# **TEtrimmer Manual**

Last updated: June 27, 2024 Author: Hang Xue; Jiangzhao Qian

### **TEtrimmer Github repository**

- 1. Getting Started
  - 1.1 Installation
  - 1.2 Test run
  - 1.3 Understand output results
- 2. Obtain input files
  - 2.1 Genome file
  - 2.2 TE consensus library
- 3. Run TEtrimmer
  - 3.1 Hardware requirement
  - 3.2 Basic run examples
  - 3.3 Basic options
  - 3.4 Advanced options
- 4. Understand output files
- 5. Additional manual curation (optional; video coming soon)
- 6. FAQs
- 7. Reference

# Getting started

#### 1.1 Installation

```
Install TEtrimmer dependencies.

We highly recommend installation with `mamba`, as it is much faster.
```

```
conda create --name TEtrimmer

conda install -c conda-forge mamba

conda activate TEtrimmer

mamba install qianjiangzhao::tetrimmer

TEtrimmer --help
```

```
Or you can install by the provided .yml file (Only for Linux).
```

```
conda install -c conda-forge mamba
git clone https://github.com/qjiangzhao/TEtrimmer.git
mamba env create -f [path to/TEtrimmer env for linux.yml]
python <path to TEtrimmer>/TEtrimmer.py --help
```

We are working on uploading the package to the Bioconda channel and dockerize it.

\*\*or\*\* See required dependencies <u>TEtrimmer dependencies</u> and <u>TEtrimmer env for linux.vml</u>.

#### 1.2 Test run

For users running TEtrimmer for the first time, we recommend testing the software with example test files. This practice ensures TEtrimmer, along with its associated databases, is correctly installed and capable of executing without errors. Download example test files from Github: test\_input.fa and test\_genome.fasta.

During the first run, a Pfam database (around 1.6 Gb) will be automatically downloaded (unless the user specifies a custom database by -pfam\_dir). This step requires a network connection and might take around 10 mins. The test run typically takes 5 to 10 minutes, depending on your computing setup.

#### Test run command:

If the above command is run successfully, your terminal will look like below:

Firstly, the Pfam database will be downloaded.

```
Pfam-A.hmm not found. Downloading... This might take some time. Please be patient

Pfam-A.hmm is downloaded and unzipped. Pfam database was stored in
/Users/hangxue/Documents/Wildermuth_Lab/TE_Trimmer/pfam_database

Pfam-A.hmm.dat not found. Downloading... This might take some time. Please be patient

Pfam-A.hmm.dat is downloaded and unzipped. Pfam database was stored in
/Users/hangxue/Documents/Wildermuth_Lab/TE_Trimmer/pfam_database
```

Finally, the progress bar should show 100% completeness.



Note: It might take a long time from 99% to 100% for your genome. TEtrimmer performs de-duplication and clustering via CD-EST-HIT which can be computationally intensive when dealing with many sequences.

## 1.3 Understand TEtrimmer output files

Your output directories should look the same as the following:

Classification and deduplication Multiple sequence alignment Single\_fasta\_files TE\_Trimmer\_for\_proof\_annotation Annotation perfect Annotation\_good Annotation\_recommend\_check Anotattion need check Clustered proof annotation Cluster1 Cluster2 Cluster3 TE\_low\_copy TE skipped TEtrimmer\_proof\_anno\_GUI annoGUI.py Sequence\_name\_mapping.txt summary.txt TE\_Trimmer\_consensus.fasta TE Trimmer consensus merged.fasta

You can compare your test results with the example <u>test\_output</u> on GitHub to make sure TEtrimmer is running successfully on your setup. Note: the results of each run will be slightly different due to the randomness in alignment and clustering steps. Explanations for each file are as below.

**summary.txt** - summary file. This file contains a summary of each processed sequence.

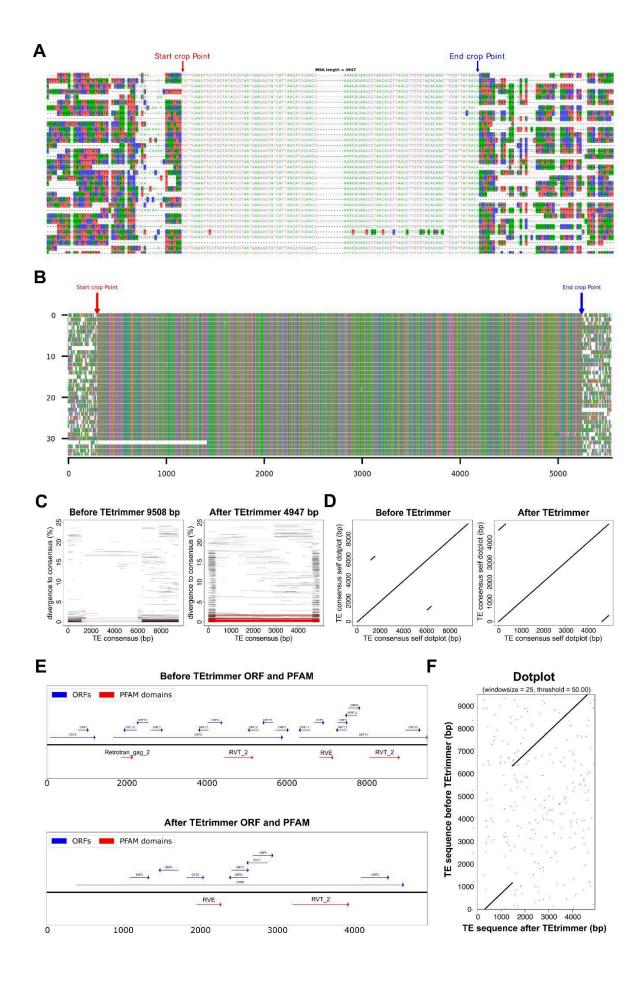
					sum	mary						
input_name	consensus_name	blast_hit_n	cons_MSA_seq_n	cons_full_blast_n	input_length	cons_length	input_TE_type	reclassified_type	terminal_repeat	low_copy	evaluation	status
rnd_6_family_3291	rnd_6_family_3291	0	NaN	NaN	447	447	Unknown	Unknown	FALSE	FALSE	NaN	skipped
rnd_1_family_667	rnd_1_family_667	27	12	9	1382	5363	Unknown	LTR/Copia	FALSE	FALSE	Need_check	processed
rnd_6_family_2561	rnd_6_family_2561	16	16	7	2508	2209	Unknown	DNA/TcMar-Fot1	FALSE	FALSE	Need_check	processed
rnd_1_family_587	rnd_1_family_587	41	12	9	771	2856	LTR/Copia	LTR/Copia	FALSE	FALSE	Need_check	processed
rnd_1_family_587	rnd_1_family_587_01	41	12	6	771	2826	LTR/Copia	LTR/Copia	FALSE	FALSE	Need_check	processed
rnd_5_family_182	rnd_5_family_182	57	54	31	433	395	Unknown	Unknown	FALSE	FALSE	Reco_check	processed
rnd_1_family_188	rnd_1_family_188	31	26	21	6453	6467	LTR/Gypsy	LTR/Gypsy	LTR	FALSE	Good	processed
rnd_1_family_588	rnd_1_family_588	534	84	416	306	3214	DNA/MULE-MuDR	DNA/MULE-MuDR	FALSE	FALSE	Reco_check	processed
rnd_1_family_470	rnd_1_family_470	157	50	42	5760	5847	Unknown	LTR/Gypsy	LTR	FALSE	Perfect	processed
rnd_1_family_470	rnd_1_family_470_01	157	17	26	5760	5051	Unknown	LTR/Gypsy	FALSE	FALSE	Need_check	processed

- input\_name: the sequence header of the original input sequence with converted special characters. If # is present in the original header, the string after # is used for the TE classification, and the string before # is kept for naming the sequence.
- consensus\_name: the sequence header of the processed consensus sequence. In the case where an input sequence can yield multiple processed consensus sequences, the subsequent "consensus\_name" will be appended by a numerical identifier at the end. (eg. "rnd\_1\_family\_470" and "rnd\_1\_family\_470\_01" are two processed consensus sequences derived from the same input sequence "rnd\_1\_family\_470").
- blast\_hit\_n: the blast hit number of the original input sequence against your genome
  file
- **cons\_MSA\_seq**: sequence number of the multiple sequence alignment (MSA) used for generating the TE consensus sequence.
- **cons\_full\_blast**: the full blast hit number of the processed consensus sequence. Full blast means the blast hits must cover more than 90% of the query sequence.
- **input\_length**: the length of the original input sequence.
- **cons\_length**: the length of the TEtrimmer processed consensus sequence.
- **input\_TE\_type**: the TE class of the original input sequence if known. Otherwise, it will be "Unknown".
- reclassified\_TE\_type: the TE class of the processed consensus sequence either inherited from the original input sequence or reclassified by TEtrimmer. For example, "rnd\_6\_family\_2561" is reclassified as "DNA/TcMar-Fot1" after processing.
- **terminal\_repeat**: the presence of either TIR or flanking LTR at both ends of the processed consensus sequence.
- low\_copy: whether the input sequence is considered as a low copy TE element. TE will be denoted as low copy when its BLASTn hit number is less than 10, but with more than 2 full-length BLASTn hits and terminal repeat (LTR or TIR).
- evaluation: the processed consensus sequence is evaluated based on the parameters shown below. The "Perfect" and "Good" ones are high-quality TE consensus sequences that require no or minimal modifications while the "Reco\_check" and "Need\_check" might need additional manual proof curation by the provided TEtrimmer GUI.

Criteria	Terminal_repeat	Classified	cons_MSA_seq	cons_full_blast	if_PFAM
Perfect	TRUE	TRUE	>=30	>=5	TRUE
Good	TRUE	Not_required	>=10	>=2	Not_required
Reco_check	Not_required	Not_required	>=20	>=2	Not_required
Need_check	Not_required	Not_required	Not_required	Not_required	Not_required

"TE_Trimmer_consensus.fasta" file. The input TE sequence will be skipped when the copy number is less than 10 and does not belong to "low copy" TEs.
<b>TE_Trimmer_consensus.fasta</b> : fasta file of TEtrimmer processed TE consensus sequences before de-duplication.
<b>TE_Trimmer_consensus_merged.fasta</b> : fasta file of processed TE consensus sequences file after de-duplication.
Sequence_name_mapping.txt: text file maps TEtrimmer modified input fasta sequence names with original names. Any occurrence of '/', '-', ':', '', ' ' and empty spaces before '#' will be converted to '_'
Multiple_sequence_alignment:
error_file.txt - stor all error messages, only present when errors were found.  All intermediated analysis files will be kept when the "debug" option is enabled.
Single_fasta_files: All sequences in the input file will be separated into single fasta files and be stored here to allow multithreads analysis.
Classification_and_deduplication:
□round*.fasta_clstr: merged sequences after round* cd-hit-est □ temp_TE_Trimmer_classified_consensus.fasta: processed consensus sequences
<b>TE_Trimmer_for_proof_annotation</b> : This folder contains files that can be used for
additional manual proof annotation.
Annotation_perfect: Four files are associated with each output TE sequence.
<b>TE_name.cluster.fa</b> : MSA file before clustering
TE_name.raw.fa: MSA file after clustering but before cleaning
TE_name.fa: MSA file used for consensus sequence generation
<b>TE_name.pdf</b> : Plot file used to evaluate output. See the explanation below.

**Status:** skipped or processed. If a sequence is skipped, it is not included in the



**Figures A, B**: these two figures give you a visual representation of the multiple sequence alignment (MSA) of the processed consensus sequences. Ideally, the MSA should have clear boundaries at both ends.

**Figure C, D**: this part gives you the comparison between the processed consensus sequence and its original input sequences using TE-Aid plots. Left panel: the genomic hits with divergence to consensus. The red color indicated high-quality hits. Right panel: a self dot-plot. (https://github.com/clemgoub/TE-Aid)

**Figure E**: this part gives you the comparison of the predicted open reading frame (ORF) and PFAM domains between consensus sequences before and after TEtrimmer analysis.

**Figure F**: This figure is the dot plot between the input sequence and processed consensus sequence. The x-axis represents the TEtrimmer processed TE consensus sequence and the y-axis is the original input sequence.

- Annotation\_good: same as the previous folder
- Annotations\_check\_recommended: same as the previous folder
- **Annotations\_check\_required**: same as the previous folder
- Clustered\_proof\_annotation: The folder copies all files from "Annotation\_perfect", "Annotation\_good", "Annotation\_check\_recommended", and "Annotation\_check\_required" and groups them into different clusters based on TE consensus sequence similarity. You can do your manual proof curation based on this folder. Theoretically, you only need to choose one set of consensus files from each cluster folder.
- TE\_low\_copy
  - TE\_name.fa: Input sequence that is classified as a low-copy element
  - **TE** name.pdf: Plots file helps to assess if it is a true low-copy TE
- **TE\_skipped** 
  - **TE** name.pdf: Plots file helps to assess if it is a true skipped TE
- TEtrimmer\_proof\_anno\_GUI
  - annoGUI.py: Graphical user interface tool to help manual proof curation. Use "python ./annoGUI.py -g [genome.fa]" to start it.

# 2. Obtain input files

#### 2.1 Genome file

Select a genome database or repository that hosts the genetic information for your chosen organism. Some commonly used databases include: National Center for Biotechnology Information (NCBI), Ensembl, UCSC Genome Browser, and European Bioinformatics Institute (EBI). Downloading your genome as a FASTA file (.fa or .fasta).

## 2.2 TE consensus library

TEtrimmer uses the TE consensus library from de novo TE annotation tools, like RepeatModeler or EDTA, as input. For this reason, you have to run RepeatModeler or other TE annotation software first. You can find details about how to use RepeatModeler at https://www.repeatmasker.org/RepeatModeler/ (Flynn et al. 2020) or EDTA (Su et al. 2021). TEtrimmer package already includes RepeatModeler. Below is an example command of running RepeatModeler. (-LTRStruct option is highly recommended to use. This increases the running time of RepeatModeler but you will get more intact TEs at the end).

```
BuildDatabase -name <genome> <genome.fa>
RepeatModeler -database <genome> -threads 20 -LTRStruct >& run.out &
```

The RepeatModeler output file <genome\_families.fa> can be used as the input TE consensus library for TEtrimmer.

## 3. Run TEtrimmer

## 3.1 Hardware requirement

System: Linux, macOS

#### RAM:

 For HPC Linux users, enough RAM needs to be assigned. We highly recommend running TEtrimmer on HPC with at least 40 threads and assigning at least 5 GB RAM to each thread.

Threads	RAM		
40	200 GB		
100	600 GB		

 Windows WSL and macOS users can use Virtual Memory. Simply assign 20 threads to push the CPU to its limits. We did tests on a Macbook Pro (2020 M1 chip, 16 GB RAM) and compared with HPC, you can find the running time here:

Query sequence number	Platform	Threads	RAM	Running time
1700	Macbook Pro M1	20	16 GB + Virtual Memory	50 hours
1700	HPC	40	200 GB	5 hours

## 3.2 Basic run examples

Note: if your <output\_directory> is not empty, TEtrimmer will create a new folder under it named using the following format "TEtrimmer\_output\_yyyymmdd\_hhmmss" (eg. TEtrimmer\_output\_20240108\_125418)

If you want to continue the analysis based on previous unfinished results in the same directory:

If you are combining files from different sources for the input file, we recommend removing duplicate sequences during processing. This step can potentially save overall run time.

## 3.3 Basic options

The basic options to run TEtrimmer are as follows:

```
--input file, -i
       Required
       string: path to TE consensus library (FASTA format). Use the output from RepeatModeler
       or EDTA et al.
--genome file, -g
       string: Path to genome file (FASTA format).
--output dir, -o
       Default: current working directory
       string: Path to the output directory
--preset, -s
       Default: conserved
       string: choose one preset config from below (see more info in section 3.5):
               conserved
              divergent
--num threads, -t
       Default: 10
       int: the thread numbers used for TEtrimmer.
```

#### --classify unknown

Default: False

boolean: Use RepeatClassifier to classify the processed consensus sequence if the input sequence is not classified or is unknown.

#### --classify all

Default: False

boolean: Use RepeatClassifier to classify every processed consensus sequence. WARNING: it will take a longer time.

#### --continue analysis, -ca

Default: False

boolean: continue from the previous unfinished TEtrimmer run and use the same output directory.

#### --dedup

Default: False

boolean: remove duplicate sequences in the input file. Recommend using this option if the user is combining results from output files from different de novo TE annotatils tools. This will help reduce total running time without losing much info. (95% length and 95% similarity)

#### --genome anno, -ga

Default: False

boolean: perform genome TE annotation using RepeatMasker with the TEtrimmer curated TE libraries.

#### --help

Default: False

boolean: show help message and exit.

## 3.4 Advanced options

#### --debug

Default: False

boolean: debug mode. This will keep all raw files. WARNING: a lot of files will be kept, especially in the Multiple\_sequence\_alignment folder.

#### --pfam\_dir, -pd

Default: None

string: Pfam database directory. TE Trimmer will download the database automatically. Only turn on this option if you want to use a local PFAM database or the automatic download fails.

#### --cons thr

Default: 0.8

float: The minimum level of agreement required at a given position in the alignment for a consensus character to be called. Positions with agreement less than the threshold will be represented by the ambiguous character "N" in the consensus sequence.

#### --mini orf

Default: 200

int: the minimum ORF length that will be predicted by TEtrimmer.

#### --max msa lines

Default: 100

int: the maximum number of sequences to be included in a multiple sequence alignment. We do not recommend increasing this number.

## --top\_mas\_lines

Default: 100

int: if the sequence number of multiple sequence alignment (MSA) is greater than <max\_msa\_lines>, TEtrimmer will first sort sequences by length and choose <top\_msa\_lines> number of sequences. Then, TEtrimmer will randomly select sequences from all remaining BLAST hits until <max\_msa\_lines> sequences are found for the multiple sequence alignment.

#### --min\_seq\_num

Default: 10

int: the minimum blast hit number required for the input sequence. If the blast hit number is below <min\_seq\_num>, this sequence will be skipped. Additionally, it serves as the minimum requirement for the number of sequences needed to form a valid cluster during multiple sequence alignment. If the blast hit or number of sequences in each cluster is below <min\_seq\_num>, this sequence will also be skipped. A higher value of this number might lead to more low copy TE skipped but more accurate results. We do not recommend decreasing this number.

#### --min blast len

Default: 150

int: The minimum sequence length for blast hits to be included for further analysis.

#### --max\_cluster\_num

Default: 2

int: The maximum number of clusters assigned in each multiple sequence alignment. Each multiple sequence alignment can be grouped into different clusters based on alignment patterns. TEtrimmer will sort clusters by size and choose the top <max\_cluster\_num> of clusters for further analysis. WARNING: using a larger number will potentially result in more accurate consensus results but will significantly increase the running time. We do not recommend increasing this value to over 5.

#### --ext\_thr

Default: 0.7

Float: the threshold to call "N" at a position. For example, if the most conserved nucleotide in a MSA column has proportion smaller than <ext\_thr>, a "N" will be called at this position. Used with <ext\_check\_win>. The lower the value of <ext\_thr>, the more likely to get longer the extensions on both ends. You can try reducing <ext\_thr> if TEtrimmer fails to find full-length TEs.

#### --ext\_check\_win

Default: 150

int: the check windows size during defining start and end of the consensus sequence based on the multiple sequence alignment. Used with <ext\_thr>. If <ext\_check\_win> bp at the end of multiple sequence alignment has "N" present (ie. positions have similarity proportion

smaller than <ext\_thr>), the extension will stop, which defines the edge of the consensus sequence.

## --ext\_step

Default: 1000

int: the number of nucleotides to be added to the left and right ends of the multiple sequence alignment in each extension step. TE\_Trimmer will iteratively add <ext\_step> nucleotides until finding the TE boundary or reaching <max\_ext>. For organisms with high divergence such as in rice, we recommend increasing this number (eg. 1500).

#### --max ext

Default: 7000

int: the maximum extension in nucleotides at both ends of the multiple sequence alignment. For organisms with high divergence such as in rice, we recommend increasing this number (eg, 9000).

#### --gap\_thr

Default: 0.4

float: If a single column in the multiple sequence alignment has a gap proportion larger than <gap\_thr> and the proportion of the most common nucleotide in this column is less than <gap\_nul\_thr>, this column will be removed from the consensus.

#### --gap\_nul\_thr

Default: 0.7

float: the nucleotide proportion threshold for keeping the column of the multiple sequence alignment. Used with the <gap\_thr> option. i.e. if this column has <40% gap and the portion of T (or any other) nucleotide is >70% in this particular column, this column will be kept.

## --crop\_end\_div\_thr

Default: 0.8

float: the crop end by divergence function will convert each nucleotide in the multiple sequence alignment into a proportion value. This function will iteratively choose a sliding window from each end of each sequence of the MSA and sum up the proportion numbers in this window. The cropping will continue until the sum of proportions is larger than <crop end div thr>. Cropped nucleotides will be converted to -.

#### --crop\_end\_div\_win

Default: 20

int: the window size used for the end-cropping process. Used with <crop\_end\_div\_thr> option.

## --crop\_end\_gap\_thr

Default: 0.1

float: the crop end by gap function will iteratively choose a sliding window from each end of each sequence of the MSA and calculate the gap proportion in this window (i.e. the number of columns that are gaps in the window size of <crop\_end\_gap\_win>). The cropping will continue until the sum of gap proportions is smaller than <crop\_end\_gap\_thr>. Cropped nucleotides will be converted to -.

```
--crop_end_gap_win
```

Default: 250

int: the window size used to crop end by gap, used with <crop end gap thr> option.

#### --start\_patterns

Default: TG

string: LTR elements always start with a conserved sequence pattern. TE Trimmer searches the beginning of the consensus sequence for these patterns. If the pattern is not found, it will extend the search of <start\_patterns> to up to 15 nucleotides from the beginning of the consensus sequence and redefine the start of the consensus sequence if the pattern is found. Note: The user can provide multiple LTR start patterns in a comma-separated list, like: TG,TA,TC (no spaces; the order of patterns determines the priority for the search).

#### --end\_patterns

Default: CA

string: LTR elements always end with a conserved sequence pattern. TE Trimmer searches the end of the consensus sequence for these patterns. If the pattern is not found, it will extend the search of <end\_patterns> to up to 15 nucleotides from the end of the consensus sequence and redefine the end of the consensus sequence if the pattern is found. Note: The user can provide multiple LTR end patterns in a comma-separated list, like: CA,TA,GA (no spaces; the order of patterns determines the priority for the search).

## 3.5 Default parameters associated with different organisms

The preset parameters below still have room for improvement and there is no guarantee that one of them will be optimal for your organism. The default (conserved) setting is considered better if your organism has relatively more conserved TE elements, i.e. the MSA of blast hits of your input sequence look clean and conserved (eg. the screenshot of TE\_name.pdf in Section 1.3). The divergent setting has a longer extension period of TE elements and lower conservation threshold when deciding the boundary of TE elements. The setting for divergent TE elements will take longer to finish the TEtrimmer run than the conserved one.

```
"conserved": {
  "cons thr": 0.8,
  "max msa lines": 100,
  "top mas lines": 100,
  "min_seq_num": 10,
  "min blast len": 150,
  "max cluster num": 2,
  "ext thr": 0.7,
  "ext_step": 1000,
  "max ext": 7000,
  "gap thr": 0.4,
  "gap nul thr": 0.7,
  "crop end div thr": 0.8,
  "crop_end_div_win": 40,
  "crop_end_gap_thr": 0.1,
  "crop end gap win": 250,
  "start_patterns": "TGT",
```

```
"end_patterns": "ACA",
 "mini_orf": 200,
 "ext_check_win": 150
"cons_thr": 0.8,
 "max_msa_lines": 100,
 "top_mas_lines": 100,
 "min_seq_num": 10,
 "min_blast_len": 150,
 "max_cluster_num": 2,
 "ext thr": 0.6,
 "ext_step": 2000,
 "max_ext": 8000,
 "gap_thr": 0.4,
 "gap_nul_thr": 0.7,
 "crop_end_div_thr": 0.7,
 "crop_end_div_win": 40,
 "crop_end_gap_thr": 0.1,
 "crop_end_gap_win": 250,
 "start_patterns": "TG",
 "end_patterns": "CA",
 "mini_orf": 200,
 "ext_check_win": 150
},
```

# 4. Understand additional output files

See common output files in Section 1.3. We provide a brief explanation of additional output files below when the "debug" option is used. For further explanation, feel free to email the authors.

**Multiple\_sequence\_alignment** - All raw files will be kept in this folder when <debug > is enabled. Explanations of selected files are shown below.

For each input sequence, it might have the associated files below

*.b: blast result file
*.b.bed: bed file converted from blast result file
*.bed_u: bed file after removing identical lines
*.bed_uf.bed: bed file after filter (only choose the top 100 long sequence for multiple
sequence alignment)
*.bed*_*_n.*: bed file or alignment file with left and right extension number (eg.
bed_1000_2000 means left 1000 extension and right 2000 extension).
*_n.fa: fasta file, use the bed file name column as sequence header
*_bcln.fa: bed file that has been cleaned. All letters in the fasta sequence not
belonging to [AGCTagct] will be deleted.
*_aln.fa – aligned fasta file
*_aln.fa_gs – aligned fasta file after gap removing
*.fa_pat_MSA.fa – pat means pattern. Distinct columns in the MSA are extracted and
used for clustering.
*.iqtree – iqtree result
<pre>*g_*.bed – bed file for cluster *</pre>
*_fm.bed: final multiple sequence alignment

HMM - This folder is used to store Hidden Markov Model file. Only created when <-hmm> is enabled.

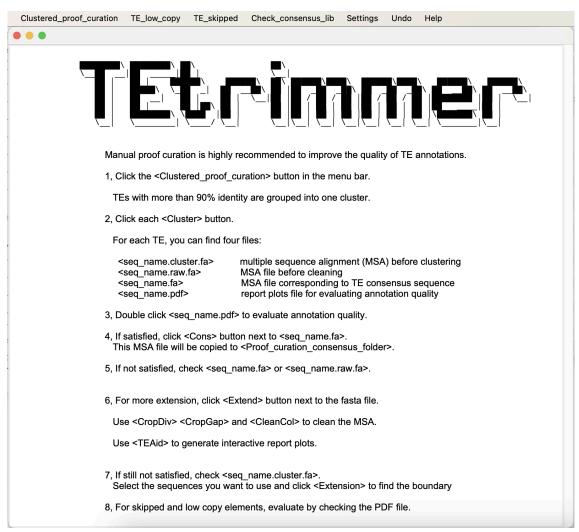
# 5. Additional proof curation (highly recommended)

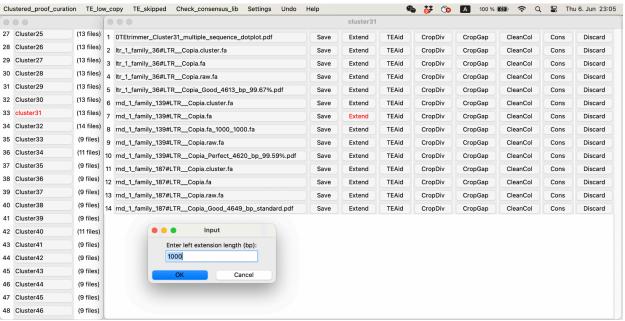
You can use the TEtrimmer graphical user interface (GUI) tool to assist your proof curation. We highly recommend doing manual proof curation to increase the result quality.

TEtrimmer GUI can be used for Linux, macOS, and Windows. But "TEAid" function can't be used for Windows systems.

```
To start TEtrimmer GUI tool:
```

```
python <path_to_TEtrimmer_output_directory>
/TEtrimmer_for_proof_curation/TEtrimmer_proof_anno_GUI/annoGUI.py -g
<genome.fa>
```





You can browse all the files by double-clicking the file name.

The MSA can be extended by clicking the "Extend" button.

The "TEAid" button is designed to generate interactive plots. Note: this is usable for macOS and Linux.

MAS cleaning functions are also integrated into the GUI named "CropDiv", "CropGap", and "CleanCol".

"CropDiv" and "CropGap" can clean MSA rows.

"CleanCol" can remove MSA gappy columns.

You can modify MSA cleaning function parameters by "Settings" in the menu bar.

# 6. FAQs

## Can I run TEtrimmer on a laptop or cluster?

TEtrimmer can be executed on both laptops and clusters. Refer to the examples in <u>section 3.1</u>. Note: If your run stops for any reason, you can use the "--continue\_analysis" option to resume from where you left off without the need to start over.

# Which preset setting should I use for my organism? [--preset conserved or divergent?]

If you have little idea about how conserved TEs are in your organism, you can start with the setting of "divergent". The "divergent" setting will probably take longer than the "conserved" setting but it might have better performance. Using the "divergent" setting on conserved TEs will not compromise the output performance; however, the opposite might be true.

## 7. Reference

- Flynn, Jullien M., Robert Hubley, Clément Goubert, Jeb Rosen, Andrew G. Clark, Cédric Feschotte, and Arian F. Smit. 2020. "RepeatModeler2 for Automated Genomic Discovery of Transposable Element Families." *Proceedings of the National Academy of Sciences* 117 (17): 9451–57.
- Ou, Shujun, Weija Su, Yi Liao, Kapeel Chougule, Jireh RA Agda, Adam J. Hellinga, Carlos Santiago Blanco Lugo et al. "Benchmarking transposable element annotation methods for creation of a streamlined, comprehensive pipeline." *Genome biology* 20 (2019): 1-18.