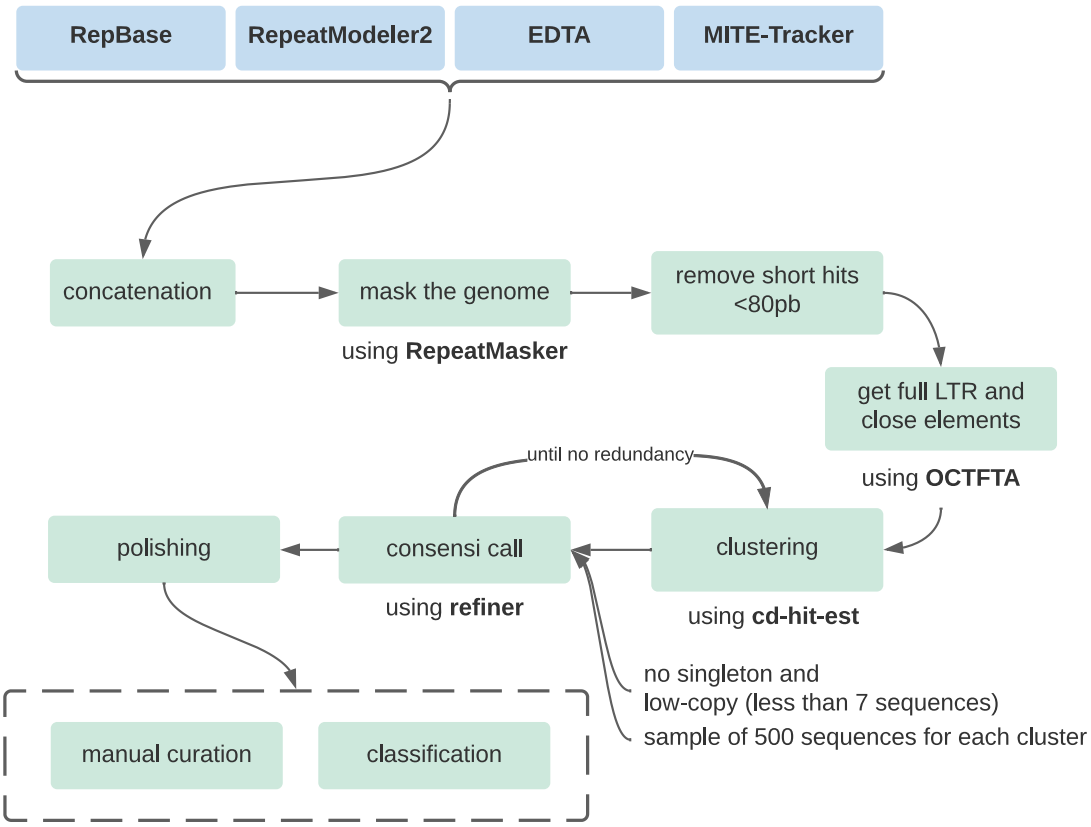

FIGURE 4 – Schéma des étapes suivies pour obtenir la librairie d'ET.

Pour obtenir la librairie d'éléments transposables on a suivi les étapes montrées sur la FIGURE 4. Pour commencer, on a concaténé les séquences obtenues avec les outils *de novo* avec le sous-ensemble de `RepBase`. Dans la suite du document, on se référera à ce fichier avec le terme de libraire « brute » ou « concaténée ».
On a ensuite utilisé le fichier obtenu, comme librairie pour masquer le génome de référence en utilisant `RepeatMasker` [18] (CODE S6) avec la valeur de `-cutoff` la plus stringente possible (250), de manière à limiter le nombre d'éventuels faux positifs et, donc le nombre de séquences à traiter par la suite.
Outre que connaître la portion génomique représentée par des ET, l'outil permet de connaître aussi les positions de toutes les instances.
Pour mieux expliquer, une séquence dans la librairie utilisée (le fichier concaténé), représente une et une seule famille d'ET sous la forme d'un consensus [3]. Or, en raison de leur nature répétée, les ET peuvent être présents en plusieurs copies à différents endroits du génome, étant chaque copie une instance (ou membre) de la famille. Dans le reste du document, « consensus » et « famille » indiqueront la même chose. De même, on pourra inter-changer les termes suivants : « copie », « instance » ou encore « membre ».

En sortie de `RepeatMasker`, on a choisi d'éliminer les fragments ayant une longueur inférieure à 80 pb, pour performer ce filtrage, on a utilisé un script customisé (LIEN 1, se référer à la section LIENS EXTERNES, dans les ANNEXES).

Les fragments restant ont été traités avec `OneCodeToFindThemAll (OCTFTA)` [40] pour reconstituer certaines copies entières. Pour clarifier, il n'est pas rare de trouver dans les bases de données, les LTR et les régions codantes correspondantes (FIGURE G2) dans deux séquences séparées. Pour cette raison,

3. Une séquence consensus est une séquence qui contient les résidus les plus fréquents pour chacune des positions d'un ensemble de séquences. Dans notre cas, il s'agit d'une séquence « moyenne » représentant une famille ET.

`OCTFTA` permet de reconstituer les solo-LTR avec leur élément interne (en se basant sur l'utilisation d'un préfixe commun pour la nomenclature du solo-LTR et de son élément interne dans la base de données). De plus, la détection des ET avec `RepeatMasker` peut aussi retourner des copies fragmentées. Cela peut arriver si le score d'alignement entre le consensus en question et la région génomique analysée descend au-dessous d'un certain seuil entre différents portions du consensus.
`OCTFTA` permet également de fusionner deux fragments se trouvant en proximité sur le génome (`--insert 80`, fusionnement de deux fragments si la distance entre eux est inférieur à 80pb). Les commandes pour `OCTFTA` se trouvent ici :Code S7.
Un script customisé (Lien 2) a été utilisé pour récupérer les séquences de toutes les copies (reconstituées ou pas) dans un seul fichier `FASTA`.

L'ensemble des copies a été utilisé ensuite pour un clustering avec `cd-hit-est` [41] (Code S8) pour réduire la redondance et pour regrouper les différentes instances provenant d'une même famille ensemble en essayant de respecter les conditions de la règle 80-80-80 (décrite ici : Paragraphe 1 - Les éléments transposables). Puis, en utilisant un script customisé (Lien 2), on a séparé les clusters *singleton* (ayant une seule séquence) et *low copy* (ayant 6 ou moins séquences) dans deux fichiers séparés et sur les clusters restant on a utilisé `Refiner` (du package `RepeatModeler2`) pour appeler une séquence consensus à partir d'un échantillonnage de 500 séquences par cluster.
Par la suite, on a réalisé plusieurs cycles `cd-hit-est`/`refiner` pour réduire un maximum la redondance.

2.2.1 Polissage

Chaque séquence a été ensuite polie avec un pipeline customisé (Lien 4) pour essayer d'étendre les consensi, quand cela était possible, en vue du génotypage (comme expliqué dans le Paragraphe 1.1 - Contexte scientifique). Un schéma explicatif du pipeline est montré en Figure 5.
Brièvement, le pipeline prend en entrée une séquence putative d'élément transposable et le génome de référence utilisé pour l'annotation et il retourne la séquence polie.
Plus en détail : pour commencer il effectue un `blastn` [42] de la séquence sur le génome, si plusieurs hits proviennent d'un même endroit génomique la séquence est marquée comme un possible Simple Sequence Repeats (SSR) et le programme se termine. Dans le cas contraire, un échantillonnage aléatoire des hits est effectué (par défaut, récupération de 100 hits dont 25 les plus longs) si nécessaire (prendre en compte tous les hits est extrêmement coûteux en termes de calcul). Ensuite les séquences associées sont extraites avec 1000pb flanquantes (100pb si la séquence de départ fait moins de 1Kb) de plus pour chaque extrémité en utilisant `bedtool slop`. Ces séquences sont donc alignées en utilisant `mafft` [43] et l'alignement obtenu est nettoyé à l'aide de `CIAlign` [44]. Cet outil permet notamment d'éliminer les « insertions » rares de l'alignement (si l'insertion est présente en moins de la moitié des séquences formant l'alignement).
À partir de l'alignement, un nouveau consensus est appelé et, si celui-ci est très similaire au consensus de départ et que les extrémités sont suffisamment couvertes par des hits, le pipeline se termine en retournant la nouveau consensus obtenu.
En revanche, si le consensus est suffisamment différent, une analyse sur la couverture des extrémités est effectuée : si elles sont très couvertes cela signifie que le consensus peut encore s'étendre donc on récupère des nucléotides supplémentaires aux extrémités, en revanche si elles ne sont pas couvertes cela signifie que probablement on a atteint la fin du consensus et qu'il faut donc tronquer les bases en excès récupéré avec l'extension précédente. La séquence obtenue est donc utilisée dans un nouveau cycle `blastn-mafft-CIAlign` et, le nouveau consensus est comparé à celui obtenu à l'itération précédente.
Lorsque la séquence d'une nouvelle itération est similaire à celle de l'itération précédente, cela signifie que le consensus a fini d'évoluer, donc le programme se termine et la dernière séquence est retournée.

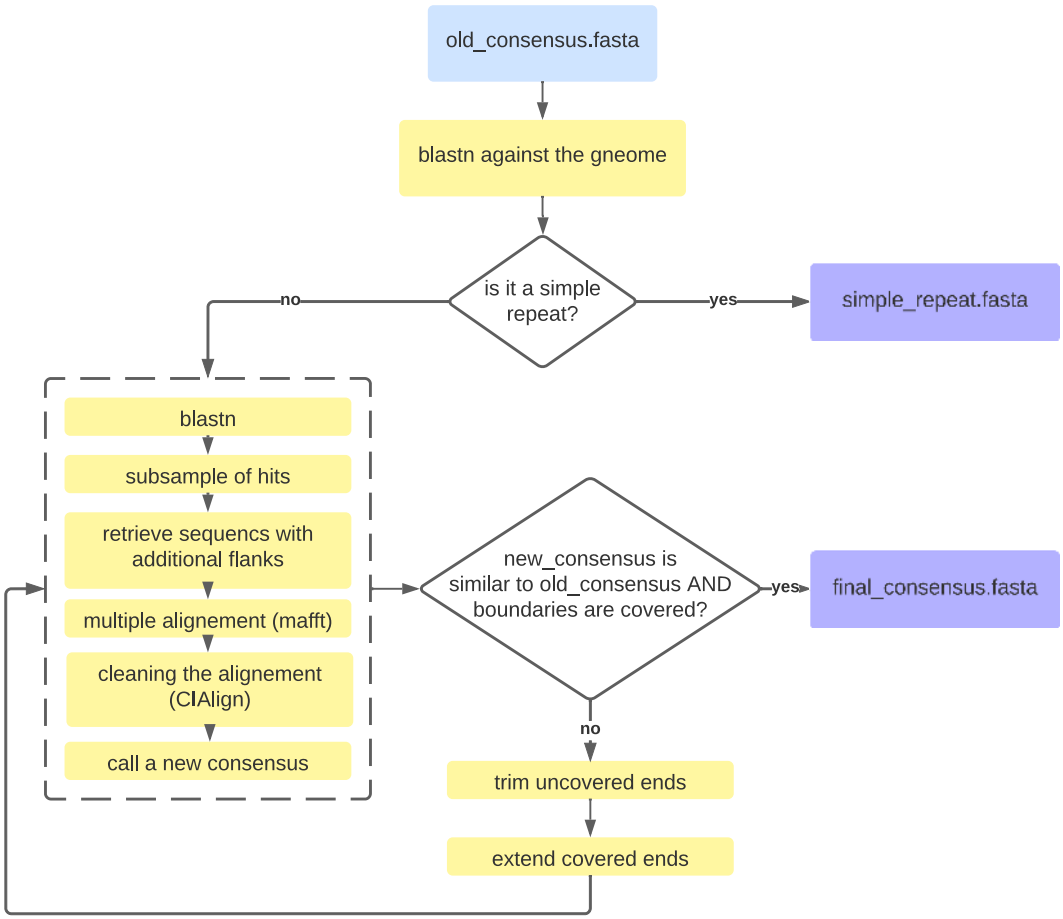

FIGURE 5 – Schéma explicatif du fonctionnement du programme pour le polissage des extrémités.

Le pipeline a été exécuté sur la totalité des séquences issues de l'étape de clutering, avec l'option `-ins 400` (permettant d'enlever les insertions rares dans l'alignement multiple issu de `mafft` jusqu'à une taille de 400pb, CODE S10). Les séquences marquées comme SSR ont été mises de côté.

2.2.2 Classification

L'ensemble des séquences a été classifié en suivant les étapes montrées en FIGURE 6. On a exécuté `RepeatMasker` (`-s -a -inv -nolow`) avec `RepBase27.03` sur notre librairie de séquences putatives. Ici, l'outil est utilisé dans le but de performer une recherche par homologie d'ADN contre ADN.
Le résultat a été filtré de manière à respecter la règle 80-80-80 (CODE S11) ; ensuite le meilleur hit de chaque séquence a été choisi (en fonction du score d'alignement) (LIEN 5) et le fichier obtenu a été traité pour attribuer des super-familles (correspondantes aux familles identifiés) à chaque séquence (LIEN 6).
En parallèle, on a utilisé `RepeatClassifier` (de la suite `RepeatModeler2`, outil qui utilise une recherche par homologie d'ADN sur de l'ADN et d'ADN traduit sur des protéines) et `RepeatProteinMask` (de la suite `RepeatMasker`, qui utilise aussi une recherche par homologie d'ADN traduit sur des protéines). Pour ce dernier on a filtré les hits en fonction d'un score minimum (>200) et d'une longueur minimale (au moins 80pb sur la séquence query) (CODE S12) et ensuite on a gardé que le meilleur hit (en fonction du score) pour chaque séquence (LIEN 5).
Une séquence a été classifiée si et seulement si le même résultat a été obtenu avec au moins deux outils sur trois (LIEN 7) dans un premier temps au niveau de la super-famille et ensuite (uniquement dans le cas où une super-famille a été identifiée), au niveau de la famille (quand possible).

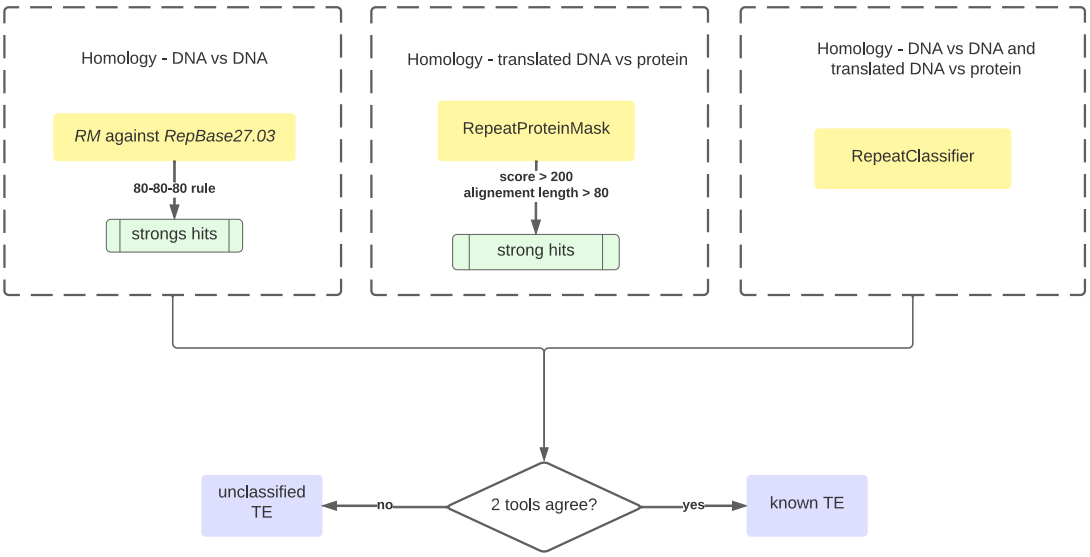

FIGURE 6 – Étapes suivies pour la classification.

Toute vérification manuelle des ET, à la fois pour le polissage et à la fois pour la classification, a été réalisé avec `TE-Aid` [45].

3 Résultats obtenus et discussion

3.1 Évaluation de l'annotation

Tout comme pour les gènes, l'annotation d'éléments transposables utilise aussi un assemblage du génome de l'espèce étudiée comme point de départ. On a choisi d'utiliser comme génome de référence l'assemblage de *Palatini et al., 2020* (AALBO_PRIMARY.1, https://www.ncbi.nlm.nih.gov/assembly/GCF_006496715.1) malgré la présence de nombreux haplotigs. L'utilisation des lectures long reads *PacBio* ainsi que sa couverture génomique[4] font de lui le candidat préconisé pour cette tâche, avec 2 197 scaffolds et un N50 corrélé de 55 702 539.

Pour l'annotation *de novo*, on a choisi d'utiliser d'un côté deux outils génériques pour maximiser la détection d'ET, `RepeatModeler2` et `EDTA`, et de l'autre un outil ET-spécifique, `MITE-Tracker`, car les *MITE*, ont été trouvés en quantité significative dans l'annotation de *Goubert et al,* 2015.

Résultats pour les outils d'annotation `RepeatModeler2`, `EDTA` et `MITE-Tracker`, pour leur caractéristiques spécifiques ont généré des librairies différentes. Pour les explorer, on a utilisé `RepeatMasker` sur chacune de celles-ci.

Librairie utilisée	**N.° séquences**	**% génome masqué**	**% Classe I** (dont LTR)	**% Classe II**	**% Non classifiés**
`EDTA`	16 191	77.53	37.94 (37.94)	38.35	1.23
`RepeatModeler2`	15 384	77.41	45.70 (30.78)	9.75	18.15
`MITE-Tracker`	10 863	31.33	-	31.33	-

TABLE 2 – Nombre de consensi et pourcentage des classes (selon la classification effectué par l'outil considéré) pour les outils d'annotation utilisés.

% génome masqué : représente le pourcentage du génome couvert par la librairie utilisée ; **% Classe I** : pourcentage de rétrotransposons ; **% Classe II** : pourcentage d'éléments transposables à ADN ; **% Non classifiés** : pourcentage d'éléments n'ayant pas reçu de classification.

Dans le TABLEAU 2 on peut observer les tailles des librairies obtenues ainsi que les pourcentages du génome qu'elles permettent de masquer. `EDTA` et `RepeatModeler2`, lesquels sont deux outils de détection famille non-spécifique, ont détecté presque deux fois plus que dans l'annotation de *Melo et al.* [14] (TABLEAU 1).
Globalement, les outils *de novo* semblent être capable de détecter des nouvelles familles d'éléments, vu la large portion de génome couverte, mais par exemple, si on tient compte du pourcentage du génome masqué avec la librairie obtenue avec `MITE-Tracker`, on se rend compte qu'on devrait s'y attendre à avoir un tiers du génome occupé par des *MITE*. Cela semblerait être improbable (par exemple, dans *Goubert et al.*, le pourcentage de MITE a été estimé à 6.04% du génome) et ça nous laisse penser que les outils *de novo* peuvent avoir un fort taux de faux positifs.
D'ailleurs, le nombre de consensi obtenu pour chaque librairie *de novo* n'est pas anodin, il suffit de penser que dans l'annotation de *Goubert et al,* ils ont identifié 5 141 séquences consensus au total. On peut donc facilement suspecter une forte redondance dans les résultats ces outils.

3.1.1 Évaluation de l'annotation *de novo*

Évaluation de la redondance des différentes librairies On a ainsi choisi de clusteriser chaque librairie *de novo* (voir SECTION 2.1.1 - ÉVALUATION DES RÉSULTATS) séparément. Les résultats sont montré dans la TABLEAU 3.

4. Fraction du génome couverte par l'assemblage.

On peut constater que `MITE-Tracker` passe de 10 863 consensi à 1 313 clusters (87,91% familles en moins par rapport au nombre de départ) ; `RepeatModeler2` passe de 15 384 à 8 215 (46.60% en moins) et *EDTA* passe de 16 191 à 14 957 (réduction du nombre de famille uniquement du 7.62%). Le fort taux de redondance pour `MITE-Tracker` pourrait être lié au fait qu'il a été utilisé sur 11 fichiers séparés (le génome a été découpé en plusieurs morceaux, pour permettre un temps de calcul raisonnable).
Ainsi, la librairie obtenue avec `MITE-Tracker` paraît être la plus redondante des trois tandis qu'`EDTA` est à l'apparence la moins redondante.
Une observation plus détaillée des clusters obtenus avec `RepeatModeler2`, a permis de faire ressortir le fait que 6 263 séquences (soit le 40.71% du nombre de départ de consensi) sont en fait des dupliqués, c'est-à-dire qu'elles sont 100% identiques avec d'autres séquences déjà présentes dans la librairie.
Cela pourrait être dû au fait que `RepeatModeler2` a été exécuté en deux fois, avec le deuxième run en modalité « *recover* » (permettant de reprendre une exécution qui a été interrompue auparavant [5]).

Outil d'annotation	Nb de séquences de départ	Nb de clusters	Nb de séquences dupliquées
`RepeatModeler2`	15 384	8 215	6 263
`EDTA`	16 191	14 957	0
`MITE-Tracker`	10 863	1 313	0

TABLE 3 – Nombre de séquences redondantes pour chaque librairie *de novo*.

Le nombre de séquences de départ correspond au nombre de consensi contenu dans chacune des librairies obtenues par les outils *de novo*. Chaque cluster regroupe plusieurs séquences, qui *a priori* proviennent de la même famille d'ET (en raison de la règle 80-80-80 appliquée pour réaliser le clustering).

Évaluation des distributions des longueurs Pour évaluer les consensi générés par ces outils, on a choisi de comparer la distribution des leurs longueurs avec celles de `RepBase`. Pour faire cela nous avons classifié les consensi en utilisant `DeepTE` [36] pour avoir une information sur la classification des séquences.
On a choisi d'utiliser `DeepTE` pour son approche innovante (réseau neuronal convolutif) et sa rapidité, mais vu qu'il s'agit d'un outil encore peu utilisé au sein de la communauté, on a choisi de le tester. On a donc essayé de classifier les séquences contenues dans `RepBase` et de voir si le logiciel renvoyait la même classification que celle proposée dans la base de données (LIEN 8). Sur les 17 365 séquences classifiées, `DeepTE` a réussi en classifier correctement 9 356 (soit le 53.87%) au niveau de la super-famille.

En FIGURE 7 (LIEN 9) on peut observer la distribution des longueurs pour les différents types d'ET (*LTR*, *DNA*, *LINE*, *SINE*, *non-LTR* et *MITE*) pour les différents outils (`RepeatModeler2`, `EDTA` et `MITE-Tracker`) comparés aux séquences de `RepBase`. On a donc une sous-figure par type d'ET, sur l'axe des abscisses on retrouve les intervalles de longueurs et en ordonnées le pourcentage de séquences pour un outil donné et un intervalle donné.
Premièrement, on peut noter la difficulté de `DeepTE` à classifier correctement les MITE. On peut le constater par le fait qu'on a des séquences provenant de `MITE-Tracker` classifiées comme LTR, LINE, SINE et non-LTR. Les MITE classifiés comme ADN sont moins problématiques, car dans la littérature ils sont souvent classifié comme des éléments non-autonomes de Classe II.
Il pourrait également s'agir de faux positif générés par `MITE-Tracker`, mais vu la longueur des ces séquences (les MITE sont connus pour être des séquences courtes, ayant moins de 800pb), un erreur dans la classification performée avec `DeepTE` paraît plus probable.
Sur une vue d'ensemble, on observe les mêmes distributions de longueurs que dans `RepBase` pour tous les outils et tous les types. Cependant, on peut remarquer que dans les intervalles ayant les longueurs les plus élevées (vers la droite), la librairie qui dans la plupart des cas possède le pourcentage le plus élévé, est `RepBase`. On peut voir cela dans les intervalles [2800-4200], [4200-5600] et [5600-7000] pour les LTR ; [2400-3200] et [3200-4000] pour les ADN ; [3200-4800] et [4800-6400] pour les LINE ; [160-320], [320-480] et [480-640] pour les SINE ; [4200-5600] et [5600-7000] pour les non-LTR et ; [480-640] pour les MITE.

5. La premier exécution de `RepeatModeler2` a été interrompue pour permettre l'exécution du programme sur une plateforme plus adéquate.

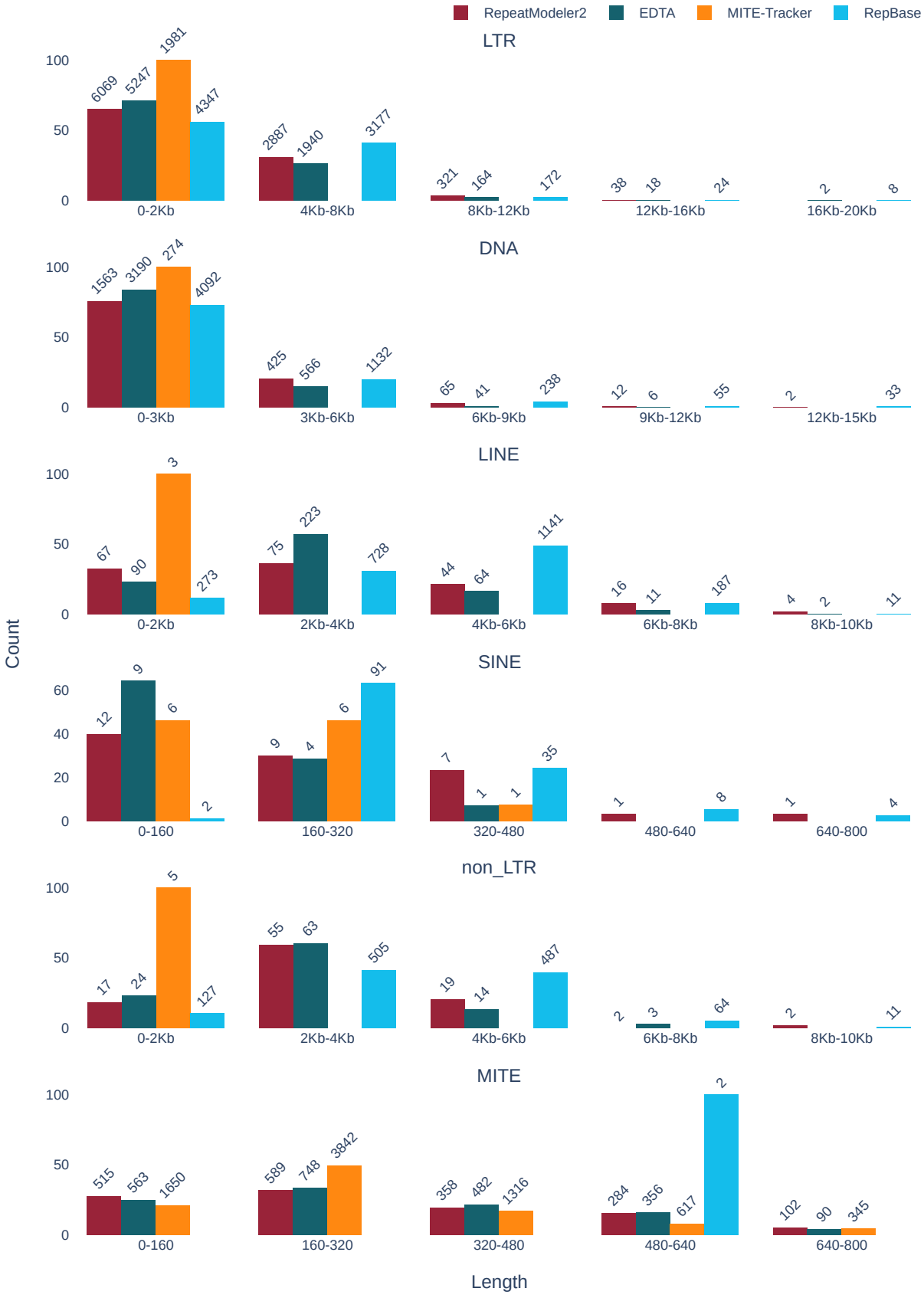

FIGURE 7 – Distribution des longueurs des consensi identifiés par les outils *de novo* et dans `RepBase`.

Chaque sous-figure représente un type d'ET (LTR, DNA, LINE, SINE, non-LTR et MITE). Sur l'axe de l'abscisse on retrouve les différents intervalles de longueur et sur l'axe des ordonnées le pourcentage des séquences contenues dans un intervalle donné pour un outil donné. Par exemple, environ le 50% des séquences classifiées comme LTR (par `DeepTE`) provenant de `RepeatModeler2`, sont contenues dans l'intervalle [0-1400].

La majorité des consensi les plus longs proviennent de `RepBase`, ainsi il semblerait que les outils *de novo* fournissent bien souvent des consensi tronqués, on montre donc encore plus le besoin d'étendre les séquences (Sous-section 2.2.1 - Polissage) (spécialement pour celles issues d'un approche *de novo*).

Pour les *LTR*, parmi les outils *de novo*, `RepeatModeler2` est l'outil qui en détecte le plus (9 315 consensi). Pour les *DNA*, c'est `EDTA` (3 803 consensi).
L'outil qui a détecté le plus de *LINE* et *non-LTR* est `EDTA` (respectivement 400 et 104 consensi) et, celui qui a annoté le plus de *SINE* est `RepeatModeler2` (30 consensi).
Enfin, pour ce qui concerne les *MITE*, on observe la quasi-absence de consensi en `RepBase` (uniquement deux séquences), ce qui nous permet de re-souligner l'importance d'utiliser un double approche *de novo*/homologie. Sans surprise, l'outil qui a annoté le plus d'éléments est `MITE-Tracker` avec 7 770 consensi.

Globalement, les séquences pour les différentes types d'ET obtenues par les outils *de novo* ont des longueurs similaires aux séquences contenues dans `RepBase`, donc on peut pas imaginer de filtrer des faux positifs parmi les consensi en fonction de ce critère.

Évaluation de la couverture génomique Les librairies générées par les outils *de novo* couvrent une large portion du génome, ce qui nous laisse penser à une possible présence de faux-positifs. Pour estimer la proportion d'éléments répétés, on a exécuté `dnaPipeTE` [13] sur des données simulées à partir de l'assemblage (pour plus de détails, voir Paragraphe 2.1.1 - Évaluation des résultats).

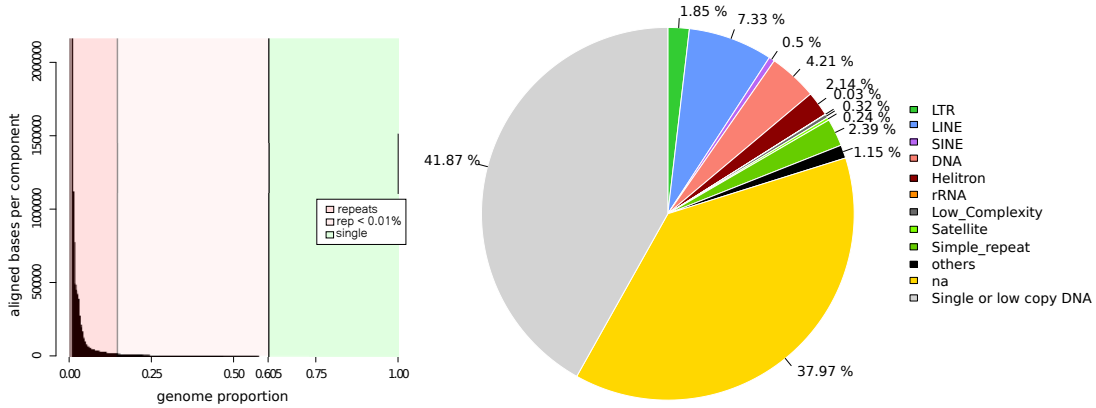

Figure 8 – Estimation de la fraction répétée du génome d'*Aedes albopictus*.

A - Nombre des bases alignées par composante en fonction de la proportion génomique. En rouge, la fraction génomique fortement répétée ; en rose, la fraction normalement répétée et enfin, en vert l'ADN non répété. **B - Estimation de la composition en éléments transposables.**

En Figure 8A, on peut voir que l'outil identifie le 60.5% du génome étant comme répété. Par rapport aux résultats montré en Tableau 2, il semblerait que `RepeatModeler2` et `EDTA` surestiment la fraction répétée du génome (respectivement de 17.03% et 16.91%). Il pourrait aussi s'agir d'une sous-estimation de `dnaPipeTE`, mais ses résultats sont plus en ligne avec les résultats précédemment obtenus (Tableau 1).
En Figure 8B, même si de manière approximative, on peut voir une nouvelle estimation des éléments transposables constituant la fraction répétée. On note principalement des LINE (7.33%), des ADN (4.21%) (ainsi que des Helitrons, 2.14%) et des LTR (1.85%). Cependant, la majorité reste non-classifié (37.97%).

On a également effectué un deuxième contrôle (Figure S1), en utilisant `RepBase` à la place de la librairie de défaut et sur un vrai jeu de données (`SRR10038864`) d'*Ae. albopictus*, en utilisant comme paramètre pour la taille du génome, l'estimation par cytométrie en flux (1,23Gb). Le pourcentage obtenu

pour la fraction répétée est de 59.4%. Ce pourcentage est très similaire à celui obtenu sur les données simulées et au même temps il ne s'éloigne pas trop des annotations d'autres auteurs (Tableau 1), il permet donc d'avoir une idée de la vraie fraction répétée du génome.

3.2 Construction de la librairie

3.2.1 Récupération des copies

Après concaténation (Figure 4) des librairies brutes, la librairie fusionnée contient **59 848** séquences, dont 42 438 provenant des outils *de novo* et 17 410 provenant de `RepBase`.
Après avoir utilisé `RepeatMasker` sur la librairie fusionnée, on obtient **8 966 074** séquences qui ont été masquées sur le génome, en couvrant 80.94% de celui-ci.
Parmi les 59 848 consensi de départ, uniquement 38 165 (63.7% de la librairie fusionnée) d'eux ont eu une correspondance sur le génome.

En sortie de `RepeatMasker` on obtient un fichier d'annotation (`.out`) lequel contient toutes les correspondances entre le génome et les séquences contenues dans la librairie brute. Ainsi, chaque ligne du fichier représente un alignement avec les informations associées : le score d'alignement (obtenu par programmation dynamique, algorithme de *Smith-Waterman*), la divergence entre la référence et la séquence de la librairie, les pourcentages d'insertion et de délétion, les positions de début et de fin de l'alignement sur la référence, le nom de la séquence qui s'est alignée, sa classification (si présente dans librairie) et enfin un identifiant unique pour une insertion donnée (ainsi plusieurs alignements qui sont considérés comme provenant d'une même insertion auront le même identifiant).

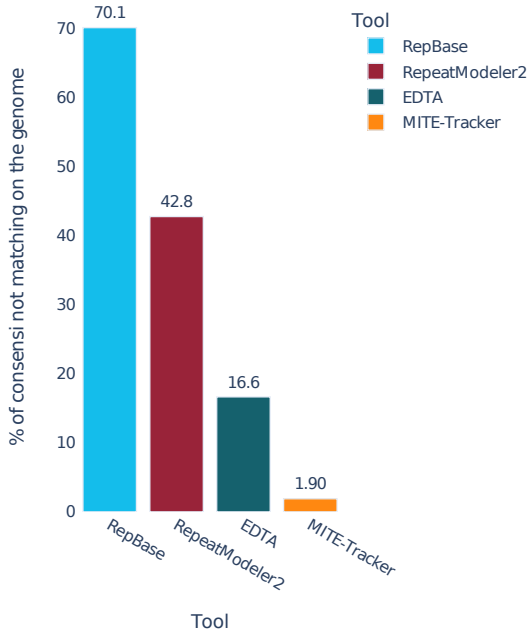

Figure 9 – Pourcentage des consensi qui n'ont pas eu de correspondance sur le génome pour chaque outil *de novo* (plus `RepBase`).

En Figure 9 (Lien 8), on peut observer le pourcentage de séquences pour chaque librairie de départ (`RepeatModeler2`, `EDTA`, `MITE-Tracker` et `RepBase`) qui n'ont pas masqué aucune portion du génome.
On observe que le 70.1% des consensi contenus dans `RepBase` n'as pas de correspondance sur le génome.

Cela est attendu en raison du fait que cette base de données contient des séquences provenant d'espèces différentes (séquences d'*Arthropodes*), du coup même si il s'agit d'espèces phylogénétiquement proches à *Ae. albopictus* il est possible qu'elles soient génétiquement distantes ou alors qu'elles possèdent un contenu en éléments transposables différent.
Parmi les outils *de novo*, `RepeatModeler2` est celui en ayant le plus de séquences sans correspondance (42.8%), suivi par `EDTA` (16.6%) et `MITE-Tracker` (1.90%). Le taux élevé de séquences qui ont pas été gardées pour `RepeatModeler2` est en partie explicable par la quantité de séquences redondantes (Tableau 3), où 6 263 (40.71%) séquences étaient des dupliqués. Du coup il reste plus que le 2.09% de la librairie n'ayant réellement pas de correspondances sur le génome. De même pour `MITE-Tracker`, il y a que le 1.90% des séquences qui ne sont pas retrouvées.
En revanche, `EDTA` montre un quantitatif important (16.6%). Cela pourrait être expliqué par le fait que `RepeatMasker` mets en « compétition » les fragments s'alignant sur une même portion de génome ; dans ce cas, uniquement le hit avec le meilleur score sera conservé par le programme. Il est donc possible que `EDTA` ait détecté un certain nombre d'éléments déjà détectés par `RepeatModeler2` et, que ces derniers avaient un meilleur score d'alignement. Il est possible que le même mécanisme se soit mis en place avec les séquences dupliquées de `RepeatModeler2`, un meilleur hit (ou dans ce cas, équivalent) était déjà présent parmi les résultats et donc les séquences dupliqués n'ont pas été intégrées dans le fichier de sortie.
Pour démontrer ce mécanisme de `RepeatMasker`, nous l'avons utilisé sur chaque librairies obtenues par les outils *de novo* séparément, de façon à vérifier le nombre de consensi ayant des match pour chacune. En absence de compétition entre les différents librairies, on s'attend à voir une diminution des séquences n'ayant pas des correspondances sur le génome. Le résultat obtenu est montré dans le tableau suivant :

Librairie	Nb de séquences n'ayant pas de match	% sur la totalité de la librairie
`RepeatModeler2`	6 440	41.86
`EDTA`	298	1.84
`MITE-Tracker`	76	0.70

Table 4 – Nombre de séquences filtré par chaque librairie indépendamment en sortie de `RepeatMasker`.

Dans le Tableau 4 on peut constater que `RepeatModeler2` a toujours un grand nombre de séquences n'ayant pas de match sur le génome (6 440), mais cela reste cohérent avec le nombre séquences dupliquées identifiées dans le Tableau 3 ; ainsi si on les exclue, on obtient uniquement $6440 - 6263 = 177$ séquences n'ayant réellement pas de match (ce qui fait 1.15% de la librairie brute de départ pour cet outil). Pour `EDTA` on peut voir que uniquement 298 séquences sont « filtrées » par `RepeatMasker` et que donc le pourcentage de la librairie n'ayant pas de match passe de 16.6% à 1.84%. On constate une réduction aussi pour `MITE-Tracker` qui passe de 1.90% à 0.70%.
En conclusion, on montre que le fait d'utiliser `RepeatMasker` sur les différents librairies séparément ou sur une librairie finale a un impact sur le résultat final ; utiliser directement une librairie fusionnée permet ainsi de réduire la redondance entre les différentes librairies de départ.

3.2.2 Reconstruction des copies entières

L'output de `RepeatMasker` a été ensuite filtré en fonction de la longueur des fragments. On a filtré tout fragment ayant moins de 80pb car on considère qu'ils ne sont pas assez longs pour apporter assez d'information dans les étapes suivantes. Le nombre de fragments passe donc de 8 966 074 à **6 596 866**.

Nous avons choisi ensuite d'utiliser `OCTFTA` pour reconstituer les solo-LTR avec leur élément interne. Notamment il se base sur la recherche de séquences ayant le même préfixe dans le nom (ex. *Gypsy*) mais avec des suffixes différentes : *LTR* pour les solo-LTR et *I* pour les éléments internes.
Étant donné qu'on a utilisé des outils différents et variés, il est possible que certains noms utilisés pour les séquences ne respectent pas ce format ; pour cette raison nous avons choisi d'utiliser le paramètre `-insert` pour fusionner deux fragments se trouvant à moins de 80pb.

Après avoir exécuté `OCTFTA`, le nombre de séquences est passé à **4 304 741** instances.

3.2.3 Clustering des copies et appel des consensi

Les séquences ont été donc clusterisées en utilisant `cd-hit-est` pour pouvoir reconstituer les familles d'éléments transposables (règle 80-80-80).
Suite au clustering, on obtient **437 732** clusters :

Nb clusters larges (7 ou plus séquences)	35 654
Nb clusters « singleton » (une seule séquence)	276 980
Nb clusters « low copy » (entre 2 et 6 séquences)	125 098
Total	**437 732**

TABLE 5 – Résultats de `cd-hit-est`.

Pour la suite de la construction de la librairie, on a choisi de ne garder que les clusters ayant un certain nombre de séquences (minimum 7) et on a choisi de séparer les restants en « singleton » (ayant une seule séquence) et en « low copy » (entre 2 et 6 séquences). On a choisi de mettre de côté les « singletons » car on estime que la majorité des ET est présent en plusieurs copies au sein du génome et, de façon similaire, les clusters « low copy » ne sont également pas sélectionnés car on les attribue généralement à des ET qui sont peu actifs et, ils peuvent être facilement confondus avec d'autre éléments génomiques qui sont présentes en plusieurs copies (ex. des gènes paralogues [6]).
Le nombre minimal de séquences au sein d'un cluster est donc directement lié à la répétitivité minimale attendue pour n'importe quelle famille ; en FIGURE 10 (LIEN 10) on peut voir comme le nombre de clusters peut varier sensiblement en fonction du nombre minimal de séquences formant les clusters.

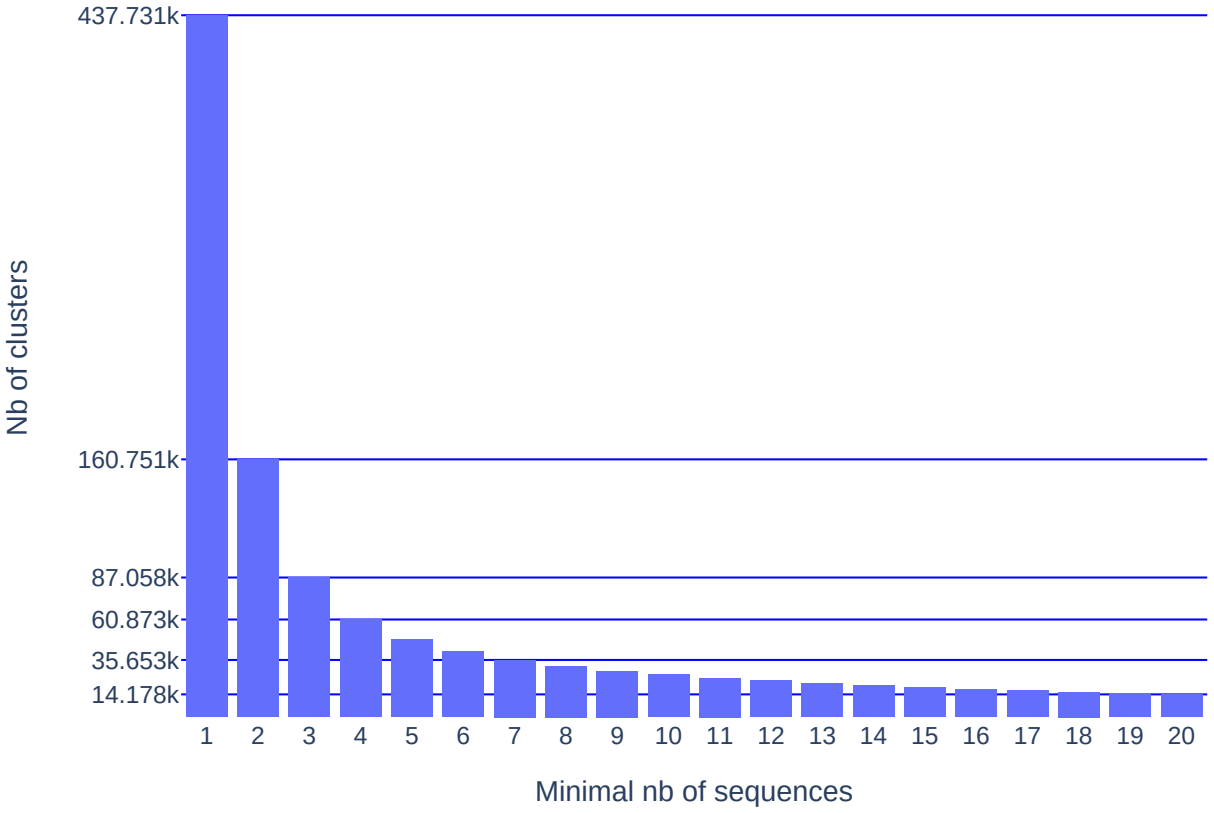

FIGURE 10 – Nombre de clusters en fonction du nombre minimal de séquences formant le cluster.

6. Gènes issus d'un événement de duplication.

On peut voir comme en sélectionnant au minimum deux séquences on réduit considérablement le nombre de clusters (de 437 761 à 160 751 clusters). En prenant 7 séquences ou plus (comme on l'a fait dans notre démarche), on obtient que 35 653 clusters et, si on passe à 20 séquences on réduit le nombre à 14 178 clusters.
On aurait pu, sans doute utiliser 20 séquences comme seuil minimal pour réduire le temps de calcul dans les étapes qui ont suivi, mais on aurait risqué de perdre certaines des familles les moins actives, étant elles présentes en un nombre limité de copies.

Ensuite, on s'est également interrogés sur la contribution de chaque librairie de départ dans la formation des clusters (clusters respectant le seuil de séquences minimales, au premier cycle), FIGURE 11. Parmi les 35 654, il y en a 191 qui sont formés uniquement par des séquences provenant de `RepBase`, 5 740 de `RepeatModeler2`, 3 821 d'`EDTA` et 423 de `MITE-Tracker`.
En détail, `RepBase` participe dans un total de 3 407 clusters, soit un peu moins d'un dixième (9.55%) du total et, permet d'avoir un ensemble fiable d'ET ; `RepeatModeler2` dans 29 206 (soit 81.92%) ; `EDTA` dans 28 015 (soit 78.57%) et ; `MITE-Tracker` dans 11 066 (soit 31.04%).
On note que l'intersection entre `RepeatModeler2` et `EDTA` est de 12 768 clusters et celle entre ces deux plus `MITE-Tracker` est de 6 959. Ces chiffres permettent de souligner l'intérêt d'une stratégie utilisant une annotation *de novo*.
Les séquences provenant de `MITE-Tracker` et qui sont retrouvées dans des clusters avec des séquences provenant d'autres outils (1 056 + 6 959) pourrait laisser penser que `MITE-Tracker` a classifié par erreur certains éléments, autres que des MITE, comme des MITE. Cela paraît évident si on pense un nombre de MITE contenus dans `RepBase` (10 séquences). Cependant, il faut garder en tête que, il s'agit d'un type d'ET qui a été décrit plus récemment que d'autres et que donc pour cela, sa représentation dans les bases de données est logiquement moindre.
De même, le nombre de clusters outil-spécifique `RepeatModeler2` et `EDTA` pourrait suggérer la présence de faux positifs, mais cela reste compliqué à déterminer, surtout que les deux utilisent des approches différentes dans la détection.

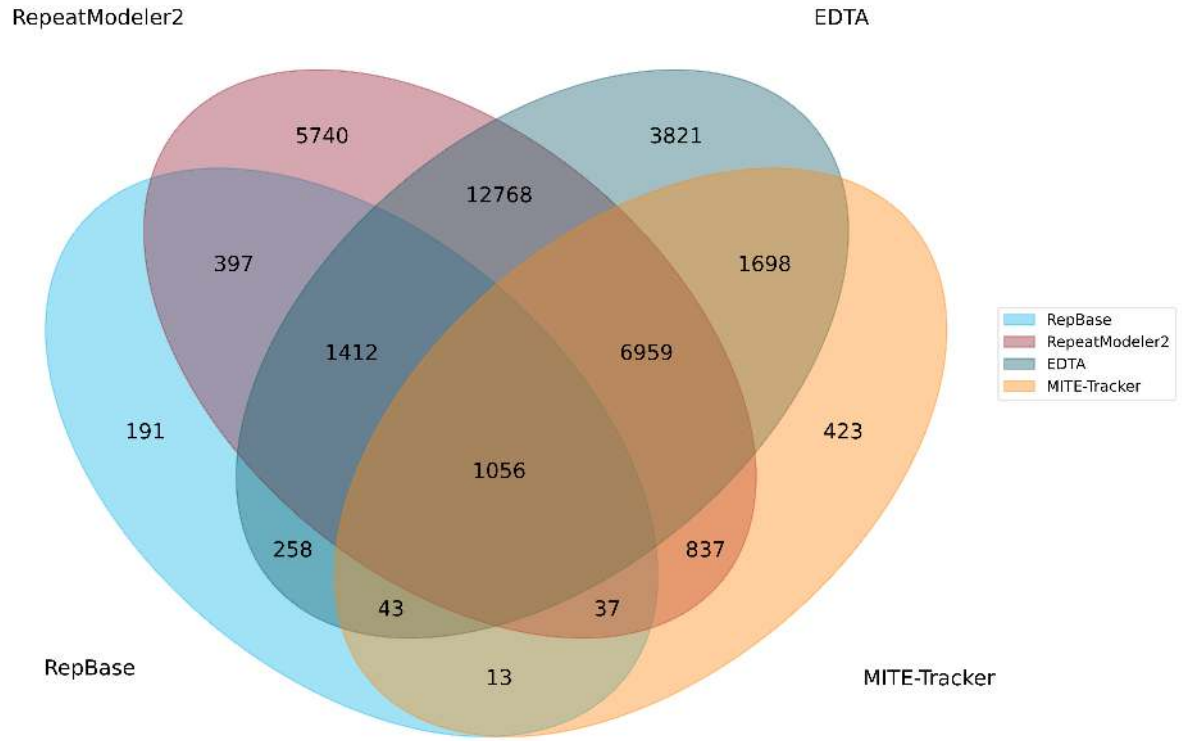

FIGURE 11 – Contribution de chaque librairie de départ dans la formation des clusters.

Pour chacun des clusters conservés, on a appelé un consensus en utilisant `Refiner`. Comme indiqué dans SOUS-SECTION 2.2, pour cette étape on n'a pas utilisé le nombre total de séquences de chaque

cluster. Le nombre de séquences conséquent de certains cluster était beaucoup trop élévé pour permettre l'exécution de `Refiner` dans un délai de temps raisonnable. On a choisi d'échantillonner un maximum de 500 séquences car c'est suffisant pour réduire le temps de calcul et puis car une analyse préliminaire a permis de montrer que ce nombre de séquences est suffisant pour avoir un consensus stable.

Enfin, on a réalisé plusieurs cycles (7) `cd-hit-est`/`Refiner` de façon à réduire un maximum la redondance. À différence du premier cycle, à chaque itération on a conservé tous les clusters et on a pas eu besoin de re-effectuer un sous-échantillonnage des séquences intra-cluster. On obtient ainsi un total de **23 009** séquences.

3.3 Polissage des extrémités

Pour améliorer la qualité des consensi on a choisi d'effectuer un polissage des extrémités. Cela peut consister en deux types d'action, l'extension ou la troncature. À cet effet nous avons écrit un pipeline décrit dans le Paragraphe 2.2.1.

Évaluation du pipeline Pour tester l'efficacité du programme, nous avons créé un petit jeu de données ne contenant que des éléments transposables connus et annotés dans `RepBase25.08` de l'espèce *Drosophila melanogaster*, car étant une espèce modèle, ses éléments transposables sont parmi ceux les mieux annotés. On a ainsi artificiellement tronqué les extrémités de ces séquences (Lien 11), 10% de la longueur originale pour chaque extrémité, et on a utilisé le programme en utilisant l'assemblage `ASM2016949v1` (obtenu par technologie de séquençage long reads - `Nanopore MinION`).
Le jeu de données utilisé contient au total 180 séquences. 179 ont été polies tandis qu'une seule séquence a été détectée comme des SSR. Parmi les 179 séquences traitées, **155** ont été étendues et **24** ont été tronquées ultérieurement. L'extension minimale (par rapport à la séquence tronquée artificiellement) a été de 0% et celle maximale de 314% , avec une médiane de 40% ; la troncature minimale a été de 1% et celle maximale de 50%, avec une médiane de 12%.

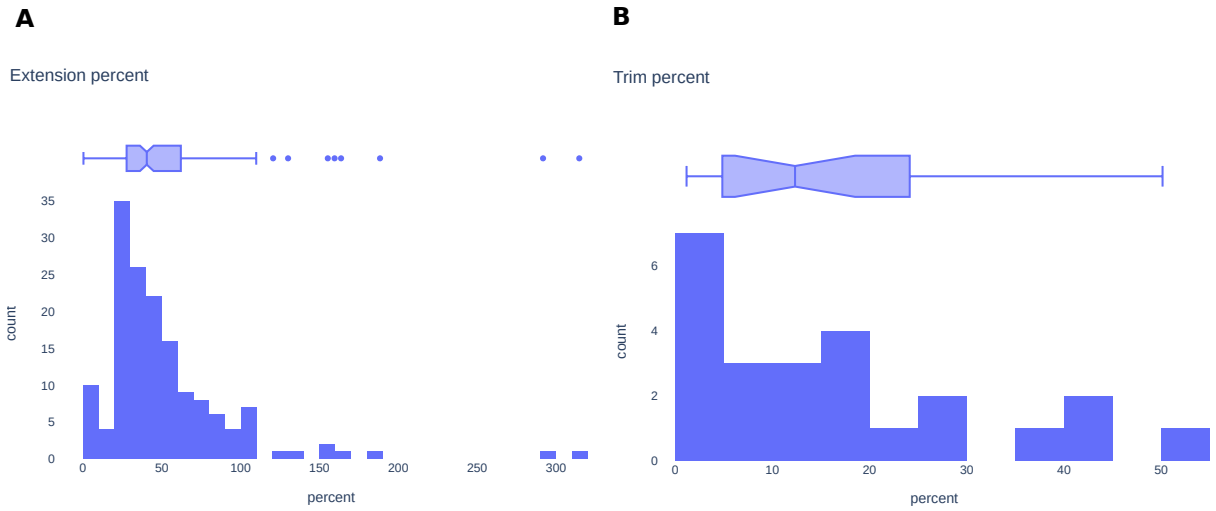

Figure 12 – Pourcentage d'extension et de troncature sur le jeu de données de test.

A - Distribution des pourcentages d'extension ; **B** - Distribution des pourcentages de troncature.

L'attendu était de n'avoir que des séquences étendues et, spécialement avec une extension moyenne de 25% (pour retrouver les extrémités tronquées artificiellement). Or, ce n'est pas le cas de nos résultats, donc on a choisi d'inspecter un peu plus les séquences tronquées de départ et les séquences traitées avec le pipeline et, d'évaluer certains critères, comme le nombre de fragments obtenus en réalisant un `blastn`

sur le génome de référence (plus le nombre est élevée, plus la famille est mieux représentée), le nombre de copies entières (étant une copie entière un fragment ayant au moins 90% de la longueur du consensus), la conservation (ou amélioration) de la détection des domaines protéiques, la divergence [7] des copies entières (moins elles sont divergentes, plus ça signifie que le consensus les représente de manière correcte) et, enfin la couverture génomique des extrémités. Pour évaluer tous ces critères, on a utilisé un outil pour la vérification manuelle des ET, `TE-Aid` [45].

Dans le Tableau 6, on peut observer les résultats de cette évaluation. On peut noter que le nombre moyen de fragments a augmenté sensiblement de 94.8 à 164.8, ce qui est plutôt bien en considérant le fait que plus de fragments sont trouvés sur le génome correspondant à la séquence consensus, mieux elle représente l'ensemble des copies.
Le nombre moyen de copies entières a aussi augmenté, mais cette fois-ci la différence est moins éclatante ; bien que il soit important, quand possible, de récupérer la totalité des copies entières, il faut préciser que dans des nombreux cas l'augmentation a été observée sur des séquences qui ont été tronquées ; or en réduisant la taille de la séquence il va de soi qu'il soit plus facile d'obtenir des copies entières.
Pour 158 séquences sur 179 on a pu retrouver (ou même améliorer) les domaines protéiques annotés par `TE-Aid`. On a pu remarquer que les consensi n'ayant pas conservé leur domaines protéiques c'était des séquences qui ont eu une forte augmentation du nombre de fragments après polissage et que ceux-ci, en particulier se chevauchent à certains endroits sur le consensus, suggérant que certaines nucléotides (en correspondance des chevauchements) sont incorrects.
Concernant la divergence des copies entières par rapport à la séquence consensus, on a constaté une diminution dans 48 cas, aucune différence pour 67 séquences et, une augmentation dans 63 cas. D'une part, la diminution de la divergence se traduit avec une meilleure description de la famille d'ET mais d'autre part, l'augmentation de la divergence couplée à l'augmentation du nombre de copies entières peut aussi être synonyme d'une meilleure représentation car, les membres les plus anciens d'une famille sont connus pour avoir une forte divergence par rapport à les copies les plus récentes.
Enfin, nous avons pu constater une amélioration de la couverture sur l'extrémité 5' dans 95 cas sur 179 et dans 90 cas sur 179 pour l'extrémité 3'.

	Avant polissage	**Après polissage**
Nombre moyen de fragments	93.8	163.8
Nombre moyen de copies entières	17.9	18.5
Conservation/Amélioration des domaines protéiques	158 sur 179	
Divergence des copies entières	48 -, 67 =, 63 +	
Amélioration couverture en 5'	95 sur 179	
Amélioration couverture en 3'	90 sur 179	

Table 6 – Évaluation du pipeline pour le polissage.

Divergence des copies entières : `"-"` indique que le nombre de copies entières après polissage ayant moins de divergence qu'avant ; `"="` indique pas de changement ; `"+"` indique une augmentation de la divergence.

Pour clarifier ce qu'on entend par amélioration de la couverture, un exemple bien réussi est montré en Figure 13. Dans un premier temps on peut noter la longueur du consensus avant et après polissage, on passe de 4 688pb à 6 426pb, ce qui fait une extension de 37%, donc c'est plus que l'attendu moyen (25%) mais ça reste tolérable.
Ensuite on peut voir que le nombre de fragments a presque doublé, de 68 à 119 fragments et que le nombre de copies à longueur entière (en rouge, sur le graphique de divergence des hits `blastn`, en haute à gauche) n'a pas bougé (5 copies).
Sur le graphique de la couverture génomique, on retrouve le même profil à une exception près : on peut

7. La divergence d'une copie du consensus traduit d'une certaine manière la distance entre les deux séquences. Lors de l'évolution, des copies différentes d'un même élément peuvent accumuler de façon différente des mutations, engendrant une divergence propre à chaque copie.
Étant la séquence consensus d'une famille, la séquence « moyenne » représentant l'ensemble des copies de cette famille, il est important de minimiser sa distance (ou donc en autre termes, la divergence) avec les membres de la famille.

noter comme la couverture sur les deux extrémités soit augmenté de façon considérable après polissage et, comme l'extension se soit arrêtée juste après ces deux pics. Ce qui c'est passé c'est la chose suivante : la séquence avant traitement présentait uniquement l'élément interne d'un LTR (comme démontré par l'annotation des domaines protéiques) et, le pipeline dans la séquence traitée a permis de récupérer les Long Terminal Repeats (LTR) proprement dit présents aux extrémités.
Cela se confirme à la fois sur le *self dotplot* (en bas à gauche), lequel montre des répétions directes aux extrémités (les traits parallèles à la diagonale), et à la fois sur l'annotation des structures et des domaines protéiques avec les flèches vertes dans le même sens.
Enfin, mais pas moins important, on retrouve l'annotation des domaines protéiques avant et après polissage (Gag + Pol).

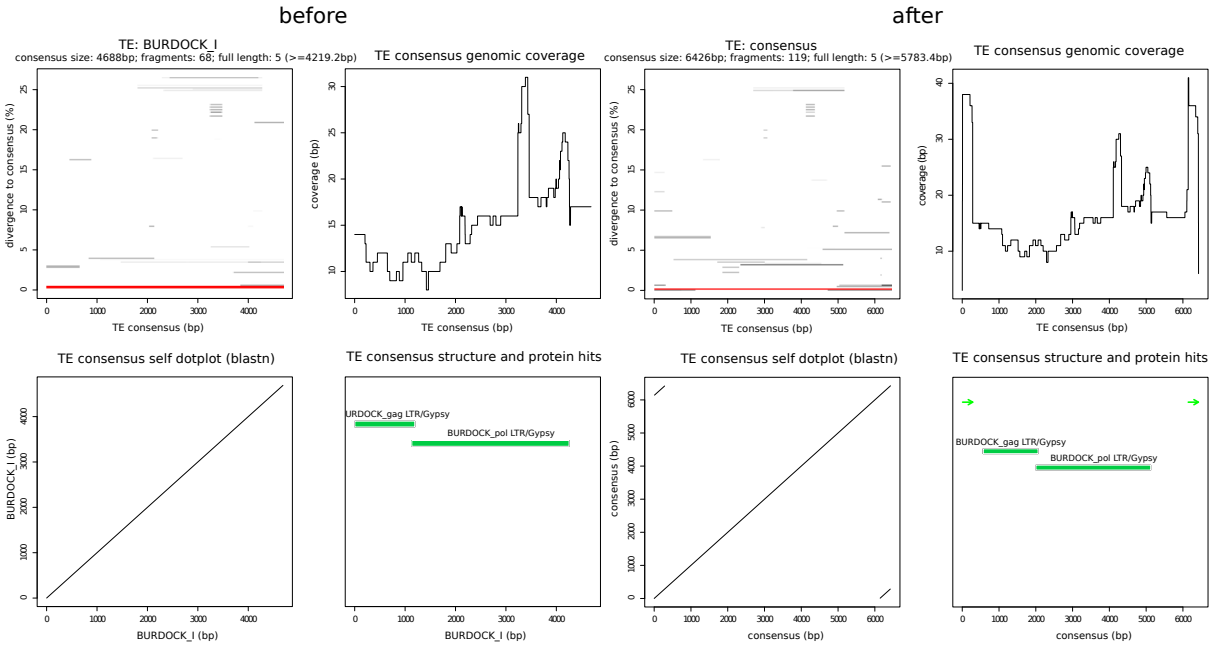

FIGURE 13 – Résultats de `TE-Aid` pour l'élément **BURDOCK_I**.

En haute à gauche : divergence et position par rapport au consensus des hits (`blastn`), en rouge les hits considérés comme des copies à longueur entière ; *en haute à droite* : couverture génomique du consensus par les hits ; *En bas à gauche* : le *self dotplot* de la séquence (`blastn` de la séquence consensus contre elle-même) ; *En bas à droite* : annotation des structures caractéristiques des ET et hits contre une bases de données de domaines protéiques (`RepeatPeps` (`RepeatMasker`)).
Before - Résultats de pour l'élément BURDOCK_I tronqué artificiellement. **After** - Résultats après polissage.

Pour retourner à l'évaluation du pipeline, il est important de signaler les choses suivantes : la plupart des meilleurs exemples étaient des éléments internes de LTR (comme en FIGURE 13) ; lors du traitement de certains solo-LTR, le programme a essayé d'aller chercher l'élément interne correspondant mais cela à généralement aboutit dans nombreux fragments se chevauchant de manière irrégulière (en raison du fait que l'élément interne peut être récupéré à la fois en allant vers la gauche à partir du solo-LTR en 3' et à la fois en allant vers la droite à partir du solo-LTR en 5') ; les séquences ayant subi une forte troncature (au délà du 30%) étaient toutes des familles *TART* ou *TARHE* (non-LTR) ; enfin, qu'une majorité des séquences testées étaient des LTR.

Résultats polissage Le pipeline a été donc utilisé pour polir les séquences contenues dans la notre librairie. Les distributions des pourcentages d'extension et de troncature sont montrées en FIGURE 14.
Sur 23 009 séquences, 19 939 ont été polies et des 3 070 restantes, 610 sont des SSR et 2 460 n'ont pas été modifiées. Parmi les séquences polies, 14 364 ont été étendues et 5 575 ont été tronquées.
L'extension minimale est de 0%, la maximale de 465% et la médiane est à 28% ; la troncature minimale est de 0%, la maximale de 93% et la médiane à 13%.

Pour vérifier le bon fonctionnement du pipeline, tout comme pour le test, nous avons utilisé `TE-Aid` sur 10 séquences. Quelques exemple sont montrés en Annexes (Figure S2 à Figure S5). Sur ceux-ci on peut constater une amélioration globale des séquences avec des extrémités bien polies (ex. Figure S2), une forte augmentation de fragments (ex. Figure S3) ou encore une nouvelle récupération de LTR (ex. Figure S5).

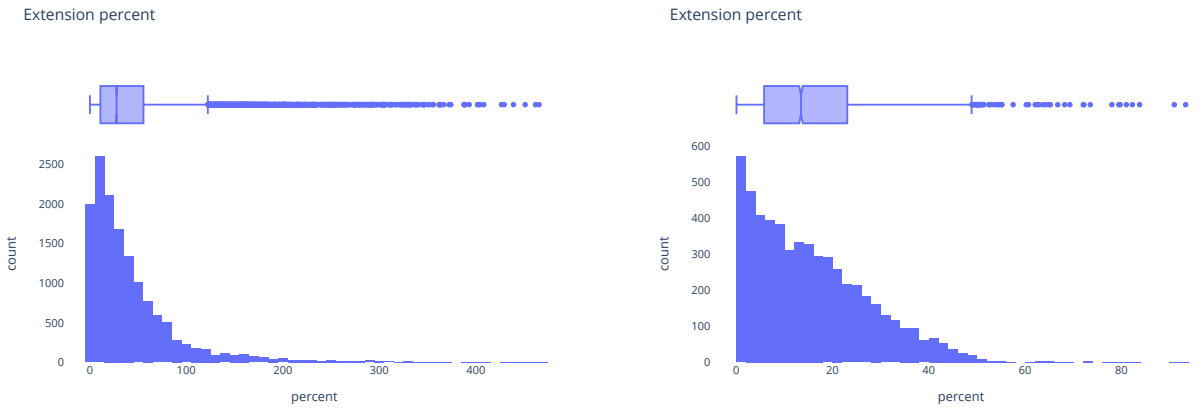

Figure 14 – Distributions des pourcentage d'extension et de troncature sur nos données.

3.4 Classification

La classification a été réalisé sur les **23 009** séquences traitées avec le pipeline pour le polissage. On a gardé donc les séquences marquées comme SSR dans le but de les conserver pour ensuite avoir une idée globale sur le contenu d'éléments répétés (du coup éléments transposables ou pas) dans le génome d'*Ae. albopictus*.

Résultats des outils de classification

- `RepeatMasker` : en sortie on obtient un total de 86 695 hits sur des consensi contenus dans `RepBase27`. Après avoir filtré en fonction de la divergence et la longueur de l'alignement sur la séquence analysée, on obtient 17 671 hits (dont 8 065 séquences uniques sur les 23 009 séquences traitées).

Classification	Effectif
DNA	2 257
LINE	1 135
LTR	3 471
MITE	37
non-LTR	418
SINE	394
Unknown	72

Table 7 – Classification selon `RepeatMasker` en utilisant `RepBase`

- `RepeatProteinMask` : on obtient un total de 160 643 hits sur les séquences de `RepeatPeps.lib`. Après filtrage sur le score d'alignement (>200) et sur la longueur de l'alignement sur la séquence

analysée (query), on obtient 4 626 séquences uniques sur les 23 009 séquences traitées.

Classification	Effectif
DNA	98
LINE	1 909
LTR	2 501
RC	6
SSR	111

Table 8 – Classification selon `RepeatProteinMask`

☛ `RepeatClassifier` : cet outil attribue directement une classification (quand possible) à la séquence traitée, pour cette raison aucune étape de filtrage est nécessaire ici. Les effectifs pour chaque groupe d'ET sont montrés dans le tableau suivant :

Classification	Effectif
DNA	951
LINE	2 335
LTR	3 092
RC	1 101
SINE	142
tRNA	190
Unknown	15 223

Table 9 – Classification selon `RepeatClassifier`

Fusionnement des résultats Comme expliqué en Section 2.2.2, on a attribué à une séquence une classification si et seulement si au moins deux outils ont eu le même résultat (d'abord au niveau de la super-famille et puis, où possible, au niveau de la famille). Les effectifs des super-familles identifiées sont montrés dans le tableau suivant :

Classification	Effectif
DNA	624
LTR	2 742
LINE	1 983
PLE	9
SINE	45
SSR	39
Total	5 442

Table 10 – Résultat combiné des différents outils de classification.

Dans ce tableau, on a délibérément omis les éléments suivants : rRNA (2 séquences), Satellites (19), snRNA (1) et tRNA(1).

Les premières 20 séquences les plus répétées ainsi que les 20 ayant le plus de couverture génomique on été vérifiées manuellement (Lien 12) et lorsqu'il a été possible de renseigner au moins la super-famille, la séquence a été rajoutée, en atteignant un total de 5 465 séquences classifiées.
En Figure 15 on peut observer les familles les plus représentées sur le génome d'*Ae. albopictus*. Parmi les familles les plus répétées (de 600 fragments jusqu'à 4 000), on retrouve tous les principales éléments

transposables mais, on note en particulier la domination des LINE sur l'ensemble du paysage des ET. D'un point de vue de la couverture génomique, comme avant on note la forte présence des LINE dans les ET les plus répresentés sur le génome.

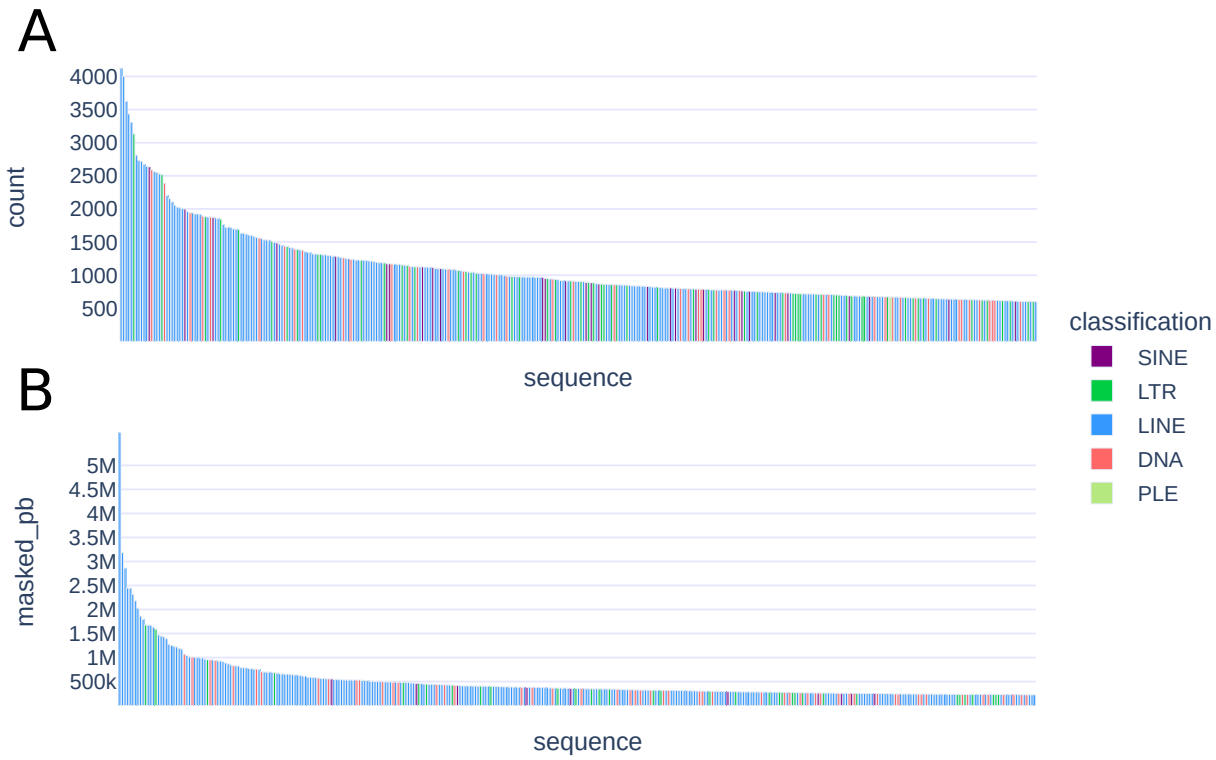

FIGURE 15 – Graphique montrant les familles plus présentes sur le génome d'*Ae. albopictus*.

A - Distribution des éléments transposables les plus répétés. Chaque barre représente une séquence de notre librairie ; en ordonnée : le nombre de fragments identifié par `RepeatMasker`. **B** - En ordonnée : le nombre de bases masquées.

Pour quantifier exactement les proportion de chaque classe d'ET, on a exécuté une dernière fois `RepeatMasker`. Les résultats obtenus sont montrés dans TABLEAU 11. On peut observer que comme dans les annotations d'autres auteurs (TABLEAU 1), la catégorie d'ET qui s'impose sur le génome est la classe I (rétrotransposons) avec un abondant 39.43% du génome. Parmi les éléments de classe I, on distingue particulièrement les LINE et les LTR.
Les éléments transposables de classe II couvrent le 16%, quantité qui est très similaire à les annotations précédentes. On peut ainsi apprécier l'effort réalisé par les outils *de novo* dans la détection des LTR (en particulier `RepeatModeler2`, qui par rapport à sa version précédente a intégré des modules dédiés à leur détection).

En FIGURE 16, on montre une représentation graphique du paysage des éléments répétés d'*Ae. albopictus*. On peut observer comme les LINE, les LTR et les ADN occupent de manière presque équivalente le génome.
Parmi les LINE, les super-familles les plus représentées sont *R1*, *RTE* et *JAM1*. Parmi les LTR, on retrouve principalement les super-familles *Bel-Pao*, *Copia* et *Gypsy*. Chez les ADN, on retrouve les super-familles *hAT* et *Sola*, mais on peut observer la difficulté des outils de classification à classifier ces éléments ; en fait la majorité des séquences a été classifiée uniquement comme ADN car un consensus au niveau suivant (super-famille ou famille) n'a pas été trouvé.

Classe	Nb d'instances	Nb de bases masqués (pb)	% du génome
Classe I	1 823 774	571 983 224	39.43
dont SINE	91 912	16 750 947	1.15
dont LINE	953 856	301 852 752	20.81
dont LTR	778 006	253 379 525	17.47
classe II	825 697	232 145 026	16.00
Non-classifiées	149 610	58 788 174	4.05
Total	2 799 081	862 916 424	59.49

TABLE 11 – Quantification des éléments transposables dans le génome d'*Ae. albopictus*.

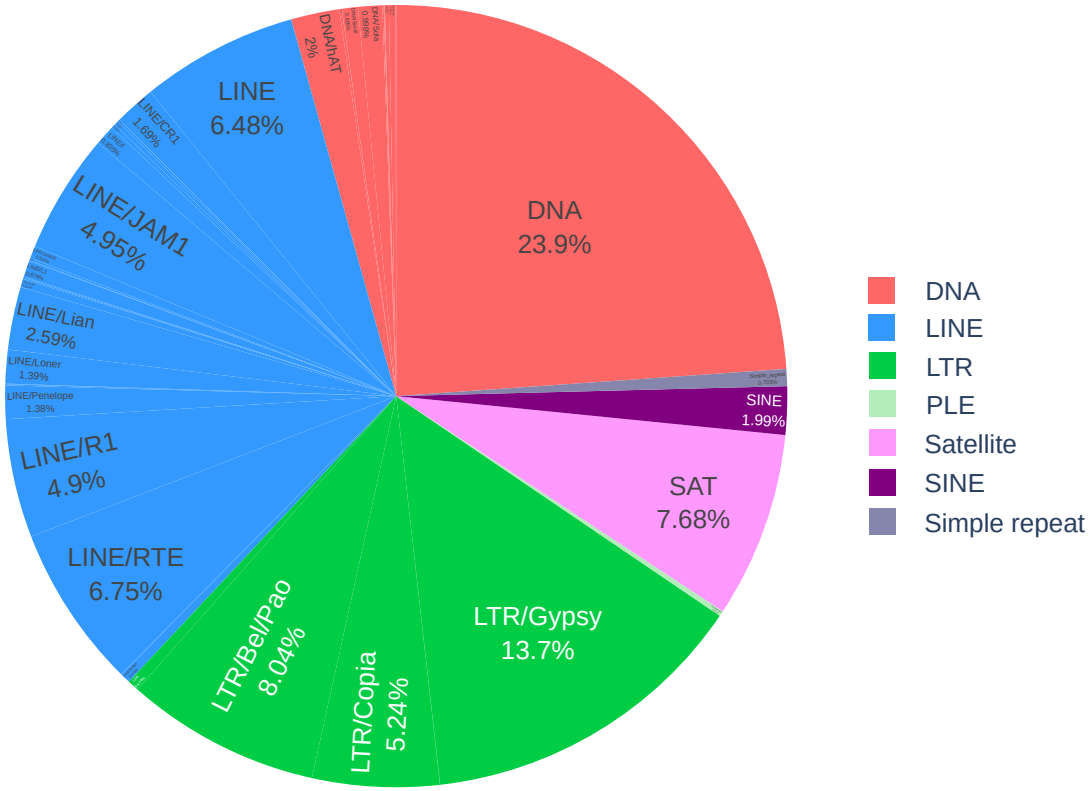

FIGURE 16 – Représentation graphique du contenu en ET d'*Ae. albopictus*.

4 Conclusion et perspectives

L'annotation Cette étape, qui a été particulièrement intense d'un point de vue de temps de calcul, a été indispensable pour la construction de la librairie finale. Dans le document, on a montré l'intérêt (Sous-section 3.2.3 - Clustering des copies et appel des consensi) d'utiliser un double approche homologie-*de novo*. On a pu noter que, d'une part l'utilisation d'un approche par homologie est indispensable pour avoir un ensemble d'éléments fiable et que, d'autre part, un approche *de novo* est nécessaire pour l'identification de nouvelles familles (Figure 11).
Le jeu de 59 848 séquences obtenu a été traité par la suite dans la construction de la librairie.

Construction de la librairie La construction d'une librairie d'ET est une analyse qui aujourd'hui est loin d'être standardisée. Il en est la preuve le fait que très souvent celle-ci consiste uniquement dans une première étape d'annotation (généralement en utilisant seulement un approche par homologie, ex. avec `RepeatMasker`) et une deuxième étape de classification. Cela permet, bien sur, d'avoir une librairie utilisable dans les analyses « downstream », mais qui sera ainsi difficilement complète et soignée. Ici, on a essayer de décrire de la meilleure façon possible le paysage des éléments transposables d'*Ae. albopictus*.
Retravailler les copies obtenues à partir des consensi (Sous-section 3.2.1 - Récupération des copies) générées par les outils d'annotation a permis de gérer aisément la redondance des séquences, en utilisant des outils de masquage du génome (comme `RepeatMasker` dans notre cas) et de clustering (`cd-hit-est`). Il a aussi permis de vérifier le support génomiques des consensi (comme montre en Tableau 4) ainsi que de réunir de manière plus parcimonieuse l'information apportée par les différents outils (en re-appelant des consensi à l'aide de `Refiner`).

La méthodologie utilisée peut cependant, être discutable sous plusieurs points de vue. Par exemple, au niveau du seuil stringent utilisé (Code S6) pour `RepeatMasker` pour récupérer les fragments correspondants aux consensi, une utilisation d'un seuil plus faible aurait permis une détection plus vaste d'éléments ; mais tout comme pour le sous-échantillonnage des séquences lors de l'appel des consensi (lors de l'utilisation de *Refiner*), il s'agit d'une contrainte technique liée au temps de calcul, de difficile résolution.
De même, on aurait pu préférer de filtrer les fragments obtenus en fonction de la longueur, après utilisation d'`OCTFTA`, mais cela aurait demandé un temps de calcul supplémentaire.
On reconnaît aussi la limite technique portée par `cd-hit-est` dans le clustering de séquences plus ou moins divergées, lequel pour avoir un résultat satisfaisant a demandé d'effectuer plusieurs clustering de suite.
Le nombre total de séquences obtenus suite à la construction de la librairie a été de 23 009.

Axes d'amélioration pour le pipeline sur le polissage Bien que le pipeline a été capable de fournir des résultats valables pour notre expérience, il peut être encore développé sur plusieurs aspects. Par exemple, on peut imaginer d'implémenter une fonction permettant d'étendre de façon plus efficace les extrémités du consensus (certaines séquences sont bien étendues mais parfois les extrémités sont moins « lissés » que dans des autres cas) en faisant des ajustement sur les paramètres en fonction des observations empiriques.
Jusqu'à maintenant, on n'a pas pris en compte dans le programme l'augmentation des fragments comme critère pour trancher entre la séquence de l'itération *i* avec celle de l'itération *i-1*. De même on pourrait imaginer de prendre en compte les ORF et, de donner des poids différents aux domaines protéiques détectés selon qu'ils soient trouvé dans le même sens que le consensus ou en anti-sens.
Le jeu de données utilisé pour le test contenait une majorité de séquences provenant de l'ordre LTR, donc il serait intéressant de tester le pipeline sur une jeu de données plus hétérogène.
Enfin, on pourrait également imaginer de concevoir une idée meilleure pour gérer la sur-extension des solo-LTR (en tenant compte de la similarité des régions flanquantes rajoutées à chaque itérations) et de tester les séquences marquées comme SSR (par exemple en essayant de les classifier pour voir si effectivement il ne s'agit pas de faux positifs).

La classification La classification des éléments transposables est une étape cruciale qui permet de connaître la composition en ET du génome analysé. Celle-ci est fortement dépendante de la qualité des bases de données utilisées ; si la base de données manque de certaines catégories d'ET, ceux-ci seront peu, voire pas du tout, classifiés. Aujourd'hui `RepBase` est ce qu'il y a de plus complet en termes de bases de données pour les éléments transposables. Malgré l'utilisation de celle-ci, on a pu remarquer une problématique liée à la classification de certain éléments non-autonomes non représentés, notamment comme les MITE, auxquels on a dédié beaucoup de temps lors de l'annotation avec l'utilisation de `MITE-Tracker`.
Comme perspective pour en améliorer la classification, on propose de classifier manuellement quelques séquences de MITE (ainsi que d'autre catégories pour lesquelles on souhaite améliorer la classification) et, de les joindre à la base de données utilisée pour la classification.

On souhaite également signaler une difficulté globale dans la classification provoquée par une utilisation non unifiée et cohérente des conventions de nomenclature des ET au sein des différentes bases de données ainsi que des différents outils de classification. Très souvent, la hiérarchie et la nomenclature proposée par *Wicker et al.* [1], laquelle représente à ce jour le standard pour la classification des ET, est rarement respectée dans sa globalité. Pour faire un exemple, dans la plupart des outils de classification, les éléments de classe II notés comme DNA, sont directement comparés à les super-familles des LINE ou des SINE ou encore à l'ordre des LTR (pour plus de détail sur la hiérarchie des ET, se référer à la Figure 2).
Pour simplifier et, dans un premier temps permettre une classification homogène entre les différentes outils de classification utilisés dans cet étude, on a préféré d'utiliser une nomenclature « faussée » ; on propose donc parmi les perspectives, de réaliser une meilleure classification, respectant la hiérarchie proposée par *Wicker et al.*, sans aucune exception.

Dans la librairie finale, on a choisi de ne garder pour le moment, que les 5 465 consensi ayant reçu une classification.
On propose ainsi, effectuer une analyse plus fine sur les consensi qui restent $(23009 - 5465 = 17544)$; en particulier, en rajoutant une étape de classification structurelle pour permettre une meilleure détection des éléments comme les TIR et les MITE (lesquels présentent des *Terminal Inverted Repeats*, qui sont facilement identifiables) ; d'effectuer un `blastx` (blast de l'ADN traduit dans les six cadres de lecture possible) contre une base de données (ex. `nr`) de domaines protéiques issus de gènes pour pouvoir exclure toute séquence pouvant être un gène (provenant d'*Ae. albopictus* mais aussi d'insertions virales ou de contaminations lors du séquençage du génome).

En conclusion, lors de ce stage nous avons construit une librairie d'éléments transposables d'*Aedes albpictus* en tenant compte des exigences qu'elle devait respecter ; à savoir l'exhaustivité (pouvoir décrire la totalité du paysage d'ET du génome) et l'intégralité des séquences elles-mêmes (en vue du génotypage sur le projet 1000 génomes).
Juger de l'exhaustivité d'une annotation n'est jamais une tâche simple, mais en comparaison avec les annotations antérieures (Tableau 1), nous avons démontré qu'une dualité entre une approche par homologie et *de novo* peut apporter de l'information supplémentaire par rapport aux deux approches utilisées séparément. Par conséquence, notre librairie permet de masquer 59.49% du génome, contre le 55.81% de l'annotation de *Palatini et al,* 2020 , dans les deux cas le même assemblage a été utilisé en donnant des résultats qui peuvent être comparés.
L'intégralité des séquences est peut être encore plus difficile à évaluer, malgré cela la méthode développée a donné des résultats satisfaisants. (Section 2.2.1 - Polissage).
Au niveau du paysage des ET, nous avons pu re-confirmer que la catégorie d'éléments la plus large en termes de proportion génomique (301Mpb) et de nombre de copies (953 856 instances totales) sont les LINEs (rétrotransposons) lesquels représentent 20.81% du génome. Parmi les LINEs nous avons pu retrouver des nombreuses familles déjà décrites chez les moustiques [46-48], comme *RTE*, *R1*, *Lian* et *Loner*. Nous retrouvons également des proportions assez similaires pour les LTR et les ADN, avec respectivement 17.47% et 16.00% de couverture sur le génome.

Références

1. Wicker, T. *et al.* A unified classification system for eukaryotic transposable elements. en. *Nature Reviews Genetics* **8.** Number : 12 Publisher : Nature Publishing Group, 973-982. ISSN : 1471-0064. http://www.nature.com/articles/nrg2165 (2022) (déc. 2007).
2. Paupy, C., Delatte, H., Bagny, L., Corbel, V. & Fontenille, D. Aedes albopictus, an arbovirus vector : From the darkness to the light. *Microbes and Infection. Forum on Chikungunya* **11,** 1177-1185. ISSN : 1286-4579. https://www.sciencedirect.com/science/article/pii/S1286457909001051 (2022) (1er déc. 2009).
3. Porretta, D., Mastrantonio, V., Bellini, R., Somboon, P. & Urbanelli, S. Glacial History of a Modern Invader : Phylogeography and Species Distribution Modelling of the Asian Tiger Mosquito Aedes albopictus. *PLOS ONE* **7.** Publisher : Public Library of Science, e44515. ISSN : 1932-6203. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0044515 (2022) (2012).
4. Reiter, P. Aedes albopictus and the world trade in used tires, 1988-1995 : the shape of things to come ? *Journal of the American Mosquito Control Association* **14 1,** 83-94 (1998).
5. Gratz, N. G. Critical review of the vector status of Aedes albopictus. *Medical and Veterinary Entomology* **18,** 215-227. ISSN : 1365-2915. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.0269-283X.2004.00513.x (2022) (2004).
6. McKenzie, B. A., Wilson, A. E. & Zohdy, S. Aedes albopictus is a competent vector of Zika virus : A meta-analysis. *PLOS ONE* **14.** Publisher : Public Library of Science, e0216794. ISSN : 1932-6203. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0216794 (2022) (2019).
7. Bourque, G. *et al.* Ten things you should know about transposable elements. *Genome Biology* **19,** 199. ISSN : 1474-760X. https://doi.org/10.1186/s13059-018-1577-z (2022) (19 nov. 2018).
8. McClintock, B. The Origin and Behavior of Mutable Loci in Maize. *Proceedings of the National Academy of Sciences of the United States of America* **36,** 344-355. ISSN : 0027-8424. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1063197/ (2022) (juin 1950).
9. Wells, J. N. & Feschotte, C. A Field Guide to Eukaryotic Transposable Elements. *Annual review of genetics* **54,** 539-561. ISSN : 0066-4197. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8293684/ (2022) (23 nov. 2020).
10. Schnable, P. S. *et al.* The B73 maize genome : complexity, diversity, and dynamics. *Science (New York, N.Y.)* **326,** 1112-1115. ISSN : 1095-9203 (20 nov. 2009).
11. Lander, E. S. *et al.* Initial sequencing and analysis of the human genome. *Nature* **409.** Number : 6822 Publisher : Nature Publishing Group, 860-921. ISSN : 1476-4687. https://www.nature.com/articles/35057062 (2022) (fév. 2001).
12. McCullers, T. J. & Steiniger, M. Transposable elements in Drosophila. *Mobile Genetic Elements* **7,** 1-18. ISSN : null. https://doi.org/10.1080/21592256X.2017.1318201 (2022) (2017).
13. Goubert, C. *et al.* De Novo Assembly and Annotation of the Asian Tiger Mosquito (Aedes albopictus) Repeatome with dnaPipeTE from Raw Genomic Reads and Comparative Analysis with the Yellow Fever Mosquito (Aedes aegypti). *Genome Biology and Evolution* **7,** 1192-1205. ISSN : 1759-6653. https://doi.org/10.1093/gbe/evv050 (2022) (avr. 2015).
14. Melo, E. S. d. & Wallau, G. L. Mosquito genomes are frequently invaded by transposable elements through horizontal transfer. en. *PLOS Genetics* **16.** Publisher : Public Library of Science, e1008946. ISSN : 1553-7404. https://journals.plos.org/plosgenetics/article?id=10.1371/journal.pgen.1008946 (2022) (nov. 2020).
15. Chen, X.-G. *et al.* Genome sequence of the Asian Tiger mosquito, Aedes albopictus, reveals insights into its biology, genetics, and evolution. en. *Proceedings of the National Academy of Sciences* **112.** Publisher : National Academy of Sciences Section : PNAS Plus, E5907-E5915. ISSN : 0027-8424, 1091-6490. https://www.pnas.org/content/112/44/E5907 (2022) (nov. 2015).
16. Palatini, U. *et al.* Improved reference genome of the arboviral vector Aedes albopictus. *Genome Biology* **21,** 215. ISSN : 1474-760X. https://doi.org/10.1186/s13059-020-02141-w (2022) (août 2020).
17. Goerner-Potvin, P. & Bourque, G. Computational tools to unmask transposable elements. en. *Nature Reviews Genetics* **19.** Number : 11 Publisher : Nature Publishing Group, 688-704. ISSN : 1471-0064. https://www.nature.co

m/articles/s41576-018-0050-x (2022) (nov. 2018).

18. Smit, A., Hubley, R. & Green, P. *RepeatMasker Open-4.0* 2013-2015. http://www.repeatmasker.org (2022).

19. Jurka, J. *et al.* Repbase Update, a database of eukaryotic repetitive elements. Anglais. *Cytogenetic and Genome Research* **110.** Publisher : Karger Publishers, 462-467. ISSN : 1424-8581, 1424-859X. https://www.karger.com/Article/FullText/84979 (2022) (2005).

20. Storer, J., Hubley, R., Rosen, J., Wheeler, T. J. & Smit, A. F. The Dfam community resource of transposable element families, sequence models, and genome annotations. *Mobile DNA* **12,** 2. ISSN : 1759-8753. https://doi.org/10.1186/s13100-020-00230-y (2022) (jan. 2021).

21. Flynn, J. M. *et al. RepeatModeler2 : automated genomic discovery of transposable element families* en. preprint (Genomics, nov. 2019). http://biorxiv.org/lookup/doi/10.1101/856591 (2022).

22. Flutre, T., Duprat, E., Feuillet, C. & Quesneville, H. Considering Transposable Element Diversification in De Novo Annotation Approaches. en. *PLOS ONE* **6.** Publisher : Public Library of Science, e16526. ISSN : 1932-6203. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0016526 (2022) (jan. 2011).

23. Ou, S. *et al.* Benchmarking transposable element annotation methods for creation of a streamlined, comprehensive pipeline. *Genome Biology* **20,** 275. ISSN : 1474-760X. https://doi.org/10.1186/s13059-019-1905-y (2022) (déc. 2019).

24. Bao, Z. & Eddy, S. R. Automated De Novo Identification of Repeat Sequence Families in Sequenced Genomes. *Genome Research* **12,** 1269-1276. ISSN : 1088-9051. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC186642/ (2022) (août 2002).

25. Price, A., Jones, N. & Pevzner, P. Price, A.L., Jones, N.C. & Pevzner, P.A. De novo identification of repeat families in large genomes. Bioinformatics 21, i351-i358. *Bioinformatics (Oxford, England)* **21 Suppl 1,** i351-8 (juill. 2005).

26. Ellinghaus, D., Kurtz, S. & Willhoeft, U. LTRharvest, an efficient and flexible software for de novo detection of LTR retrotransposons. *BMC Bioinformatics* **9,** 18. ISSN : 1471-2105. https://doi.org/10.1186/1471-2105-9-18 (2022) (jan. 2008).

27. Ou, S. & Jiang, N. LTR_retriever : A Highly Accurate and Sensitive Program for Identification of Long Terminal Repeat Retrotransposons. *Plant Physiology* **176,** 1410-1422. ISSN : 0032-0889. https://doi.org/10.1104/pp.17.01310 (2022) (fév. 2018).

28. Xu, Z. & Wang, H. LTR_FINDER : an efficient tool for the prediction of full-length LTR retrotransposons. *Nucleic Acids Research* **35,** W265-W268. ISSN : 0305-1048. https://doi.org/10.1093/nar/gkm286 (2022) (juill. 2007).

29. Shi, J. & Liang, C. Generic Repeat Finder : A High-Sensitivity Tool for Genome-Wide De Novo Repeat Detection. *Plant Physiology* **180,** 1803-1815. ISSN : 0032-0889. https://doi.org/10.1104/pp.19.00386 (2022) (août 2019).

30. Su, W., Gu, X. & Peterson, T. TIR-Learner, a New Ensemble Method for TIR Transposable Element Annotation, Provides Evidence for Abundant New Transposable Elements in the Maize Genome. English. *Molecular Plant* **12.** Publisher : Elsevier, 447-460. ISSN : 1674-2052. https://www.cell.com/molecular-plant/abstract/S1674-2052(19)30062-0 (2022) (mars 2019).

31. Xiong, W., He, L., Lai, J., Dooner, H. K. & Du, C. HelitronScanner uncovers a large overlooked cache of Helitron transposons in many plant genomes. en. *Proceedings of the National Academy of Sciences* **111.** Publisher : National Academy of Sciences Section : Biological Sciences, 10263-10268. ISSN : 0027-8424, 1091-6490. https://www.pnas.org/content/111/28/10263 (2022) (juill. 2014).

32. *National Center for Biotechnology Information* en. https://www.ncbi.nlm.nih.gov/ (2022).

33. Boyle, J. H. *et al.* A Linkage-Based Genome Assembly for the Mosquito Aedes albopictus and Identification of Chromosomal Regions Affecting Diapause. *Insects* **12,** 167. ISSN : 2075-4450. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7919801/ (2022) (fév. 2021).

34. Crescente, J. M., Zavallo, D., Helguera, M. & Vanzetti, L. S. MITE Tracker : an accurate approach to identify miniature inverted-repeat transposable elements in large genomes. *BMC Bioinformatics* **19,** 348. ISSN : 1471-2105. https://doi.org/10.1186/s12859-018-2376-y (2022) (oct. 2018).

35. Fu, L., Niu, B., Zhu, Z., Wu, S. & Li, W. CD-HIT : accelerated for clustering the next-generation sequencing data. *Bioinformatics* **28,** 3150-3152. ISSN : 1367-4803. https://doi.org/10.1093/bioinformatics/bts565 (2022) (déc. 2012).

36. Yan, H., Bombarely, A. & Li, S. DeepTE : a computational method for de novo classification of transposons with convolutional neural network. *Bioinformatics* **36,** 4269-4275. ISSN : 1367-4803. https://doi.org/10.1093/bioinformatics/btaa519 (2022) (août 2020).

37. Huang, W., Li, L., Myers, J. R. & Marth, G. T. ART : a next-generation sequencing read simulator. *Bioinformatics* **28,** 593-594. ISSN : 1367-4803. https://doi.org/10.1093/bioinformatics/btr708 (2022) (fév. 2012).

38. Danecek, P. *et al.* Twelve years of SAMtools and BCFtools. *GigaScience* **10,** giab008. ISSN : 2047-217X. https://doi.org/10.1093/gigascience/giab008 (2022) (fév. 2021).

39. Quinlan, A. R. & Hall, I. M. BEDTools : a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26,** 841-842. ISSN : 1367-4803. https://doi.org/10.1093/bioinformatics/btq033 (2022) (mars 2010).

40. Bailly-Bechet, M., Haudry, A. & Lerat, E. "One code to find them all" : a perl tool to conveniently parse RepeatMasker output files. *Mobile DNA* **5,** 13. ISSN : 1759-8753. https://doi.org/10.1186/1759-8753-5-13 (2022) (mai 2014).

41. Li, W. & Godzik, A. Cd-hit : a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* **22,** 1658-1659. ISSN : 1367-4803. https://doi.org/10.1093/bioinformatics/btl158 (2022) (juill. 2006).

42. Camacho, C. *et al.* BLAST+ : architecture and applications. *BMC Bioinformatics* **10,** 421. ISSN : 1471-2105. https://doi.org/10.1186/1471-2105-10-421 (2022) (déc. 2009).

43. Katoh, K., Misawa, K., Kuma, K.-i. & Miyata, T. MAFFT : a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research* **30,** 3059-3066. ISSN : 0305-1048. https://doi.org/10.1093/nar/gkf436 (2022) (juill. 2002).

44. Tumescheit, C., Firth, A. E. & Brown, K. CIAlign : A highly customisable command line tool to clean, interpret and visualise multiple sequence alignments. en. *PeerJ* **10.** Publisher : PeerJ Inc., e12983. ISSN : 2167-8359. https://peerj.com/articles/12983 (2022) (mars 2022).

45. Goubert, C. *et al.* A beginner's guide to manual curation of transposable elements. *Mobile DNA* **13,** 7. ISSN : 1759-8753. https://doi.org/10.1186/s13100-021-00259-7 (2022) (mars 2022).

46. Tu, Z., Isoe, J. & Guzova, J. A. Structural, genomic, and phylogenetic analysis of Lian, a novel family of non-LTR retrotransposons in the yellow fever mosquito, Aedes aegypti. en. *Molecular Biology and Evolution* **15,** 837-853. ISSN : 0737-4038, 1537-1719. https://academic.oup.com/mbe/article-lookup/doi/10.1093/oxfordjournals.molbev.a025989 (2022) (juill. 1998).

47. Biedler, J. & Tu, Z. Non-LTR Retrotransposons in the African Malaria Mosquito, Anopheles gambiae : Unprecedented Diversity and Evidence of Recent Activity. *Molecular Biology and Evolution* **20,** 1811-1825. ISSN : 0737-4038. https://doi.org/10.1093/molbev/msg189 (2022) (nov. 2003).

48. Boulesteix, M. & Biémont, C. Transposable elements in mosquitoes. Anglais. *Cytogenetic and Genome Research* **110.** Publisher : Karger Publishers, 500-509. ISSN : 1424-8581, 1424-859X. https://www-karger-com.docelec.univ-lyon1.fr/Article/FullText/84983 (2022) (2005).

49. Van Rossum, G. & Drake, F. L. *Python 3 Reference Manual* ISBN : 1441412697 (CreateSpace, Scotts Valley, CA, 2009).

50. Wall, L., Christiansen, T. & Orwant, J. *Programming perl* (" O'Reilly Media, Inc.", 2000).

51. GNU, P. *Free Software Foundation. Bash (3.2. 48) [Unix shell program]* 2007.

52. Benson, G. Tandem repeats finder : a program to analyze DNA sequences. *Nucleic Acids Research* **27,** 573-580. ISSN : 0305-1048. https://doi.org/10.1093/nar/27.2.573 (2022) (jan. 1999).

53. *UCSC Genome Browser Home* https://genome.ucsc.edu/index.html (2022).

54. Neph, S. *et al.* BEDOPS : high-performance genomic feature operations. *Bioinformatics* **28,** 1919-1920. ISSN : 1367-4803. https://doi.org/10.1093/bioinformatics/bts277 (2022) (juill. 2012).

55. *NINJA : a tool for large-scale neighbor-joining phylogeny inference - Home* http://wheelerlab.org/software/ninja/ (2022).

56. Mistry, J., Finn, R. D., Eddy, S. R., Bateman, A. & Punta, M. Challenges in homology search : HMMER3 and convergent evolution of coiled-coil regions. *Nucleic Acids Research* **41,** e121. issn : 0305-1048. https://doi.org/10.1093/nar/gkt263 (2022) (juill. 2013).

57. Rognes, T., Flouri, T., Nichols, B., Quince, C. & Mahé, F. VSEARCH : a versatile open source tool for metagenomics. en. *PeerJ* **4.** Publisher : PeerJ Inc., e2584. issn : 2167-8359. https://peerj.com/articles/2584 (2022) (oct. 2016).

58. Chapman, B. & Chang, J. Biopython : Python tools for computational biology. *ACM SIGBIO Newsletter* **20,** 15-19. issn : 0163-5697. https://doi.org/10.1145/360262.360268 (2022) (août 2000).

59. *NumPy. Learn Python at Python.Engineering* https://python.engineering/cat/numpy/ (2022).

60. McKinney, W. pandas : a Foundational Python Library for Data Analysis and Statistics. en, 9.

61. Parisot, N. *et al.* The transposable element-rich genome of the cereal pest Sitophilus oryzae. *BMC Biology* **19,** 241. issn : 1741-7007. https://doi.org/10.1186/s12915-021-01158-2 (2022) (nov. 2021).

62. Li, H. *lh3/seqtk* original-date : 2012-03-23T23 :24 :13Z. Fév. 2022. https://github.com/lh3/seqtk (2022).

63. Ou, S. & Jiang, N. LTR_FINDER_parallel : parallelization of LTR_FINDER enabling rapid identification of long terminal repeat retrotransposons. *Mobile DNA* **10,** 48. issn : 1759-8753. https://doi.org/10.1186/s13100-019-0193-0 (2022) (déc. 2019).

64. Zhang, R.-G., Wang, Z.-X., Ou, S. & Li, G.-Y. *TEsorter : lineage-level classification of transposable elements using conserved protein domains* en. Rapp. tech. Section : New Results Type : article (bioRxiv, oct. 2019), 800177. https://www.biorxiv.org/content/10.1101/800177v1 (2022).

Annexes

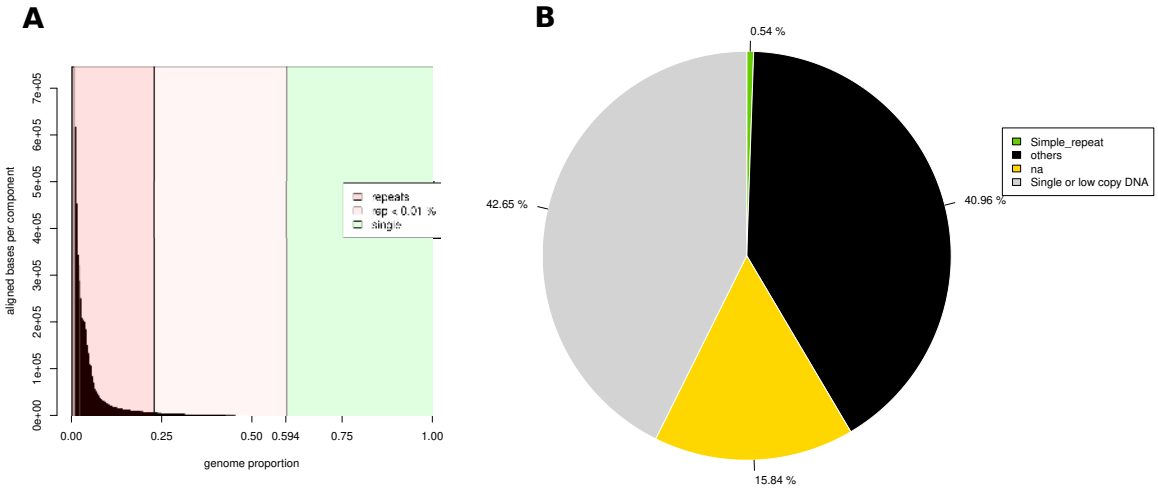

FIGURE S1 – Estimation de la fraction répétée sur un jeu de données réel.

A - Nombre des bases alignés par composante en fonction de la proportion génomique. En rouge, la fraction génomique fortement répétée ; en lilas, la fraction normalement répétée et enfin, en vert l'ADN non répété. **B - Estimation de la composition en éléments transposables.** À différence de la FIGURE 8, ici on obtient la grande tranche noire (correspondant aux éléments identifiés) car la librairie utilisée (`RepBase`), n'est pas formaté de la même manière que la librairie proposée par défaut dans `dnaPipeTE`.

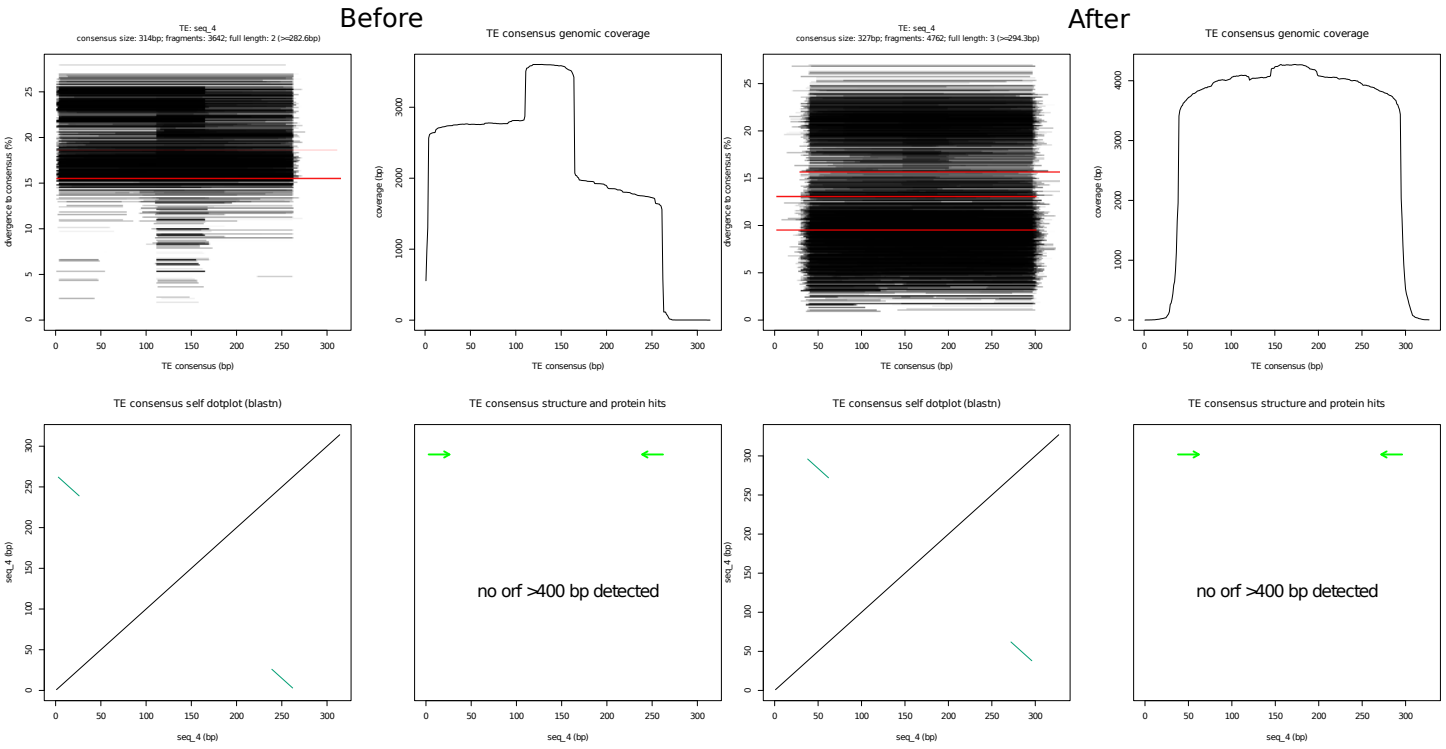

FIGURE S2 – Séquence 4.

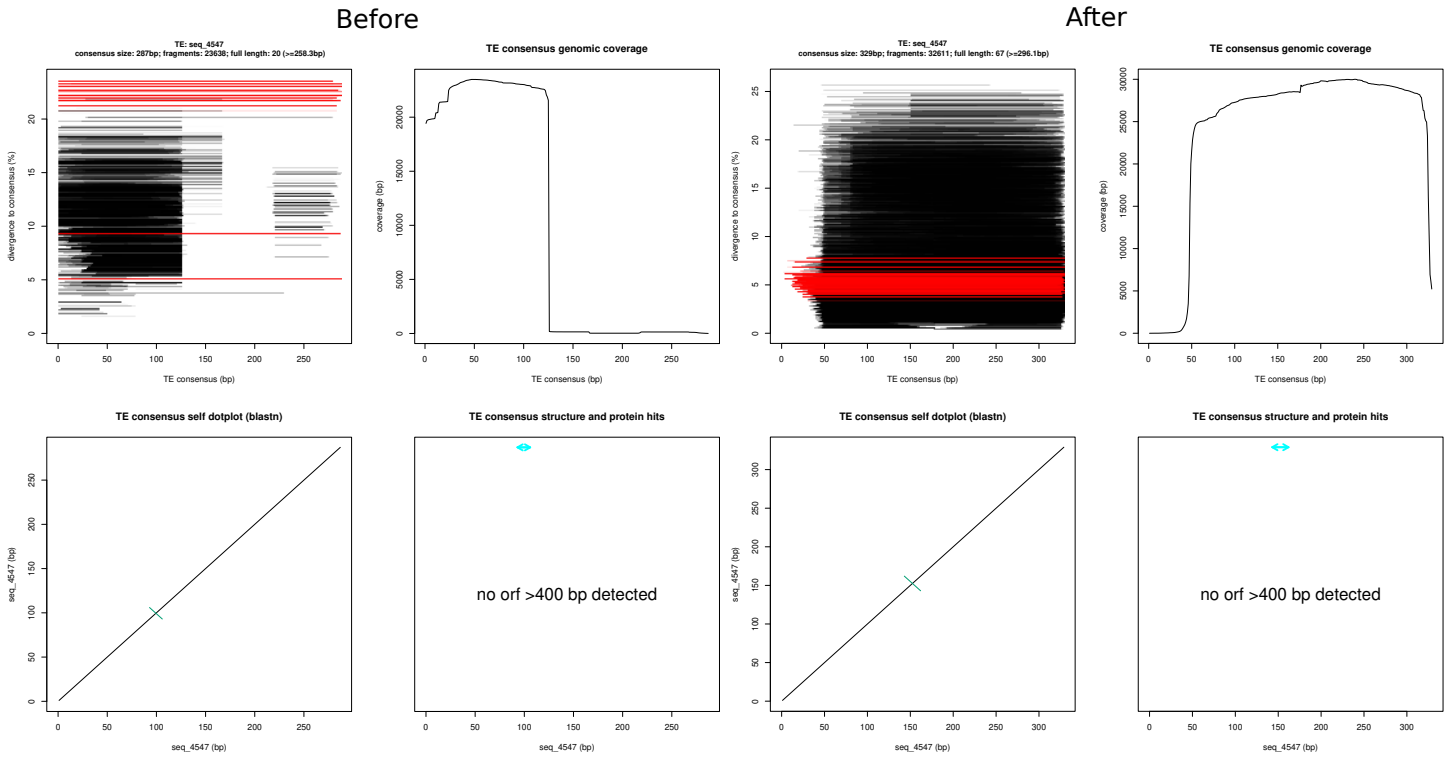

FIGURE S3 – Séquence 4547.

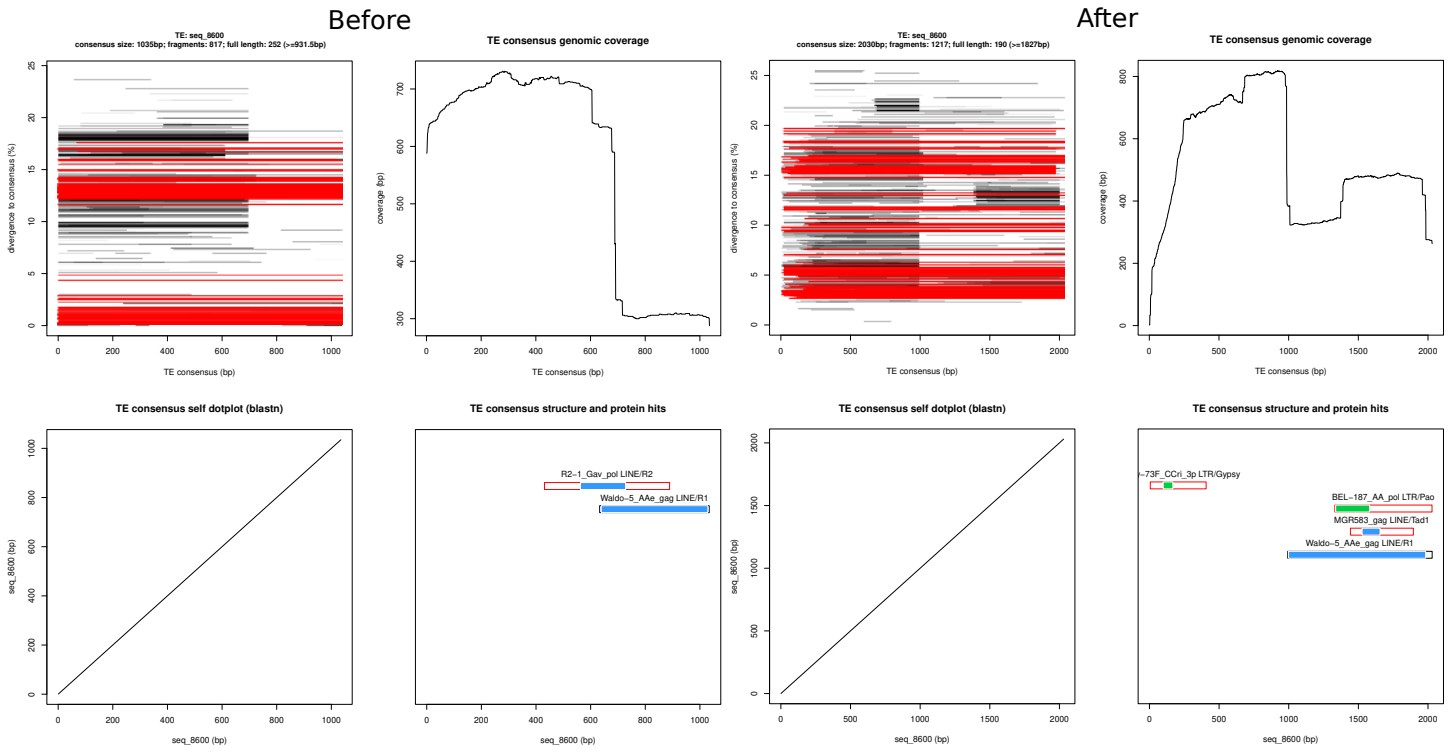

FIGURE S4 – Séquence 8600.

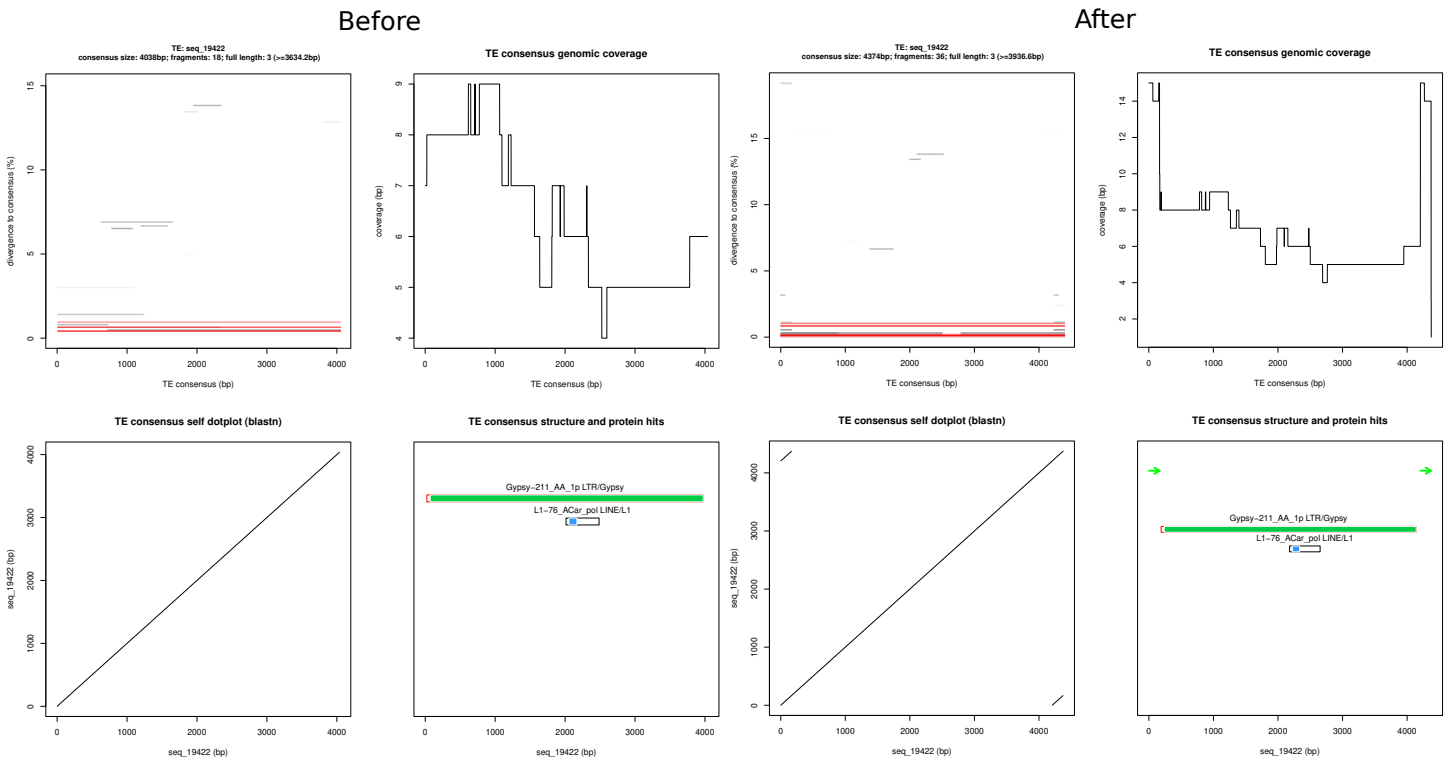

FIGURE S5 – Séquence 19422.

	Date de publication	GenBank accession	Méthode d'assemblage	Longueur totale	Téchnologie de séquençage
Chen et al.	23/02/2016	GCA_001444175.2	`SOAPdenovo - v2.04`	1 923 476 627	`Illumina HiSeq 2000`
Palatini et al.	28/06/2019	GCA_006496715.1	`Canu - v1.7`	2 538 387 871	`PacBio Sequel`
Boyle et al.	20/04/2021	GCA_018104305.1	`Lep-Anchor v. March-2020`	1 450 641 013	PacBio

	Nb de scaffold	N50 scaffold	L50 scaffold
Chen et al.	154 782	201 017	2 851
Palatini et al.	2 197	55 702 539	13
Boyle et al.	574	10 094 208	43

TABLE S1 – Comparaison des différents assemblages génomiques d'*Ae. albopictus* déposés sur le NCBI.

```
# BuildDatabase generate a database for a given reference genome, it is installed with RepeatModeler2
BuildDatabase -name Aalb $DATA_DIR/GCA_006496715.1_Aalbo_primary.1_genomic.fna

# NINJA_DIR: path to folder containg binaries for NINJA
# the export is needed to avoid this error https://github.com/Dfam-consortium/RepeatModeler/issues/134
export $NINJA_DIR

RepeatModeler -database Aalb -pa $nb_cpus -LTRStruct
```

Code S1 – Commandes `RepeatModeler2`.

```
# activation of the conda environment provided by the author
conda activate EDTA

# EDTA needs header with length shorter than 13 characters
awk -F " " '{ if ($0 ~ /^>/) { print ">"$6;} else { print $0}}' GCA_006496715.1_Aalbo_primary.1_genomic.
    fna  | sed -e 's/,//' > GCA_006496715.1_Aalbo_primary.1_genomic.short_header.fna


# run of EDTA_raw.pl for each transposable elements type (ltr|tir|helitron)
perl $EDTA_DIR/EDTA_raw.pl --genome $DATA_DIR/GCA_006496715.1_Aalbo_primary.1_genomic.short_header.fna --
    type ltr|tir|helitron --overwrite 0 -t $nb_cpus
# --overwrite 0: it allows to merge results from differents runs without deleting the results previously
    obtained

# merging and finalising the annotation, it must be run in the same folder as previously commands
EDTA --genome $DATA_DIR/GCA_006496715.1_Aalbo_primary.1_genomic.short_header.fna \
     --overwrite 0 \
     --cds $DATA_DIR/GCF_006496715.1_Aalbo_primary.1_cds_from_genomic.fna \
     --anno 1 \
     --exclude $DATA_DIR/GCF_006496715.1_Aalbo_primary.1_genomic.bed \
     -t $nb_cpus
```

Code S2 – Commandes `EDTA`.

```
# activate conda environnment provided by the author
conda activate MITE-Tracker

# split of the reference in 11 files
faSplit sequence $DATA_DIR/GCA_006496715.1_Aalbo_primary.1_genomic.fna 11 Aalb

cd $MITE_DIR

# run the program in candidates mode on the 11 files
python3 -m MITETracker -g $DATA_DIR/{$nb_file}.fa -w $nb_cpus -j Aalb{$nb_file} --task candidates

mkdir results/cons
cat results/Aalb_*/candidates.csv > results/cons/candidates.csv
cat results/Aalb_*/candidates.fasta > results/cons/candidates.fasta

cd ..

# run the program in the cluster mode
python3 -m MITETracker -g none -w $nb_cpus -j cons --task cluster --min_copy_number 4
```

Code S3 – Commandes `MITE-Tracker`.

```
# output: simulated.bam
art_illumina -ss HSXn -sam -i $ASSEMBLY\_DIR/GCA\_006496715.1\_Aalbo\_primary.1\_genomic.fna -p -l 150 -f
    20 -m 200 -s 10 -o simulated

# sort the bam file
samtools sort -n -m 16G -@ 16 -o simulated.sorted.bam simulated.bam

# recover the fastq files
bedtools bamtofastq -i simulated.sorted.bam -fq simulated1.mapped.fastq -fq2 simulated2.mapped.fastq
```

Code S4 – Commande `Simulation fastq files`.

```
python3 dnaPipeTE.py -input /simulated1.mapped.fastq.gz \
        -output output/ \
        -RM_lib RepeatMasker.lib \ # library contained as default in RepeatMasker
        -genome_size 2570101930 \ # size of the assembly
        -genome_coverage 0.1 \
        -sample_number 2 \
        -RM_t 0.2 \
        -cpu 32
```

Code S5 – Commande `dnaPipeTE`.

```
RepeatMasker -pa 64 -s -a -inv -nolow -gff \
    -dir . \
    -lib $WORK_DIR/TE_Aealb.raw.fa \ # concatenation of the novo tools results and RepBase
    -cutoff 250 $ASSEMBLY\_DIR/GCA_006496715.1_Aalbo_primary.1_genomic.fna
```

Code S6 – Commande `RepeatMasker`.

```
build_dictionary.pl --rm Aalb.out --unknown --fuzzy > dico_fuzzy.txt
one_code_to_find_them_all.pl --rm Aalb.out --ltr dico_fuzzy.txt --fasta --flanking 100 --strict --unknown
    --insert 80
```

Code S7 – Commande `OneCodeToFindThemAll`.

```
cd-hit-est -i TE_copies.fasta -o consensi.fasta -c 0.8 -n 4 -M 10000 -T 54 -d 0
```

Code S8 – Commande `cd-hit-est`.

```
python3 $SCRIPTS/parse_cdhit.py --cdhit consensi.fasta.clstr --fasta copies.fasta -o resultats/ -t 16
```

Code S9 – Commande `Refiner`.

```
polishTE -i file.fasta -g $genome -ins 400 -o $seq_dir
```

Code S10 – Commande pour le pipeline de polissage.

```
tail -n +4 file.out | awk '{if ($2<=20) print}' | awk '{if($7-$6>80) print}' > file.out.filtered
```

Code S11 – Filtrage de l'output de RepeatMasker.

```
awk '{if ($2>200) print}' file.annot | awk '{if ($6-$5>80) print}' > file.annot.filtered
```

Code S12 – Filtrage de l'output de RepeatProteinMask.

Liens externes

Lien 1 https://github.com/TommasoBarberis/TE_Aalb/blob/main/scripts/library_construction/RMout_filter.sh
Lien 2 https://github.com/TommasoBarberis/TE_Aalb/blob/main/scripts/library_construction/octfta_to_fasta.sh
Lien 3 https://github.com/TommasoBarberis/TE_Aalb/blob/main/scripts/library_construction/parse_cdhit.py
Lien 4 https://github.com/TommasoBarberis/polishTE
Lien 5 https://github.com/TommasoBarberis/TE_Aalb/blob/main/scripts/library_construction/parse_rmout.py
Lien 6 https://github.com/TommasoBarberis/TE_Aalb/blob/main/scripts/library_construction/classification/classif_SF.py
Lien 7 https://github.com/TommasoBarberis/TE_Aalb/blob/main/scripts/library_construction/classification/merge_classif.py
Lien 8 https://github.com/TommasoBarberis/TE_Aalb/blob/main/notebooks/bench_deepte.ipynb
Lien 9 https://github.com/TommasoBarberis/TE_Aalb/blob/main/notebooks/denovo_tools_check.ipynb
Lien 10 https://github.com/TommasoBarberis/TE_Aalb/blob/main/notebooks/repeated_fraction.ipynb
Lien 11 https://github.com/TommasoBarberis/TE_Aalb/blob/main/scripts/library_construction/utils/trim_end.py
Lien 12 https://github.com/TommasoBarberis/TE_Aalb/tree/main/manual_curation