TETrimmer Manual

Last updated: Feb 28, 2024
Author: Hang Xue; Jiangzhao Qian

TETrimmer Github repository

1. Getting started

1.1 Installation

Install Conda package for TETrimmer. Note: TETrimmer installation requires python=3.10

```
conda create --name tetrimmerenv python=3.10
conda activate tetrimmerenv
conda install -c conda-forge mamba
conda install qianjiangzhao::tetrimmer
tetrimmer --help
```

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

Or install the required dependencies as listed here (https://github.com/qjiangzhao/TETrimmer/blob/main/TETrimmer_dependencies) and clone the github repository for TETrimmer.

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 downloaded by default (unless the user specifies a custom database by -pfam_dir). This step requires 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:

```
tetrimmer [-i <path to test_input.fa>] [-g <path to
test_genome.fasta>] [-o <output_directory>] [-t 10]
[--classify_unknown]
```

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

Firstly, Pfam database will be downloaded.

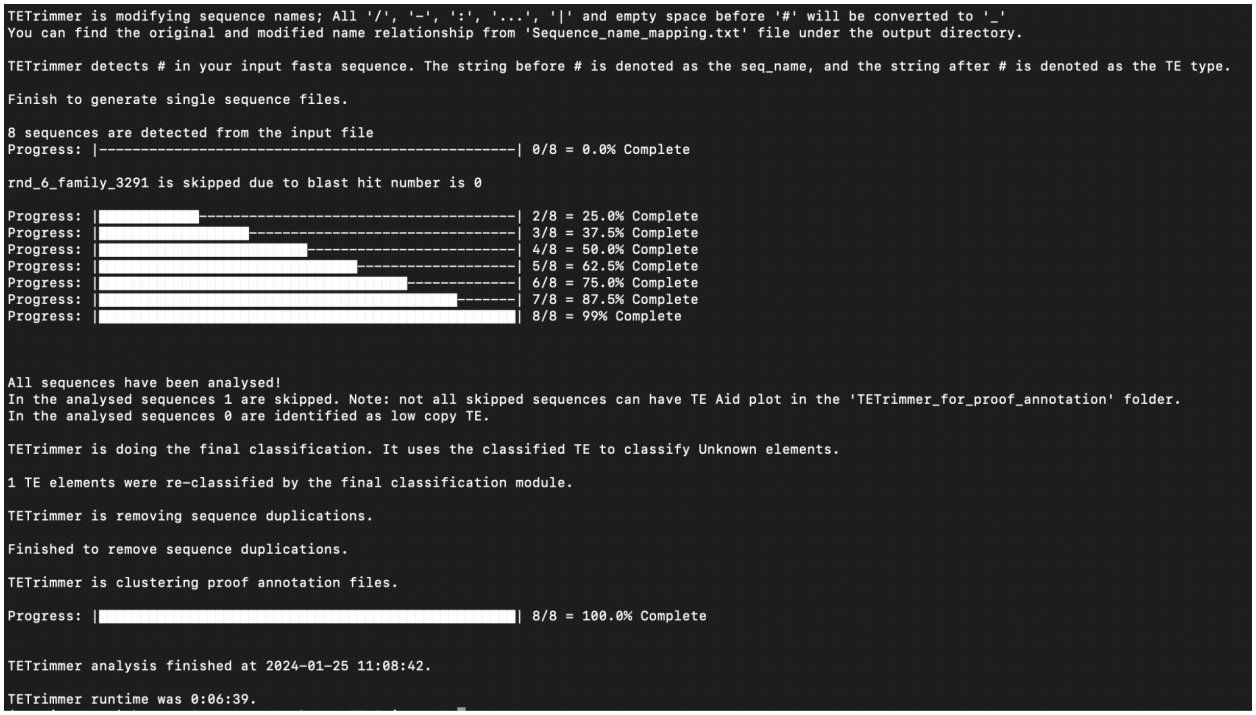
```
Pfam-A.hmm not found. Downloading... This might take some time. Please be patient


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.



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

📄Sequence_name_mapping.txt

📄summary.txt

📄TE_Trimmer_consensus.fasta

📄TE_Trimmer_consensus_merged.fasta

You can compare your test results with example test_output on github to make sure TETrimmer is run successfully on your own 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.

summary

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, only the string before # is kept for the naming purpose.

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 extended by appending 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: number of sequences that are clustered together and give rise to the processed consensus sequences by multiple sequence alignment (MSA)

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. For example, the original input sequence classification of rnd_6_family_2561 is Unknown.

reclassified_TE_type: the TE class of the processed consensus sequence either inherited from the original input sequence or reclassified. For example, rnd_6_family_2561 is reclassified as DNA/TcMar-Fot1 after processing.

terminal_repeat: the presence of either TIR or 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 blast hit number is less than 10, but with more than 2 full blast hits and terminal repeat (LTR or TIR).

evaluation: the processed consensus sequence is evaluated based on 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 curation.

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

Status: skipped or processed. If a sequence is skipped, it is not included in the TE_Trimmer_consensus.fasta. The input TE sequence will be skipped when the copy number is less than 10.

📄**TE_Trimmer_consensus.fasta** - fasta file of processed TE consensus sequences before de-duplication.

📄**TE_Trimmer_consensus_merged.fasta** - fasta file of processed TE consensus sequences file after de-duplication.

📁**Multiple_sequence_alignment** -

- 📄**error_file.txt** - Error file to store all error messages, only present when error were found. Note: some errors do not affect the final outcome of the TETrimmer.

📁**Single_fasta_files** - All sequences in the input file will be separated to single fasta files and be stored here for the purpose of parallel processing.

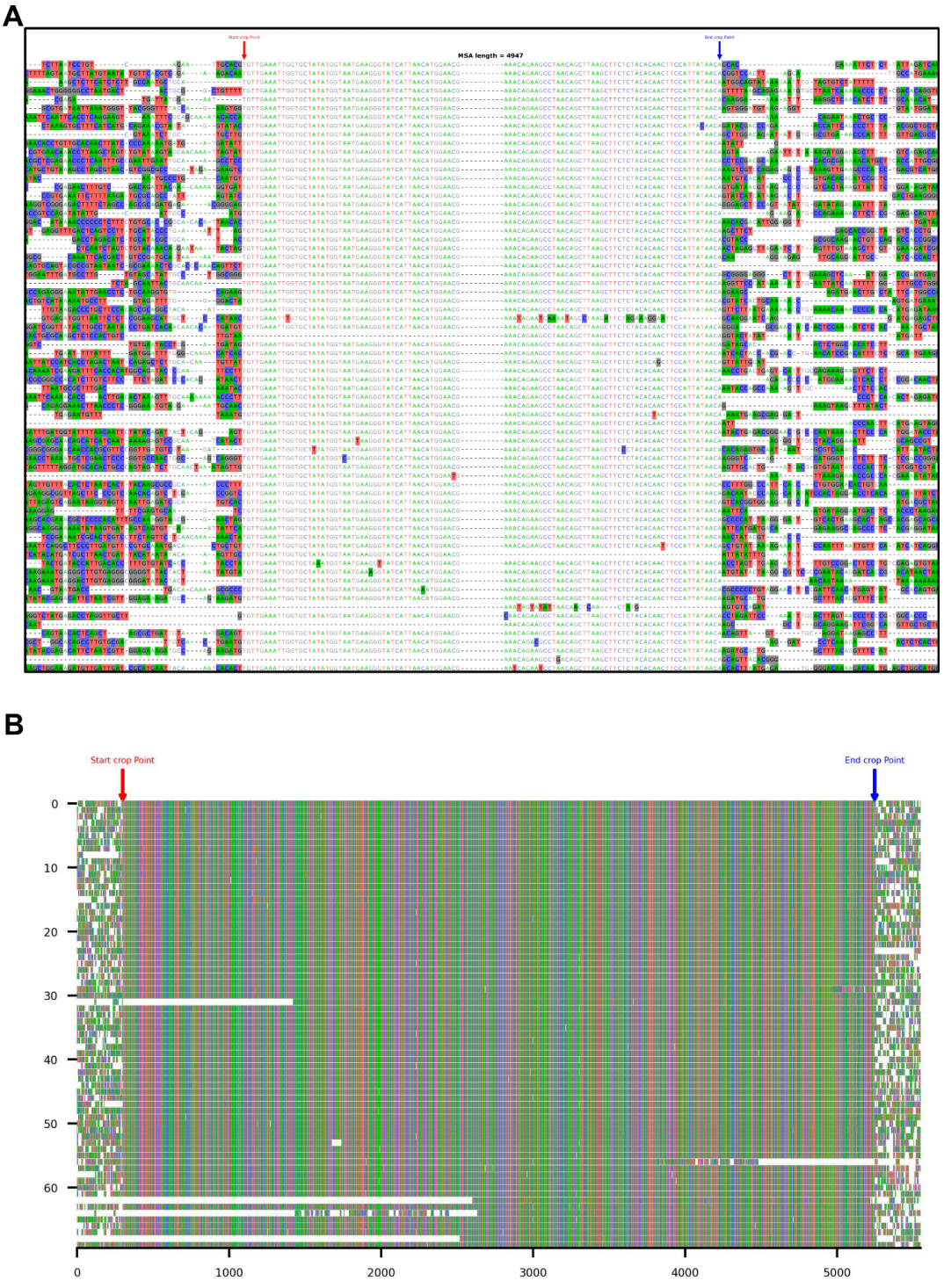
📁**Classification_and_deduplication**

- 📄_round*.fasta_clstr: merged sequences after round* cd-hit-est
- 📄 temp_TE_Trimmer_classified_consensus.fasta: processed consensus sequences that has known TE classification
- 📄 temp_TE_Trimmer_unknown_consensus.fasta: processed consensus sequences that has Unknown TE classification
- 📁 temp_repeatmasker_classification: result files after running RepeatMasker to classify unknown sequences (temp_TE_Trimmer_unknown_consensus.fasta) by classified sequences *(temp_TE_Trimmer_classified_consensus.fasta)

- 📄 TETrimmer_consensus_no_low_copy.fasta – TE consensus library before deduplication but don't include low copy element. Used to cluster manual proof annotation files
- 📄 TETrimmer_consensus_no_low_copy.fasta_cd.fa – cd-hit-est fasta output
- 📄 TETrimmer_consensus_no_low_copy.fasta_cd.fa.clstr – cd-hit-est output containing clustering information.

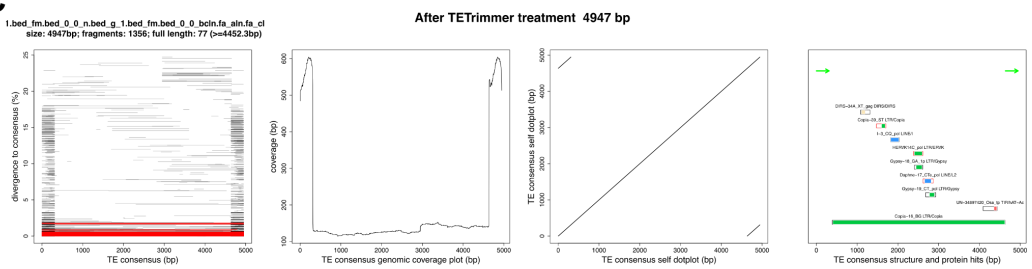
📁**TE_Trimmer_for_proof_annotation** - This folder contains files that can be used for additional manual proof annotation.

- 📁**Annotation_perfect** - For each sequence, four files are associate with it (anno.fa; fa; pdf;bed).
  - 📄**TE_name.anno.fa** - Multiple sequence alignment file before cleaning.
  - 📄**TE_name.bed** – Bed file used for further extension.
  - 📄**TE_name.fa** - Multiple sequence alignment file after cleaning and boundary definition.
  - 📄**TE_name.pdf** - Plot file used to evaluate output. See the explanation below**.**



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.

C



D

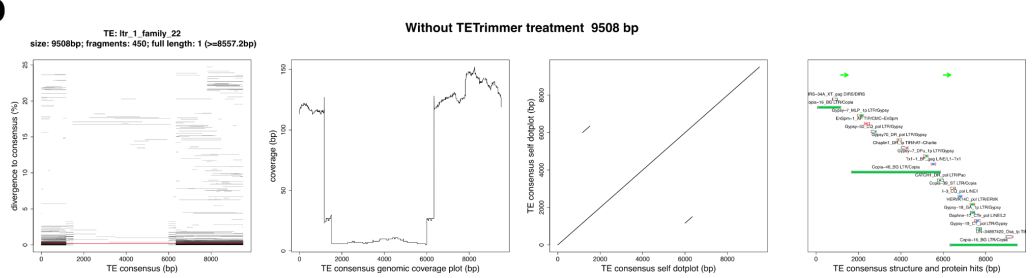
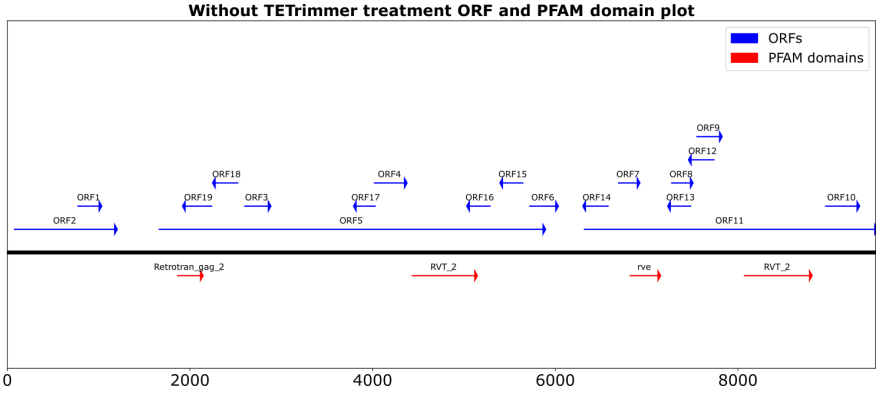


Figure C, D: this part gives you the comparison between processed consensus sequence (top) and its original input sequences (bottom) using TE-Aid plots.

- First panel: the genomic hits with divergence to consensus. Red color indicated high quality hits.
- Second panel: the genomic coverage of the consensus.
- Third panel: a self dot-plot.
- Last Panel: a structure analysis includes: TIR and LTR suggestions, open reading frames (ORFs) and TE protein hit annotation. See more explanation of TE-Aid plots at https://github.com/clemgoub/TE-Aid.

E



F

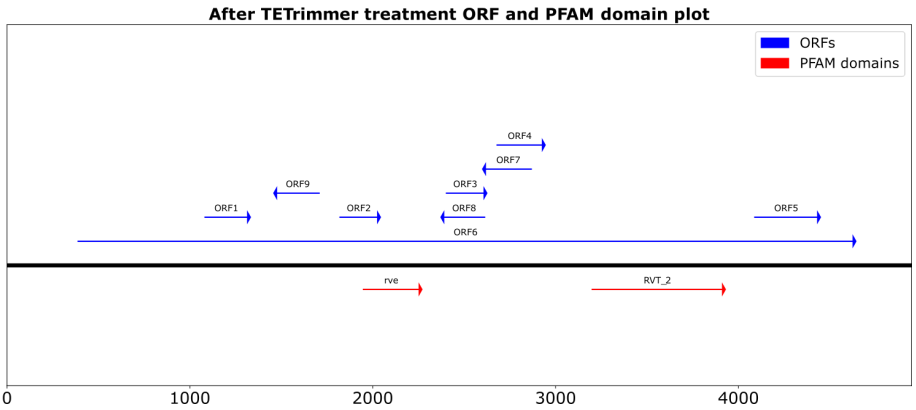


Figure E, F: this part gives you the comparison of predicted open reading frame (ORF) and PFAM domains between consensus sequence (top) and its original input sequences (bottom)

G

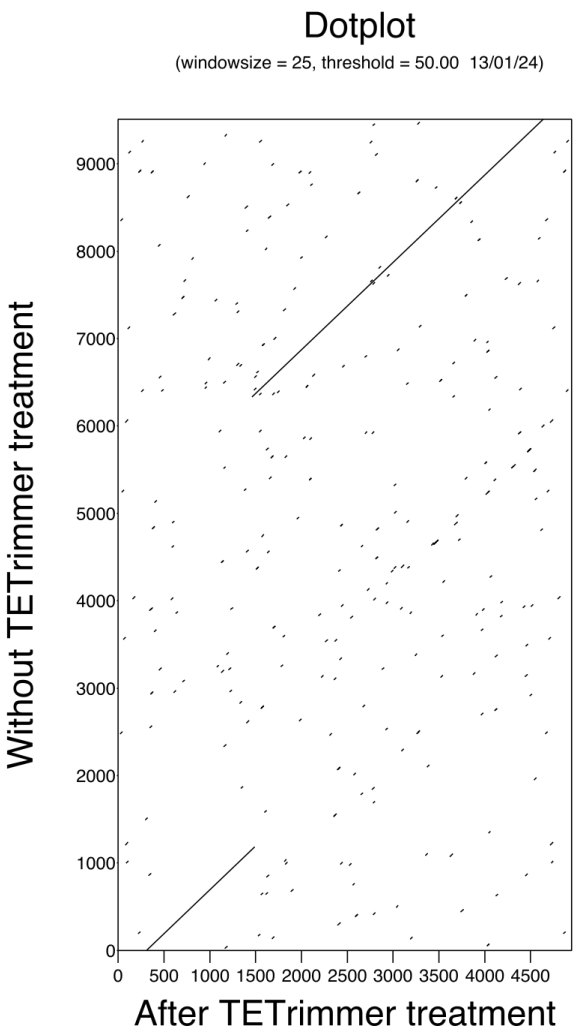


Figure G: This figure is the dotplot between the input sequence and processed consensus sequence. The x axis represents the TETrimmer processed TE consensus sequence and the y axis is the original input sequence.

📁**Annotation_good** - same as previous folder
📁**Annotations_check_recommended** - same as previous folder
📁**Annotations_check_required** - same as previous folder
📁**Clustered_proof_annotation** – The folder copies all files from “Annotation_perfect”, “Annotation_good”, “Annotation_check_recommended”, and “Annotation_check_required" and group them to different clusters based on TE consensus sequence similarity. You can do your manual proof annotation 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

📄**TE_name.pdf** – TE-Aid plot and Pfam plot. This can help to assess if it is a true low copy TE.

📁**TE_skipped**

📄**TE_name.pdf** – TEAid plot and Pfam plot. This can help to assess if it is reasonable to skip those elements. You can rescue the skipped TE by manual proof annotation.

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, 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 at (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 mygenome mygenome.fa

RepeatModeler -database mygenome -threads 20 -LTRStruct >& run.out &
```

The RepeatModeler output file <mygenome_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 user, 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

- PC 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)

```
tetrimmer [-i <mygenome_families.fa>] [-g <mygenome.fa>] [-o
<output_directory>] [-t 20]
```

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

```
tetrimmer [-i <mygenome_families.fa>] [-g <mygenome.fa>] [-o
<output_directory>] [-t 20] [--continue_analysis]
```

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.

```
tetrimmer [-i <mygenome_families.fa>] [-g <mygenome.fa>] [-o
<output_directory>] [-t 20] [--dedup]
```

3.3 Basic options

The basic options to run TETrimmer are as follows:

--input_file, -i
    Required
    string: path to TE consensus file (FASTA format). Use the output from RepeatModeler or EDTA et al.

--genome_file, -g
    Required
    string: Path to genome file (FASTA format).

--output_dir, -o
    Default: current working directory
    string: Path to 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 threads numbers used for TETrimmer.

--hmm
    Default: False
    boolean: Generate HMM files for each processed consensus sequence.

--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 previous unfinished TETrimmer run and would 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
  },
  "divergent": {
    "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. 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. Explanation 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 belong 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
- 📄 *._g_*.bed – bed file for cluster *
- 📄*_fm.bed: final multiple sequence alignment

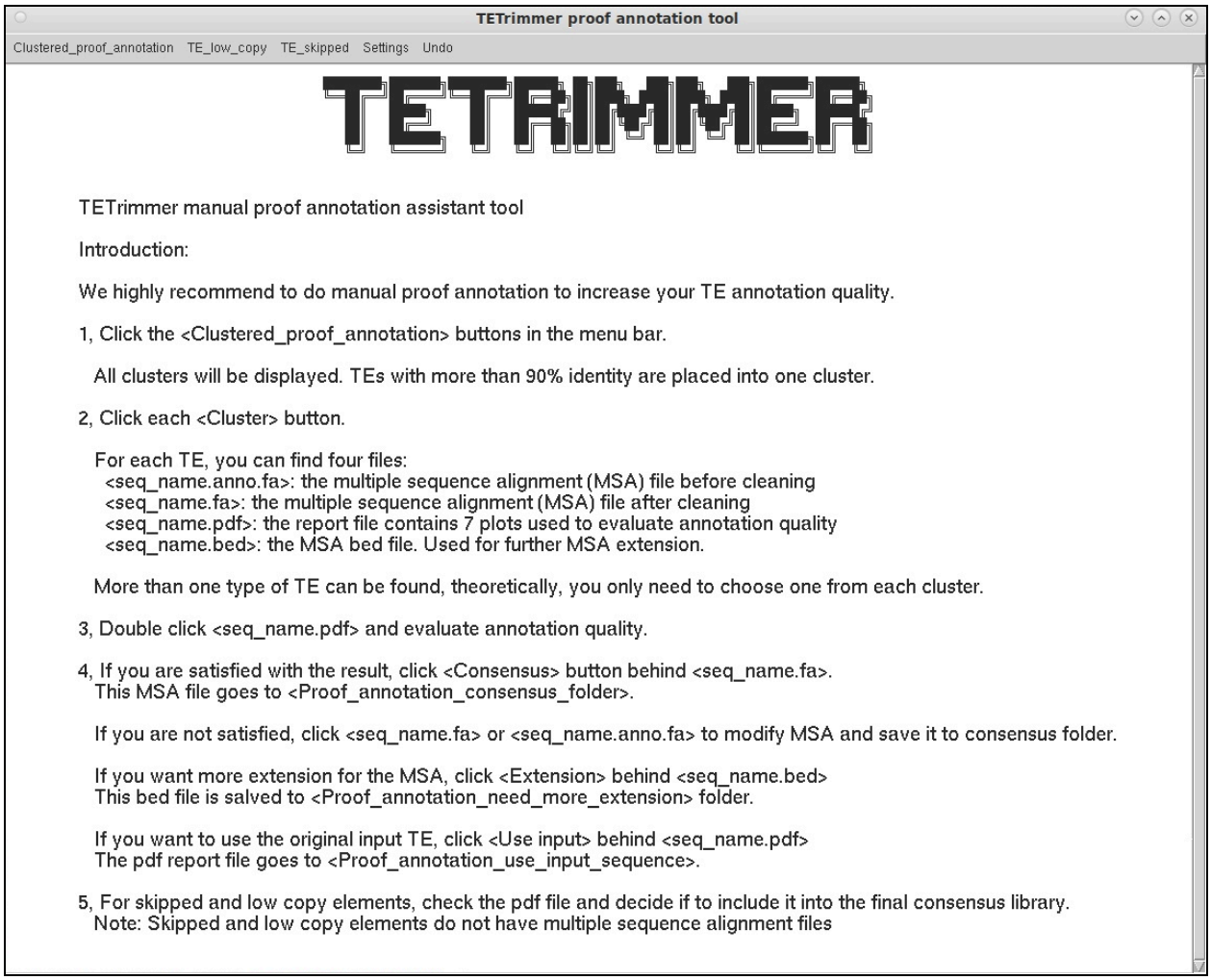
📁HMM - This folder is used to store Hidden Markov Model file. Only created when <-hmm> is enabled.

5. Additional manual curation (highly recommended ; video coming soon)

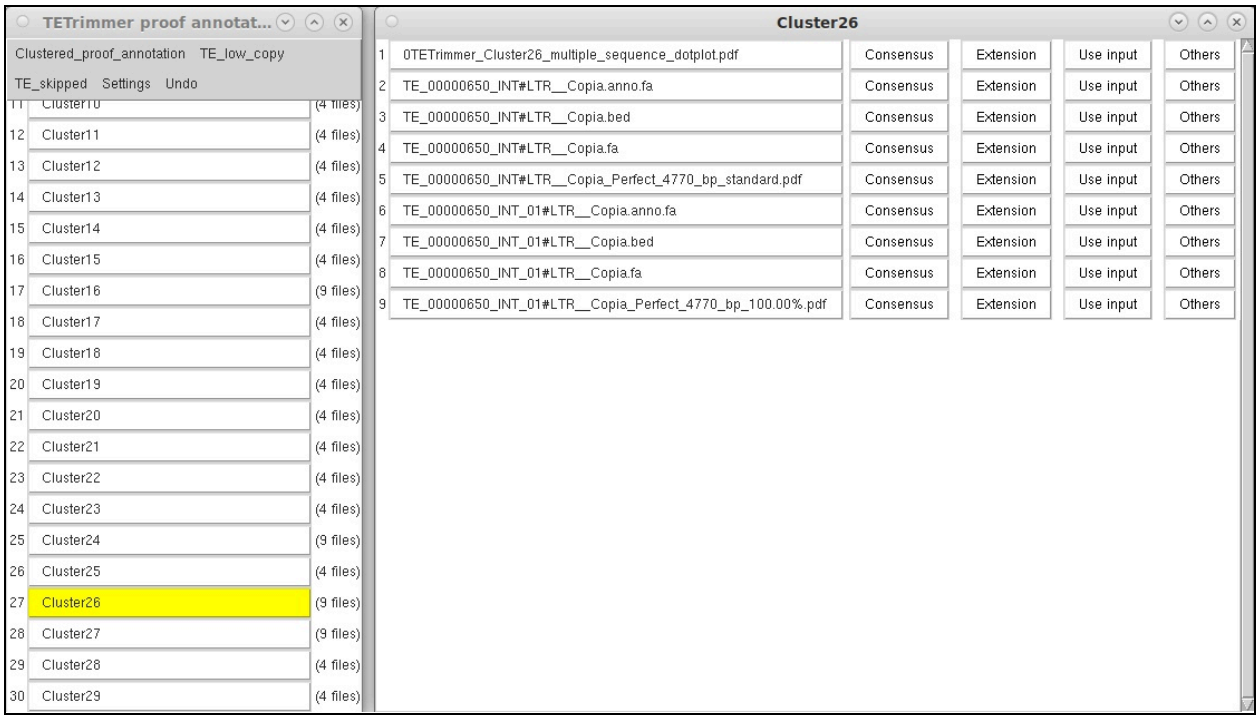
You can use this graphical user interface tool to assist your proof annotation. We highly recommend doing manual proof annotation to increase the result quality. A more detailed video of how to use this GUI will be uploaded soon.
Note: This GUI might not be compatiable with macOS system.

Proof annotation command:

```
python <path to your output_directory>
/TETrimmer_for_proof_annotation/TETrimmer_proof_anno_GUI/annoGUI.py
```



For example, click "Clustered_proof_annotation" and “Cluster” button to show all files in each cluster.



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 section 3.1. 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.