Annotation and Tracking of Transposable Elements in the *C. elegans* Genome

Annotation of Transposable Elements using Transposon Ultimate

Transposon Ultimate (TEUlt) is an annotation pipeline developed by Riehl et al. in 2022 that employs the use of many individual transposon annotation tools to create a consolidated and thorough dataset of transposon annotations (paper here). The following steps outline the process used to obtain the transposable element (TE) annotations utilizing the TEUlt pipeline with some alterations.

Files Needed

- [ ] ANNOTATION.ipynb
- [ ] Genome fastas and/or chr_lengths.txt
- [ ] TE annotations in .gff3 format from TEUlt

Annotation Steps

TEUlt pipeline instructions are located here

1. Set up TEUlt environment
2. Annotate genome using TEUlt annotation pipeline "reasonaTE"
    - We annotated the genome using the TEUlt wrappers of the following tools:
        - helitronScanner
        - mitefind
        - mitetracker
        - repeatmodel (RepeatModeler)
        - repMasker (RepeatMasker)
        - sinescan
        - tirvish
        - transposonPSI
        - NCBICDD1000
3. If necessary, conduct manual annotations and use source code alterations to integrate the annotations into the TEUlt generated annotations
    1. We ran the following tools without using the TEUlt wrappers and then added the output files into the TEUlt file system (a la the directions outlined in the TEUlt documentation for running LTRpred located here) before proceeding to the "parse annotations" step of the TEUlt steps
        - MUST
        - LTR Retriever (utilizes the outputs of LTR Harvest and LTR Finder as inputs)
        - SINE Finder (ran CHUNK-WISE because I ran into issues with the recursive memory limit when using the conda wrapper for the large *C. elegans* genome)
    2. LTR Retriever is not used in the original TEUlt pipeline, but we chose to include annotations from this tool based on assessments of its low rate of false positives as compared to other LTR

annotation tools. The edited source code includes code alterations to the original TEUlt pipeline to include annotations from this program.

4. Parse annotations using TEUlt
5. Run TEUlt reasonaTE pipeline on the genome annotations to obtain final TE annotations from TEUlt
6. At this point, analysis of the TE annotations was conducted using Python code outside of the TEUlt pipeline. This code is located in the ANNOTATION jupyter notebook.
7. Size filter out TEs whose sizes lie outside of canonical literature ranges (some of the programs in TEUlt annotated HUGE portions of the chromosomes as TEs)
    - this is a tricky parameter to define, because there is a line to walk between not eliminating tandem repeats or nested TEs, but also filtering out the ridiculously sized TEs
    - ended up using mid-stringent size parameters to filter out the really giant annotations
    - DNA TE size parameters:
        - Helitron | 20000 bp
        - CMC | 20000 bp
        - Zator | 15000 bp
        - hAT | 20000 bp
        - Sola | 7000 bp
        - Tc1/Mariner | 20000 bp
        - MITE | 1000 bp
        - Novosib | 3000 bp
    - Retrotransposon size parameters
        - Gypsy | 20000 bp
        - Copia | 20000 bp
        - LINEs | 10000 bp
        - SINEs | 600 bp
        - ERV | 8000 bp
8. After size filtering, the set of TE annotations was considered complete and used for the subsequent TE tracking procedure.

Tracking the Movement of Transposable Elements

In order to locate and determine movement of TEs, we determined the overlap of TE annotations with SNPs in the CB genome relative to the N2 genome to find unique and trackable TE copies. Then, alternative sequences were generated by substituting in the alternative allele of a SNP into the N2 TE sequence. We then searched through the CB TEUlt annotations for the alternative sequences. This section of the tracking process was conducted in the TRACKING jupyter notebook.

Files Needed

- [ ] TRACKING.ipynb
- [ ] Whole genome fastas from TEUlt (sequence.fasta) - these will need to be renamed if working with multiple genomes!
- [ ] chr_lengths.txt
- [ ] final TE annotations in .gff3 format (either from TEUlt or post-size filtering)

- [ ] bedtools intersect outfile (see Tracking Step #1)
- [ ] SNPs .vcf

Tracking Steps

1. Run the bedtools intersect tool (link here) to analyze the overlap between TE annotations and SNPs.
    - We chose to run this tool with both the -wa and -wb options so that both the original TE annotations and SNP annotations were written to the outfile, so that alternate sequences could be generated later on.
    - example bedtools intersect cmd: `bedtools intersect -wa -wb -a N2_filter.gff3 -b "CB.aligned.to.N2.SNPs.final.seqnames.vcf" > N2TEs_CBSNPs_intersect.txt`
    - this command is able to be run in the TRACKING jupyter notebook using a magic command to run bash commands in a jupyter cell
2. Import the bedtools intersect out file, and the other needed files listed above into the TRACKING jupyter notebook
3. Assign SeqIO sequence records to each TE based on the N2 chromosome it is located on so that the TE sequence can be extracted later
4. Generate alternate sequences expected to see in the CB genome by replacing the reference allele with the alternate allele using the overlapped SNPs and TE annotation information
5. Search through the TEs annotated by TEUlt in the CB genome to determine if any of the unique TE copies are present in the annotations
6. If a single unique TE copy is present more than once in the CB genome, split up the list of CB TE IDs into individual lines in the output files
7. run the `create_tsv_file` command as below to generate the .tsv file needed for determining whether the detected TE movement is due to transposition or caused by alignment differences
    - `create_tsv_file(matching_df, False, True, True)`

Determining "Real" Movement of Transposable Elements

Next, the locations of the unique TE copies in the CB genome were adjusted for alignment position using alignment files in the shatter format, to determine if they had actually transposed, or if locational discrepancies were caused by alignment differences.

Files Needed

- [ ] DETERMINE-ACTUAL-MOVEMENT.ipynb
- [ ] shatter alignment files
- [ ] matching TEs .tsv file from TRACKING notebook

Alignment Adjustment Steps

1. Import the shatter alignment files and matching_TEs.tsv file generated in Tracking Step #7 above
2. Compare TE start positions with the alignment adjusted start positions from the shatter files
3. Sort the TEs based on whether or not they actually moved, and whether movement, if any, was interchromosomal or intrachromosomal

4. Create dataframe of only TEs that actually moved
5. Create bedfile and RIdeogram compatible .tsv file of the TEs that actually moved for visualization

Other Important Notes

- The complete edited source code, along with .md files explaining changes made to the code to integrate LTR Retriever and avoid an error during TE clustering is in the /source_code folder
-