MUSTv2

```
MITE Uncovering SysTem    version 2.4.001
--------------------(c) Fengfeng Zhou (FengfengZhou@gmail.com) 2013-12-28
------------------------------------------------------------------------------
./MUST_Pipe.pl <input.fasta> <output.MITE.dat> <DirTemp> [options]
Note:
If any options are provided, the user have to input the values for all the options.
Option list:
-<Thread_Num>             [10]
-<MIN_TIR_length>         [8]
-<MAX_TIR_length>         [50]
-<MIN_DR_length>          [2]
-<MAX_DR_length>          [30]
-<MIN_MITE_length>        [100]
-<MAX_MITE_length>        [600]
-<FIXED_FLANKING_length>  [50]
-<Mutation_Rate>          [0.80]
------------------------------------------------------------------------------
------------------------------------------------------------------------------
```

HelitronScanner

```
HelitronScanner v1.0
Software can be run in three different modes...
  1) scanHead (scan DNA sequence for start points of transposons)
     helitronscanner scanHead -g dna.fasta -bs 0 -o headresult.txt
  2) scanTail (scan DNA sequence for end points of transposons)
     helitronscanner scanTail -g dna.fasta -bs 0 -o tailresult.txt
  3) pairends (group start and end points to identify helitron annotations)
     helitronscanner pairends -hs headresult.txt -ts tailresult.txt -o helitrons_result.txt
(default head.lcvs and tail.lcvs from author are used)
```

MiteTracker

```
usage: MITETracker.py [-h] -g GENOME -j JOBNAME [-w WORKERS]
                      [--mite_max_len MITE_MAX_LEN] [--lcc LCC]
                      [--mite_min_len MITE_MIN_LEN]
                      [--align_min_len ALIGN_MIN_LEN]
                      [--tsd_min_len TSD_MIN_LEN] [--tsd_max_len TSD_MAX_LEN]
                      [--FSL FSL] [--min_copy_number MIN_COPY_NUMBER]
                      [--task TASK]
```

MiteFinderII

```
USAGE: MiteFinder  [-threshold DOUBLE] [-disable_mismatch] [-version] [-help] [-pattern_scoring STRING] [-
fragnment_length INTEGER] -input STRING [-output STRING]
DESCRIPTION: An application for detecting miniature inverted-repeat transposable elements on a genome-wide scale.
VERSION: MiteFinder 1.0.006
ALL MANDATORY AND OPTIONAL ARGUMENTS:
-threshold              Threshold of removing mite candidates with low-confidence score.
                        Default is 0.5.
-disable_mismatch       Logical. It can disable the detection of mismatch base pairs if 1,
                        otherwise 0. Default is 0.
-version                Show the current version.
-help                   Show help information.
-pattern_scoring        The path of a scoring file.
-fragnment_length       Length of fragnment. Default is 10000.
-input                  A mandatory option. The path of an input file.
-output                 The path of an output file. Default is ./default_output.fsa
```

SineFinder

```
  sine_finder [options] <fastafile_name>
  OPTIONS:
     -h     (help:) display this message.
     -d     (description:) display a short description of the program.
     -v     (version:) print version number.
     EVALUATION PARAMETERS:
     -T (seqwise|chunkwise)
            (run type:) the way how infiles are processed (default: seqwise).
            'seqwise': each sequence is loaded and searched for patterns.
               For large sequences 'chunkwise' is the better choice.
            'chunkwise': only fragments (chunks) of the sequence are
               loaded and processed. The size is defined by option
               -C. To ensure that matches do not get lost by sequence
               splitting an overlap should be specified (option -O).
     -t <integer>
            TSD mismatch tolerance (default:2).
     -w <integer>
           word size TSD seed starts search with (default:5).
     -p <integer>
            penalty for a nucleotide mismatch in TSD search (default: 1).
     -s <integer>
            TSD score cutoff (default:10).
     -o (F|R|FR)
            direction of TSD search, allowed orientation (default:F).
     (only chunkwise processing:)
     -C <integer>
            (chunksize:) size of each fragment loaded and processed
            individually. Use only when -T is set to chunkwise (default: 100000).
     -O <integer>
            (overlap:) overlap of fragments treated by chunkwise processing.
            Use only when -T is set to chunkwise (default:8000).
     OTHER OPTIONS:
     -f (fasta|csv|both)
          (file type:) file type result is written to (default:fasta).
     -V    (verbose:) display program call.
sine_finder [options] <fastafile_name>
use -h for a detailed description of command line arguments
DESCRIPTION:
sine_finder.py is a script for the analysis of sequence files for putative members of the SINE repeat family. The
program is written in Python (http://www.python.org/). Besides an installation of Python (at least version 2.5) there
are no other requirements. The script contains a basic MotifFinder class, which performs a pattern search by regular
expressions. A derived SINEFinder class extends the functionality to examine TSDs (target site duplications). Search
patterns can easily be modified. The sine_finder may be used in 3 ways:
- command line mode by calling with arguments
- interactive mode by calling without arguments
- as a module by including it in other scripts
Sequences to be analysed should be in FASTA format:
         - a record starts with an ID line having '>' as first character,
         - descriptive data may follow the ID delimited by spaces,
         - sequence might be wrapped and displayed over several lines,
         - multiple sequence entries are allowed,
         - empty lines between records are allowed.
For datasets with very long sequences the option 'chunk-wise' may be used. Then at once sequences are only read and
processed in segments of a specified length (chunks). To avoid loss of matches at split sites chunks can overlap (size
of overlap is definable, duplicate matches are removed).
STRATEGY
The script performs a search in two steps:
1. A pattern search. The pattern is defined as a regular expression, which is a textual description for a set of
strings. It consists of groups or subpatterns so that the result contains the sequence matches for these as well. As
example: the a-box pattern '[GA][CGA]TGG' may result in ACTGG or GATGG whereas a spacer pattern like '.{25,50}'
matches any sequence as long as it has a length between 25 and 50 bp.
2. A TSD cannot be found with a regular expression, so in the pattern search regions have to be defined where the TSDs
are expected. In a second step these matches were examined for TSDs. Each result of the pattern search produces two
TSD expectation regions which are compared by a fuzzy matching. Doing this a window of the size of MIN_WORDSIZE over
sequence 1 and a search for exact counterparts of these 'words' in sequence 2 is performed. Matches and their
coordinates are returned. These serve as seed for the following procedure where the given matches are extended to the
right and to the left until MISMATCH_TOLERANCE is reached. Terminal mismatches are clipped. (For additional
information on regular expressions and their syntax: http://docs.python.org/library/re.html)
```

TirVish

```
Usage: gt tirvish [option ...] -index INDEXNAME
Identify Terminal Inverted Repeat (TIR) elements,such as DNA transposons.

-index          specify the name of the enhanced suffix array index (mandatory)
                default: undefined
-seed           specify minimum seed length for exact repeats
                default: 20
-mintirlen      specify minimum length for each TIR
                default: 100
-maxtirlen      specify maximum length for each TIR
                default: 1000
-mintirdist     specify minimum distance of TIRs
                default: 500
-maxtirdist     specify maximum distance of TIRs
                default: 10000
-mat            specify matchscore for extension-alignment
                default: 2
-mis            specify mismatchscore for extension-alignment
                default: -2
-ins            specify insertionscore for extension-alignment
                default: -3
-del            specify deletionscore for extension-alignment
                default: -3
-xdrop          specify xdropbelowscore for extension-alignment
                default: 5
-similar        specify TIR similarity threshold in therange [1..100%]
                default: 85.00
-overlaps       specify no|best|longest|all
                default: best
-mintsd         specify minimum length for each TSD
                default: 2
-maxtsd         specify maximum length for each TSD
                default: 11
-vic            specify the number of nucleotides (to the left and to the right)
                that will be searched for TSDs around 5' and 3' boundary of
                predicted TIRs
                default: 60
-hmms           profile HMM models for domain detection (separate by spaces,
                finish with --) in HMMER3 format
                Omit this option to disable pHMM search.
-pdomevalcutoff global E-value cutoff for pHMM search
                default 1E-6
-pdomcutoff     model-specific score cutoff
                choose from TC (trusted cutoff) | GA (gathering cutoff) | NONE
                (no cutoffs)
                default: GA
-refseqs        specify the name of the gene sequences to scan for inside
                candidates
                default: undefined
-seqids         use sequence descriptions instead of sequence numbers in GFF3
                output
                default: yes
-md5            add MD5 hashes to seqids in GFF3 output
                default: no
-help           display help for basic options and exit
-help+          display help for all options and exit
-version        display version information and exit

Report bugs to https://github.com/genometools/genometools/issues.
```

LtrHarvest

```
Usage: gt ltrharvest [option ...] -index <indexname>
Predict LTR retrotransposons.

-index      specify the name of the enhanced suffix array index (mandatory)
            default: undefined
-range      specify range in the input sequence(s) in which LTR pairs are
            searched
            default: 0 0
-seed       specify minimum seed length for exact repeats
            default: 30
-minlenltr  specify minimum length for each LTR
            default: 100
-maxlenltr  specify maximum length for each LTR
            default: 1000
-mindistltr specify minimum distance of LTR startpositions
            default: 1000
-maxdistltr specify maximum distance of LTR startpositions
            default: 15000
-similar    specify similaritythreshold in range [1..100%]
            default: 85.00
-mintsd     specify minimum length for each TSD
            default: 4
-maxtsd     specify maximum length for each TSD
            default: 20
-motif      specify 2 nucleotides startmotif + 2 nucleotides endmotif: ****
            default: undefined
-motifmis   specify maximum number of mismatches in motif [0,3]
            default: 4
-vic        specify the number of nucleotides (to the left and to the right)
            that will be searched for TSDs and/or motifs around 5' and 3'
            boundary of predicted LTR retrotransposons
            default: 60
-overlaps   specify no|best|all
            default: best
-xdrop      specify xdropbelowscore for extension-alignment
            default: 5
-mat        specify matchscore for extension-alignment
            default: 2
-mis        specify mismatchscore for extension-alignment
            default: -2
-ins        specify insertionscore for extension-alignment
            default: -3
-del        specify deletionscore for extension-alignment
            default: -3
-v          verbose mode
            default: no
-tabout     show 'old' tabular output instead of GFF3 on stdout
            default: yes
-seqids     use sequence descriptions instead of sequence numbers in GFF3 output
            default: no
-md5        add MD5 hashes to seqids in GFF3 output
            default: no
-longoutput additional motif/TSD output
            default: no
-out        specify FASTA outputfilename
            default: undefined
-outinner   specify FASTA outputfilename for inner regions
            default: undefined
-gff3       specify GFF3 outputfilename
            default: undefined
-help       display help for basic options and exit
-help+      display help for all options and exit
-version    display version information and exit

For detailed information, please refer to the manual of ltrharvest.
Report bugs to https://github.com/genometools/genometools/issues.
```

SineScan

```
SYNOPSIS: SINE_Scan_process.pl <options>

Options:
-g      string          Genomic file, the file's suffix must be 'fasta', 'fa' or 'fas'.
                        [Required]
-d      string          Directory which the pipeline uses. Intermidiate and final results files
                        will be put in this directory. [Required]
-o      string          Genome ID used in final output. An example(the author's advice):
                        genus_species_strain(or version). [Required]
-z      string          The directory where final results are written. [Default: ./]
-k      integer         The number of cpu numbers. [Default: 2]
-s      integer         Mode of running SINE_Scan. This pipeline has three modules, you can run
                        them all or seperately. To run the entire pipeline, type 123. To run
                        the first two modules, type 12. [default: 123]
-i      string          FASTA format file containing SINE candidates for verification. This
                        file is needed if you skip Module One to run Module Two.
                        [Default: NULL]
-a      string          FASTA format file containing SINE candidates for classification and
                        homology search. This file is needed if you skip Module One and Two to
                        run Module Three directly. [Default: NULL]
-n      integer         Minimum copy number of SINEs in genomic data. [Default: 5]
-F      integer         Length of sequences flanking to SINE ends. This region will be anlyzed
                        to determine SINE insertion events. [Default: 60]
-E      integer         Length of SINE ends used to determine SINE insertion events.
                        [Default: 25]
-c      float           Minimum identity of SINEs within a family. This value should be >= 0.8.
                        [Default: 0.8]
-p      float           Minimum proportion of matched region of two SINEs within a family.
                        [Default: 0.8]
-b      float           Identity percentage of a shorter sequence of two sequences for blast
                        filter. [default: 0.8]
-t      float           Minimum identity to define a SINE matching a RNA well. [Default: 0.6]
-r      integer         Minimum percentage of a RNA matching a SINE, which defines a SINE
                        matching a RNA well. [default: 60]
-D      float           Minimum differences between SAQ of TE ends and flanking regions.
                        [default: 0.3]
-S      float           Maximum SAQ of flanking regions. [default: 0.6]
-I      float           Minimum SAQ of TE ends. [default: 0.75]
-C      float           Minimum identity in one column to define a conserved site.
                        [Default: 0.8]
-l      integer         Maximum distance between two conserved sites. [Default: 4]
-L      integer         Minimum length of a highly conserved block. [Default: 10]
-f      float           Minimum percent of conserved sites in highly conserved block.
                        [Default: 0.6]
-w      integer         Continue to run the pipeline from the point it stops. The value of -w
                        only could be assigned 2 or 3, which means the pipeline begins from
                        Module Two or Three. Use this parameter only based on an aborted
                        previous run. If this paramter is used, no other parameter is needed.
-h      string          Show this help

An example : perl SINE_Scan_process.pl -g genome_file(fasta) -d workdir -s 123 -o species_name;
```

RepeatModeler

```
No database indicated
    RepeatModeler - Model repetitive DNA
SYNOPSIS
      RepeatModeler [-options] -database <XDF Database>
DESCRIPTION
    The options are:
    -h(elp)
        Detailed help
    -database
        The name of the sequence database to run an analysis on. This is the
        name that was provided to the BuildDatabase script using the "-name"
        option.
    -pa #
        Specify the number of parallel search jobs to run. RMBlast jobs will
        use 4 cores each and ABBlast jobs will use a single core each. i.e.
        on a machine with 12 cores and running with RMBlast you would use
        -pa 3 to fully utilize the machine.
    -recoverDir <Previous Output Directory>
        If a run fails in the middle of processing, it may be possible
        recover some results and continue where the previous run left off.
        Simply supply the output directory where the results of the failed
        run were saved and the program will attempt to recover and continue
        the run.
    -srand #
        Optionally set the seed of the random number generator to a known
        value before the batches are randomly selected ( using Fisher Yates
        Shuffling ). This is only useful if you need to reproduce the sample
        choice between runs. This should be an integer number.
    -LTRStruct
        Run the LTR structural discovery pipeline ( LTR_Harvest and
        LTR_retreiver ) and combine results with the RepeatScout/RECON
        pipeline. [optional]
    -genomeSampleSizeMax #
        Optionally change the maximum bp of the genome to sample in all
        rounds of RECON (default=243000000).
CONFIGURATION OVERRIDES
    -repeatmasker_dir <string>
        The path to the installation of RepeatMasker.
    -abblast_dir <string>
        The path to the installation of the ABBLAST sequence alignment program.
    -recon_dir <string>
        The path to the installation of the RECON de-novo repeatfinding program.
    -genometools_dir <string>
        The path to the installation of the GenomeTools package.
    -cdhit_dir <string>
        The path to the installation of the CD-Hit sequence clustering package.
    -ltr_retriever_dir <string>
        The path to the installation of the LTR_Retriever structural LTR analysis package.
    -rmblast_dir <string>
        The path to the installation of the RMBLAST sequence alignment program.
    -mafft_dir <string>
        The path to the installation of the MAFFT multiple alignment program.
    -trf_prgm <string>
        The full path including the name for the TRF program ( 4.0.9 or higher )
    -ninja_dir <string>
        The path to the installation of the Ninja phylogenetic analysis package.
    -rscout_dir <string>
        The path to the installation of the RepeatScout ( 1.0.6 or higher ) de-novo repeatfinding program.
SEE ALSO
        RepeatMasker, RMBlast
COPYRIGHT
     Copyright 2005-2019 Institute for Systems Biology
AUTHOR
     RepeatModeler:
       Robert Hubley <rhubley@systemsbiology.org>
       Arian Smit <asmit@systemsbiology.org>
     LTR Pipeline Extensions:
       Jullien Michelle Flynn jmf422@cornell.edu
```

RepeatMasker

```
RepeatMasker version 4.1.1
#########################################################################
RepeatMasker
Developed by Arian Smit and Robert Hubley
Please refer to: Smit, AFA, Hubley, R. & Green, P "RepeatMasker" at
http://www.repeatmasker.org
#########################################################################

RepeatMasker is a program that screens DNA sequences for interspersed
repeats and low complexity DNA sequences. The output of the program is
a detailed annotation of the repeats that are present in the query
sequence as well as a modified version of the query sequence in which
all the annotated repeats have been masked (default: replaced by
Ns). Sequence comparisons in RepeatMasker are performed by the program
cross_match, an efficient implementation of the Smith-Waterman-Gotoh
algorithm developed by Phil Green, or by WU-Blast developed by Warren
Gish.

This help file discusses the following topics:
0     Basic input and output
1     Options
1.1   Species and contamination check options
1.2   Options effecting which repeats get masked
1.3   Speed and search parameters
1.4   Output and formatting
1.5   ProcessRepeats options
2     Methodology and quality of output
2.1   Methodology
2.2   Scoring matrices
2.3   Databases
2.4   Sensitivity and speed
2.5   Selectivity and matches to coding sequences
2.6   Low complexity DNA and simple repeats
3     How to read the results
3.1   The annotation (.out) file
3.2   Alignments
3.3   The summary (.tbl) file
4     Applications
4.1   Use in database searches
4.2   Identification of DNA source and bacterial insertions
4.3   DateRepeats - Masking lineage-specific repeats for genomic alignments
4.4   Use with gene prediction programs and other applications
5       References

0  INPUT and OUTPUT
Input format:
Sequences have to be in the ' FASTA format':
>sequencename all kind of info
AGCGATCGCATCGAGCGCATTCGCATGGGG
>sequencename2 all kind of info
GCCCATGCGATCGAGCTTCGCTAGCATAGCGATCA
The program accepts FASTA format with errors and raw sequence files,
but does not work with other formats like GenBank, Staden, etc..
You can use RepeatMasker on a file containing multiple FASTA format
sequences and on multiple sequence files at the same time:
RepeatMasker *.fasta

This command will mask all files that end with .fasta in the current
directory and give separate reports for each file. Note that if you
have multiple small sequences it is considerably faster to run
RepeatMasker on one batch file than on many single sequence files. The
summary file will be more informative as well. However, analysis on
single files (when larger than 2 kb each) can be slightly more
accurate, since GC levels for each sequence will be calculated and
used to choose appropriate parameters.


Standard output:
RepeatMasker returns a .masked file containing the query sequence(s)
with all identified repeats and low complexity sequences masked. These
masked sequences are listed and annotated in the .out file. The masked
sequences are printed in the same order as they are in the submitted
file, whereas the sequences are presented alphabetically in the
annotation table. The .tbl file is a summary of the repeat content of
the analyzed sequence.
```

1 OPTIONS

1.1 Species options
-species <query species>   Indicate source species of query DNA
-lib [filename]   Allows the use of a custom library

contamination checking options
-is_only   only clips E coli insertion elements out of FASTA and .qual files
-is_clip   clips IS elements before analysis (default: IS only reported)
-no_is   skips bacterial insertion element check
-rodspec   only checks for rodent specific repeats (no RepeatMasker run)
-primspec   only checks for primate specific repeats (no RepeatMasker run)

For detailed explanation of the contamination detection options, see
"4.2 Identification of DNA source" below.

-spec
Interspersed repeats mostly are copies of transposable elements in
different states of erosion. Thus, dependent on the time of activity
of the source transposable element, interspersed repeats generally are
specific to a (clade of) species, and different redatabase
(http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html). In
principal, all unique clade names occurring in this database can be
used. Examples are:
-species "sus scrofa"
-species chimpanzee
-species arabidopsis
-species canidae
-species mammals

Capitalization is ignored, multiple words need to bound by apostrophes.
RepeatMasker builds one or more repeat consensus files the first time
a species/group has been chosen, or when a new database has been
downloaded. These will be written in a subdirectory of the Libraries
directory named after the date of the repeat database version and the
Latin name of the clade. For example, "-species monocotyledons"
creates the file
"..../RepeatMasker/Libraries/20040616/liliopsida/specieslib".
Currently, only for mammalian species multiple files are created,
bearing names like "shortcutlib" and "longlib", which the queries are
compared to sequentially.

The creation of these files takes some time (a few seconds sometimes),
but the next times RepeatMasker is run on the same species these
existing files will be used. When Wu-BlAST is used as the search
engeine (see 1.3), blastable libraries are built, again as a one time
event for each species.

After multiple database updates, the libraries could hog some space,
and you may consider deleting the older
"..../RepeatMasker/Libraries/<date>" directories.
The files contain all repeats of the RepeatMasker database that have
been found in the genome of the given species, or have been found in a
related species and are thought to predate the speciation time of the
two. For example, -species gorilla, will create a gorilla repeat file
that is almost as big as the human file, because almost all repeats in
human predate the 6-10 million years that separates us from the
gorilla, though none of the consensus sequences have been derived from
Gorilla DNA. A repeat file for hyraxes, for which order no repeats
have been submitted to the database yet, will contain all repeats
found in the human genome that are thought to be older than the origin
of most mammalian orders.

If a group of species is indicated, all repeats are included that are
found in any species belonging to this clade. Thus, "-species diptera"
leads to comparison against repeats found in the genomes of any
diptera species, currently primarily represented by fruitfly and
mosquitoes, and "-species murinae" compares the query to all known
murine repeats, including rat and mouse.

Not all "common" English names occur in the taxonomy database. For
example, "chimp", "squirrels", "grasses", or "carnivores" are not
present. The program will suggest functional names using Soundex, with
oftentimes unexpected results. Using Latin names is always safest.

famdb.py

famdb.py provides an interface to the Dfam and RepeatMasker libraries
in FamDB format, and allows you to see what species are covered and
how many repeats are assigned to them. For example:

`./famdb.py -i Libraries/Dfam.h5 lineage --ancestors --descendants human --format totals`

shows that there are 8 lineage-specific repeats and 1337 ancestral
(e.g. hominids, primates, mammals, and insertion artifacts).

You can run `./famdb.py --help` for more information about the available
commands.

These are the numbers and bp of repeat consensus sequences (excluding
simple repeats and RNAs) as of May 2009 for the best represented clades

species	# of consensi	total bp
All mammals combined	3081	4253979
Primates *	585	902148
Rodents *	606	931299
Carnivores *	130	158362
Perissodactyls *	130	220814
Ruminants *	112	130320
Bats *	131	112724
Marsupials	554	863923
Monotremes	102	159182
Birds	425	644078
Amphibia (mostly frog)	230	428828
Teleost fish	1140	2807233
Tunicates	134	368438
Sea urchins	211	560185
Flies	306	906766
Mosquitos	363	914943
Other insects	356	1080649
Nematodes	461	698036
Flatworms	209	641758
Cnidarians	911	3057775
Fungi	256	695278
Arabidopsis	544	1460558
Other dicot plants	742	2563646
Rice	575	1430176
Maize / corn	439	1566688
Other monocot plants	303	912057
Algae	186	533952

* Only order-specific elements; these genomes are also matched to 400+
consensus sequences for elements active before the origin of orders.

-lib

The majority of species are of course not yet covered in the repeat
databases and many are far from complete, but you may have your own
collection. At other times you may want to mask or study only a
particular type of repeat.

For these types of siutations, you can use the -lib option to
specify a custom library of sequences to be masked in the query. The
library file needs to contain sequences in FASTA format. Unless a full
path is given on the command line the file is assumed to be in the
same directory as the sequence file.

The recommended format for IDs in a custom library is:

>repeatname#class/subclass
or simply
>repeatname#class

In this format, the data will be processed (overlapping repeats are
merged etc), alternative output (.ace or .gff) can be created and an
overview .tbl file will be created. Classes that will be displayed in
the .tbl file are 'SINE', 'LINE', 'LTR', 'DNA', 'Satellite', anything
with 'RNA' in it, 'Simple_repeat', and 'Other' or 'Unknown' (the
latter defaults when class is missing). Subclasses are plentiful. They
are not all tabulated in the .tbl file or necessarily spelled
identically as in the repeat files, so check the RepeatMasker.embl
file for names that can be parsed into the .tbl file.

You can combine the repeats available in the RepeatMasker library
with a custom set of consensus sequences. To accomplish this
use the famdb.py tool:

`./famdb.py -i Libraries/RepeatMaskerLib.h5 families --format fasta_name --ancestors --descendants 'species name'`

The resulting sequences can be concatenated to your own set of sequences in a
new library file.

1.2 Masking options (options that determine what kind of repeats are masked)

```
-cutoff [number] sets cutoff score for masking repeats when using -lib
               (default cutoff 225)
-nolow         does not mask low complexity DNA or simple repeats
-l(ow)         same as nolow (historical)
-(no)int       only masks low complex/simple repeats (no interspersed repeats)
-alu           only masks Alus (and 7SLRNA, SVA and LTR5)(only for primate DNA)
-div [number]  masks only those repeats that are less than [number] percent
               diverged from the consensus sequence
```

-cutoff
When using a local library you may want to change the minimum score for reporting a match. The default is 225, lowering it below 200 will usually start to give you significant numbers of false matches, raising it to 250 will guarantee that all matches are real. Note that low complexity regions in otherwise complex repeat sequences in your library are most likely to give false matches.

-nolow / -l(ow)
With the option -nolow or -l(ow) only interspersed repeats are masked. By default simple tandem repeats and low complexity (polypurine, AT-rich) regions are masked besides the interspersed repeats. For database searches the default setting is recommended, but sometimes, e.g. when using the masked sequence to predict the presence of exons, it may be better to skip the low complexity masking.

-noint / -int

When using the -noint or -int option only low complexity DNA and simple repeats will be masked in the query sequence.
Inexact simple repeats may be spanned and hidden by an interspersed repeat annotation. In particular, most A-rich simple repeats derived from the poly A tails of SINEs and LINES are merged with the annotation of the SINE or LINE (i.e. you can't tell there is a simple repeat). Thus, if you're interested in finding the location of potentially polymorphic simple repeats, this option is recommended.

-norna
Because of their close similarity to SINEs and the abundance of some of their pseudogenes, RepeatMasker by default screens for matches to small pol III transcribed RNAs (mostly tRNAs and snRNAs). When you're interested in small RNA genes, you should use the -norna option that leaves these sequences unmasked, while still masking SINEs.

-alu
-div
You can limit the masking and annotation to (primate) Alu repeats with the -alu option and to a subset of less diverged (younger) repeats with the option -div. For example,

"RepeatMasker -div 20 -mus mysequence"

will mask only those rodent repeats and simple repeats that are less than 20% diverged from the consensus sequence and

"RepeatMasker -div 10 -alu mysequence"

will mask Alus that are less than 10% diverged from the Alu consensus sequences and no other repeats.

The -div option may be used to limit the masking to those repeats that are specific to a species group for use in subsequent comparison of orthologous genomic loci. Notice that a more sophisticated method to mask lineage-specific repeats (currently only in mammals) is now available with the script DateRepeats (4.3).

1.3 Options effecting speed and search parameters

```
-q             Quick search; 5-10% less sensitive, 3-4 times faster than default
-qq            Rush job; about 10% less sensitive,
-s             Slow search; 0-5% more sensitive, 2.5 times slower than default.
-pa(rallel) [number]
               Number of processors to use in parallel (only works for
                 batch files or sequences larger than 50 kb)
```

```
-engine [crossmatch|wublast|decypher]
               Select a non-default search engine to use.  If not specified
                 RepeatMasker will use the default configured at install time.
-w(ublast)     Use WU-blast, rather than cross_match as engine
                 **DEPRECATED** Use -engine [crossmatch|wublast|decypher] now.
-frag [number] Maximum sequence length masked without fragmenting
                 (default 40000).
-gc [number]   Use matrices calculated for 'number' percentage background
                  GC level.
-gccalc        Program calculates the GC content even for batch files/small
                  sequences.
-nocut         Skips the steps in which repeats are excised.
-noisy         Prints cross_match progress report to screen (defaults to
                 .stderr file)

-s -q -qq
RepeatMasker can be run at four different sensitivity/speed levels,
with the option -q providing quick (less sensitive) and -s slow
(sensitive) results compared to default. The option -qq has been added
for when you're in a frightful hurry. Each higher gear is about 2-3
times faster, and 90% as sensitive as the next lower gear. See "2.4
Sensitivity and Speed" below for details


-w(ublast)
  **DEPRECATED** See -engine.

-engine [crossmatch|wublast|decypher]
By default, RepeatMasker uses the search engine configured
during installation as the default.  To use the non-default
search engine you can specify it with the -engine parameter.

Before June 2004, the script MaskerAid (written by Joey Bedell, Ian
Korf and Warren Gish at the St Louis Washington University Genome
Center) was necessary to use WU-BLAST with RepeatMasker, but that
functionality is now built in. RepeatMasker includes a search engine
object that allows relatively straightforward integration of other
search engines. Currently only WU-BLAST has the flexibility to accept
all cross_match options.

For longer sequences, default RepeatMasker runs with WU-BLAST take
about as long as cross_match powered runs at -qq settings (see "2.4
Sensitivity and speed"). The speed settings have relatively little
effect on the speed when using WU-BLAST, with the fastest settings
1.25-1.75 as fast as the slowest settings, while the sensitivity
increases significantly.  Thus, I recommend to always run RepeatMasker
in sensitive (-s) or default mode when using WU-BLAST. I've made the
difference in parameters between sensitive and default settings larger
at -w settings, to make these speed options more meaningful and gain
more sensitivity (with little cost in speed).

Even with these more extreme parameters, the sensitivity can't quite
reach that of the sensitive settings using cross_match, but it comes
very close, and the huge difference in speed make this option
very attractive.

The output format with the -w option is identical to default and
scores are comparable, as the same complexity adjustment is applied.
The only difference is that, when using the wublast option, hyphens
in the sequence are retained (in default mode all non-letters were
deleted from the sequence).  WU-BLAST uses hyphens to indicate
insurmountable barriers and alignments will not span hyphens.


-pa(rallel)
For sequences over 50 kb long or files wit multiple sequences,
RepeatMasker can use multiple processors. When you type:

RepeatMasker -par 10 <file>

A batch file of sequences will run with up to 10 sequences at the
time, until all sequences are done, while a file with one large
sequence will analyze the sequence in up to 10 fragments at the same
time. The minimum fragment size is 25 or 33 kb, the maximum 66 kb (all
sequences over 100 kb are divided in 33-66 kb fragments). For the
batch files no minimum size exists. Thus,

If <file> contains:    RM runs in parallel:
one 60 kb sequence     two 30 kb fragments
one 400 kb sequence    ten 40 kb fragments
one 1 Mb sequence     ten 50 kb fragments, twice
ten 500 bp sequences   ten 500 bp sequences
```

two 500 kb sequences   ten 50 kb fragments, twice

Processing of the detected matches takes place after all batches or fragments have been cross-matched with the databases.
Beware that, generally, you have a limited number of processor IDs allotted. RepeatMasker uses 4 PIDs for each parallel job, so if you're allotted 64 user PIDs, you can 'only' run 16 fragments/batches in parallel.

-frag
Even when the -par option is not used, RepeatMasker transparently fragments sequences over 40 kb in fragments of equal sizes with 1 kb overlaps. Similarly, sequence batches containing more than 51 kb are subdivided in batches of 40 kb or less. The -frag option sets the maximum fragment and batch size

The only visible effect of the fragmentation is in the alignment files, where alignments at the edges of the fragments can be duplicated and/or truncated. The 1 kb overlap between fragments almost guarantees that there is no loss in sensitivity at the edges. Fragmentation initially was implemented to allow the size of sequences and sequence batches to be unlimited. Cross_match can be very memory intensive when SW alignments have to be performed in large matrices. This may happen with short minmatch and large bandwidth settings. Note that RepeatMasker should not croak when cross_match runs out of memory; it will redo the failed search with a higher word length or smaller bandwidth until it succeeds. However, this will lead to gradually less sensitive comparisons.

Fragmentation also can improve repeat detection when a genomic sequence contains large regions of DNA with significantly different GC levels (isochores), since sets of scoring matrices are chosen based on the GC level of a fragment.

-gc
-gccalc
Neutral mutation patterns differ significantly depending on the GC richness of a locus and we have calculated optimal scoring matrices for the alignment to consensus sequences in a range of background GC levels (see 2.2). Usually, RepeatMasker calculates the percentage of the sequence consisting of Gs and Cs and uses the appropriate matrices. However, the program defaults to using 'average' 43% GC matrices when the query is shorter than 2000 bp or a batch file is analyzed. This is because short sequences can diverge greatly from the GC level of the locus. For example, CpG islands and exons are more GC rich than the surrounding DNA, whereas a LINE-1 element can be more AT rich than the background. In a batch file, RepeatMasker analyses all sequences together with the same matrices. The percentage GC in all the sequences combined may be inappropriate for some sequence entries; using high GC level matrices in AT rich sequences (and vice versa) may result in false masking.

One can override this behavior in two ways:
With the option -gc you can set the GC level to a certain percentage:

RepeatMasker -gc 37 mybatchofsequences.fa

lets the program use matrices appropriate for 37% GC background. The batch could, for example, contain ESTs from a single locus with a known GC level.
Alternatively, the -gccalc option forces RepeatMasker to use the actual GC level of a short sequence or the average GC level of a batch of sequences. The latter sequences, for example, may be contigs in a sequencing project.

-nocut
The option -nocut skips a step in the default procedure for human and rodent queries, in which full-length younger insert are spliced out of the query to reconstruct a pre-insertion situation. RepeatMasker is generally more sensitive and efficient including the deletion step as it can unearth older repeats that were interrupted by these younger elements.

1.4 Output options

-a   shows the alignments in a .align output file; -ali(gnments) also works
-inv   alignments are presented in the orientation of the repeat (with option -a)

```
-cut    saves a sequence (in file.cut) from which full-length repeats are excised
        (temporarily disfunctional)
-small  returns complete .masked sequence in lower case
-xsmall returns repetitive regions in lowercase (rest capitals) rather than masked
-x      returns repetitive regions masked with Xs rather than Ns

-poly   reports simple repeats that may be polymorphic (in file.poly)
-ace    creates an additional output file in ACeDB format
-gff    creates an additional General Feature Finding format output
-u      creates an untouched annotation file besides the manipulated file
-xm     creates an additional output file in cross_match format (for parsing)

-fixed  creates an (old style) annotation file with fixed width columns
-no_id  leaves out final column with unique ID for each element
-e(xcln) calculates repeat densities (in .tbl) excluding runs of >25 Ns in query

-noisy  prints cross_match progress report to screen (defaults to .stderr file)


-a / -ali(gnments)
-inv
Alignments are saved in a .align file when using the option -a. They
are shown in the orientation of the query sequence, unless you use the
option -inv as well, which will return alignments in the orientation
of the repeats (see 3.2 Alignments).


-cut
The -cut option to RepeatMasker is not supported in this release.  It
will be rolled into a new annotation utility in the near future.  If
you need this functionality sooner please send an email to Robert
Hubley ( rhubley@systemsbiology.org ).  Thanks for your patience.

The option  made the program save a file "file.cut" which contains
an intermediate sequence in the masking progress. In this sequence all
full-length elements, young LINE-1 3' ends, and close to perfect simple
repeats were deleted.


-x
When -x is used the repeat sequences are replaced by Xs instead of
Ns. The latter allows one to distinguish the masked areas from
possibly existing ambiguous bases in the original sequence. However,
when running BLAST searches (and maybe other programs) Xs are deleted
out of the query and the returned BLAST matches will have position
numbers not necessarily corresponding to that of the original
sequence.


-xsmall
When the option -xsmall is used a sequence is returned in the .masked
file in which repeat regions are in lower case and non-repetitive
regions are in capitals.


-poly
You can get a list of potentially polymorphic microsatellites with the
option -poly. This is simply a subset of the list in .out, with
dimeric to tetrameric repeats less than 10 % diverged from perfection.


-xm
When using the -xm option an additional output file (.out.xm) is
created that contains the same information as the .out file (excluding
the low-complexity/simple DNA), but then in the original cross_match
format. This output is harder to read but there are programs that
require the exact cross_match output format.


-u
The script ProcessRepeats adjusts the original RepeatMasker output so
that the annotation more closely reflects reality. With the option -u
a .ori.out file is created that contains the original (but sorted)
cross_match summary lines.


-ace
With the -ace option the script creates an .ace file. This is merely a
suggestion. The columns in the table currently are:

Motif_homol <repeat-name> RepeatMasker(method) <percent divergence>
```

<start in query> <end in query> <orientation> <start in consensus>
<end in consensus>

-gff
The script creates a .gff file with the annotation in 'General Feature
Finding' format. See http://www.sanger.ac.uk/Software/GFF for
details. The current output follows a Sanger convention:

<seqname> RepeatMasker Similarity <start in query> <end in query>
<percent divergence> <orientation> . Target "Motif:<repeat-name>"
<start in consensus> <end in consensus>

In this line, 'RepeatMasker' becomes 'RepeatMasker_SINE' if the match
is against an Alu. I don't know why.

-fixed
Since April 1999 the column widths in the annotation table are
adjusted to the maximum length of any string occurring in a column;
this allows long sequence names to be spelled out completely.
Previously, a fixed column width table was returned, which can still
be obtained by using the -fixed option. Parsing should not be effected
by this change of default behavior, as the same number of columns with
the same formatted text are still separated by white space.

-no_id
Since September 2000 a column displaying a unique number (ID) for each
integrated element is printed by default. This used to be optional
(-id). Fragments of a single element, separated from each other by
subsequent insertions of other elements, deletions or recombinations,
carry the same number. This feature allows better interpretation of
the data and should greatly help proper graphical display of the
repeats.

The column follows all other columns, except for the (rare) indication
that an annotation overlaps another annotation (*). This change, which
was announced in the previous release, should not hinder most parsing
scripts. If it causes problems, the old format can be retrieved with
the option -no_id.

-excln
The percentages displayed in the .tbl file are calculated using a
total sequence length excluding runs of 25 Ns or more. This is useful
when analyzing draft sequences that are often concatenated contigs
separated by (sometimes very) long stretches of Ns. This option can
be used with ProcessRepeats as well. The number of Ns in long runs in
the query are apparent in the .tbl file, and you only need to run
ProcessRepeats with the option on the .cat file.

-noisy
RepeatMasker used to print the voluminous cross_match progress reports
to the screen. Since the Dec 1998 version this output is stored in a
.stderr file and a more informative much smaller progress report is
printed to the screen. The option -noisy allows one to see the
cross-match reports coming by on the screen (yeah).

1.5 ProcessRepeats options

When you have already run RepeatMasker and want to recreate the .out
or .tbl file, you only need to rerun ProcessRepeats on the .cat
file(s), which will take just a small fraction of the time required to
rerun RepeatMasker. Such a situation can occur when you've
accidentally deleted the .out or .tbl file or want additional or
differentially formatted output files. Note that alignment files
cannot be created unless RepeatMasker was run with the -a option and
that the original .tbl and .out file will be overwritten unless you
rename them.

```
ProcessRepeats -species mus -nolow -gff -excln myhumongousmousesequence.cat
```

Repeat matches are processed differently for different query species,
so the -species mus option is necessary. With the -nolow option, the
.out file will not contain information on simple repeats and low
complexity DNA anymore. The -gff option creates an additional output
file in GFF format, and the -excln option displays the density of

repeats in the .tbl file as a percentage of those bp that are not
contained in long stretches of Ns.

The options/flags for ProcessRepeats are:

-species <query species>   Identical as for the RepeatMasker script

-lib   skips most of processing, does not produce a .tbl file unless the
       custom library is in the >name#class format.

-nolow does not display simple repeats or low_complexity DNA in the annotation
-noint skips steps specific to interspersed repeats, saving lots of time
-u   creates an untouched annotation file besides the manipulated file
-xm   creates an additional output file in cross_match format (for parsing)
-ace   creates an additional output file in ACeDB format
-gff   creates an additional Gene Feature Finding format
-poly creates an output file listing only potentially polymorphic simple repeats
-no_id leaves out final column with unique number for each element (was default)
-fixed creates an (old style) annotation file with fixed width columns
-excln calculates repeat densities excluding long stretches of Ns in the query
-orf2   results in sometimes negative coordinates for L1 elements; all L1 subfamilies
       are aligned over the ORF2 region, sometimes improving interpretation of data
-a   shows the alignments in a .align output file

2 METHODOLOGY AND QUALITY OF OUTPUT

2.1 Methodology

RepeatMasker compares the query sequence against one or more files of
FASTA sequences. The sequences in the libraries provided with
RepeatMasker are consensus sequences derived from alignment of
multiple copies of interspersed or satellite repeats. For interspersed
repeats, a consensus tends to approach the sequence of the
transposable element from which the repeat is derived.

Both cross_match and WU-blast perform their Smith-Waterman (SW)
alignments by first identifying exact word matches and restricting the
alignment to a band or matrix surrounding this exact
match(es). Overlapping matrices are merged. The speed settings of
RepeatMasker are purely changes in the minimum word length from which
an alignment can be seeded and, in some cases, changes in the width of
the band. A wider bandwidth allows more gaps in the alignment and,
more importantly, increases the likelihood that neighboring matrices
overlap.

Cross_match does a low complexity adjustment of the raw SW score. When
WU-blast is used, the RepeatMasker script performs this adjustment. Low
complexity matches are the primary cause of false matches, and this
adjustment contributes significantly to the high selectivity of
RepeatMasker (see 2.5)

As a result of the existence of many related consensus sequences in
the database, usually multiple repeats match one region in the query
at the same time. Generally, cross_match and WU-blast report to the
script only those matches that are less than 80-90% overlapped by a
higher scoring match. This implies that, at first approximation, names
are assigned to repeats based on the highest SW score. Given
appropriate consensus sequences and alignment parameters, this is
intuitively correct as well. However, the scripts have a lot of code
to improve on this first approximation, primarily to deal with partial
matches.

The cut-off SW score above which matches are reported is empirically
derived (see '2.5 selectivity' below). Note that there is no cut-off
divergence level; reported matches can be less than 60% identical.

The alignments parameters -substitution matrices, and gap initiation
and extension penalties- are derived from data harbored in multiple
alignments of a special subset of interspersed repeats. The derived
matrices are theoretically optimal for a series of conditions (see
below). The gap penalties are sub-optimal, primarily because gap
lengths have a non-linear distribution and are poorly represented by a
single gap-extension penalty.

For primate, rodent and other mammalian DNA, the query is compared to
consecutive subsets of repeat libraries. For primates, perfect simple
repeats, full-length Alus, full-length short interspersed repeats, and
young L1 3' ends are first (and in that order) clipped from the

sequence to expose underlying older elements. Subsequently, the query is compared to most repeats, a set of ancient elements under especially sensitive settings, a large set of long retroviral sequences under faster settings (to save time), and AT-rich L1 3' ends that may have been discarded earlier as low complexity matches. Finally, simple repeats and low complexity regions are masked.

2.2 Scoring matrices

We have calculated statistically optimal scoring matrices for the alignment of neutrally diverging (non-selected) sequences in human DNA to their original sequence. These matrices have been in use since the May 1998 release. The matrices were derived from alignments of DNA transposon fossils to their consensus sequences. A series of different matrices are used dependent on the divergence level (14-25%) of the repeats and the background GC level (35-53%, neutral mutation patterns differ significantly in different isochores).

These matrices are (close to) optimal for human genomic sequences longer than 10 kb, for which length the GC level usually is representative of the isochore in which the sequence lives. However, the GC level of small fragments can diverge a lot from the surrounding (e.g. a fragment spanning a CpG island, a GC rich exon or an AT-rich LINE-1 element) and RepeatMasker defaults to using matrices derived for a 43% GC background when a sequence is shorter than 2000 bp or when a batch file is submitted. When the appropriate background GC level is known, this can be entered with the -gc option.

(Note that these matrices are an integral portion of RepeatMasker and are covered under the same restrictions as the scripts and databases as described in the signed software agreement).

2.3 Repeat databases

The RepeatMasker program are distributed with a copy of the Dfam database (www.dfam.org). Dfam is a small but growing "open" databases of Transposable Element seed alignments, profile Hidden Markov Models and consensus sequences.

RepeatMasker is also compatible with the RepBase database managed by the Genetic Information Research Institute and requires a license to use. Up until 2019 we maintained the "Repbase RepeatMasker Edition" libraries as co-editor of RepBase Update. For newer versions of RepBase users will need to use the sequences in FASTA format with RepeatMasker's "-lib" option.

2.4 Sensitivity and speed

The program can be run at four levels of sensitivity. The only difference between these settings is the minimum match or word length in the initial (not quite) hashing step of the cross_match program (see the cross_match/PHRAP documentation). For mammalian queries, he "slow" setting will find and mask 0-5% more repetitive DNA sequences than by default, whereas the "quick" settings miss 5-10%, and the "rush" (-qq) settings may miss 10-25% of the sequences masked by default. The alignments may extend more or be somewhat more accurate in the more sensitive settings as well.

Following are benchmark times for random 1 Mbp of sequences of a variety of different species run in parallel on 4 Pentium4 2.4Ghz processors with 3 GB RAM with June 2004 RepeatMasker databases. The percentage of the query masked is given in parentheses.

		cross_match			
Species	WUBlast (Def)	Rush	Quick	Default	Slow
Human	02:54 (39.26)	01:54 (33.91)	05:05 (36.85)	22:15 (39.92)	57:54 (40.58)
Human-reversed	01:09 (1.98)	01:05 (2.00)	03:39 (2.06)	18:44 (2.07)	53:37 (2.09)
Chimpanzee	03:00 (40.83)	01:50 (35.24)	04:45 (38.70)	20:22 (41.59)	53:14 (42.24)
Mouse	03:31 (54.02)	01:47 (48.65)	04:21 (51.74)	18:54 (54.15)	47:26 (55.18)
Rat	04:46 (66.07)	02:05 (62.07)	04:32 (63.84)	19:41 (65.97)	48:23 (67.20)
Dog	02:24 (34.62)	01:32 (29.15)	03:07 (32.44)	12:29 (35.09)	30:14 (35.69)
Arabidopsis	01:01 (3.02)	00:51 (2.95)	04:41 (3.00)	46:52 (3.12)	1:46:53 (3.13)
Ciona savigny	01:25 (15.64)	01:02 (13.12)	01:30 (14.45)	06:13 (15.90)	15:24 (16.30)
C. elegans	02:35 (22.63)	01:38 (20.84)	02:39 (22.52)	12:12 (23.21)	25:15 (23.59)
Drosophila	01:59 (47.21)	01:23 (43.08)	02:30 (45.60)	15:49 (47.51)	39:24 (48.38)

Chicken	00:42 (6.52)	00:35 (6.18)	00:58 (6.42)	04:59 (6.53)	11:48 (6.58)
Fugu	00:35 (5.89)	00:34 (5.40)	00:49 (5.70)	03:51 (5.89)	09:20 (6.05)

The human-reversed sequence is the "human" sequence reversed but not complemented. 2% of this sequence is (properly) masked as simple repeats or low complexity DNA.

Note that for many non-mammalian species the slower settings do not dramatically increase the percentage recognized as interspersed repeats. Most of the repeats in the databases for these species are relatively young and thus are easily detected. This particular 1Mbp Arabidopsis sequence is an extreme example, where at slow settings in almost two hours only 1800 bp more is masked than at rush settings in 51 seconds (the Arabidopsis database is large).

The speed is also dependent on the repeat content of the sequence. For human sequences, Alu rich sequences are analyzed fastest, LINE rich sequences somewhat slower, repeat poor regions slower still, and long satellite regions can take a while.

If you have several shorter sequences it is much faster to run RepeatMasker on a batch file (all sequences in one file). On above computer, in the rush mode (cross_match), a batch of 10 5 kb sequences is analyzed in 23 seconds, 20 5kb in 34 sec., etc.

The user time for larger sequences or sequence batches (50 kb and up) is linearly related to the length of the query due to the fragmentation of the query sequence.

The increase in speed by using multiple processors is dependent on the usage of the computer and the above-mentioned non-linear relationships of sequence length and processing time. However, under the right circumstances, using 2 processors can increase the speed close to twofold, because the most time-consuming processes are performed in parallel.

2.5 Selectivity and matches to coding sequences

The cutoff Smith-Waterman scores for masking interspersed repeats are conservative, since masking of one short potentially interesting region generally is more harmful than not masking a number of hard to find matches. If there are any false matches, they tend to have scores close to the cutoff, which is 225 for most repeats, 300 for the low-complexity LINE-1 search*, and 180 for the very old MIR, LINE2 and MER5 sequences.

* most LINE-1s are detected with a 225 cut-off, but in one step in RepeatMasker the low-complexity score adjustment is turned off to find ancient A-rich L1 elements.

With each release, we test for the occurrence of false matches in randomized and in inverted (but not complemented) DNA including a range of isochores from 36% to 54% GC. To retain seeds for Smith Waterman alignments, sequences are randomized at the 10 bp word level. Note that the inverted sequences retain the low complexity and simple repeat patterns of the original sequences. Even at sensitive settings, for which false matches are most likely, the 1998-2004 versions of RepeatMasker have reported no (false) matches at all to interspersed repeats in the randomized or inverted sequences. No simple repeats were reported in the randomized queries.

In a 1999 test, RepeatMasker returned only a single probably false match (71 bp) when analyzing a batch of 4440 coding regions in human mRNAs (7.2 Mb) at sensitive settings. The coding regions were collected from GenBank, based on annotations, filtered for the presence of complete ORFs and initiator methionines, and made more or less non-redundant. When each coding region was analyzed individually using the -gccalc option, 5 matches (414 bp, 0.006%) were falsely masked (156 bp at default speed, 76 bp at quick settings). In this analysis each sequence was analyzed with matrices chosen based on the actual GC level, even for very short sequences, while in the batch analysis of the coding regions the 'average' 43% GC matrices were used.

The 1998 and later versions of RepeatMasker show somewhat more false masking when a pre-1998 version of cross_match is used. These are primarily the result of improper assumptions of the background nucleotide frequency used in the scoring matrix calculation when

adjusting for the complexity of a match. Specifically, a very GC rich region in an AT-rich isochore (like an exon) may improperly match a GC rich repeat, since the scores for C/G matches are higher in the used scoring matrix than for AT matches (calculated for this AT rich background) whereas the old cross_match assumed that a 50% GC background in these calculations and equal scores for A/T and G/C matches have been given. The new version of cross_match reads the correct nucleotide background level from the matrix used.

2.6 Simple repeats and low complexity DNA

Low-complexity DNA

By default, along with the interspersed repeats, RepeatMasker masks low-complexity DNA. Simple repeats (micro-satellites) can originate at any site in the genome, and therefore have an interspersed character. Other low-complexity DNA, primarily poly-purine/ poly-pyrimidine stretches, or regions of extremely high AT or GC content will result in spurious matches in some database searches as well (especially in the ungapped BLASTN searches). For example, extremely AT-rich regions consistently will give very low probability matches to mitochondrial DNA in BLASTN searches. The settings are very stringent, and we think that few if any sequences informative in database searches are masked as low-complexity DNA. However, you can skip the low-complexity DNA masking using the option -nolow or -l(ow).

Under the current settings a 100 bp stretch of DNA is masked when it is >87% AT or >89% GC, a 30 bp stretch has to contain 29 A/T (or GC) nucleotides. The settings are slightly more stringent than the original settings, partly because the gapped BLAST programs are less sensitive to short regions of low complexity then the old gapless BLAST. In coding regions I have not yet found extensive regions (>10 bp) masked as low complexity DNA that would not be masked by the combined XNU and SEG filters routinely used in BLASTX.

Annotation of simple repeats

Although RepeatMasker does a good job in masking simple repeats to avoid spurious matches in database searches, it is not written to find and indicate all possibly polymorphic simple repeat sequences. Only di- to pentameric and some hexameric repeats are scanned for and simple repeats shorter than 20 bp are ignored. The -poly option prints out a separate list of simple repeats of < 10% divergence from a perfect repeat. However, even long perfect repeats may not be presented in this list; e.g. two perfect 40 bp long (CA)n repeats interrupted by 10 Ts are aligned in one piece and may be reported as having > 10% divergence from the consensus. Many perfect hexameric or longer unit repeats will be listed as more or less diverged smaller unit repeats and may not appear in the .polyout file.

Also note that, in the default output, simple repeats expanded from the poly A tails of Alus and LINE-1 are now included in the Alu or LINE-1 annotation. This cleans up the annotation a bit and lets the stand-alone poly A regions stand out (they may indicate the presence of a processed pseudogene). However, even perfect simple repeats in such tails will be hidden in the .out file.

A program optimized to quickly find all dimeric to pentameric repeats is sputnik, available at http://espressosoftware.com/pages/sputnik.jsp.

Local duplications, tandem repeats and satellites.

Gary Benson's program "Tandem Repeat Finder" (another catchy name) currently is the standard for finding satellites and all other direct repeats (http://tandem.bu.edu/trf/trf.html).
Any local duplications (tandem, inverted, interrupted) can be detected with the program miropeats (http://www.genome.ou.edu/miropeats.html), which presents this similarity information graphically.

3 HOW TO READ THE RESULTS

3.1 The annotation (.out) file

The annotation file contains the cross_match summary lines. It lists all best matches (above a set minimum score) between the query sequence and any of the sequences in the repeat database or with low complexity DNA. The term "best matches" reflects that a match is not

shown if its domain is over 80% or 90% contained within the domain of
a higher scoring match, where the "domain" of a match is the region in
the query sequence that is defined by the alignment start and
stop. These domains have been masked in the returned masked sequence
file. In the output, matches are ordered by query name, and for each
query by position of the start of the alignment.

Example:

```
  SW  perc perc perc  query    position in query     matching repeat      position in  repeat
score div. del. ins.  sequence begin  end  (left)    repeat  class/family   begin end (left) ID
...
 1320 15.6  6.2  0.0  HSU08988  6563 6781 (22462) C  MER7A   DNA/MER2_type    (0)  337  104  20
12279 10.5  2.1  1.7  HSU08988  6782 7718 (21525) C  Tigger1 DNA/MER2_type    (0) 2418 1486  19
 1769 12.9  6.6  1.9  HSU08988  7719 8022 (21221) C  AluSx   SINE/Alu         (0)  317    1  17
12279 10.5  2.1  1.7  HSU08988  8023 8694 (20549) C  Tigger1 DNA/MER2_type  (932) 1486  818  19
 2335 11.1  0.3  0.7  HSU08988  8695 9000 (20243) C  AluSg   SINE/Alu         (5)  305    1  18
12279 10.5  2.1  1.7  HSU08988  9001 9695 (19548) C  Tigger1 DNA/MER2_type (1600)  818    2  19
  721 21.2  1.4  0.0  HSU08988  9696 9816 (19427) C  MER7A   DNA/MER2_type  (224)  122    2  20
```

This is a sequence in which a Tigger1 DNA transposon has integrated into a MER7 DNA transposon copy. Subsequently two
Alus integrated in the Tigger1 sequence. The first line is interpreted as such:

```
  1320     = Smith-Waterman score of the match, usually complexity adjusted
        The SW scores are not always directly comparable. Sometimes
        the complexity adjustment has been turned off, and a variety of
        scoring-matrices are used dependent on repeat age and GC level.

  15.6     = % divergence = mismatches/(matches+mismatches) **
  6.2      = % of bases opposite a gap in the query sequence (deleted bp)
  0.0      = % of bases opposite a gap in the repeat consensus (inserted bp)
  HSU08988 = name of query sequence
  6563     = starting position of match in query sequence
  6781     = ending position of match in query sequence
  (22462)  = no. of bases in query sequence past the ending position of match
  C        = match is with the Complement of the repeat consensus sequence
  MER7A    = name of the matching interspersed repeat
  DNA/MER2_type = the class of the repeat, in this case a DNA transposon
             fossil of the MER2 group (see below for list and references)
  (0)      = no. of bases in (complement of) the repeat consensus sequence
             prior to beginning of the match (0 means that the match extended
             all the way to the end of the repeat consensus sequence)
  337      = starting position of match in repeat consensus sequence
  104      = ending position of match in repeat consensus sequence
  20       = unique identifier for individual insertions

  An asterisk (*) following the final column (see below example)
  indicates that there is a higher-scoring match whose domain partly
  (<80%) includes the domain of the current match.
```

** This has changed in August 2001: cross_match output gives the
percent mismatches/(matches+mismatches+unaligned bases in query). I
didn't think this definition is otherwise commonly used and most users
will assume the divergence level would be
mismatches/(matches+mismatches).

Note that the SW score and divergence numbers for the three Tigger1
lines are identical. This is because the information is derived from a
single alignment (the Alus were deleted from the query before the
alignment with the Tigger element was performed). The ProcessRepeats
script makes educated guesses if any pair of fragments is derived from
the same element or not; if so, the fragments will have the same ID in
the last column, in this example it figured that the MER7A fragments
represent one insert.

Here is another example that shows how much trouble ProcessRepeats
takes to defragment elements and how the ID can be useful in
interpreting the results:

```
 7120 19.9 0.6 0.3 NT_001227  85631  87837 (19816) + L1PA16    LINE/L1       1 1885 (4964)  123
 2503 14.9 6.5 0.7 NT_001227  87839  88241 (19412) + MSTA      LTR/MaLR      1  428    (0)  100
  867 12.9 2.7 0.0 NT_001227  88242  88388 (19265) + MSTA-int  LTR/MaLR      1  151 (1500)  100 *
 5219 19.5 2.9 0.6 NT_001227  88386  89342 (18311) + MSTA-int  LTR/MaLR    629 1607   (44)  100
 8003  3.5 0.8 0.0 NT_001227  89362  90773 (16880) C L1PA3     LINE/L1     (0) 6155   4745  103
 7677  3.5 0.0 0.0 NT_001227  90795  94059 (13594) C L1PA3     LINE/L1     (0) 6155   2872  104
 9050  6.5 0.4 0.1 NT_001227  94060  95127 (12526) C MER11C    LTR/ERVK    (0) 1071      1  106
 7677  3.5 0.0 0.0 NT_001227  95128  97101 (10552) C L1PA3     LINE/L1  (3282) 2873    900  104
 5619  7.8 0.3 0.9 NT_001227  97097  97865  (9788) C L1PA3     LINE/L1  (5370)  776     13  104 *
  320 16.9 0.0 1.7 NT_001227  97876  97934  (9719) + MSTA-int  LTR/MaLR   1594 1651    (0)  100
 1475 19.0 4.8 5.6 NT_001227  97935  98255  (9398) + MSTA      LTR/MaLR      1  323   (48)  100
 2322 14.4 0.8 1.6 NT_001227  98256  98629  (9024) + THE1C     LTR/MaLR      1  371    (0)  112
10051 12.9 3.5 4.3 NT_001227  98630 100221  (7432) + THE1C-int LTR/MaLR      1 1580    (0)  112
```

```
 2359 15.7 0.3 1.9 NT_001227 100224 100598  (7055) + THE1C     LTR/MaLR      3  371    (0)  112
 1475 19.0 4.8 5.6 NT_001227 100599 100646  (7007) + MSTA      LTR/MaLR    323  371    (0)  100
 1360 19.4 8.2 1.7 NT_001227 100662 100955  (6698) + MSTA      LTR/MaLR    114  426    (0)  113
11892 24.7 1.9 2.0 NT_001227 100968 101243  (6410) + L1PA16    LINE/L1    1881 2143 (4706)  123
 2062 11.9 8.4 0.0 NT_001227 101244 101563  (6090) C L1PA12    LINE/L1    (10) 6164   5818  116
11892 24.7 1.9 2.0 NT_001227 101564 105425  (2228) + L1PA16    LINE/L1    2137 5989  (860)  123
  257  0.0 0.0 2.9 NT_001227 105436 105469  (2184) + (TAA)n    Simple        2   34    (0)  118
 2189 18.2 0.2 0.7 NT_001227 105470 105893  (1760) + L1PA16    LINE/L1    6062 6483  (386)  123
  255  6.1 0.0 0.0 NT_001227 105896 105928  (1725) + (TA)n     Simple        1   33    (0)  120 *
  369  0.0 0.0 0.0 NT_001227 105928 105968  (1685) + (GA)n     Simple        2   42    (0)  121
  305 18.8 0.0 1.0 NT_001227 105971 106066  (1587) + (TA)n     Simple        2   96    (0)  122
 1589 21.2 1.6 1.1 NT_001227 106068 106449  (1204) + L1PA16    LINE/L1    6485 6868    (1)  123
```

This entire 20,819 bp block of sequence is comprised by an L1PA16 (#123), in which 7 or 8 elements have integrated (it is unclear to me if the MSTA #113 is a separate integration or a tandem duplication). There are at least four layers, with MER11 (#106) inserted in L1PA3 (#104) inserted in MSTA (#100, maybe in #113) inserted in L1PA16. L1PA16 is already primate specific, so that all these insertions took place during primate evolution.

The ID column helps much in deciphering the events. It also should be a basis for graphic display of RepeatMasker output.

3.2 Alignments

When using the -a option, a .align file is created that contains alignments of your query sequence to the matching repeat consensus sequences. The alignments are given in the same order as listed in the .out file. They are always in the orientation of the query; you can use the -inv option to produce all alignments in the orientation of the consensus sequence.

The alignments are in the cross_match/SWAT format, in which mismatches rather than matches are indicated (transitions with an i and transversions with a v). The description line preceding the alignment is similar to that seen in the .out file. In the example of an alignment below, an old retrovirus-like LTR (MLT1H) has been interrupted by the more recent insertion of a short DNA transposon (MADE2):

```
384 28.89 9.24 2.17 chr1_4622259_4622561 21 77 (225) MLT1H#LTR/MaLR 23 88 (461) 5

  chr1_4622259_       21 TGGCC-CAATTCTTTACCTCTC--TGCCTCTTGTGCCTTTTG-------G 60
                            -     ? ii   i   --  iv  i-i    i    -------i
  MLT1H#LTR/MaL       23 TGGCCACAATTMTCCACCCCTCCCTGTATCC-ATGCCCTTTGCAATGTGA 71

  chr1_4622259_       61 CTTTGCCATTTCTTCTA 77
                            vii i  i i
  MLT1H#LTR/MaL       72 CTTTGCAGCTCCTCCCA 88

Transitions / transversions = Transitions / transversions = Unknown
Gap_init rate = Unknown


557 11.25 0.00 0.00 chr1_4622259_4622561 78 157 (145) C MADE1#DNA/Mariner (0) 80 1 3

  chr1_4622259_       78 TTAGGTTGGTGCAAAAGTAATTGTGGGTTTTAGCATTTAAAGTAATACCA 127
                                                 i  v    iv    ?        iv
C MADE1#DNA/Mar       80 TTAGGTTGGTGCAAAAGTAATTGCGGTTTTTGCCATTRAAAGTAATGGCA 31

  chr1_4622259_      128 AAAACCACAACTACTTTTGCACCAACCTAA 157
                            i   i
C MADE1#DNA/Mar       30 AAAACCGCAATTACTTTTGCACCAACCTAA 1

Transitions / transversions = Transitions / transversions = Unknown
Gap_init rate = Unknown


384 28.89 9.24 2.17 chr1_4622259_4622561 158 283 (19) MLT1H#LTR/MaLR 89 218 (331) 5

  chr1_4622259_      158 TAGTAAAAGCAGAGGATAAT-----ATTCCTTGTCTTTGGGTTTGTCATG 202
                           i -- i i ii   vv  v ----- ii     vv i    i v i v
  MLT1H#LTR/MaL       89 CA--AGAGGTGGAGTCTATTTCCCCACCCCTTGAATCTGGGCTGGCCTTG 136

  chr1_4622259_      203 TGACTCTCTTTGGCCATGGGAACATAGGCAAAAATGACT-TGTGCCCCTT 251
                            iv         vvi   ii -    i  i    v-      vv
  MLT1H#LTR/MaL      137 TGACTTGCTTTGGCCAATAGAATGT-GGCAGAAGTGACGGTGTGCCAGTT 185

  chr1_4622259_      252 CTGAGCCCCGGCCTTGAGAGGTCTT-CATGCTT 283
```

```
                                iv     ii     i   -  i
  MLT1H#LTR/MaL      186 CTGAGCCTAGGCCTCAAGAGGCCTTGCACGCTT 218
```

Transitions / transversions = Transitions / transversions = Unknown
Gap_init rate = Unknown

Note that the description line is identical for the first and third alignment. Before the query was compared to the MLT1H consensus, RepeatMasker had recognized the MADE1 element and had removed it from the query sequence to more or less reconstruct the pre-MADE1-insertion situation. Thus, position 21 to 283 of the query could be aligned to the MLT1H consensus in a single piece. Since 2004 (RepeatMasker3.0 and up), such alignments are broken up to present all matches in serial order.

Alignments are especially useful for designing PCR primers in a region full of repeats. It is possible to design primers contained in a common repeat that still work in a whole genome, when the 3' end is in a region that is very different from the consensus.

Discrepancies between alignments and the .out file

Discrepancies between alignments and annotation result from the adjustments made by the ProcessRepeats script to produce more legible annotation. This annotation also tends to be closer to the biological reality than the raw cross_match output.

For example, adjustments often are necessary when a repeat is fragmented through deletions, insertions, or an inversion. Many subfamilies of repeats closely resemble each other, and when a repeat is fragmented these fragments can be assigned different subfamily names in the raw output. ProcessRepeats often can decide if fragments are derived from the same integrated transposable element and which subfamily name is appropriate (subsequently given to all fragments). This can result in discrepancies in the repeat name and matching positions in the consensus sequence (subfamily consensus sequences differ in length).

In many cases matches are fused into one annotation. To give a few common examples:

- In large sequences that are analyzed in fragments consecutive fragments overlap and repeats in these overlaps will appear twice (partially or wholly) in the alignment file but are merged in the .out file.

- A-rich simple repeats originated from the poly A tail of Alus and LINE-1s are incorporated in the annotation of the Alu or LINE-1.

- There is an 'endless' number of subfamilies for retroposons which can not all be represented in the databases and sometimes an element is matched by overlapping pieces of two related subfamilies (which will be merged).

- You may find large discrepancies in position numbering if an element includes tandem repeat units. For example, MER109 contains multiple ~300 bp repeat units that can lead to overlapping matches. In the annotation such matches are fused.

- Simple repeats or satellites that are longer than the number of units represented in the repeat library will be represented by multiple, genrally overlapping alignments in the .align file, but only a single annotation line in the .out file.

Specific LINE problems:

Some other discrepancies between alignments and annotations are specific to LINE-like elements. These repeats usually do not appear as complete elements in the consensus database. For LINE-1, this is mostly due to the contrast in conservation over the length of its sequence during its evolution in the mammalian genome; the ~3 kb ORF2 region of LINE-1 has been very conserved, whereas the untranslated regions and ORF1 to a lesser degree have evolved very fast. Thus the 3' end or 5' end of an ancient LINE-1 does not even remotely resemble that of the currently active LINE-1, whereas the coding region for reverse transcriptase is closely related. Thus, many subfamilies have been defined for both the 5' and 3' UTRs (48 and 55, resp.) of LINE-1 elements in human DNA, whereas only 11 ORF2 entries are present in the

database. Besides the fact that some 3' ends have multiple defined 5' ends, and vice versa, the program would become very slow when each query is compared to 55 full length (6 to 8 kb) LINE-1 elements. Thus, LINE-1 elements are presented in the database in 3 pieces, and the ProcessRepeats script puts these pieces together. As a result both the names of the repeats and position numbering in the consensus sequence are generally different in the alignments than in the output file. The LINE2 elements are likewise broken up in 3' UTRs for different subfamilies and one 5'UTR-ORF2 region.

Between LINE-1 subfamilies, the 3' UTR ranges from 500 bp to over 2000 bp (in L1MC/D3), and the length of the 5' UTR is even more variable, even between subfamilies that show strong similarity in the 3' UTR. To allow the LINE-1 fragments to be put together, all position numbers in older LINE-1 subfamilies are normalized relative to the position of ORF2 (the conserved part of LINE-1) in a complete L1PA2 element. Since some older elements have much longer 5' UTRs or ORF1-ORF2 linker regions than L1PA2, this often results in the assignment of negative position numbers for the 5' end of LINEs. Since the March2000 release, such positions and all positions in fragments thought to be part of the same LINE-1 insert are readjusted to count from the 5' end (which is not necessarily the very 5' end of the LINE-1 source gene, as these are hard to derive for old elements). One problem with this approach is that positions are not adjusted in detached 3' fragments that are somehow not recognized by the program as originating from the same insertion. Thereby, the common origin of the 5' fragments and 3' fragments may become completely obscured. You can use the option '-orf2' of ProcessRepeats to retrieve an output in which all LINE-1s are numbered so that position 1 of ORF2 is aligned (resulting in occasionally negative positions).

3.3 The summary (.tbl) file

The summary file is pretty much self-explanatory. Below is an example.

```
================================================
file name: AC027410.fa
sequences:            1
total length:    152192 bp (148791 bp excl N-runs)
GC level:         39.59 %
bases masked:     88734 bp ( 59.64 %)
================================================
                number of      length   percentage
                elements*    occupied  of sequence
------------------------------------------------
SINEs:              195        45195 bp    30.37 %
      ALUs          178        43249 bp    29.07 %
      MIRs           17         1946 bp     1.31 %

LINEs:               54        31173 bp    20.95 %
      LINE1          36        24602 bp    16.53 %
      LINE2          18         6571 bp     4.42 %
      L3/CR1          0            0 bp     0.00 %

LTR elements:        13         5833 bp     3.92 %
      MaLRs           8         4079 bp     2.74 %
      ERVL            0            0 bp     0.00 %
      ERV_classI      5         1754 bp     1.18 %
      ERV_classII     0            0 bp     0.00 %

DNA elements:        17         4459 bp     3.00 %
      MER1_type      12         1903 bp     1.28 %
      MER2_type       4         2466 bp     1.66 %

Unclassified:         0            0 bp     0.00 %

Total interspersed repeats:    86660 bp    58.24 %


Small RNA:            2          124 bp     0.08 %

Satellites:           0            0 bp     0.00 %
Simple repeats:      22         1151 bp     0.77 %
Low complexity:      22          799 bp     0.54 %
================================================

* most repeats fragmented by insertions or deletions
  have been counted as one element
Runs of >20 Ns in query were excluded in % calcs
```

The query species was assumed to be Pan troglodytes
RepeatMasker version 20040617 , default mode
run with cross_match version 0.990329
RepBase Update 9.04, RM database version 20040617

AC027410 was a draft sequence, with individual contigs separated by poly N linkers. In this case, the option -excln was used, so that these strings of Ns were ignored for the percent calculations.

The classification in this table is well defined (see my reviews in COGD) and forms a good basis for visual presentation and tabulation of the repeats in your study.

We've been able to classify almost all human repeats, most of them even in subclasses. The totals for the classes often are higher than the sum of the subclasses, because not all elements fit in a subclass and minor subclasses are not listed separately in the table (e.g. for the human table the Mariner, Tc2, Piggybac, Zaphod, and Arthur families of DNA transposons). The HAL1 element, derived from LINE-1, is added to the LINE-1 total in this table.

Note that the "MER" subclasses have no relationship to each other. The term MER (MEdium Reiterated repeats) was introduced for purely administrative purposes to give the beast a name. The MER1 and MER2 groups were named after the first member of these groups identified as an interspersed repeat in our genome. In the literature they're also known as the Tigger and Charlie groups.

The nomenclature of mammalian repeats derived from retrovirus-like elements is different from older versions. I've now divided this class up in the traditional class I, class II (ERVK), class III (ERVL) retroviruses and the ERVL-derived but very distinct non-autonomous MaLR elements. Since 'class III' is not an accepted classification yet, for now this class is called ERVL. The large MER4-group of non-autonomous LTR elements merges seamlessly with class I endogenous retroviruses, making it hard to define, and is now incorporated in the latter group. The ERV classes are most readily distinguished by the size of the insertion site duplication: 4 in class I, 6 in class II, and 5 in class III, though there are some exceptions to this rule. My LTR classification is not based target size duplication sizes, but on the encoded proteins in the internal sequences or, if these are not known, on matches to LTRs with internal sequences.

As described above, the ProcessRepeats script tries very hard to find out which repeat fragments were derived from the same insertion event of a transposable element, but there still will be a slight overestimate of the copy numbers.

There may be slight differences in the number of "bases masked" and the sum of the bases annotated in this .tbl file. At this moment bases are masked based on the unprocessed matches (as they are in the .cat file) and most of the discrepancies are accounted for by unmasked regions between flanking identical simple repeats, annotated as one stretch if fewer than 10 bases separate them, and fragments of repeats shorter than 10 bp which are not annotated but are masked.

4 APPLICATIONS

4.1 Use in database searches

RepeatMasker is most commonly used to avoid spurious matches in database searches. Generally this step is strongly recommended before doing BLASTN or BLASTX equivalent searches with mammalian DNA sequence.

The most common concern is of course if RepeatMasker ever masks coding regions.
We found that false matches in coding regions are extremely rare, but did identify 38 genuine fragments of interspersed repeats (4214 bp) in the (annotated) coding regions of the 4440 human mRNAs (7.2 Mb) analyzed (excluding annotated coding sequences of LINE-1 elements and endogenous retroviruses). We verified matches with lower scores by comparing the translation products to close homologous or redundant entries in the database (the repeat matching regions always were exactly missing). In the majority of these cases, the sequences appear

to be improperly annotated or to represent either artificially or naturally defective mRNAs (e.g. alternatively spliced exons comprised of a small fragment of a repeat). Genuine overlaps of interspersed repeats with coding sequences usually involve terminal regions of the ORFs. Since the transposable element derived region is unique to the protein in that (group of) species, the masking does not interfere with database searches.

However, some cautionary comments are necessary. First, a few active cellular genes are derived from transposable elements (see my list of 50 in our genome in Lander et al. 2001). Some of these genes will be partially masked by a (related) transposon in the repeat database. EST and cDNA matches beyond the masked region should alert you.

Also remember that, currently only for mammals, RepeatMasker screens for small RNA (pseudo)genes because of their similarity to SINEs. The number of matches to small RNAs are listed in the overview table; (close to) exact matches are possibly active genes, although related active genes not in the database may show diverged matches. If you're interested in (small) RNA genes, you should use the -norna option to leave these sequences unmasked, while SINEs will remain masked.

A final caution relates to the fact that 3' UTRs of transcripts are about as dense in interspersed repeats as intergenic regions are. Thus, many ESTs are completely masked as repetitive DNA. I recommend that, when you compare a genomic sequence against the EST database or use ESTs as a query in nucleotide searches, you search with the unmasked sequence as well; use a long minimum match (word length/ word size) like 40 bp to identify exact matches and avoid most background. Unfortunately the maximum word length that can be used in the NCBI BLASTN program is 18 (due to memory limitations).

4.2 Identification of DNA source (contamination detection)

Bacterial insertion elements

Bacterial insertion sequences (IS elements) often pop up in foreign sequences, as their activity in the E. coli is not always successfully suppressed during cloning. As late as 2002, human entries in the 'finished' section of GenBank contained over a hundred IS elements.

With each run, RepeatMasker includes a quick check for bacterial insertion elements that may have inserted during cloning. You can turn this off with the -no_is option. The -is_only option limits the run to this check only.

When a full-length element is found and a target site duplication is confirmed, its location is both reported to the screen and stored in a .alert file. The latter also contains information of possible mouse<->human contamination.

-is_clip, -is_only

With the -is_only and is_clip options, the detected IS and one of the flanking repeats is clipped out to restore the pre-cloning artifact situation before comparison with the repeat databases. The original query FASTA file will remain unchanged. An insertion-sequence-clipped, but otherwise unmasked query sequence is printed to <file>.withoutIS.

With either of these options, a properly adjusted quality string is printed to a file with the suffix .qual.withoutIS when a corresponding PHRED quality file (.qual) is in the same directory. Note that these names won't be such that the clipped sequence and quality file form a pair for subsequent cross_match/PHRAP work. They need to be renamed, as I assume one wants to do anyway.

Most but not all IS elements can be precisely cut out. The element may be at the edge of a sequence, or (rarely) the element may have inserted improperly, lacking target site duplications or missing terminal bases (internal deletion products are generally handled okay). These matches are reported, but are left untouched even in _is_only or is_clip mode.

The location of any IS element is both reported to the screen and stored in an .alert file. The latter also contains information of possible mouse<->human contamination.

Here are the specifics of IS element insertions:

```
IS1     8-10 bp duplication
IS2     5 bp duplication; published sequence was too short
IS3     3 bp duplication
IS4     No examples of clonal artifacts; no dup site info
IS5     4 bp duplication; preferred target TCTAGA
IS10    9 bp duplication; extreme preference for CGCTNAGCN; published
        sequence for IS5 & 10 were too long, included preferred target site
IS30    2 bp duplication
IS150   3 bp dup, with one exception (4 bp); strong pref for CAGNNTGGGGCY
IS186   10 or 11 bp dup
Tn1000  5 bp duplication;
```

Human, mouse, or rat sequence contamination or mix-up.

A straightforward way to distinguish murine and human DNA is by checking for either rodent-specific or primate specific repeats. Likewise, rodent or primate contamination in any other mammalian or non-mammalian background can be picked up as well. If your lab has, say, a rat and a pink fairy armadillo sequencing project, rat DNA in a supposedly armadillo sequence can be picked up quite reliably, depending on the length of the query.

When the option -rodspec or -primspec is used, RepeatMasker only checks the query against a small library of repeats that have not (yet) been observed in the 'other' species. The locations of the matches are printed to <file>.alert. This function will be expanded to other mammals, when these species are starting to be sequenced in earnest.

I've checked for the specificity of the reported matches quite extensively. Whenever two or more types of repeats are reported, the odds are that the alert is correct. Very occasionally, a single reported match could be a false alert. This is especially possible when a 'new' mammalian species is analyzed, because, unbeknownst to me, a related repeat may have amplified in such a genome.

Other species contamination.

When a supposedly mammalian clone is of non-mammalian origin, very few if any interspersed repeats will be reported by RepeatMasker. Rodent or primate genomic sequences are on average 40-50% dense in recognizable interspersed repeats, so that any stretch of genomic DNA of significant length (say 30 kb or more) showing less than 10% density in interspersed repeats is of suspect origin. An automated alert for such a situation is not included, as query sequences of coding regions or transcripts, generally of very low repeat density, would constantly cause an alert.

4.3 DateRepeats - Masking lineage-specific repeats for genomic alignments

Since June 2003 each repeat consensus in the mammalian repeat libraries has a phylogenetic label. The interspersed repeat is expected to be found in all species belonging to the specified genus, family, order, etc. The label is based on the presence or absence of repeats at orthologous sites in different genomes and on the average divergence of repeat copies from their derived consensus sequence. This phylogeny will become much more accurate and refined over time.

The tag allows RepeatMasker to compare queries only to repeats expected to be found in the query species, without having to provide a library for each species. For example, a rat query currently is compared to repeats tagged with Rattus, Murinae, Muridae, Rodentia, Eutheria, and Mammalia.

It also allows one to mask only those interspersed repeats that have arrived in a genome after the speciation of two species. For optimal alignment of genomic sequences of two species, 'ancestral repeats' that are located at orthologous sites in both genomes (unless deleted) should not be masked, whereas 'lineage-specific' repeats should be masked or clipped out. An experimental version of this RepeatMasker feature has been used in the alignment of the mouse to the human genome (Waterston et al. 2002). By clipping out rather than masking the lineage-specific repeats the aligning fraction for the mouse genome could be increased from 35% to 40% (see also Schwartz et al 2003, NAR 31:3518-24, and Thomas et al 2003, Nature 424:788-93).

As of the September 2003 version, the RepeatMasker package contains a
script "DateRepeats" that takes a RepeatMasker .out file and creates
annotation with added column(s) indicating if a repeat is expected to
be present in the indicated 'other species' as well as a sequence with
lineage-specific repeats masked only. The script currently works only
for a few mammalian species (human, mouse, rat, cat, dog, cow, pig,
horse, rabbit), but refinement is inevitable.

```
DateRepeats <repeatmasker .out files> -query <species1> -comp
<species2> [-comp <species3> -mask <species2>]

The required flags are:
-q -query <species1> the species that has been analyzed
-c -comp <species2>  other mammalian species; can be used multiple times, adding extra
                     columns to the annotation in a  <file.out_species2_species3_etc>

Optional parameters are:
-m -mask <species> produces a sequence file with all lineage specific repeats masked, i.e.
                   those predicted to be in the -query and absent in the -mask species
                   (sequence <file> and RepeatMasker <file.out> files must be in same directory)
                   <species> must correspond to one of the -comp species
-a -aggressive     also mask those repeats unclear to be lineage specific or ancestral
-n -nolow          does not mask (micro)satellites or low complexity DNA, which are
                   generally lineage specific, but hard to date
(-a and -n have no effect unless -m is used)

In the first release of this script the <species> for -q, -c, and -m
are limited to human, mouse, rat, cat, dog, cow, pig, horse and
rabbit.
```

For example the command line:
DateRepeats chr3_4000001_4005000.out -q mouse -c rat -c human
prints the following output to chr3_4000001_4005000.out_rat_human:

SW score	perc div.	perc del.	perc ins.	query sequence	position in query begin	end (left)		matching repeat	repeat class/family	position in repeat begin	end (left)		rat ID	hum	
12436	19.0	1.7	7.7	chr3_400	9	536	(4464) +	L1_Mur2	LINE/L1	1850	2408	(3469)	1	X	0
2728	5.2	0.0	3.9	chr3_400	537	896	(4104) +	ORR1A0	LTR/MaLR	1	346	(0)	2	0	0
12436	19.0	1.7	7.7	chr3_400	897	2441	(2559) +	L1_Mur2	LINE/L1	2408	3665	(2212)	1	X	0
5229	7.2	0.0	1.1	chr3_400	2442	3134	(1866) +	L1Md_F2	LINE/L1	5877	6580	(2)	3	0	0
12436	19.0	1.7	7.7	chr3_400	3135	4259	(741) +	L1_Mur2	LINE/L1	3665	4856	(1021)	1	X	0
394	20.9	0.0	0.9	chr3_400	4260	4351	(649) +	Lx2	LINE/L1	6892	6996	(1)	4	X	0

The X indicates that the repeat is expected to be present at
orthologous sites, while the O predicts an absence. None of the above
repeats are found in the human genome. Notice that a mouse-specific
ORR1 (#2) and L1 (#4) has inserted into a rodent specific L1 (#1).

A few lines of the output for
DateRepeats chr19.fa.out -q rat -c mouse -c human -m mouse -a -n
are

1199	17.9	11.2	1.8	chr19	313706	313990	(58909535) C	Lx9	LINE/L1	(9)	7635	7324	377	X	0
23	0.0	0.0	0.0	chr19	314125	314147	(58909378) +	AT_rich	Low_complexity	1	23	(0)	378	-	-
726	17.5	1.3	8.9	chr19	314152	314308	(58909217) +	B1_Rn	SINE/Alu	2	146	(2)	379	?	0
228	32.8	0.0	7.4	chr19	314355	314476	(58909049) C	B3	SINE/B2	(77)	139	27	380	X	0

The chr19.fa.masked_vs_mouse file contains a sequence appropriately
masked for alignment against mouse. It has repeats 377 & 380
masked. The -n flag leaves the AT-rich region unmasked, while the -a
flag forced it to mask repeat 379 as well. B1_Rn is rat specific, but
the 17.5% divergence from the consensus is much higher than the
average divergence level of non-functional DNA since the rat-mouse
split. It therefore gets the "?" assignment. The rules for assigning
question marks are arbitrary (2-fold lower divergence than expected,
1.5x higher than expected for a repeat at the boundary).

4.4 Use in gene prediction and other applications

Predicting genes from a masked sequence has several problems. First,
one should use the option -nolow to avoid masking low complexity
regions and trinucleotide repeats in coding regions. But even with
only interspersed repeats masked, gene prediction programs may fail to
identify exons correctly. As pointed out above, sometimes tail ends of
coding regions may have originated from transposable elements. Some
gene prediction programs suggest the extend of 3' UTRs. These will be
often overestimated in masked DNA, as many genuine poly A signals are
located in interspersed repeats. Finally, even if no coding regions

have been masked, splice sites may be compromised; e.g. the polypyrimidine region that contributes to an acceptor splice site may be contained within a repeat.

Thus, I generally recommend to run a gene prediction program on unmasked DNA (as well) and compare the predicted genes and exons with the RepeatMasker output. Some gene prediction program allow you to force certain exons out of the predictions (e.g. often the old ORFs of LINE-1 elements and endogenous retroviruses are included in genes). Work is also in progress at several sites to incorporate RepeatMasker into gene prediction programs, in which cases matches to repeats are weighted in along with the other parameters used.

Other uses

Many people mask repeats before designing primers or oligo probes from sequence data. I've often been told that primers/probes designed from regions unmasked by RepeatMasker have a much better success rate. A cautionary note here is that unmasked regions not necessarily are unique in the genome (e.g. many lower copy repeats are not in the database yet) and experiments should be performed as if no filtering against repeats has been done. The alignments can help in designing primers from sequences that are completely masked. Regions that diverge much from the consensus are less likely to misbehave than others.

RepeatMasker is sometimes used during assembly of large genomic sequences. This procedure probably is most useful in very Alu rich regions; in that situation I recommend to only mask the Alus, and maybe limit the masking to those Alus less than 15% diverged (-div 15).

There are plenty of other uses, e.g. analysis of repeats can reveal a lot about the evolution of a locus (deletions vs. insertions, inversions, approximate time of these events). When you're doing that you're a specialist and don't need any help from this help file (maybe from some of the literature sited below though).

5 REFERENCES

Reference for RepeatMasker

We appreciate it if you could refer to the web page (Smit,AFA & Green,P RepeatMasker at http://www.repeatmasker.org) or otherwise to Smit, AFA & Green, P., unpublished.

The EMBL format of the RepBase Update database contains references for individual repeats, as well as annotation with respect to divergence level, affiliation, copy number, etc. Much if not most of the information in this database is not published elsewhere. It can be accessed at http://www.girinst.org/.

We are trying to keep the nomenclature of the interspersed repeats in the output of RepeatMasker identical to that of the reference database. In most cases the names correspond to those most commonly used in the literature.

There is much too much literature out there to list these days. My own views on the repeat structure in mammalian genomes have most recently been described in the following papers:

Waterston et al. (2002) Initial sequencing and comparative analysis of the mouse genome. Nature. 420(6915):5 20-62.

Lander E. S., et al. (2001). Initial sequencing and analysis of the human genome. Nature 409(6822): 860-921.

Smit, A.F.A. (1999) Interspersed repeats and other mementos of transposable elements in mammalian genomes. Curr Opin Genet Devel 9 (6), 657-663.

If you have ideas for improvements or found a problem, drop a note at rhubley@systemsbiology.org or asmit@systemsbiology.org

/***