Chapter 1

An Overview of Best Practices for Transposable Element Identification, Classification, and Annotation in Eukaryotic Genomes

Fernando Rodriguez and Irina R. Arkhipova

Abstract

Transposable elements (TEs) exert an increasingly diverse spectrum of influences on eukaryotic genome structure, function, and evolution. A deluge of genomic, transcriptomic, and proteomic data provides the foundation for turning essentially any non-model eukaryotic species into an emerging model to study any and all aspects of organismal biology, ultimately shaping future directions for biomedical, environmental, and biodiversity research. However, identification and annotation of the mobile genome component still lags behind the standards accepted for host gene annotation. To achieve the objective of providing every genome project with a comprehensive description of its mobilome component in addition to the standard genic and transcriptomic datasets, each step of TE identification, classification, and annotation should be focused on improving TE boundary designation, reducing identification error rates, and providing accurate information on the type and integrity of TE insertions. Here, we offer practical advice for generating TE models in de novo assemblies for non-model organisms, provide step-by-step instructions to guide inexperienced TE annotators through some of the commonly utilized TE analysis pipelines, and entertain suggestions for tool improvement which could be implemented by interested developers.

Key words Retrotransposons, DNA transposons, Consensus sequences, Repetitive DNA, De novo repeat identification, Repeat library, Manual curation

1 Introduction

The transitions from first- to second- to third-generation sequencing technologies resulted in whole-genome sequencing (WGS) descending from megaproject levels down to routine lab work carried out as the necessary first step in exploration of biological properties for any given species. However, in contrast to the progress in gene annotation achieved and standardized over the past decade, a much more inferior quality of annotation for other genome components precludes unimpeded understanding of genomic regions that do not constitute part of the host proteome in its

Miguel R. Branco and Alexandre de Mendoza Soler (eds.), *Transposable Elements: Methods and Protocols*,
Methods in Molecular Biology, vol. 2607, https://doi.org/10.1007/978-1-0716-2883-6_1,

traditional sense but nevertheless may comprise the bulk of genomic DNA and yield abundant proteins readily detectable in proteomics experiments. Repetitive regions of the genome, of which transposable elements (TEs) comprise a major fraction, represent a rich source of structural variation contributing to differences between cells, tissues, individuals, populations, species, and higher taxonomic units, which make up the tree of life.

Gene annotation typically provides a breakdown of a transcriptional unit into features such as CDS (exons), introns, 5' and 3' UTRs, and various combinations of these features into alternative isoforms. Unfortunately, this is not the case with TE annotations, which typically do not assess the completeness of their ORFs and often do not show actual TE-host boundaries. It is not customary to annotate TE-encoded proteins, as they are not considered an "official" part of the host proteome, and their ORFs generally decay faster than host protein-coding ORFs, due to reduced selection pressures. Despite this, TE-encoded ORFs and their fragments continue to flood proteome annotations, including not only nr_protein but also the allegedly curated RefSeq database, due to poor quality of repeat-masking and the inability of protein-predicting pipelines to distinguish between gene-derived and TE-derived ORFs, especially if these are fragmentary or of unknown nature. With further improvements in assembly quality rising to the chromosome scale, it is imperative to improve the standards for TE detection and annotation, from simple designation of a sequence as repetitive to a comprehensive annotation which defines precise host-TE boundaries, marks TE transcripts including UTRs in active TEs, and highlights TE coding capacity or lack thereof. Notably, de novo TE identification remains critical even for closely related species or different morphospecies of a species complex, if their nucleotide sequence divergence exceeds a few per cent. On top of that, horizontal TE transfer can affect even the most closely related species.

No single algorithm is sufficient to achieve comprehensive TE identification, and therefore the most popular tools represent a combination of several computational strategies, which take advantage of different TE characteristics to complement each other's deficiencies. Our goal here is to summarize the user's practical tips, with the ultimate focus on the quality of the end results, and to convey the need to correct the most frequent errors observed in the output of the popular pipelines. Individual labs publishing their own genome projects are too often tempted to take these outputs at face value, relying on software reputability and skipping "sanity checks." However, software deficiencies should not be used as an excuse for failure to inspect program outputs, which ultimately results in poor annotation quality for genes as well as TEs.

Different workflows have been proposed for the de novo identification of TEs in sequenced genomes. Most are based on identification of genomic repeats or characteristic structures shared by a group of TEs and involve subsequent classification into DNA or RNA TE classes, with further breakdown into orders and superfamilies. The best-known model organisms already have high-quality curated TE libraries, which can be used for TE annotation in newly sequenced strains or cultivars through automated TE analysis. However, detection and identification of TEs in newly sequenced species is still a challenging and time-consuming task.

Several tools are available for de novo detection of repeat content and TE insertions in understudied genomes lacking a previously defined TE library of consensus sequences. Although there is no single pipeline or program that can be regarded as a universal standard, the packages REPET [1] and RepeatModeler2 [2] are among the most popular. These de novo pipelines employ a similar strategy for TE detection, relying on a multicopy nature and/or structural characteristics of repeated elements for TE identification. Each de novo detection process features a configurable flow diagram (system of programs/scripts) with a set of parameters that can be adjusted (clustering process, homology search engine etc.). Each of these pipelines outputs TE consensus sequences, which however would differ for a given genome if different tools are applied. Success can depend on many factors for each assembly, such as genome size, total repeat content, and/or assembly fragmentation. Although there are no universal metrics comparing outputs from each program, recent studies began to systematize and apply such metrics for software evaluation [3, 4]. Generally, implementation of several methodologies to search for distinct TE families is the best approach for obtaining comprehensive representation of TE diversity in each genome. Other programs use k-mer counts to identify genomic repeats, e.g [5]; these programs are much faster and require less computational power, but are memory-consuming, can report TE coordinates only, and are not described here, although boundary refinement began to be implemented for some TE types [4]. Table 1 outlines the components of the most widely used packages for TE identification/classification, which yield output TE sequences, and lists some of their strengths and downsides.

2 Materials

2.1 Databases

Successful TE annotation relies on the use of curated repeat databases for TE classification and identification of divergent TEs with broader taxon specificity. Table 2 lists the most widely used repeat databases which are systematically updated. *See* **Notes 1–3** for more detailed background information on each database. Note that the composition of each database is determined by research interests of database keepers and external submitters, the amount and quality

Table 1
Overview of representative software packages for de novo TE detection and classification

De novo pipelines	Search/align algorithms	Built-in classifiers	Notes
RepeatModeler2 [2]	RepeatScout [19] RECON [12] LTRharvest [25] LTR_retriever [20]	RepeatClassifier	Standard tool for mammals and most vertebrates Need to detect and add MITEs separately Can misclassify ERVs and SINEs in invertebrates LTR modules are additional Frequent N's in consensi
REPET [1]	BLASTER [11] RECON [12] GROUPER [11] PILER [13] LTRharvest [25] MAP [14]	PASTEC [16] based on Wicker classification system [17]	Good for plants, invertebrates and fungi Confidently detects MITEs Most LARD and TRIM are false positives, i.e. segmental duplications Multicopy host genes are abundant, need Uniprot scan for cleanup
EDTA [3]	LTRharvest [25] LTR_FINDER_parallel [26] LTR_retriever [20] GRF [27] TIR-Learner [28] HelitronScanner [29]	Tutorial in [30]	Extensively benchmarked Metrics include FDR, sensitivity, specificity, accuracy, precision Deconvolutes nested TEs Option to add RepeatModeler (Smit 2008)
FasTE [31]	EDTA [3] Tutorial in [30]	DeepTE [32] with user manual in the supplement	DeepTE uses machine learning and hmmscan to classify TEs. Number of superfamilies is limited by PFAM domain choices
DARTS [33]	RPS-BLAST MMseqs2	No classifier	Search only with LTR-RT ORF CDD and PFAM profiles Should be expanded to include CD/PFAM for each TE type

of resources available for submission and curation, and the user-friendliness of the submission interface. Figure 1 illustrates the breakdown of each database into repeat classes and broad taxonomic categories. It shows that the representativity of Dfam is optimized for vertebrate and especially mammalian genome research, while RepetDB can be recommended for plant and fungal genomes and Repbase for invertebrates. Additionally, users can choose more specialized databases covering specific taxonomic groups or TE types. The collaborative platform TE Hub [6] contains an extended list of repeat databases, with target taxonomic groups and repeat types (https://tehub.org/en/resources/repeat_databases).

Table 2
Characteristics of the most widely used repeat databases (as of February 2022)

Database	URL	Data accessibility	Organisms	Repeat Types	No. of elements	version	Reference
Repbase	http://www.girinst.org/repbase/	Subscription	Eukaryotes (1300 spp.)	Eukaryotic transposon	69,736	27.01	[8]
RepetDB	http://urgi.versailles.inra.fr/Data/Transposable-elements/REPETDB	Open access	Plants, fungi (54 spp.)	Eukaryotic transposon	101,963	2022	[9]
Dfam	http://www.dfam.org/	Open access	Eukaryotes (595 spp.)	Eukaryotic repeat	285,542[a]	3.5	[10]

[a]15,444 curated families

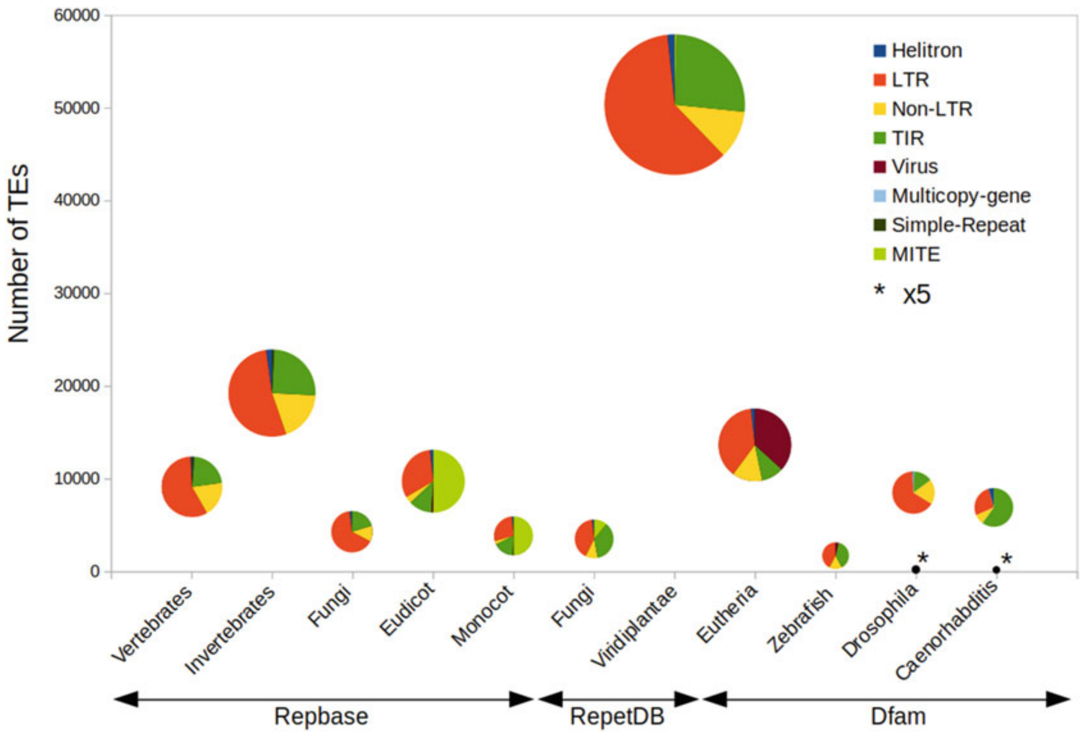

Fig. 1 TE class abundance and taxonomic diversity in selected eukaryotic repeat databases. Databases include Repbase, RepetDB and Dfam. The area of each pie chart representing a taxonomic group in each database is proportional to its TE representation. Datasets and metadata were extracted from each database as of 02/22/2022, not including unambiguous and unknown class annotations. In Dfam, only curated families are presented. Due to the small dataset size in *Drosophila* and *Caenorhabditis* (195 and 155, respectively), a 5× size chart is shown (*)

Table 3
List of bioinformatic tools required for de novo TE detection, as described in Methods

Name	URL	TE analysis	Reference
REPET	http://urgi.versailles.inra.fr/Tools/REPET	De novo detection; homology; structure; annotation	[1]
RepeatMasker	http://repeatmasker.org	Repeat masking; annotation; visualization	[18]
RepeatModeler2	https://github.com/Dfam-consortium/RepeatModeler	De novo detection; classification; structure; seed alignment	[2]
RepeatScout	http://bix.ucsd.edu/repeatscout/	De novo detection	[19]
Censor	https://www.girinst.org/downloads/software/censor/	Repeat masking; annotation	[21]
PASTEClassifier	https://urgi.versailles.inra.fr/Tools/PASTEClassifier	Classification	[16]
LTRharvest	http://genometools.org/pub/	LTR-RT structure	[25]
LTR_retriever	https://github.com/oushujun/LTR_retriever	LTR-RT structure, cleanup	[20]
One_code_to_find_them_all	http://doua.prabi.fr/software/one-code-to-find-them-all	Repeat re-annotation	[7]

2.2 Software

A summary of the principal software packages for de novo TE identification is provided in Table 3. *See* **Notes 4–7** for background information on each package.

2.3 REPET Package Dependencies

REPET is available through a containerized version, but if that fails these software tools should be locally installed to be able to run the REPET pipeline.

1. Blast from NCBI (ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/) (*see* **Note 9**).
2. AB-Blast (new name for WU-Blast, http://www.advbiocomp.com/blast.html).
3. Censor (https://www.girinst.org/downloads/software/censor/).
4. CrossMatch (http://phrap.org/consed/consed.html) (for RepeatMasker).
5. RMBlast (repeatmasker.org/RMBlast.html) (for RepeatMasker).
6. Mreps (http://mreps.univ-mlv.fr/howto.html).
7. Piler (http://drive5.com/piler/).

8. Recon (http://www.repeatmasker.org/RepeatModeler/RECON-1.08.tar.gz).
9. RepeatModeler2 (http://repeatmasker.org/RepeatModeler)
10. ESL-Shuffle (http://hmmer.org/download.html). This utility is part of the HMMER package.
11. TRF (https://tandem.bu.edu/trf/trf.download.html).
12. RepeatScout (http://bix.ucsd.edu/repeatscout/).
13. RepeatMasker (http://repeatmasker.org/RMDownload.html).
14. Repbase (https://www.girinst.org/repbase/). Repbase sequences (REPET edition) need to be placed or symlink created (ln -s) in the top-level project directory.
15. LTRharvest from GenomeTools (http://genometools.org/pub/).

3 Methods

Although the most widely used packages such as RepeatModeler/RepeatMasker and REPET can be utilized in their entirety from de novo library construction to TE classification and genome annotation, different components of these packages can also be combined for these purposes, taking advantage of the components from different packages and minimizing the disadvantages to achieve optimal performance. While researchers working on mammalian or plant genomes should adhere to the dominating pipelines used in model organisms, we found it most helpful to combine parts of the two packages by evaluating library completeness and repeat classification in non-model invertebrates through manual curation.

3.1 TEdenovo Detection

In this section, we present in a simplified form the main steps for TE*denovo*, the first part of the REPET package [1], which we found most efficient in constructing TE libraries for non-model invertebrate genomes, despite many false positives. A comprehensive REPET tutorial with in-depth explanation of all parameters and options is found at https://urgi.versailles.inra.fr/Tools/REPET/TEdenovo-tuto.

REPET TE*denovo* uses a multistep approach for TE detection to produce a classified repeat library.

One of the drawbacks of using REPET is the number of dependencies and the cohesive programmatic environment it needs to properly run the pipeline (python, python modules, MySQL database, ncbi-blast and /or ncbi-blast+, and other external scripts). For large genomes, the use of the batch-queuing system of a local cluster is highly recommended to lower the processing time, and processing of giga-genomes may encounter run-time and memory issues (*see* also **Note 8**).

For initial TE*denovo* identification, the simplest approach is the use of the containerized REPET package and all its dependencies from a Docker image that can be downloaded and run locally (https://hub.docker.com/r/urgi/docker_vre_aio). Alternatively, standard installation will need a system environment with other bioinformatic tools that do not come with REPET package (*see* Subheading 2.3).

Once the installation is complete, it is necessary to setup a working environment with a few variables to define the path to the directory where REPET has been installed ($REPET_PATH):

```
export REPET_PATH=$HOME/bin/REPET/
export PATH=$REPET_PATH/bin:...:$PATH
```

After the environment is configured, REPET needs a project directory and a project name. The input sequence (assembly as a fasta file) should be included within the project folder and named as <project_name>.fa.

Fasta files have several requirements: 15-character limit on the name of the file (and the project name); make sure there is a line break at least once every 60 bases and no illegal characters in the fasta headers. To satisfy these criteria, three code lines can be useful:

1. Line break with SeqKit (https://bioinf.shenwei.me/seqkit/).

```
seqkit seq -w 60 Genome.fasta > Genome.fna
```

2. Rename fasta headers.

```
awk '/^>/{print ">Sp_ctg" ++i; next}{print}' < Genome.fna >
Genome.fst
```

3. Convert to all-capital letters.

```
awk 'BEGIN{FS=" "}{if(!/>/){print toupper($0)}else{print
$1}}' Genome.fst > Genome.fa
```

The configuration file needs to be edited to accommodate individual settings. Two main areas need to be updated. One is [repet_env], for which the user needs to set MySQL database configuration. Another is [project], which needs the name of the actual project (i.e., the name of the fasta file) plus the complete path to the project directory (fasta file location).

These settings will suffice for execution of the REPET pipeline, which is a multistep process totaling eight steps. Each step can be run after the previous step is fully completed, or the steps can be combined in a bash script where we need to provide a project name (export PROJECT_NAME=<project_name>), and run each step using ${PROJECT_NAME} (*see* **Note 10**).

```
#eg.
export PROJECT_NAME=DmelChr4

# Step 1
rm -rf ${PROJECT_NAME}_db/
TEdenovo.py -P ${PROJECT_NAME} -C TEdenovo.cfg -S 1

# Step 2
rm -rf ${PROJECT_NAME}_Blaster/
TEdenovo.py -P ${PROJECT_NAME} -C TEdenovo.cfg -S 2 -s Blaster

# Add the rest of steps needed for the analysis. An example of
configuration file (Tedenovo.cfg) and a custom script using
all the steps are available through GitHub:
https://github.com/cascoamarillo/REPET_TEdenovo
```

After completing TE*denovo* steps, which can take up to several days depending on the genome size/repeat content and the level of multi-threading, the output directory will contain a set of folders with results produced by each process. The library of classified, nonredundant consensus TE sequences would be in the output folder:

```
${PROJECT_NAME}_Blaster_GrpRecPil_Struct_Map_TEclassif_Fil-
tered_MCL
```

and the fasta file with TE consensus sequences in:

```
${PROJECT_NAME}_denovoLibTEs_filtered_MCL.fa
```

The initial consensus sequences derived from any automatic TE identification software will represent the putative TE families present in a genome. However, producing a library of high-quality full-length consensus sequences inevitably requires some degree of manual curation. *See* **Note 11**.

3.2 TE Classification

After the de novo TE detection step, the next steps involving the output consensus sequences are similar in execution to outputs of other programs (e.g., RepeatModeler2 or EDTA). At this point, the output is a multi-fasta file with fasta headers (">") containing specific information about each TE. REPET classifies the consensus according to their features detected at Step 5 (consensus feature detection). Classification is performed by the PASTEClassifier (or PASTEC, included in the REPET package), which classifies consensus sequences in several groups using Wicker's classification (*see* **Note 4**). If PASTEC doesn't find matching features, the consensus is unclassified (using the term "NoCat"). Potential chimeras are marked by "-chim_" in the header. The terminology used at this classification step (and provided in the fasta header for each TE) differs from classification and terminology provided by other databases/classifiers which basically use the Repbase system. A table with REPET classification codes and transposon names can be found in https://tehub.org/classification/wicker. RepeatMasker and RepeatModeler can accept custom libraries for further processing, but the recommended format for ID is ">repeatname#class/subclass," so that each TE can be computed in each category for further analysis (tables, proportions, genome annotations.).

The classification tool RepeatClassifier from RepeatModeler2 (*see* **Note 6**) can be run independently on a given TE consensus library and classify each repeat based on a homology module which compares de novo TE consensi to both RepeatMasker Repeat Protein Database and RepeatMasker libraries (Repbase and/or Dfam). First, using TRF and RepeatMasker, it will try to identify simple-repetitive DNA (low complexity and tandem repeats) that might be included in the TE consensus file from the de novo step. Next, it compares against TE proteins using blastx algorithms (ncbi blastx or AB-Blastx/WU-Blastx). The template used as protein reference can be found in $REPEATMASKER_DIR/Libraries/RepeatPeps.lib. Finally, it will use blastn (ncbi RMBlastn or AB-Blastn/WU-Blastn) for comparison to the nucleotide database used by RepeatMasker ($REPEATMASKER_DIR/Libraries/RepeatMasker.lib). RepeatClassifier will output the fasta consensus file (*.fa.classified) with the addition of #class/subclass to each one of the existing fasta headers. If it does not find a matching feature, the repeat family will be labeled as "#Unknown." *See* also **Note 12**.

```
# Run RepeatClassifier to generate a .classified file
RepeatModeler2 /RepeatClassifier -consensi TE_consensus.fa
```

The use of the combined fasta header containing both REPET and RM categories is quite helpful in settling TE classifications. If both categories agree, the classification stands. However, in case of disagreement, inspection of the entry is recommended, as it is highly likely that the consensus is chimeric and/or corresponds to a non-TE sequence. *See* **Note 13** for consensus inspection and manual curation tips.

3.3 TE Annotation

This step involves annotation of TE copies back onto the genome assembly, using a custom library built in the previous section (TE*denovo*), with some degree of curation. Using the consensus sequences and the homology search engine, TE locations will be annotated in the genome, including TE class/family origin.

In this section, we describe the use of RepeatMasker (*See* **Note 5**), a free tool widely used for TE identification in genomes, which masks repetitive sequences, including low-complexity sequences and interspersed repeats. RepeatMasker can search for repeats in a query sequence using TE sequences from a repeat library (Dfam, Repbase, or custom built).

RepeatMasker installation will need a simple Unix/Linux environment with PERL (>v5.8.0) and the binary "trf" (Tandem Repeat Finder in http://tandem.bu.edu/trf/trf.html) copied in its directory. At least one of the following search engines needs to be installed:

1. AB-Blast (new name for WU-Blast, http://www.advbiocomp.com/blast.html).
2. Cross_match (http://www.phrap.org/phredphrapconsed.html).
3. Decypher (http://www.timelogic.com/decypherblast.html).
4. RMBlast (http://www.repeatmasker.org/RMBlast.html).
5. nhmmer is a part of HMMER release (http://hmmer.org).

The first four engines use a core BLAST algorithm in their methodology. Either of them can be used when a (multi-)fasta input is provided as a repeat library. After installation, the user needs to configure RepeatMasker before its test run, by manually editing the configuration file of RepeatMasker (RepeatMaskerConfig.pm) or using the PERL script "configure" in the directory and following the instructions.

Once all dependencies are settled, RepeatMasker can be run in parallel using different search engines, to evaluate which performs best, e.g., in terms of sensitivity and processing times. Use the "-h (elp)" option to get a detailed set of options in the RepeatMasker command line, or open the file /RepeatMasker/repeatmasker.help with a text editor. A general set of parameters to annotate repeats in a eukaryotic genome can be:

```
RepeatMasker -e ncbi -lib TE-library.fa -s -nolow -no_is
-gccalc -cutoff 200
```

The option "-lib" allows the use of a custom library, which is produced through de novo repeat identification and curation. Sequences must be in (multi-)fasta format. It can contain information about the repeat class/subclass, by adding

“>repeatname#class/subclass” after the repeat name (*see* Subheading 3.2) or simply >repeatname#class. The initial TE*denovo* library can also be combined with the appropriate repeat library subset (e.g., plants, invertebrates, vertebrates) from other databases (e.g. Repbase) or with a pre-existing TE library from a closely *related* species, to create an expanded set of consensus sequences. However, use of the entire Repbase instead of a subset would yield a high fraction of false positives.

By default, RepeatMasker will return three output files: a masked sequence (genome) file, an annotation file, and an overview table.

1. The masked sequence (.masked) will contain the genome or query sequence(s) with all identified repeats (transposons and low complexity sequences) masked (i.e., replaced by “N” in the sequences).
2. The annotation or map file (.out) consists of a list of all repeats identified in the genome. The file lists the matches detected by the search engine (under the parameters used) between the genome and the library, with each line representing a match (TE fragment).
3. Table file (.tbl) is a plain text format, which summarizes the number of identified repeats grouped into TE superfamilies, nucleotide length occupancy, and percent genome occupied by each superfamily. To get a complete summary report using a custom library (TE*denovo*), repeats should be classified according to RepeatMasker standards (“>repeatname#class/subclass”). Otherwise, repeats will appear as “unclassified” in the summary file, although they will be masked and listed in the other two files (.masked and .out).

   In addition to the three standard output files, other output options in different formats can be obtained for downstream applications:

```
-a(lignments)
```

4. This option writes the alignments in the supplementary .align output file. Basically, it stores all alignments (repeats vs. genome) from the search engine in a cross_match format. This format is useful for posterior processing steps, such as repeat landscape plotting.

```
-gff
```

5. This option will create an additional general feature format (. gff) output (https://m.ensembl.org/info/website/upload/gff.html), a simple tab-delimited text file for describing genomic features. GFF files can be used for genome annotation and to represent custom tracks in genome browser visualizations. *See* also **Note 14**.

3.4 Post-processing for RepeatMasker Outputs

After consensus sequences have been mapped onto the genome, TE copies can still become split into several fragments. A full TE annotation would need to evaluate all copies, truncated or not, by putting together the identified hits which belong to the same copy from a given genomic location. This can happen in case of multiple insertions/deletions in the sequence or when the library contains distinct consensus sequences which actually belong to the same full-length TE (i.e., LTR-retrotransposons). Bailly-Bechet et al. [7] have developed a Perl tool (available in http://doua.prabi.fr/software/one-code-to-find-them-all) that parses the RepeatMasker .out file and determines a more accurate number of TE copies and their location. Its implementation can be particularly useful for LTR-retrotransposons, with multiple hits from the split of the consensus into the internal portion of the element and the LTR (long terminal repeat), as is typical for Repbase entries. The first step lists all LTR-retrotransposons found by RepeatMasker and associates the hits corresponding to the internal and LTR portions. The script build_dictionary.pl will build the list from the .out file, matching both portions (internal and LTR), based on name similarity:

```
./build_dictionary.pl --rm RepeatMasker.fa.out > LTR-dictio-
nary_output.txt
```

The second script (one_code_to_find_them_all.pl) takes the LTR dictionary produced by build_dictionary.pl and the RepeatMasker .out file and compares the positions and orientation of each hit from the same TE family. Briefly, this will determine if two hits, located on the same chromosome/scaffold, can be considered fragments from the same copy based on orientation, distance and overlap between them.

```
./one_code_to_find_them_all.pl --rm RepeatMasker.fa.out --ltr
LTR-dictionary_output.txt
```

In the end, the script generates an "updated" summary file (.copynumber.csv) with information on the number of copies/fragments, total base pairs, and coverage for each TE family. Additionally, the program reports all cases of LTR-retrotransposons, non-LTR retrotransposons, and DNA transposons identified and computed during the process (.ltr.csv and .transposons.csv files).

3.5 Repeat Divergence Landscapes

One of the most useful graphical representations in RepeatMasker is the repeat landscape or TE divergence plot. Repeat landscapes depict the relative abundance of repeat classes in the genome versus the Kimura divergence from the consensus, serving as a proxy for divergence time. Once the library is classified (*see* Subheading 3.2), an interspersed repeat landscape is produced using the RepeatMasker output (.align) after converting the alignments into a coverage and divergence plot file (.divsum). The scripts calcDivergenceFromAlign.pl and createRepeatLandscape.pl are found in RepeatMasker directory /utils.

It is necessary to setup these tools in your environment:

1. RepeatClassifier from RepeatModeler (https://repeatmasker.org/RepeatModeler)
2. RepeatMasker (https://repeatmasker.org/RMDownload.html)
3. TwoBit (https://hgdownload.soe.ucsc.edu/admin/exe)

Below is a detailed simple step by step guide.

1. You can modify these variables and run it as bash script.

```
export genome="Genome"
export TElib="Genome_denovoLibTEs_filtered_MCL.fa"
```

2. Run RepeatClassifier to generate a .classified file.

```
/RepeatModeler/RepeatClassifier -consensi ${TElib}
```

3. Create .2bit file required for createRepeatLandscape.pl.

```
faToTwoBit ${genome}.fa ${genome}.2bit
twoBitInfo ${genome}.2bit stdout | sort -k2rn > ${genome}.
chrom.sizes
```

4. Run RepeatMasker: with option -a for alignment output. The option creates an additional file (.align) with TE-genome alignments in cross_match format. *See* also **Note 15**.

```
RepeatMasker -a -nolow -no_is -e ncbi -lib ${TElib}.classified
${genome}.fa
```

5. Run script calcDivergenceFromAlign.pl using output file (.align) from RepeatMasker.

```
perl /RepeatMasker/util/calcDivergenceFromAlign.pl -s ${gen-
ome}.divsum ${genome}.fa.align
```

6. Landscape plot is generated using the output file ${genome}.divsum which contains the Kimura divergence table.

```
perl /RepeatMasker/createRepeatLandscape.pl -div ${genome}.
divsum -twoBit ${genome}.2bit > ${genome}.html
```

The output is an interactive graphic .html file (to be opened with a web browser), which summarizes the content of repeats in the genome: a stacked histogram with % divergence numbers and a colored pie chart with TE genomic proportions. Ultimately, a user-preferred custom visualization setup can be done with R (ggplot2) using the results from createRepeatLandscape.pl and the histogram information at the bottom of the output file.

Occasionally, users may be interested in a particular type of repeat that does not correspond to the standard classes (SINE, LINE, LTR, DNA, satellite) and subclasses within RepeatMasker. In this case, the standard processing format in RepeatMasker can be used to plot repeats with specific name as class in a final repeat landscape. To perform this, format each ID header (>repeatname#custom-class) according to a custom class/subclass schema. Although RepeatMasker table (.tbl) would not output their genome coverage values, they are listed and annotated in the .out and .align files. Before creating the plot with createRepeatLandscape.pl, the script needs to be edited to track the groups from the custom library, by renaming the variables in "my $graphLabels" with the re-branded classes and the html code color of choice.

4 Notes

1. **Repbase** [8] is a database of consensus sequences for eukaryotic TEs. It also contains simple repeat sequences, including satellite/microsatellite sequences, as well as multicopy genes (rRNA, tRNA, snRNA) and integrated viruses of eukaryotes. Founded by Jerzy Jurka in the early 1990s, it has been continuously updated with curated TE family consensus sequences from newly sequenced eukaryotic genomes. Consensus sequences are reconstructed for each TE family in newly assembled genomes and can be used for identification of similar TEs using a homology search engine (RepeatMasker, Censor) and

for classification of de novo generated repeat libraries. According to [8], over 70% of database entries represent complete consensus sequences. Repbase fasta-formatted sequences can be directly used to detect repeats in the genome of interest. TE library used by RepeatMasker or REPET is similar to Repbase-provided, except for some formatting and additional annotation-supportive files. Repbase is maintained by the Genetic Information Research Institute (https://www.girinst.org/). Although multiple consensus TE sequences have been provided by researchers around the globe, Repbase access to those sequences since April 2019 is only available under a private/institutional subscription system, effectively discouraging outside submissions, as researchers would be expected to donate sequences to be kept behind the paywall. This results in skewed representation of certain TE groups.

2. **RepetDB** [9] is a database of TE reference sequences, mostly composed of transposons from plants and fungi as major taxonomic groups. It contains 101,963 consensus sequences from 54 genomes (version 2022) which were detected, classified, and annotated using the REPET framework from WGS projects or comparative genome analyses and is meant to be used for TE annotation in other genome projects. Additional TEs of interest can be inserted into RepetDB via TE*denovo* and TEannot from the REPET package [1]. Keep in mind that TE orders under-represented in higher plants are virtually missing, so that its use cannot be recommended for animal species.
3. **Dfam** [10] is an open database for DNA repeat families based on hidden Markov model (HMM) profiles constructed from multiple sequence alignments (seed alignments), each containing a set of representative members of TE family, as well as the consensus sequences for each family. The initial entries in Dfam were represented by a Repbase-derived library with humans, mouse, zebrafish, fly, and nematodes as major species. Although Dfam incorporates, like other classification schemes, a hierarchical TE system, it does not display a fixed-rank hierarchy (class-order-superfamily), as the definitions are dynamic and can change for a specific class, order, family, or subfamily of TEs, with the growing diversity of TEs and organisms represented. Dfam is growing in numbers of annotation data and curated TE entries, with 285,542 TE lineage specific models across 595 species (Dfam 3.5).
4. **REPET** [1] is a software package combining different programs into two pipelines: TE*denovo* and TEannot. They use a set of algorithms for clustering of interspersed repeats, detecting and classifying TE families as a consensus sequence. Efficient TE detection in TE*denovo* is based on a 3-step approach: genome-wide self-alignments, clustering of sequences into TE

families, and determining consensus sequences from multiple alignments. TE*denovo* consists of programs BLASTER [11] for genome self-alignment: RECON [12], GROUPER [11] and PILER [13] for clustering; and MAP [14] or MAFFT [15] for multiple sequence alignment. First, the genome is used simultaneously as the subject and query for genome-wide alignments (all-by-all blast). Once the self-alignment matches in the genome are determined, a clustering process groups all sequences belonging to the same TE family. Each cluster yields a multiple sequence alignment and its consensus sequence. Transposon annotation (TEannot) pipeline performs identification and annotation of TE copies back onto the genome. REPET also includes the classification tool PASTEClassifier (Pseudo Agent System for Transposable Elements Classification) [16] searching for structural features and homology to classify TEs according to the Wicker system [17].

5. **RepeatMasker** program [18] screens DNA sequences for interspersed repeats and low-complexity DNA sequences. It was created in 1998 by A. Smit. The latest release is version 4.1.2 (March 2021) and is always under constant update and bug fix reporting. RepeatMasker, which needs a simple Unix/Linux environment with PERL (>v5.8.0), is one of the most frequently used tools for TE annotation. It can identify and mask repeat sequences in a genome using a search engine, including AB-Blast/WU-Blast, crossmatch, RMBlast, Decypher and nhmmer (https://www.repeatmasker.org/), and a TE library as a query. RepeatMasker has been shown to be very efficient and fast, even for large genomes. The drawback of the homology-based approach is that it can only detect sequences already present in a TE library, but cannot detect new transposons. Along with the computer version, a RepeatMasker webserver (http://repeatmasker.org/cgi-bin/WEBRepeatMasker) can screen DNA sequences (fasta format) against the Repbase-derived RepeatMasker library of repetitive elements or against the Dfam database (*see* Subheading 2.1). One of RepeatMasker output files (.out) can be parsed with the tool **One_code_to_find_them_all** [7] to reduce fragmentation of TE copies in the genome and compute quantitative information for TE families.

6. **RepeatModeler2** [2] is an automated TE*denovo*/classification pipeline that uses two algorithms for repeat discovery: **RepeatScout** [19] and **RECON** [12]. Both algorithms, with different sensitivity for repeats, are combined into RepeatModeler2, and followed by the creation of a repeat consensus library. RepeatModeler2 also incorporates an LTR module for structural LTR detection, implementing the algorithm from

LTR_retriever [20]. LTR_retriever, like other LTR detection tools, identifies LTR-retrotransposons based on structural features. However, unlike other tools, LTR_retriever can filter out false positive results. Although we are not describing the RepeatModeler pipeline for TE*denovo* analysis in this chapter, the package contains a homology-based classification module based on the Repbase classification scheme (RepeatClassifier) which can be used to compare a repeat library provided by any TE*denovo* software against a known repeat database (*see* Subheading 3.2).

7. **Censor** [21], like RepeatMasker, relies on homology-based methods, allowing identification and masking of repetitive sequences homologous to any particular repetitive element for use in downstream applications. CENSOR is available from the Genetic Information Research Institute (GIRI) for local installation (https://www.girinst.org/downloads/software/censor/). Censor can use WU-Blast (for speed and higher sensitivity) or NCBI BLAST as its search engine. It can conduct direct DNA-DNA searches, as well as any combination of protein-DNA or translated DNA searches, using the appropriate BLAST modules. Censor is also available as web-based service at https://www.girinst.org/censor/ if using a small sequence query (up to 2 Mb). However, it yields highly fragmented outputs with different parts of a single query matching different TE sequences. Censor has two scripts, one designed to run with the licensed version of WU-Blast from Washington University (before it was sold to Advanced Biocomputing, LLC) and censor.ncbi with modified settings to work with ncbi-blast. WU-Blast, now AB-Blast, is no longer available for free for academic users. If you do not have a fully licensed WU-Blast in your system, the only way to run censor is with censor.ncbi.
8. Some of the REPET processes implement multi-threading. It is best to have a workstation or PC with a few cores to allocate the run (at least four cores). It is important to edit configuration files to specify this number.
9. REPET v2.0 uses blast 2.2.26 (ftp://ftp.ncbi.nlm.nih.gov/blast/executables/legacy/LATEST) by default, although it can be set to use blast+ (ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/). In contrast, REPET v3.0 uses blast+, although blast legacy can also be used.
10. In the configuration file (TEdenovo.cfg), parameters can be edited to adapt the search to different genomes and workstations. Blast search engine can be chosen in [self_align] for legacy blast, blast+ or AB-Blast/WU-blast [blast: ncbi, blastplus or wu]. If the genome shows ploidy increase (three or

more chromosome sets, meaning three or more gene homolog sets), different numbers can be set in "minNbSeqPerGroup" (e.g., if the genome is tetraploid, use "minNbSeqPerGroup: 5" instead of "minNbSeqPerGroup: 3") in [cluster_HSPs] parameters to minimize host gene representation. This, however, would come at a cost of skipping TEs present in 3–4 copies. After using different values in minNbSeqPerGroup, one way to measure the potential contribution of host genes to TE*denovo* output is by comparing the proportion of Unknown TEs at the classification step (*see* Subheading 3.5).

11. For novices who are looking for a step-by-step manual curation protocol, we recommend a recent article by Goubert et al. [22]; those seeking more computationally advanced approaches can refer to guidelines in [23, 24].

12. RepeatClassifier runs a simple step for each ID, comparing to protein and nucleotide databases using blastx and rmblastn algorithms. Depending on the size of the library, it can run into memory issues (depending on the PC/workstation being used) and may take a long time to finish. To overcome this, it is recommended to split the input fasta file into smaller files and run multiple jobs independently (e.g., if a library is > 5 Mb in size) and subsequently merge all of them into one (*.fa.classified) for further analysis.

13. In addition to quality control toolkits suggested in **Note 11**, several useful QC procedures can be offered that use the already available stand-alone programs as well as web-based tools with graphical outputs, which can be used even by students with little prior experience. A suspected chimeric/indel-bearing TE consensus can be visually inspected using blast formatting option "-outfmt 1" or its equivalent "query-anchored with dots for identity" at https://blast.ncbi.nlm.nih.gov/Blast.cgi. A genuine TE consensus should not display major discontinuities along its entire sequence length, or at least should yield a prominent discontinuous subset (in case of *trans*-proliferation of incomplete structural variants). Mis-assemblies would show abrupt discontinuities, while secondary insertions often show discontinuities overlapping by a few base pairs, originating from a target-site duplication. Also, select element types such as PLEs can show inherent discontinuities, i.e., the automated consensus may represent different parts of the same element joined together in tandem or inverted orientation, which arise during insertion and would be flanked by a target site duplication in the genome. Such discontinuities should be resolved by splitting and assessing the completeness of each part, using the expectations of domain composition from Wicker classification.

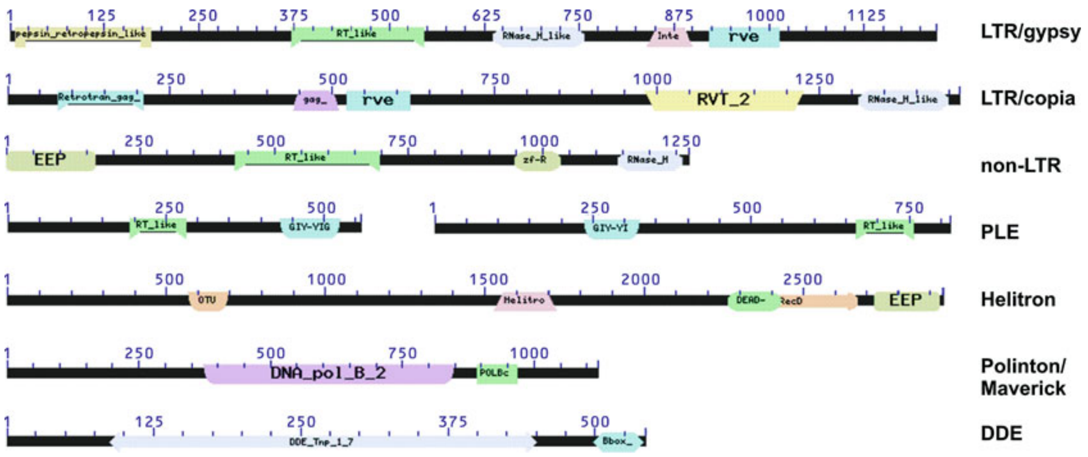

Fig. 2 Examples of characteristic domain architectures displayed by autonomous TE-encoded proteins/polyproteins. Domain architectures are represented in the graphical output of the NCBI Conserved Domain Architecture Retrieval Tool (CDART) after using TE-encoded ORF or TE DNA consensus as a query in a CD-search. For DNA TEs, only one of many types of DDE transposases is shown, in this case a piggyBac. Numbers correspond to amino acids, drawn at varying scales. Actual lengths can vary

Completeness of autonomous TEs is best assessed by evaluating their expected length (which should be within certain limits imposed by their coding capacity), as well as the domain composition via CD-search or HMM search, with detailed instructions provided in [22]. Graphical outputs provided by web interfaces can illustrate typical domain architectures characterizing each TE order. Figure 2 shows characteristic domain composition patterns for representatives of each autonomous TE order, obtained through the NCBI web interface (https://www.ncbi.nlm.nih.gov/Structure/cdd/wrpsb.cgi). Even though some domains can be optional, a TE cannot function as an autonomous unit if it is missing the most essential enzymatic core, such as a reverse transcriptase, integrase, transposase, or polymerase domain. If these domains are missing, perform an HMM search to ensure TE completeness.

14. For a previous RepeatMasker annotation run without providing the -gff option, the script in /RepeatMasker/util/rmOutToGFF3.pl can be used to convert the .out file into .gff (version 3).

15. Other options might be useful while running the RepeatMasker step. For plotting a repeat landscape of a de novo produced TE library, the option "-nolow" will not mask low complexity DNA or simple repeats. When dealing with curated eukaryote assemblies from which potential bacterial sequences have been removed, the check for bacterial insertion elements (option "-no_is") may be skipped.

16. **Future perspectives**. TEs exist as DNA sequences; however, their recognition and classification could be better achieved through assessment of their coding potential. Refinement of DNA sequences based on inferring a majority-rule consensus generally works well for DNA TEs harboring DDE-type transposases, which typically amplify from a single active copy entering the genome and yield essentially a star phylogeny with an intact transposase ORF of the progenitor element in the center. However, longer DNA TEs and most RNA TEs can often form distinct lineages within the same species, which show sufficient nucleotide sequence homology in their main ORF, but can diverge towards the less conserved ends, often yielding phylogenetically distinct subfamilies requiring separate extension of their noncoding regions. Also, DNA TEs and even some RNA TEs may be interrupted by introns. Subfamily divergence has been well studied in mammalian LINEs, but typically yields high degrees of fragmentation in de novo consensi for other non-LTR or PLE elements, when TE sequences adjacent to the consensus ORF are better extendable for one lineage than for another related lineage. Moreover, reconstitution of DNA sequences not guided by the underlying coding capacity often yields interrupted ORFs, which are more difficult to classify due to indels and frameshifts, as well as 5'-truncated consensus sequences for LINEs and PLEs missing functional domains located N-terminally to reverse transcriptase.

On this view, a promising direction for software development is scanning of the genome with profiles of TE-specific conserved protein domains, followed by extraction of homologous regions extended in both directions and by boundary definition for the multiply aligned regions. Protein-based profile searches are more sensitive than DNA-based, and prioritizing such profiles would facilitate detection of autonomous transposition-competent TE sequences, which may then be used to detect related nonautonomous elements. A step in this direction has recently been undertaken for LTR retrotransposons, which were retrieved with high sensitivity by the DARTS tool employing RPS-BLAST as a search engine and using a collection of 16 CDD and PFAM profiles corresponding to reverse transcriptase, RNase H, integrase, and protease domains of LTR-retrotransposons [33]. This approach can be expanded to other TE orders (DIRS, nLTR, PLE, DDE, Helitron, Polinton/Maverick), and a stepwise library buildup based on carefully curated sets of TE-specific domain profiles for each order would simultaneously facilitate TE classification.

The most recent comprehensive overview of the advantages and disadvantages of different approaches to de novo discovery of TEs can be found in [34]. Overall, there is much potential for development of newer and faster TE annotation tools, which would eventually bring the standards for TE annotation closer to those required of genes, so that it would be automatically provided along with gene annotations for each new genome.

Acknowledgments

Work in the laboratory is supported by grant R01GM111917 from the US National Institutes of Health to I.A.

References

1. Flutre T, Duprat E, Feuillet C, Quesneville H (2011) Considering transposable element diversification in de novo annotation approaches. PLoS One 6:e16526
2. Flynn JM et al (2020) RepeatModeler2 for automated genomic discovery of transposable element families. Proc Natl Acad Sci U S A 117:9451–9457
3. Ou S et al (2019) Benchmarking transposable element annotation methods for creation of a streamlined, comprehensive pipeline. Genome Biol 20:275
4. Valencia JD, Girgis HZ (2019) LtrDetector: a tool-suite for detecting long terminal repeat retrotransposons de-novo. BMC Genomics 20:450
5. Girgis HZ (2015) Red: an intelligent, rapid, accurate tool for detecting repeats de-novo on the genomic scale. BMC Bioinform 16:227
6. Elliott TA et al (2021) TE Hub: a community-oriented space for sharing and connecting tools, data, resources, and methods for transposable element annotation. Mob DNA 12:16
7. Bailly-Bechet M, Haudry A, Lerat E (2014) "One code to find them all": a perl tool to conveniently parse RepeatMasker output files. Mob DNA 5:13
8. Bao W, Kojima KK, Kohany O (2015) Repbase update, a database of repetitive elements in eukaryotic genomes. Mob DNA 6:11
9. Amselem J et al (2019) RepetDB: a unified resource for transposable element references. Mob DNA 10:6
10. Storer J, Hubley R, Rosen J, Wheeler TJ, Smit AF (2021) The Dfam community resource of transposable element families, sequence models, and genome annotations. Mob DNA 12:2
11. Quesneville H, Nouaud D, Anxolabéhère D (2003) Detection of new transposable element families in Drosophila melanogaster and Anopheles gambiae genomes. J Mol Evol 57 (Suppl 1):S50–S59
12. Bao Z, Eddy SR (2002) Automated de novo identification of repeat sequence families in sequenced genomes. Genome Res 12:1269–1276
13. Edgar RC, Myers EW (2005) PILER: identification and classification of genomic repeats. Bioinformatics 21(Suppl 1):i152–i158
14. Huang X (1994) On global sequence alignment. Comput Appl Biosci 10:227–235
15. Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30:772–780
16. Hoede C et al (2014) PASTEC: an automatic transposable element classification tool. PLoS One 9:e91929
17. Wicker T et al (2007) A unified classification system for eukaryotic transposable elements. Nat Rev Genet 8:973–982
18. Smit AFA, Hubley R, Green P (2015) Repeat-Masker Open-4.0. 2013–2015 http://www.repeatmasker.org
19. Price AL, Jones NC, Pevzner PA (2005) De novo identification of repeat families in large genomes. Bioinformatics 21(Suppl 1):i351–i358
20. Ou S, Jiang N (2018) LTR_retriever: a highly accurate and sensitive program for identification of long terminal repeat retrotransposons. Plant Physiol 176:1410–1422
21. Kohany O, Gentles AJ, Hankus L, Jurka J (2006) Annotation, submission and screening of repetitive elements in Repbase: RepbaseSubmitter and Censor. BMC Bioinform 7:474
22. Goubert C et al (2022) A beginner's guide to manual curation of transposable elements. Mob DNA 13:7
23. Storer JM, Hubley R, Rosen J, Smit AFA (2021) Curation guidelines for de novo

generated transposable element families. Curr Prot 1:e154

24. Carey KM et al (2021) PolyA: a tool for adjudicating competing annotations of biological sequences. bioRxiv:2021.2002.2013.430877
25. Ellinghaus D, Kurtz S, Willhoeft U (2008) LTRharvest, an efficient and flexible software for de novo detection of LTR retrotransposons. BMC Bioinformatics 9:18
26. Ou S, Jiang N (2019) LTR_FINDER_parallel: parallelization of LTR_FINDER enabling rapid identification of long terminal repeat retrotransposons. Mob DNA 10:48
27. Shi J, Liang C (2019) Generic Repeat Finder: a high-sensitivity tool for genome-wide de novo repeat detection. Plant Physiol 180:1803–1815
28. Su W, Gu X, Peterson T (2019) TIR-Learner, a new ensemble method for TIR transposable element annotation, provides evidence for abundant new transposable elements in the maize genome. Mol Plant 12:447–460
29. Xiong W, He L, Lai J, Dooner HK, Du C (2014) HelitronScanner uncovers a large overlooked cache of Helitron transposons in many plant genomes. Proc Natl Acad Sci U S A 111:10263–10268
30. Su W, Ou S, Hufford MB, Peterson T (2021) A tutorial of EDTA: extensive De Novo TE annotator. Methods Mol Biol 2250:55–67
31. Bell EA et al (2022) Transposable element annotation in non-model species: the benefits of species-specific repeat libraries using semi-automated EDTA and DeepTE de novo pipelines. Mol Ecol Resour 22:823–833
32. Yan H, Bombarely A, Li S (2020) DeepTE: a computational method for de novo classification of transposons with convolutional neural network. Bioinformatics 36:4269–4275
33. Biryukov M, Ustyantsev K (2021) DARTS: an algorithm for domain-associated RetroTransposon search in genome assemblies. Genes (Basel) 13:9
34. Storer J, Hubley R, Rosen J, Smit AFA (2022) Methodologies for the *de novo* discovery of transposable element families. Genes (Basel) 13:709