# Repeat Library Construction-Advanced

This page describes an advanced method of repetitive element identification and classification. A basic understanding of repetitive elements is assumed. With this method, MITEs (miniature inverted transposable elements) and LTR (Long terminal repeat) elements, are first searched with structural approaches (MITE-hunter and LTRharvest). Numerically, MITEs represent the most abundant transposable elements while LTR elements occupy most genomic mass in plants. The structural approaches are followed by a de novo approach where most repetitive sequences are collected. Basic instructions for generating a species specific repeat library suitable for repeat masking prior to protein coding gene annotation with MAKER can be found here Repeat Library Construction--Basic.

*Content contributed by Ning Jiang (http://www.hrt.msu.edu/ning-jiang/), Megan Bowman (http://bioinformaticsandbiostaticscore.vai.org/team/), and Kevin Childs (https://plantbiology.natsci.msu.edu/directory/kevin-childs/).*

A combination of structural-based and homology-based approach is used to maximize the opportunity for repeat collection.

This protocol requires BioPerl (http://www.bioperl.org/), MUSCLE (http://www.drive5.com/muscle/), RepeatMasker (http://www.repeatmasker.org/), Hmmer (http://hmmer.janelia.org/) and NCBI-BLAST (ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/). Please download the programs and modify your PATH wherever relevant.

## Contents



# 1. MITEs (miniature inverted repeat transposable elements)

- MITEs are collected using MITE-Hunter (http://target.iplantcollaborative.org/mite_hunter.html) with all default parameters.
- Among the output files, “Step8_*.fa” and “Step8_singlet.fa” are combined as the putative MITE sequences. This will be named as MITE.lib.

# 2. LTR (long terminal repeat) retrotransposons

- In the plant genomes characterized so far, LTR retrotransposons represent the largest genomic mass among all repeats. As a result, it is very important to collect this type of element with high confidence.
- Here, the elements are collected using LTRharvest (http://www.genometools.org/tools/gt_ltrharvest.html) and filtered by LTRdigest (http://www.genometools.org/tools/gt_ltrdigest.html) and other custom programs. Both LTRharvest and LTRdigest are in a package called GenomeTools (https://www.computer.org/csdl/trans/tb/2013/03/ttb2013030645-abs.html).

A fully automated package for building a library for LTR elements is expected to be available with next update.

### 2.1. Collection of relatively recent LTR retrotranposons

The rationale for initial collection of recent LTR elements is that these elements contain less nested insertions and mutations as well as recombination, thus the chance for false positives is relatively low.

#### *2.1.1. Collection of candidate elements with LTRs that are 99% or more in similarity using LTRharvest*

**Note: FASTA sequence IDs (sequence names) must NOT have "_" or "." in them for this pipeline to function properly. Please format accordingly prior to running LTR harvest.**

Commands:

```
DIR1/gt suffixerator -db seqfile -indexname seqfileindex -tis -suf -lcp -des -ssp –dna
```

```
DIR1/gt ltrharvest -index seqfileindex -out seqfile.out99 -outinner seqfile.outinner99 -gff3 seqfile.gff99 -minlenltr 100 \
-maxlenltr 6000 -mindistltr 1500 -maxdistltr 25000 -mintsd 5 -maxtsd 5 -motif tgca -similar 99 -vic 10  > seqfile.result99
```

Please note all the content in this command should be in one row although it appears in two rows here. Please leave out the "\" at the end of first row if you copy it. This applies to all the long commands below.

DIR1= path where gt is located

The current protocol is developed using GenomeTools-1.5.5.

seqfile = file containing genomic sequence in fasta format. Designate the entire path of this file or ensure that it is in the current working directory.

These two commands will enable the collection of sequences with following features:

- Two terminal repeats which are >= 99% similar, ranging from 100 bp tp 6000 bp;
- The two terminal repeats end with "TG…CA";
- The size of entire element ranges from 1.5 kb to 25 kb;
- The element must be flanked by a 5bp TSD (target site duplication).
- The TSD is within 10 bp from the end of the element.

#### *2.1.2. Using LTRdigest to find elements with PPT (poly purine tract) or PBS (primer binding site)*

Commands:

```
DIR1/gt gff3 -sort seqfile.gff99 > seqfile.gff99.sort
```

```
DIR1/gt ltrdigest -trnas ~/bin/LTR/eukaryotic-tRNAs.fa seqfile.gff99.sort seqfileindex > seqfile.gff99.dgt
```

The "eukaryotic-tRNAs.fa" can be downloaded from Genomic tRNA database (http://gtrnadb.ucsc.edu/). The output "seqfile.gff99.dgt" provides the location of PPT or PBS in each element, if any. The CRL_Step1.pl script processes the output of LTRdigest so that only elements with PPT or PBS are selected. In addition, at least 50% of the PPT or PBS sequences are located in the internal regions and the distance between LTR and PPT or PBS should be no more than 20 bp. This will ensure the boundary of the LTR is correct.

DIR_CRL = path where CRL scripts and other custom scripts are located

If any script is not in executable format, use "chmod +x" to make it executable.

Make sure Bioperl is active before running CRL scripts. For any input file, if it is not in the working directory, please move it to the working directory or specify the path.

Command:

```
perl DIR_CRL/CRL_Step1.pl  --gff <path to seqfile.gff.dgt file>
```

Example:

```
perl  DIR_CRL/CRL_Step1.pl --gff  seqfile.gff99.dgt
```

Output: CRL_Step1_Passed_Elements.txt

CRL_Step1_Passed_Elements.txt is a text file with sequence IDs:

Example:

214_200514

739_87362

96_405925

80_527347

### *2.1.3. Further filtering of the candidate elements*

Even after the above procedures, considerable amounts of false positive elements are still detected. The three most important sources of false positives are:

- Tandem local repeats such as centromeric repeats.
- Local gene clusters derived from recent gene duplications
- Two other transposable elements located in adjacent regions.

A common feature of these sequences is that the alignment between the two “LTRs” often extends beyond the boundary of “LTR”. So we retrieve the flanking sequences of the two LTRs. If the flank sequences are also alignable, the candidate element will be excluded.

The CRL_Step2.pl script removes element sequences containing significant sequencing gaps (more than 50 Ns), then 50bp flanking sequences on both sides of the remaining LTRs are retrieved.

Command:

```
perl DIR_CRL/CRL_Step2.pl --step1 <path to CRL_Step1_Passed_Elements.txt > --repeatfile <path to seqfile.out file>  \
--resultfile <path to seqfile.result file> --sequencefile <path to seqfile> --removed_repeats  <CRL_Step2_Passed_Elements.fasta>
```

The “seqfile.out” and “seqfile.result” are from 2.1.1. For example, seqfile.out99 and seqfile.result99.

Example:

```
perl DIR_CRL/CRL_Step2.pl --step1 CRL_Step1_Passed_Elements.txt  --repeatfile seqfile.out99 --resultfile seqfile.result99 \
--sequencefile seqfile --removed_repeats   CRL_Step2_Passed_Elements.fasta
```

Outputs:

CRL_Step2_Passed_Elements.fasta

Repeat_*.fasta files

The CRL_Step2_Passed_Elements.fasta file is a FASTA file containing the element sequences without significant sequencing gaps. The Repeat_*.fasta files are small sequence files containing the 50bp flanking upstream or downstream sequence.

Example:

>chr04002_1_718645_lLTR50bp_upstream_24

CCTACCACTAGCGTCAACCTCTCTTATAACTTGCACTGTGGTCGGCCATT

>chr04002_1_718645_rLTR50bp_upstream_24

CAGTCCTTCCGAATCTCGGGGTCGAGATTCCGTTTAAGGGAGGTAGGTTT

Move the Repeat_*.fasta files and CRL_Step2_Passed_Elements.fasta into a new directory called fasta_files. The CRL_Step3 script needs to be run in the fasta_files directory.

Commands:

```
mkdir  fasta_files
mv  Repeat_*.fasta   fasta_files
mv  CRL_Step2_Passed_Elements.fasta  fasta_files
cd  fasta_files
```

Next, the CRL_Step3.pl script aligns the Repeat_*.fasta files with default parameters. Specifically, the upstream sequence of 5' LTR was aligned with that of 3' LTR of each candidate. So were the downstream sequences. The resulting output (flankingseqfile.afa) of MUSCLE contains the two sequences in fasta format where gaps are indicated by "-". Comparison of each corresponding position of the two sequences would reveal, among all positions of the two sequences, the number of positions associated with identical nucleotides, different nucleotide, and indels. Based on the above comparison, sequence identity of the two sequences was calculated, which equals number of identical positions divided by alignable positions. Alignable positions were derived from total positions minus gaps or indels. If the flanking sequences on either side is alignable (25 identical bps, or half of the total nucleotides, with at least 60% identity excluding gaps), the element is excluded.

Here is a sample output from flankingseqfile.afa: For simplicity, the alignment is short but the principle remains the same.

```
>sequpstrean5LTR
AGCACGAGAAAACAA----
>sequpstrean3LTR
----AGTGAAAACAAAAGG
```

With this example, there is a total 15 nt in each sequence. The total number of alignment position is 19 (11 alignable positions with 8 nt gaps). Among the 11 alignable positions, two nucleotides are different and 9 are identical. So the identity is 9/11 = 82% which is > 60%. Since more than half of the nucleotides are identical and the identity is > 60%, the two sequences are considered alignable and this element is considered a false positive.

Note: the pidentity and seq_c parameters must be integers.

Make sure MUSCLE is active before running CRL_Step3.pl.

Command:

```
perl DIR_CRL/CRL_Step3.pl --directory <DIR/fasta_files> --step2 <CRL_Step2_Passed Elements file> --pidentity <percent identity> \
--seq_c <number of identical nucleotides>
```

Example:

```
perl DIR_CRL/CRL_Step3.pl --directory  /home/jiang/LTR/fasta_files --step2 CRL_Step2_Passed_Elements.fasta --pidentity 60 \
--seq_c 25
```

Output: CRL_Step3_Passed_Elements.fasta

The CRL_Step3_Passed_Elements.fasta file is a FASTA sequence file containing element sequences that have passed the percent identity and number of identical nucleotides thresholds. This file will be located in the fasta_files directory.

Move CRL_Step3_Passed_Elements.fasta to the parental directory before proceeding to next step.

Commands:

```
mv  CRL_Step3_Passed_Elements.fasta  ..
cd  ..
```

#### *2.1.4. Identify elements with nested insertions*

- Retrotranposons are frequently nested with each other or inserted by other elements. If left unidentified, it will cause misclassification and other complications. To detect those elements, LTR sequences from candidate elements retained after steps in 2.1.3 are used to mask the putative internal regions. If LTR sequences are detected in the internal regions, it is considered as elements nested with other insertions. ******
- If internal regions of elements match sequences in MITE.lib (see 1.), they are also considered as elements with nested insertions. ******
- The internal regions of elements are also used to search against a transposase database of DNA transposons. If the internal sequence has significant matches with any DNA transposase, it is considered as an element containing nested insertions. ******

Command:

```
perl DIR_CRL/ltr_library.pl --resultfile <seqfile.result file> --step3 <CRL_Step3_Passed_Elements.fasta> --sequencefile <seqfile>
```

Example:

```
perl DIR_CRL/ltr_library.pl --resultfile seqfile.result99  --step3 CRL_Step3_Passed_Elements.fasta --sequencefile seqfile
```

Output: lLTR_Only.lib

The lLTR_Only.lib output is a FASTA file containing the sequence of the left (5'end) LTR sequence.

Combine lLTR_Only.lib and MITE.lib to form a library to mask the internal regions of putative LTR elements

Command:

```
cat lLTR_Only.lib  MITE.lib  > repeats_to_mask_LTR99.fasta
```

RepeatMasker Command:

```
DIR_RM1/RepeatMasker -lib repeats_to_mask_LTR99.fasta   -nolow  -dir .  seqfile.outinner99
```

DIR_RM1 = path where RepeatMasker is located

-dir . = will write to the current directory, please note there is space between “-dir” and “.”

-nolow = ignor simiple repeats (because in this step nested insertion is the focus)

Output: seqfile.outinner99.masked

The seqfile.outinner99.masked file contains the seqfile.outinner99 file with sequences that match MITEs or LTRs masked with Ns.

Elements with nested insertion will be removed.

Command:

```
perl DIR_CRL/cleanRM.pl  seqfile.outinner.out  seqfile.outinner.masked  >  seqfile.outinner.unmasked
```

Example:

```
perl DIR_CRL/cleanRM.pl  seqfile.outinner99.out  seqfile.outinner99.masked  >  seqfile.outinner99.unmasked
```

Output: seqfile.outinner99.unmasked

If the internal region of the putative candidate element is very short, it is likely to be a false positive. These elements will be excluded.

Command:

```
perl DIR_CRL/rmshortinner.pl  seqfile.outinner99.unmasked  minimum_length_inner (suggested value 50bp) > seqfile.outinner99.clean
```

Example:

```
perl DIR_CRL/rmshortinner.pl  seqfile.outinner99.unmasked  50 > seqfile.outinner99.clean
```

Output: seqfile.outinner99.clean

Next, use BLASTX and the script outinner_blastx_parse.pl to identify any additional outinner sequence with nested insertions of putative autonomous DNA transposons and eliminate them from subsequent steps.

Commands for BLASTX:

```
DIR_BLAST/bin/makeblastdb -in DIR2/Tpases020812DNA  -dbtype prot
```

“Tpases020812DNA” represents the DNA transposase protein sequence file

DIR2 is the directory where Tpases020812DNA is located

```
DIR_BLAST/bin/blastx  -query  seqfile.outinner99.clean -db DIR2/Tpases020812DNA  -evalue 1e-10 -num_descriptions 10 \
-out seqfile.outinner99.clean_blastx.out.txt
```

Command for processing BLASTX result:

```
perl  DIR_CRL/outinner_blastx_parse.pl --blastx <blastx output file> --outinner <full path of seqfile.outinner sequence>
```

Example:

```
perl  DIR_CRL/outinner_blastx_parse.pl --blastx  seqfile.outinner99.clean_blastx.out.txt  --outinner  seqfile.outinner99
```

Output: passed_outinner_sequence.fasta

### *2.1.5 Building examplars*

- LTR elements are often very repetitive in the genome, and it will be computationally challenging to retain all individual elements in the repeat library. To reduce the redundancy, representative sequences (examplars) are chosen. ******

The CRL_Step4.pl script formats the passed outinner sequence from last step and creates the LTR sequence file used for the all vs all BLASTN search, using NCBI-BLAST.

```
perl DIR_CRL/CRL_Step4.pl --step3 <CRL_Step3_Passed_Elements.fasta> --resultfile <seqfile.result file> \
--innerfile <passed_outinner_sequence.fasta > --sequencefile <seqfile>
```

Example:

```
perl DIR_CRL/CRL_Step4.pl –step3 CRL_Step3_Passed_Elements.fasta --resultfile seqfile.result99 \
--innerfile passed_outinner_sequence.fasta  --sequencefile seqfile
```

Outputs:

lLTRs_Seq_For_BLAST.fasta

Inner_Seq_For_BLAST.fasta

Command for All vs. All BLASTN For lLTR_Seq_For_BLAST.fasta

```
DIR_BLAST/bin/makeblastdb -in lLTRs_Seq_For_BLAST.fasta  -dbtype nucl
```

```
DIR_BLAST/bin/blastn -query lLTRs_Seq_For_BLAST.fasta -db lLTRs_Seq_For_BLAST.fasta -evalue 1e-10 -num_descriptions 1000  \
-out lLTRs_Seq_For_BLAST.fasta.out
```

For Inner_Seq_For_BLAST.fasta

```
DIR_BLAST/bin/makeblastdb -in  Inner_Seq_For_BLAST.fasta  -dbtype nucl
```

```
DIR_BLAST/bin/blastn -query Inner_Seq_For_BLAST.fasta -db Inner_Seq_For_BLAST.fasta  -evalue 1e-10 -num_descriptions 1000 \
-out  Inner_Seq_For_BLAST.fasta.out
```

DIR_BLAST = path where NCBI BLAST program is located

The current protocol is developed using ncbi-blast-2.4.0+

Next, the CRL_Step5.pl script processes the BLASTN results as follows:

The element with the most matches (cutoff at 80% identity in 90% of the element length) is considered as the first exemplar. Thereafter, this element and its matches are excluded from the group and a second round BLASTN search is conducted with the remainder of the elements, leading to the generation of the second exemplar. This process is repeated until all elements were excluded.

Since the internal sequences of LTRs are usually more conserved than LTR sequences, selection of exemplars are first conducted with the internal sequences of the elements without nested insertions. Thereafter the LTR sequences correspond to the exemplars of internal regions are also chosen. The sequences of LTR exemplars are then used to mask all LTR sequences of all candidate elements; those sequences that are masked (cutoff 80% identity in 90% of the sequence length) are excluded. The remainder of LTR sequences is used to generate a new pool of exemplar sequences.

All the exemplar sequences generated are combined together and called LTR99.lib.

Command:

```
perl DIR_CRL/CRL_Step5.pl --LTR_blast <path of the LTR blast result File> --inner_blast <path of the inner blast result File> \
--step3 <CRL_Step3_Passed_Elements.fasta> --final <LTR99.lib> --pcoverage <percent coverage> --pidentity <percent identity>
```

If not set by the user, the defaults for these scripts are set to 90% coverage and 80% identity.

Example:

```
perl DIR_CRL/CRL_Step5.pl --LTR_blast lLTRs_Seq_For_BLAST.fasta.out --inner_blast Inner_Seq_For_BLAST.fasta.out  \
--step3 CRL_Step3_Passed_Elements.fasta --final LTR99.lib --pcoverage 90 --pidentity 80
```

Outputs: LTR99.lib

The LTR99.lib output file is a FASTA file containing the exemplar sequences.

### 2.2. Collection of relatively old LTR retrotransposons

Collection of relatively old LTRs is enabled by reducing the similarity between LTRs to 85% (default value of LTRharvest) and not associated with terminal sequence motif.

Commands:

```
DIR1/gt suffixerator -db seqfile -indexname seqfileindex -tis -suf -lcp -des -ssp –dna
```

You may skip the above step if you already conducted this step in 2.1.

```
DIR1/gt ltrharvest -index seqfileindex -out seqfile.out85 -outinner seqfile.outinner85 -gff3 seqfile.gff85 -minlenltr 100 \
-maxlenltr 6000 -mindistltr 1500 -maxdistltr 25000 -mintsd 5 -maxtsd 5 -vic 10  > seqfile.result85
```

- Since the terminal sequence motif is not specified, only elements with terminal sequences with patterns that are previously reported are retained.
- Procedures described in 2.1.2 are conducted to identify elements with PPT or PBS.

The list of elements is processed through the procedures described in 2.1.3 to 2.1.5 using scripts CRL_Step2 to CRL_Step5 and the resulting exemplar sequences are called LTR85.lib.

Prior to this process, remove or rename all the files associated with 2.1 to avoid confusion or misuse.

Since this set of elements contain elements collected in 2.1. (LTR99.lib) or elements highly similar to them, the examplar sequences are masked by LTR99.lib and all elements that are significantly masked (cutoff at 80% identity in 90% of the element length) are excluded.

```
DIR_RM/RepeatMasker -lib LTR99.lib -dir . LTR85.lib
```

Output: LTR85.lib.masked

The removed_masked_sequence.pl script will remove the LTR99.lib sequences masked from the LTR85.lib dataset.

Command:

```
perl DIR_CRL/remove_masked_sequence.pl --masked_elements <LTR85.lib.masked> \
--outfile <path to final LTR library you create from LTR85.lib>
```

Example:

```
perl DIR_CRL/remove_masked_sequence.pl  --masked_elements  LTR85.lib.masked  --outfile  FinalLTR85.lib
```

Hereafter this library will be referred as FinalLTR85.lib

Combine LTR99.lib, FinalLTR85.lib to obtain allLTR.lib

Command:

```
cat LTR99.lib FinalLTR85.lib > allLTR.lib
```

# 3. Collecting repetitive sequences by RepeatModeler (http://www.repeatmasker.org/RepeatModeler.html)

The genomic sequence is masked by (allLTR.lib + MITE.lib = allMITE_LTR.lib).

Commands:

```
cat allLTR.lib MITE.lib > allMITE_LTR.lib
```

```
DIR_RM/RepeatMasker -lib allMITE_LTR.lib -dir . seqfile
```

Output: seqfile.masked

If the seqfile is very large, it would take forever to mask it. In this case, you may split it into small files (e.g. 50 Mb or smaller) to conduct RepeatMasker and combine the masked files together.

Use the following script to remove the masked elements:

```
perl DIR_CRL/rmaskedpart.pl  seqfile.masked  50  >  umseqfile
```

“50” means in the output sequence file that are 50 bps in each row. It is feasible to set any numbers that you would like.

The remaining sequence is then used as the input for RepeatModeler:

```
DIR_RD/BuildDatabase -name umseqfiledb -engine ncbi <path to umseqfile>
```

```
nohup DIR_RD/RepeatModeler -database umseqfiledb >& umseqfile.out
```

Output: consensi.fa.classified

Please note the output files are located in the RepeatModeler folder (folder named with “RM_...”).

DIR_RD = path where RepeatModeler is located

Among the sequences generated by RepeatModeler, some are associated with identities and others are not. The script repeatmodeler_parse.pl separates elements without identities into repeatmodeler_unknowns and the others in repeatmodeler_identities.

```
perl DIR_CRL/repeatmodeler_parse.pl --fastafile <consensi.fa.classified> --unknowns <repeatmodeler_unknowns.fasta> \
--identities  <repeatmodeler_identities.fasta>
```

Example:

```
perl DIR_CRL/repeatmodeler_parse.pl --fastafile consensi.fa.classified --unknowns repeatmodeler_unknowns.fasta  \
--identities repeatmodeler_identities.fasta
```

Outputs:

repeatmodeler_unknowns.fasta

repeatmodeler_identities.fasta

Sequences in repeatmodeler_unknowns are searched against a transposase database and sequences matching transposase are considered as transposons belonging to the relevant superfamily and are incorporated into repeatmodeler_identities and excluded from repeatmodeler_unknowns. Once filtering is complete using the script transposon_blast_parse.pl the libraries ModelerUnknown.lib and ModelerID.lib are created.

```
DIR_BLAST/bin/makeblastdb -in DIR2/Tpases020812  -dbtype prot
```

“Tpases020812” represents the transposase protein sequence file which is located in DIR2

```
DIR_BLAST/bin/blastx -query repeatmodeler_unknowns.fasta -db DIR2/Tpases020812  -evalue 1e-10 -num_descriptions 10 \
-out modelerunknown_blast_results.txt
```

```
DIR_CRL/transposon_blast_parse.pl: --blastx <blastx output file> --modelerunknown <full path of the modeler unknown file>
```

Example:

```
perl DIR_CRL/transposon_blast_parse.pl --blastx modelerunknown_blast_results.txt --modelerunknown repeatmodeler_unknowns.fasta
```

Outputs:

identified_elements.txt

unknown_elements.txt

Rename/combine files from RepeatModeler using following commands:

```
mv  unknown_elements.txt  ModelerUnknown.lib
cat  identified_elements.txt  repeatmodeler_identities.fasta  > ModelerID.lib
```

# 4. Exclusion of gene fragments

All repeats collected so far are used to search against a plant protein database where proteins from transposons are excluded. Elements with significant hits to genes are removed, along with 50 bp upstream and downstream of the blast hit. Remaining sequence that is less than 50 bp is removed completely. Outputs from this script are elements with no significant blast hits to the protein database and the remaining sequence from elements with blast hits that is greater than 50 bp. A package (ProtExcluder (http://www.hrt.msu.edu/uploads/535/78637/ProtExcluder1.2.tar.gz) and manual (http://www.hrt.msu.edu/uploads/535/78637/ProtExcluder1.2Manual.docx)) for conducting this task is available.

```
DIR_BLAST/bin/blastx -query ModelerUnknown.lib -db alluniRefprexp070416  -evalue 1e-10 -num_descriptions 10 \
-out ModelerUnknown.lib_blast_results.txt
```

BLAST all libraries against the protein database.

ProtExcluder command Example:

```
DIR_PE/ProtExcluder.pl  -option  ModelerUnknown.lib_blast_results.txt  ModelerUnknown.lib
```

DIR_PE = the directory where ProtExcluder is located.

Output: ModelerUnknown.libnoProtFinal

Please see ProtExcluder manual for detailed instruction.

Run this on each of the repeat libraries. Alternatively, you may combine all the repeat libraries and only run it once.

After exclusion of putative gene fragments, MITE.lib, allLTR.lib, and ModelerID.lib are combined together to form KnownRepeats.lib.

KnownRepeats.lib was combined with Modelerunknown.lib to form allRepeats.lib.

It is conceivable that KnownRepeats.lib does not contain all repeats (repeats numbers are underestimated); allRepeats.lib may contain sequences from novel gene families (repeat numbers are overestimated).

CRL and other custom scripts are available here (http://www.hrt.msu.edu/uploads/535/78637/CRL_Scripts1.0.tar.gz).

All transposase protein database is available here (http://www.hrt.msu.edu/uploads/535/78637/Tpases020812.gz).

DNA transposase protein database is available here (http://www.hrt.msu.edu/uploads/535/78637/Tpases020812DNA.gz).

The transposase database is composed of transposase protein sequence from RepeatMasker package (RepeatMasker-open-3-3-0), and that from Kennedy et al. 2011 (http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-130,), in addition to transposase sequences identified in Jiang lab.

Plant protein database is available here (http://www.hrt.msu.edu/uploads/535/78637/alluniRefprexp070416.gz).

The plant protein database contains those from swissprot plant protein (ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/taxonomic_divisions/uniprot_sprot_plants.dat.gz, dated Dec. 14, 2011) and those from NCBI Refseq plants (ftp://ftp.ncbi.nih.gov/refseq/release/plant/plant.*.protein.faa.gz , *=1 to 12, dated Nov. 17, 2011). To reduce the annotation errors in the database, the protein sequences were filtered. First, the protein sequences were searched against NCBI EST database (TBLASTN, e=10-5), only sequences with a match were retained. Second, the remaining sequences were searched against the transposase database (BLASTP e=10-5) mentioned above, sequences with matches were excluded. Finally, the rice protein sequences were compared with verified transposons (such as Pack-MULEs) in the rice genome. If the protein sequence matched a transposon perfectly and was the only perfect match in the genome, the relevant protein sequence was excluded. Although elements such as Pack-MULEs contain true gene sequences, the annotation (the protein sequence in the database) often extends to non-gene sequences such as terminal inverted repeat or sub-terminal repeat, which are not true plant proteins and would cause great complications. As a result, it is essential to exclude them.

Reference for this page:

Campbell, M. S., Law, M., Holt, C., Stein, J. C., Moghe, G. D., Hunagel, D. E., Lei, J., Achawanantakun, R., Jiao, D., Lawrence, C. J., Ware, D., Shiu, S-H., Childs, K. L., Sun, Y., Jiang, N, Yandell, M. 2014. MAKER-P: a tool-kit for the rapid creation, management, and quality control of plant genome annotations. Plant Physiology 164 513-524.

---