Univeristé Evry Val d'Essonne

Licence 3e année Biologie

Étude des transposons chez Arabidopsis thaliana

Auteur	*Email*	*Id*
FAUNIERES Diane	diane-clotilde.faunieres@univ-evry.fr	20150944
TELLIER Virgile	virgile.tellier@univ-evry.fr	20153479

7 mai 2018

Table des matières

1 Introduction

L'étude des génomes a révélé l'existence d'éléments transposables aussi connus sous le nom de transposons. Notre projet porte sur les éléments transposables présents dans le génome de Arabidopsis thaliana. Cette espèce a été une des premières à être séquencée. Les chercheurs se sont portés sur ce choix car son génome est de petite taille et donc plus simple à analyser. De plus, Arabidopsis est facilement cultivable, possède un cycle de reproduction court et est peu coûteux à entretenir. Un transposon est un élément d'ADN répété du génome qui possède la capacité de se multiplier et de se déplacer d'un endroit à un autre sur un même brin d'ADN ou sur un autre brin. Au fur et à mesure de leur multiplication et de leur déplacement ceux-ci peuvent être de moins en moins semblable à l'élément transposable de base. L'analyse du génome de Arabidopsis thaliana à travers l'informatique va nous permettre de trouver la localisation de ces transposons, leur état par rapport à une séquence consensus, et les effets qu'ils pourraient avoir sur les reste des gènes.

2 Les Transposons

Les transposons nécessitent des enzymes telles que l'intégrase et la transposase afin de pouvoir s'insérer ou se deleter d'un génome. Un grand nombre de classes d'entre eux sont par nature autonomes. Les mouvements et réplications des transposons implique des modifications dans la séquence chromosomique, telles que les inversions, délétions et duplications. On observe deux types de déplacements possibles chez un transposon :

— conservateur : sa séquence est transférée d'un site à un autre au sein du génome.

— réplicatif : sa séquence s'insère dans un autre site, la séquence de base restant sur le site original.

Deux copies d'éléments transposables appartiennent à la même famille quand elles ont la même structure, possèdent le même mécanisme de transposition et proviennent les unes des autres. Ils peuvent aussi avoir la possibilité d'interagir par transcomplétion par exemple.

2.1 Éléments à ARN (dits rétrotransposons) / Classe I

Leur cycle de réplication suit généralement le modèle suivant : l'ADN du transposon est transcrit en ARN, puis rétro-transcrit en ADNc[1]. Ce dernier est ensuite inséré dans l'ADN de la cellule hôte[1].

2.1.1 Avec LTR

Les rétrotransposons à LTR tiennent leur nom dû aux extrémités de leur séquence dite Long Terminal Repeat (LTR). Ces derniers de part et d'autre du transposons contiennent le promoteur en 5' et la séquence terminatrice en 3'[1]. Leurs gènes sont en général les suivants :

- gag : son produit s'associe avec les transcrits du LTR afin de former des particules virales.
- pol : responsable de l'intégrase et la transcriptase, nécessaires pour les déplacements et les réplications.
- env : code pour une protéine d'enveloppe héritée des rétrovirus.

2.1.2 Sans LTR

À la différence des éléments transposables avec LTR, la synthèse d'ADNc se fait directement à l'insertion de la nouvelle copie. D'abord un brin du site cible est clivé par une endonucléase, laissant l'ARN s'y attacher. Un brin d'ADN est rétro-transcrit, et après rétraction de l'ARN, le deuxième brin est synthétisé[2].

Parmi les rétrotransposons sans LTR, nous pouvons distinguer les LINEs et les SINEs. Les LINEs, ou Long Interspersed Nuclear Element, sont autonomes grâce à leur production d'un rétro-transposase et d'un enzyme à activité nucléase, et possèdent un promoteur en 5' et une queue poly-A en 3'. Les SINEs, ou Short Interspersed Nuclear Elements, à la différence des LINEs, ne possèdent pas de séquence codant pour une rétro-transcriptase. Ils sont donc dépendants des mécanismes des autres éléments transposables pour leur propagation.

2.2 Éléments à ADN / Classe II

Les transposons de classes II ne passent pas par l'intermédiaire d'un ARN afin de se transposer. Ils se basent sur un mécanisme de couper-insérer, grâce à plusieurs transposases. Ces transposons peuvent être dupliqués, spécifiquement pendant la phase S du cycle cellulaire. Cela se produit quand un site donneur a déjà été répliqué, mais pas le site accepteur[3].

3 Extraction des données

La première étape de ce projet consiste à extraire les données de nos différents fichiers et construire les axes d'étude de notre projet. Nous allons au passage vouloir construire une base de données afin de faciliter l'exploitation et la lecture des résultats de notre recherche. L'ensemble des

données finales est disponible sur GitHub [10.4], ainsi que le conteneur Docker sur Dropbox [10.1].

La commande suivante permet de récupérer le nombre de copies de transposons par famille et super-famille d'éléments transposables.

```
awk '{print $5}' TAIR10_Transposable_Elements.txt | sort| uniq -c | sed -e 's/ *//'
    -e 's/ /\t/' | tr -s ' ' '\t'
```

Notre choix d'étude s'est porté sur les familles suivantes :

— VANDAL14 de la super-famille DNA/MuDR possède 25 copies

— ATCOPIA30 fait partie de LTR/Copia et présente 33 copies

— ATLINE1_2 appartient à LINE/L1 et apporte 41 copies

Nous pouvons maintenant récupérer les séquences consensus de ces familles à partir du fichier Athaliana_RepBase_EMBL.txt, ainsi que les annotations associées.

À l'aide de Python [9.2], nous pouvons maintenant utiliser les outils *extractseq* de Emboss et *blastall* de Blast+ afin d'extraire les séquence de chaque transposons et les aligner contre leur référence afin de créer nos fichiers blast.

D'autre part, un autre alignement entre les séquences consensus et les transposons est généré avec l'outil *water* de Emboss.

Pour préparer nos fichiers de données pour la base de données, les fichiers ont été formatés en *.tsv* à l'aide Python [9.2], sauf dans le cas des gènes. Ces derniers ont été obtenus à partir d'un fichier contenant les protéines connues d'Arabidopsis thaliana [9.2].

Au final, il nous reste à créer une base de données sous PostgreSQL, et à entrer l'ensemble des fichiers obtenus auparavant.

Ce processus d'extraction de données n'a évidement pas été effectué d'une seule traite. Nous avons du recréer nos fichier de nombreuses fois pour de multiples raisons (données manquantes, non conformité avec les attentes du projet, commandes obsolètes ne produisant pas les résultats attendus...). Cela n'a rien de surprenant pour un projet informatique de traitement de données.

4 Alignements et annotations

4.1 Alignement avec Smith–Waterman

L'algorithme Smith-Waterman est un algorithme basé sur la programmation dynamique. Il produit un alignement local, c'est-à-dire qu'il va tenter de trouver les zones de forte homologie

en priorité [2]. En revanche, cet algorithme retourne le meilleur alignement possible. À cause de cela, nous n'allons pas nous baser sur Smith-Waterman pour nos annotations. En effet, de part la nature des transposons, il est possible de ne pas trouver les alignements recherchés, ou même ceux de moindre qualité.

4.2 Comparaison avec Blast

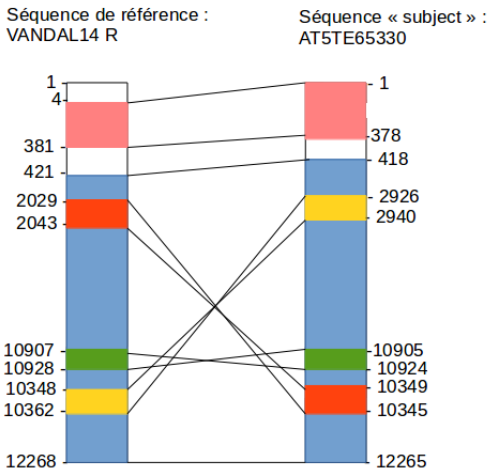
L'algorithme Blast est de même un algorithme produisant des alignements locaux [3]. Il est basé sur des heuristiques et des matrices de score. Le grand avantage de cet algorithme est qu'il retourne tous les alignements statistiquement significatifs, au lieu du meilleur alignement. Cet algorithme correspond à nos attentes, il va nous permettre d'annoter nos copies des transposons, de vérifier s'ils sont fonctionnels et leur impact les gènes de notre organisme.

```
# BLASTN 2.2.26 [Sep-21-2011]
# Query: ATCOPIA30_I AC005970 Internal region of the ATCOPIA30 LTR retrotransposon.
# Database: ATCOPIA30.txt
# Fields: Query id, Subject id, % identity, alignment length, mismatches, gap openings, q. start, q. end, s. st
\TCOPIA30_I     AT2TE10355      100.00  1588    0       0       2282    3869    1588    1       0.0     3106
\TCOPIA30_I     AT2TE10355      100.00  1122    0       0       1088    2209    2782    1661    0.0     2224
\TCOPIA30_I     AT2TE10355      100.00  681     0       0       353     1033    3517    2837    0.0     1255
\TCOPIA30_I     AT2TE10355      100.00  269     0       0       1       269     3869    3601    2e-121   426
\TCOPIA30_I     AT2TE10355      100.00  16      0       0       3149    3164    706     721     0.010   32.2
\TCOPIA30_I     AT2TE10355      100.00  16      0       0       475     490     3380    3395    0.010   32.2
\TCOPIA30_I     AT2TE10355      100.00  14      0       0       493     506     3364    3377    0.16    28.2
\TCOPIA30_I     AT2TE10355      100.00  12      0       0       1804    1815    547     536     2.4     24.3
```

L'alignement des copies nous permet alors de voir précisément quelle portion de la séquence de référence est intacte chez notre copie ou non et si il y a eut des délétions.

Exemple : Schéma de l'alignement de AT5ATE65330 avec la copie de référence VANDAL14

AT5ATE65330 faisant partie des copies que nous avons considéré comme étant complète au vue de ces résultats.



4.3 Annotation des copies

L'intérêt de l'annotation que ce soit pour les gènes ou pour les copies de transposons est d'identifier les gènes, les copies de transposons ainsi que leur niveau de conservation. L'annotation permet aussi l'identification de répétitions, et la prédiction de gène codant pour des protéines et des ARNs.

Afin de bien annoter nos copies, il faut, selon le type de nos copies, faire attention à des caractéristiques différentes. En effet, pour les rétro-transposons à LTR, il nous faut veiller à ce que les 2 séquences LTR soient conservées, car c'est dans celles-ci que se trouve le promoteur permettant le passage par l'intermédiaire ARN. Une délétion dans ces régions causerait ainsi la "non-fonctionnalité" de la copie du rétrotransposon. Pour les copies issues d'une famille de type DNA/MuDR ou LINE, il ne faut pas de délétion majeur au sein de la copie.

Grâce à BlastAll et les fichiers obtenus nous pouvons donc annoter chaque copies ainsi que faire le tri de celles-ci. Nous les avons alors répertoriés dans plusieurs catégories :

— Complète : La copie du transposons possède une longueur d'alignement très proche, voire parfaitement identique à la longueur de la copie de référence de la famille du transposon.
— Incomplète : La copie du transposons possède des alignements avec un pourcentage élevé tout le long de la séquence de référence de la famille du transposons, mais il manque quelques portions(gaps).
— Delete : La copie du transposon ne correspond qu'à un fragment de la copie de référence de la famille.
— Fonctionnelle : La copie possède une double correspondance pour la séquence LTR de la référence (en 3' et en 5'), attestant de la présence d'une séquence LTR plus ou moins conservée à chaque extrémité de la copie, tout en ayant un alignement avec la région interne. (Uniquement pour les éléments de type LTR).

5 Conception de la base de données

Une étape de notre projet étant de créer un base de données afin de réunir les informations nécessaires concernant les chromosomes, les gènes, les copies et leur famille, ainsi que les alignements. Cela nous offrira par la suite la possibilité d'effectuer des requêtes sur celle-ci et recouper les différentes données obtenues au cours du projet.

5.1 Schéma Entité-Association

La première étape pour la conception de notre base de données est de construire un schéma Entité-Association. Nous avons remarqué que nous travaillons avec cinq entités (gène, famille, chromosome, copie, alignement) et 2 associations :

— Une entre chromosome et gène.
— Une entre chromosome et copie.

À partir de cette étape, il a été possible de produire un schéma Entité-Association [4].

5.2 Écriture du code SQL

Le code SQL[9.3] a été écrit dans le but de créer une base de données grâce au logiciel de gestion PostgreSQL.

Les différentes difficultés rencontrées à ce niveau ont été les suivantes :
— Formatage des données pour avoir des fichiers conformes.
— Organisation du code en lui-même, l'ordre de création des tables et l'utilisation de clés auto-incrémentées n'étant pas trivial.
— Le logiciel PostgreSQL est volatile lorsque configuré incorrectement.

Le logiciel MySQLWorkBench a permit un correction d'éventuels problème d'écriture ou de cohérence, ainsi que la génération d'une image du schéma EA.

6 Requêtes et Résultats

Nous cherchons à savoir si nos copies de transposons peuvent affecter les fonctions de gènes se trouvant à proximité.

Pour bien effectuer nos recherches nous devons tout d'abord définir à quelle distance en amont et en aval du gène la présence de l'élément transposable aurait des conséquences sur le gène. Nous savons qu'Arabidopsis thaliana est une espèce eucaryote et donc que sa régulation est celle d'un organisme eurcaryote. En aval, nous savons que les éléments transposable auraient des conséquences sur le gène si ils se situaient dans la région promotrice proximale. Nous avons effectué des recherches à ce sujet, nous donnant -200 bp avant le début du gène pour la région promotrice proximale et -1960 bp pour la région contenant les enhancers. En amont du gène, le dernier exon possède, au sein de sa séquence, le codon stop. Ainsi, nous pouvons donc laisser en amont du gène une marge de la taille approximative d'un exon que nous avons trouvé de 296 bp.

Nous avons dû également faire attention aux chromosomes sur lesquels se trouvaient les gènes et les copies de nos éléments transposables afin que nos résultats soient cohérents.

Requête permettant de trouver les id des copies qui sont "complètes" tout types de copies confondues :

```
1 select id_t from copietrans where annotations='complete';
```

Résultats :

```
bi=# select * from copietrans where annotations='complete';
    id_t    |  dbt_t   |  fin_t   | sens_c |   id_f   |   type   | taille_t | annotations
------------+----------+----------+--------+----------+----------+----------+-------------
 AT1TE09080 |  2788430 |  2800701 | +      | VANDAL14 | DNA/MuDR | 12272    | complete
 AT1TE23740 |  7352930 |  7365197 | -      | VANDAL14 | DNA/MuDR | 12268    | complete
 AT2TE54910 | 12566155 | 12578422 | +      | VANDAL14 | DNA/MuDR | 12268    | complete
 AT3TE40530 |  9723884 |  9736151 | +      | VANDAL14 | DNA/MuDR | 12268    | complete
 AT5TE65330 | 18119471 | 18131741 | +      | VANDAL14 | DNA/MuDR | 12271    | complete
(5 rows)
```

Requête permettant de trouver les informations concernant les alignements dont la longueur est supérieur à 300 et dont le pourcentage de correspondance dans l'alignement est égale à 100% :

```
1 select * from alignement where longueur_al>300 and percent=100;
```

Résultats :

```
 id |   id_cop1   |  id_cop2   | percent | longueur_al | mismatch | gap | q_start | q_end | s_star | s_end
----+-------------+------------+---------+-------------+----------+-----+---------+-------+--------+-------
  1 | ATCOPIA30_I | AT2TE10355 |     100 |        1588 |        0 |   0 |    2282 |  3869 |   1588 |     1
  2 | ATCOPIA30_I | AT2TE10355 |     100 |        1122 |        0 |   0 |    1088 |  2209 |   2782 |  1661
  3 | ATCOPIA30_I | AT2TE10355 |     100 |         681 |        0 |   0 |     353 |  1033 |   3517 |  2837
(3 rows)
```

Requête permettant de trouver les copies dites "complètes" se trouvant exclusivement à l'intérieur d'un gène :

```
1 select * from copietrans, gene, copietsurchromo, genesurchromo, chromosome where
    gene.num_chromo=num.copietrans and annotations='complete' and fin_t<fin_g and
    dbt_t>dbt_g;
```

-> Résultats : Ici, si l'on trouve un résultat, cela signifie qu'une des copies se trouve alors dans entièrement dans un gène et qu'elle est donc susceptible d'affecter ce dernier.

Requête permettant de trouver les copies pouvant affecter les gène en se situant à proximité (en aval à maximum -200 bp) d'un gène et pouvant donc se trouver dans la région promotrice proximale :

```
1 select id_t from gene, copietrans where gene.num_chromo=copietrans.num and dbt_g>
    fin_t and((dbt_g-fin_t)<200);
```

-> Résultats : Dans le cas, on l'on trouverait des résultats, ici, cela signifierait que par sa

présence la copie pourrait affecter le gène en étant présent dans sa séquence promotrice proximale.

Requête permettant de trouver les copies pouvant affecter les gènes en se situant à proximité (en aval et en amont) ou à l'intérieur d'un gène :

```
1 select * from copietrans natural join chromosome, gene where gene.num_chromo=num.
    copietrans and dbt_g>fin_t and((dbt_g-fin_t)<200) or (fin_g<fin_t and ((fin_t-
    fin_g)<296)) or (fin_t<fin_g and dbt_t>dbt_g);
```

-> Résultats : Comme nous avons pu le voir quelques requêtes plus haut, il y a peu de nos copies qui sont complètes, avec les autres restrictions que cette requête leur impose, nous n'avons pas de résultats pour cette requête. Montrant ainsi que nos copies d'éléments transposables ne sont pas susceptibles d'affecter un gène se trouvant sur le même chromosome.

Requête permettant de trouver toutes les copies alignées avec la séquence de référence :

```
1 select * from alignement where id_cop2='AT2TE10355';
```

-> Résultats :

id	id_cop1	id_cop2	percent	longueur_al	mismatch	gap	q_start	q_end	s_star	s_end
1	ATCOPIA30_I	AT2TE10355	100	1588	0	0	2282	3869	1588	1
2	ATCOPIA30_I	AT2TE10355	100	1122	0	0	1088	2209	2782	1661
3	ATCOPIA30_I	AT2TE10355	100	681	0	0	353	1033	3517	2837
4	ATCOPIA30_I	AT2TE10355	100	269	0	0	1	269	3869	3601
5	ATCOPIA30_I	AT2TE10355	100	16	0	0	3149	3164	706	721
6	ATCOPIA30_I	AT2TE10355	100	16	0	0	475	490	3380	3395
7	ATCOPIA30_I	AT2TE10355	100	14	0	0	493	506	3364	3377
8	ATCOPIA30_I	AT2TE10355	100	12	0	0	1804	1815	547	536
9	ATCOPIA30_I	AT2TE10355	100	12	0	0	759	770	1029	1040
10	ATCOPIA30_I	AT2TE10355	100	12	0	0	2396	2407	1463	1474
11	ATCOPIA30_I	AT2TE10355	100	12	0	0	638	649	1867	1878
12	ATCOPIA30_I	AT2TE10355	100	12	0	0	48	59	1902	1913
13	ATCOPIA30_I	AT2TE10355	100	12	0	0	3323	3334	2066	2055
14	ATCOPIA30_I	AT2TE10355	100	12	0	0	2830	2841	3100	3111
15	ATCOPIA30_I	AT2TE10355	100	12	0	0	1992	2003	3221	3232
16	ATCOPIA30_I	AT2TE10355	100	12	0	0	1957	1968	3811	3822
17	ATCOPIA30_I	AT2TE10355	100	11	0	0	1937	1947	717	727
18	ATCOPIA30_I	AT2TE10355	100	11	0	0	2304	2314	1498	1508
19	ATCOPIA30_I	AT2TE10355	100	11	0	0	2362	2372	1556	1566
20	ATCOPIA30_I	AT2TE10355	100	11	0	0	631	641	1652	1662
21	ATCOPIA30_I	AT2TE10355	100	11	0	0	3143	3153	1923	1933

7 Conclusion

Ce projet avait pour but de nous familiariser avec les techniques de bio-informatique, nous permettant ainsi de combiner nos compétences de biologie avec celles acquises en informatique.

Nous devions, à partir de nos 3 familles de copies choisies, regarder grâce à la base de données que nous avons conçu si certaines de nos copies pouvaient affecter un ou plusieurs gène de l'espèce étudiée, ici, Arabidopsis thaliana. Nous avons pu voir grâce aux alignements qu'une copie pouvait être très conservée, délétée d'une portion plus ou moins grande, ou incomplète voire même beaucoup trop courte. En interrogeant notre base de données, nous avons pu apporter un début de réponse aux différentes questions de ce projet.

Dans le cas des copies issues de nos familles, nous avons peu de copies complètes et dans le cas des éléments transposables à LTR aucune de fonctionnelle. Cela a un impact conséquent sur la qualité des nos résultats. Il aurait été nécessaire d'annoter l'ensemble des copies pour avoir des résultats significatifs.

8 Perspectives

La continuation de ce projet passe par plusieurs voies.

La première est l'extension de la base de données dans de multiples directions. Nous pourrions ajouter de multiples espèces afin d'étudier les transposons à travers le monde vivant. De plus, une étude continue pourrait fournir des informations sur les mouvements des transposons en fonction du temps. Il est aussi possible de recroiser cette base de données avec des maladies connues, afin d'étendre les connaissances du domaine médical.

Nous pouvons considérer la recherche et le développement de méthodes alternatives autres que les alignements afin d'étudier les transposons. La recherche à travers les patterns propres aux éléments transposables en se basant sur des critères connus est une piste.

Enfin, il est tout à fait plausible que ce projet puisse aboutir à une étude sur l'efficacité des différents transposons vis à vis du transfert et de l'insertion de gènes, ainsi que la conservation de l'intégrité de leur séquence à travers le temps. Cela pourrait fournir un outil alternatif à crispr-cas9.

9 Annexes

9.1 Figures

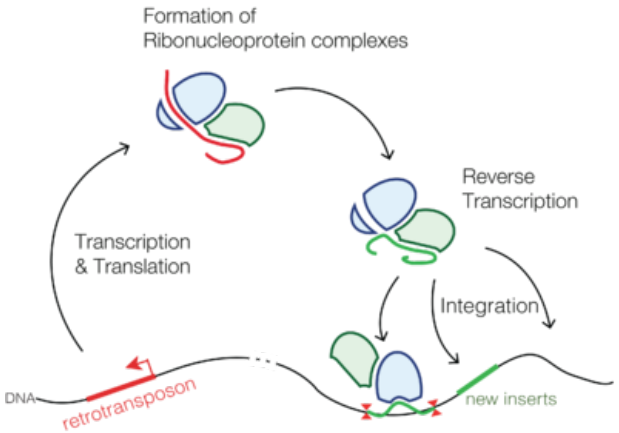


FIGURE 1 – Schéma simplifié du mécanisme de mouvement des rétrotransposons.

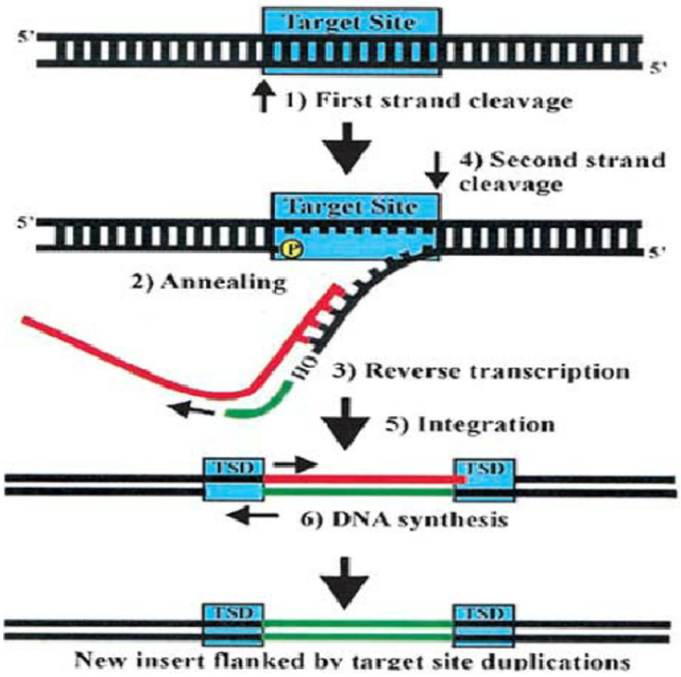


FIGURE 2 – Schéma du mécanisme rétrotransposons sans LTR (d'après Ostertag et Kazazian 2001).

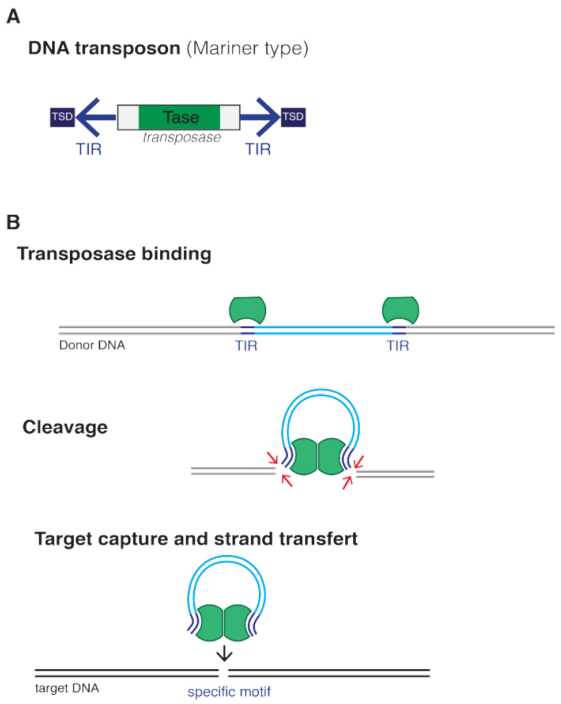


FIGURE 3 – A : Les séquences TIR(inverted tandem repeats) sont de part et d'autre du transposon. B : Mécanisme de transposition. Les transposases utilisent les TIRs comme séquences cibles.

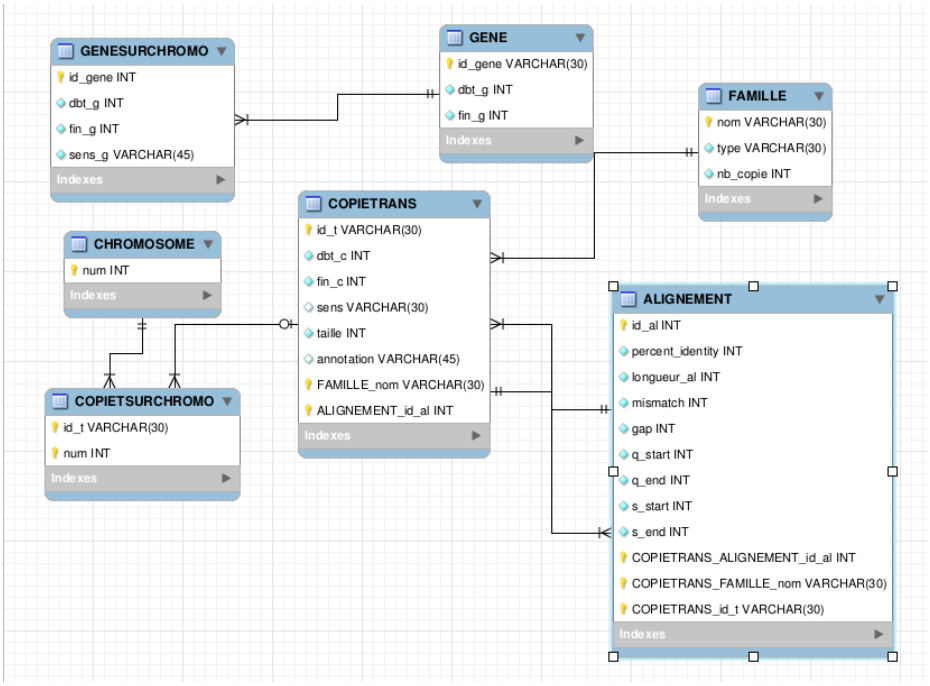


FIGURE 4 – Schéma Entité-Association de la base de donnée du projet.

9.2 Scripts Python

```
#!/usr/bin/python3
from subprocess import call
from subprocess import check_output
import datetime

fam = "ATCOPIA30"
family = fam+".txt"

f = open(family,'w')

zones = check_output(["grep",fam,"TAIR10_Transposable_Elements.txt"]).decode("utf-8
    ").split('\n')

now = datetime.datetime.now()

for i in zones:
        if len(i) < 1:
                break
        i = i.split("\t")
        start = i[2]
        stop = i[3]
        X = i[0][2]
        ref = i[0]
        command = []
        command.append("extractseq")
        filename = "".join(["sequence",i[0][2],".fasta"])
        command.append(filename)
        command.append("-r")
        region = "".join([start,"-",stop])
        command.append(region)
        command.append("t.tmp")
        command.append("-sid1")
        command.append(ref + "␣"+family+ "␣"+str(i[1]) +"␣" + str(now))
        call(command)
        call(["cat","t.tmp"], stdout = f)

call(["rm","t.tmp"])
f.close()

mkdb = ["formatdb","-i",family,"-p","F"]
call(mkdb)

fasta = fam.lower()+".fasta"
blast = ["blastall","-m","9","-p","blastn","-d",family,"-o",fam+".blast","-i",fasta
    ]
```

```
call(blast)
```

Listing 1 – Extraction des transposons du fichier TAIR10_Transposable_Elements.txt, puis alignement contre une base de donnée Blast générée à partir des séquences concensus

```
#!/usr/bin/python3

inputfile = "TAIR10_TE.fas"
outputfile = "copie.tsv"

f = open(inputfile,'r')

line = f.readline()

buff = []

while line:

        if ">" in line:
                line = line.split('|')
                buff.append(line)




        line = f.readline()

f.close()

w = open(outputfile, 'w')

for line in buff:
        w.write("\t".join([line[0][1:],line[2],line[3],line[1], line[4], line[5],
    line[6][:-3]])+'\n')

w.close()
```

Listing 2 – Script Python modèle afin de générer des fichiers *.tsv* pour la base de données

```
#!/usr/bin/python3

pepfile = "TAIR10_pep_20101214_updated.txt"

f = open(pepfile,'r')

gene = {}

line = f.readline()

```

```
11 while line:
12         if line[0] is not ">":
13                 line = f.readline()
14                 continue
15         line = line.split("|")
16         name = line[0]
17         name = name.split(">")[1]
18         name = name.split(".")[0]
19 
20         info = line[3]
21         info = info.split(":")[1]
22         forw = info.split(" ")[1]
23         info = info.split(" ")[0]
24         info = info.split("-")
25         first = int(info[0])
26         last = int(info[1])
27         print(info)
28 
29 
30         if name not in gene:
31                 gene[name] = (first,last,forw)
32         else:
33                 if forw is "FORWARD":
34                         if gene[name][0]>first:
35                                 gene[name] = (first,gene[name][1],forw)
36                         if gene[name][1]<last:
37                                 gene[name] = (gene[name][0],last,forw)
38                 else:
39                         if gene[name][0]<first:
40                                 gene[name] = (first,gene[name][1],forw)
41                         if gene[name][1]>last:
42                                 gene[name] = (gene[name][0],last,forw)
43 
44         line = f.readline()
45 
46 f.close()
47 
48 w = open("dico2.tsv",'w')
49 
50 for ent in gene:
51         w.write("\t".join([ent,str(ent[2]),str(gene[ent][0]),str(gene[ent][1]),str(
    gene[ent][2])])+"\n")
```

Listing 3 – Extraction des noms et positions des gènes à partir du fichier TAIR10_pep_20101214_updated.txt

9.3 Code SQL

```
1  CREATE TABLE CHROMOSOME(
2  num VARCHAR(30),
3  taille_c INT NOT NULL,
4  CONSTRAINT pk_chromosome PRIMARY KEY(num));
5
6  CREATE TABLE FAMILLE(
7  id_f VARCHAR(30),
8  nb_copie INT NOT NULL,
9  type_f VARCHAR(30),
10 CONSTRAINT pk_famille PRIMARY KEY(id_f));
11
12 CREATE TABLE COPIETRANS(
13 id_t VARCHAR(30),
14 dbt_t INT NOT NULL,
15 fin_t INT NOT NULL,
16 sens_c VARCHAR(30),
17 id_f VARCHAR(30),
18 type VARCHAR(30),
19 taille_t VARCHAR(30),
20 annotations VARCHAR(30),
21 CONSTRAINT pk_copietrans PRIMARY KEY(id_t),
22 CONSTRAINT fk_famille FOREIGN KEY(id_f) REFERENCES FAMILLE(id_f));
23
24 CREATE TABLE GENE(
25 id_g VARCHAR(30),
26 num_chromo VARCHAR(30),
27 dbt_g INT NOT NULL,
28 fin_g INT NOT NULL,
29 sens_g VARCHAR(30),
30 CONSTRAINT pk_gene PRIMARY KEY(id_g));
31
32
33 CREATE TABLE ALIGNEMENT(
34 ID BIGSERIAL PRIMARY KEY,
35 id_cop1 VARCHAR(30) NOT NULL,
36 id_cop2 VARCHAR(30) NOT NULL,
37 percent FLOAT,
38 longueur_al INT NOT NULL,
39 mismatch INT,
40 gap INT,
41 q_start INT,
42 q_end INT,
43 s_star INT,
44 s_end INT,
45 CONSTRAINT fk_alignement2 FOREIGN KEY(id_cop2) REFERENCES COPIETRANS(id_t));
```

```
46
47 CREATE TABLE GENESURCHROMO(
48 num VARCHAR(30),
49 id_g VARCHAR(30),
50 CONSTRAINT fk_chromosome FOREIGN KEY(num) REFERENCES CHROMOSOME(num),
51 CONSTRAINT fk_gene FOREIGN KEY(id_g) REFERENCES GENE(id_g),
52 CONSTRAINT pk_genesurchromo PRIMARY KEY(num, id_g));
53
54 CREATE TABLE COPIETSURCHROMO(
55 id_t VARCHAR(30),
56 num VARCHAR(30),
57 CONSTRAINT pk_copietsurchromo PRIMARY KEY(num, id_t),
58 CONSTRAINT fk_chromosome1 FOREIGN KEY(num) REFERENCES CHROMOSOME(num),
59 CONSTRAINT fk_copietrans FOREIGN KEY(id_t) REFERENCES COPIETRANS(id_t));
60
61 copy chromosome from '/home/fnrs/Bureau/ProjetBI/fichierbd/X.tsv';
62 copy gene from '/home/fnrs/Bureau/ProjetBI/fichierbd/dico2.tsv';
63 copy famille from '/home/fnrs/Bureau/ProjetBI/fichierbd/family2.tsv';
64 copy copietrans from '/home/fnrs/Bureau/ProjetBI/fichierbd/copie_3.tsv';
65
66 CREATE TABLE temp (id_cop1 VARCHAR(30), id_cop2 VARCHAR(30) NOT NULL,
67 percent FLOAT, longueur_al INT NOT NULL, mismatch INT, gap INT, q_start INT, q_end
       INT, s_star INT, s_end INT);
68 COPY temp FROM '/home/fnrs/Bureau/ProjetBI/fichierbd/ATCOPIA30_4.blast';
69 INSERT INTO ALIGNEMENT SELECT nextval('alignement_id_seq'),* FROM temp;
70 DROP TABLE temp;
```

Listing 4 – Code SQL de création de la base de données et insertion des données du projet.

10 Matériels et Méthodes

10.1 Environnement de Travail

L'ensemble du travail a été effectué sous deux ordinateurs. Le premier utilisant le système d'exploitation Ubuntu 16.04.4 LTS 64 bits, qui a été simulé avec un conteneur Docker sous Windows 10 :

- Conteneur Ubuntu [lien]
- Docker [lien]
- Le conteneur Docker créé est disponible à [lien]

Le deuxième possèdant similairement Ubuntu 14.04 LTS.

Toutes les installations sur Ubuntu ont été effectuées grâce à APT (Advanced Package Tool).

10.2 Logiciels

Les logiciels suivants ont été utilisé pour traiter les données :

— Gestion de base de données : PostgreSQL 10.4 [lien]

— Génération de l'image du schéma Entité-Association et intégrité de la base de données [lien]

— Alignement de séquences : BLAST+ [lien]

— Package d'outils bio-informatiques : Emboss 6.3.0 [lien]

10.3 Langages de programmation

Les langages et outils suivant ont été utilisé pour créer des commandes et des scripts afin de traiter nos données :

— Commandes courrantes de UNIX et bash [lien]

— Le langage Python3 [lien]

— Le langage Awk [lien]

10.4 Fichiers et données du projet [3]

Liens vers les différents fichiers utilisés dans le projet :

— TAIR10_Transposable_Elements.txt : Liste des transposons [lien]

— TAIR10_TE.fas : Séquences de chaque transposons [lien]

— TAIR10_pep_20101214 : Liste de l'ensemble des protéines [lien]

— sequenceA.fasta : Séquence du chromosome A de Arabidopsis thaliana [1] [2] [3] [4] [5]

— L'ensemble des fichiers est disponible à [lien]

Références

[1] https://www.sciencedirect.com/science/article/pii/S0960982212004459.

[2] Gustav Wieds. "Bioinformatics explained : Smith-Waterman." In : (Mai 2007).

[3] http://etutorials.org/Misc/blast/Part+III+Practice/Chapter+5.+BLAST/5.2+The+BLAST+Algorithm/.