Chapter 2

Using and Understanding RepeatMasker

Sébastien Tempel

Abstract

RepeatMasker is a program that screens DNA sequences for interspersed repeats and low-complexity DNA sequences. In this chapter, we present the procedure to routinely use this program on a personal computer.

Key words: Repeats, Mobile genetic elements, Transposon, Retrotransposon, In silico

1. Introduction

RepeatMasker is a free application created by A.R.A Smith in 1998. The latest version 3-2-9 is available since January 7, 2010. RepeatMasker screens one or more genomic sequences in FASTA format and detects transposable elements, satellites and low-complexity DNA sequences. It is the most use d software for the detection of transposable elements. Although it was not the subject of scientific publication, it is particularly used for genome annotation and for studying families of multiple transposable elements.

RepeatMasker is written in PERL. It uses a search engine like Basic Local Alignment Search Tool (BLAST; (1)) with a library of transposable elements as the query and the input sequence as the database. RepeatMasker reads "raw results" from the alignment tool, manipulates them based on some criteria and writes them in the RepeatMasker standard outputs. There are currently two versions of RepeatMasker: RepeatMasker that runs on the user's computer (UNIX or LINUX computer) and WEBRepeatMasker that runs on the Institute for System Biology server. WEBRepeatMasker is a Web interface of RepeatMasker (Fig. 1). They both accept the same input sequences, have some options and outputs in common, but the RepeatMasker has more outputs and options.

Yves Bigot (ed.), *Mobile Genetic Elements: Protocols and Genomic Applications*, Methods in Molecular Biology, vol. 859,
DOI 10.1007/978-1-61779-603-6_2,

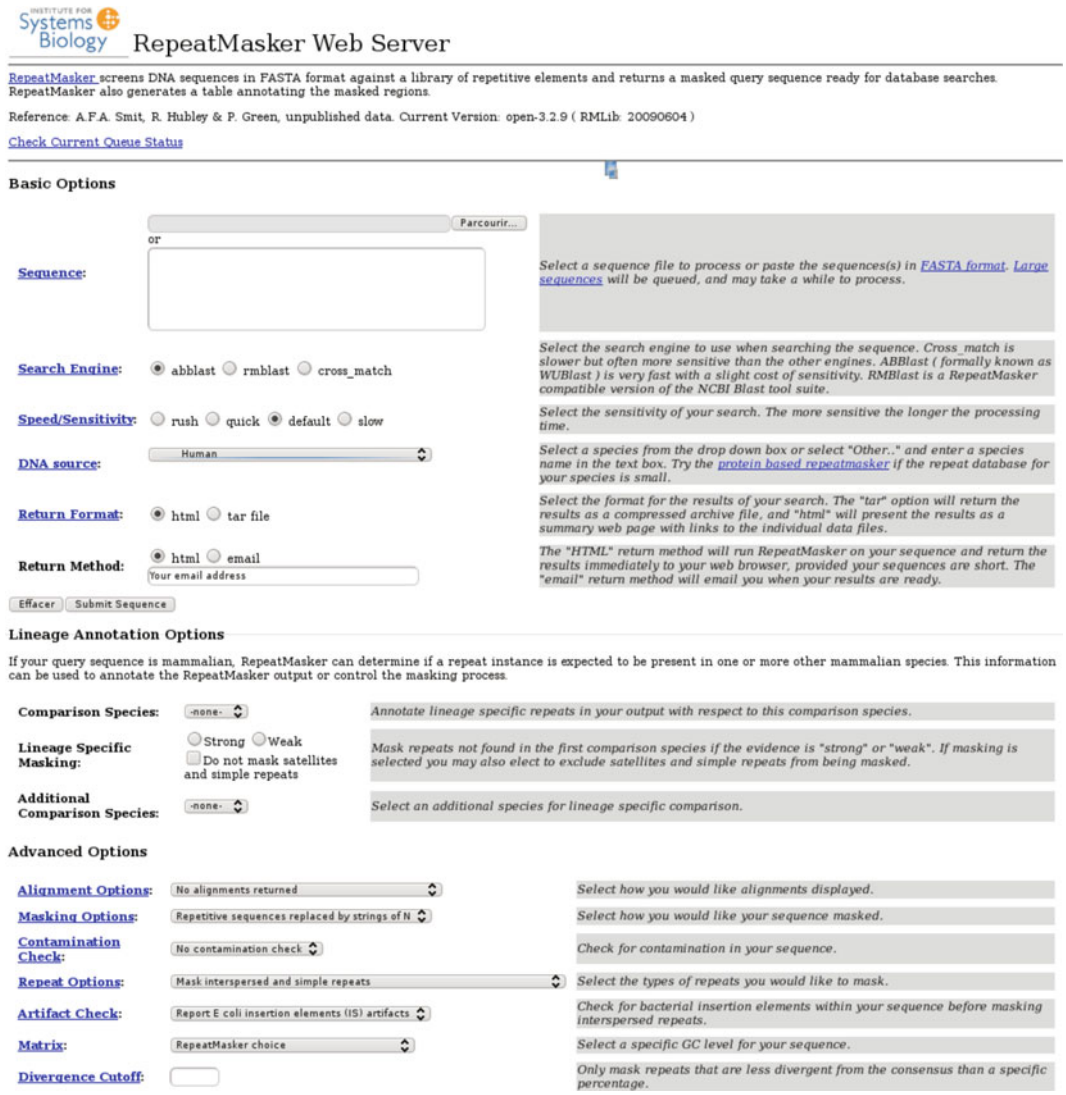

Fig. 1. Web interface of RepeatMasker: WEBRepeatMasker.

2. Installation of RepeatMasker

RepeatMasker is not limited to one input sequence and there is no limitation of the sequence size. WEBRepeatMasker has a limit of 10 Mbp for deferred processing with an email reply and 100 kbp for immediate processing. The user needs to install RepeatMasker on his personal computer if he wants to use it on long sequences (longer than 10 Mbp) or other output formats. The user can download the last version of RepeatMasker at http://www.repeatmasker.org/RMDownload.html and uncompress it. RepeatMasker runs only on Unix and Linux systems.

Before configuring the software, the user also needs to install a library and a minimum of three pre-requirements: PERL, Tandem Repeat Finder (TRF; (2)), and a search engine sotfware. All these files must be installed before the first run and the configuration of RepeatMasker.

The user must download the RepeatMasker library file (repeatmaskerlibraries-XXX.tar.gz, XXX corresponds to the date when the last version was created) from the GIRI Web site (http://www.girinst.org/repbase/index.html) after the user has requested an account opening. Uncompress the file in the "Libraries" subdirectory inside RepeatMasker directory. RepeatMasker needs PERL 5.8.0 or one of its update for the installation. This is usually installed by default on Unix systems. Tandem Repeat Finder detects the tandem repeat in a FASTA sequence. After the software installed, copy the binary file in the same directory than RepeatMasker binaries and rename it to "trf." The search engine library is one of the four alignment tools: AB-BLAST (new name for WU-BLAST, http://www.advbiocomp.com/blast.html), Cross_match (http://www.phrap.org/phredphrapconsed.html), Decypher (http://www.timelogic.com/decypherblast.html), or RMBLAST (http://www.repeatmasker.org/RMBlast.html).

The four alignment tools use the BLAST algorithm at the core of their methods.

3. Configuration of RepeatMasker

After the installation of Tandem Repeat Finder (2) and the alignment software, the user must configure RepeatMasker before the first run. There are two ways to configure it: manually editing the configuration file of RepeatMasker (RepeatMaskerConfig.pm) or using the PERL script 'configure' in RepeatMasker directory that automatically writes the configuration file. Here, I assume that the user will use the PERL script. First, the script asks the user to enter some necessary paths as follows:

```
**PERL INSTALLATION PATH**

Enter path [ /usr/bin/perl ]:
```

Confirm the path to the PERL binaries. The path inside the brackets corresponds to the path written in RepeatMaskerConfig.pm. Usually, PERL is installed at '/usr/bin/perl', it is only necessary to confirm it (*press Enter*).

```
**REPEATMASKER INSTALLATION PATH**

Enter path [ /usr/local/bin/RepeatMasker ]:
```

Enter the complete path to where you install RepeatMasker. If the user does not know the complete path of RepeatMasker, he can use the Unix command 'pwd' in the software's directory.

```
**TRF INSTALLATION PATH**

Enter path [ ]:
```

Enter the complete path to where you put the TRF binary. Normally it is the same path as RepeatMasker.

Add a Search Engine:

```
1. CrossMatch: [ Un-configured ]

2. RMBlast-NCBI Blast with RepeatMasker extensions: [

Un-configured ]

3. WUBlast/ABBlast (required by DupMasker): [ Un

configured ]

4. DeCypher (TimeLogic): [ Un-configured ]

5. Done
```

Enter Selection: 3

```
**ABBlast/WUBlast BLASTP INSTALLATION PATH**

Enter path [ ]: /usr/local/bin/WUBLAST

Do you want ABBlast/WUBlast to be your default

search engine for Repeatmasker? (Y/N) [ Y ]:
```

In this part of the script, choose which alignment tool you want use. In this example, I choose WUBLAST (number 3). The script prompts for the path of AB-BLAST. Confirm AB-BLAST as the default alignment tool. The script will return to the Search Engine menu. Enter another search engine or finish the configuration.

4. Running RepeatMasker with Basic Options

On WEBRepeatMasker, load the sequence file (maximum 100 kbp) by entering the sequence into the indicated text field or browsing the file with the 'Load' button (Fig. 1). Choose the results format: 'html' or 'tar file'. The 'html' choice shows the results as Web page and the 'tar file' choice sends a compress file that contains all output files. Select an appropriate genome for your sequence with the "DNA source" button (Fig. 1).

If the user chooses to use the RepeatMasker from the command line, it works as follows:

```
RepeatMasker [-options] <seqfiles(s) in FASTA format>
```

Enter the RepeatMasker name following by the options and the name of the input file. The output files are automatically created. The different options of RepeatMasker (and WEB RepeatMasker) are presented in the Subheading 5.

4.1. Input Format Sequence

The input sequence must be in FASTA format. FASTA format looks like this:

```
>Sequence1
ACGTGCGCGATCGCCTGCTAGGCGTACGTCGCAGGCGATCGATGTGCTAGATCAG
ATGACA
>Sequence2 human
GGGCTAGATTAGCACCACATACATCGCTCA
```

The FASTA format of a sequence always start with the symbol '>'. This symbol is the beginning of the sequence name. This name can be one letter or number, one word or one sentence. The sequence name is limited to one line (it finishes with return symbol). The size of the sequence is not limited and could be written in one or more lines. The FASTA sequence finishes with a blank line or a new start symbol '>'.

4.2. Output Results

WEBRepeatMasker and RepeatMasker returns three output files for each query: a masked file, a map file and an overview table file.

4.2.1. Map File

The map file is a list of all repeats identified in the query sequence by the alignment software. Figure 2 shows the map file format. The format of map file is the cross_match summary lines (Figs. 2 and 3). RepeatMasker converts the other search engine output into this format. The map file lists all best matches (above a minimum score) between the query sequence and any of the sequences in the repeat database or with low-complexity DNA. A match of a repeat A could be inserted in the match of a repeat B. If this inserted match has a lower score than its container, RepeatMasker hides it. The matches in the list masked in the map file are masked in masked sequence file (see Subheading 4.2.2). In the map file, matches are ranked by position from the start of the alignment.

4.2.2. Masked Sequence

RepeatMasker returns a .masked file containing the query sequence(s) with all identified repeats and low-complexity sequences masked. All nucleotides contained in a repeat in the map file are replaced by "N" in the sequences (Fig. 4). Some options allow modifying the format and the category of repeats that are masked (described in Subheading 5).

±	score	% div.	% del.	% ins.	query sequence	----position in query---- begin	end	(left)	C +	matching repeat	repeat class/family	-position in repeat-- (left) begin	end	begin (left)	linkage id/graphic
±	282	14.5	0.0	7.2	chr1_Arabidopsis	1	105	(3499825)	C	ATREP18	DNA	(1142)	649	561	1
±	201	9.4	0.0	0.0	chr1_Arabidopsis	1066	1097	(3498833)	+	(C)n	Simple_repeat	1	32	(0)	2
±	26	34.6	0.0	0.0	chr1_Arabidopsis	3305	3330	(3496600)	+	AT_rich	Low_complexity	1	26	(0)	3 *
±	29	88.4	0.0	0.0	chr1_Arabidopsis	3308	3350	(3496580)	+	AT_rich	Low_complexity	1	43	(0)	4
±	267	7.9	0.0	0.0	chr1_Arabidopsis	4291	4328	(3495602)	+	(TA)n	Simple_repeat	2	39	(0)	5
±	279	0.0	0.0	0.0	chr1_Arabidopsis	8669	8699	(3491231)	+	(TC)n	Simple_repeat	1	31	(0)	6
±	221	16.9	1.6	3.3	chr1_Arabidopsis	9970	10030	(3489900)	+	(TA)n	Simple_repeat	2	61	(0)	7
±	297	10.9	0.0	0.0	chr1_Arabidopsis	11915	11960	(3487970)	+	(CAT)n	Simple_repeat	2	47	(0)	8
±	186	4.3	0.0	0.0	chr1_Arabidopsis	13346	13368	(3486562)	+	(GA)n	Simple_repeat	2	24	(0)	9
±	26	81.5	0.0	0.0	chr1_Arabidopsis	15266	15319	(3484611)	+	AT_rich	Low_complexity	1	54	(0)	10
±	26	57.7	0.0	0.0	chr1_Arabidopsis	15296	15321	(3484609)	+	AT_rich	Low_complexity	1	26	(0)	11
±	180	0.0	0.0	0.0	chr1_Arabidopsis	16804	16823	(3483107)	+	(TTG)n	Simple_repeat	2	21	(0)	12
±	1523	6.5	0.4	0.8	chr1_Arabidopsis	17010	17256	(3482674)	+	ATREP3	RC/Helitron	1	246	(1851)	13 *
±	8985	6.0	0.8	1.7	chr1_Arabidopsis	17256	18642	(3481288)	+	ATREP20	RC/Helitron	27	1471	(673)	14
±	552	4.4	0.0	0.0	chr1_Arabidopsis	18661	18728	(3481202)	+	ATREP20	RC/Helitron	2077	2144	(0)	14
±	28	87.6	0.0	0.0	chr1_Arabidopsis	18796	18900	(3481030)	+	AT_rich	Low_complexity	1	105	(0)	15
±	198	0.0	0.0	0.0	chr1_Arabidopsis	20510	20531	(3479399)	+	(TA)n	Simple_repeat	2	23	(0)	16
±	198	0.0	0.0	0.0	chr1_Arabidopsis	22621	22642	(3477288)	+	(TTTTTA)n	Simple_repeat	6	27	(0)	17
±	333	0.0	0.0	0.0	chr1_Arabidopsis	37736	37772	(3462158)	+	(GAA)n	Simple_repeat	1	37	(0)	18
±	225	0.0	0.0	0.0	chr1_Arabidopsis	41361	41385	(3458545)	+	(TA)n	Simple_repeat	2	26	(0)	19
±	34	77.1	0.0	0.0	chr1_Arabidopsis	42444	42491	(3457439)	+	AT_rich	Low_complexity	1	48	(0)	20
±	35	73.2	0.0	0.0	chr1_Arabidopsis	44763	44818	(3455112)	+	AT_rich	Low_complexity	1	56	(0)	21
±	209	19.1	0.0	0.0	chr1_Arabidopsis	46523	46564	(3453366)	+	CT-rich	Low_complexity	139	180	(0)	22
±	271	24.7	0.0	0.0	chr1_Arabidopsis	50433	50517	(3449413)	+	(CAG)n	Simple_repeat	1	85	(0)	23
±	5236	6.5	0.7	1.1	chr1_Arabidopsis	55676	56576	(3443354)	+	SIMPLEHAT1	DNA/hAT	1	1059	(0)	24
±	234	0.0	0.0	0.0	chr1_Arabidopsis	62347	62372	(3437558)	+	(GAA)n	Simple_repeat	2	27	(0)	25

Fig. 2. An example map file from the start of *Arabidopsis thaliana* chromosome 1.

282	= Smith-Waterman score of the match, (complexity adjusted, dependent on repeat age and GC level)
14.5	= % of divergence = mismatches/(matches+mismatches)
0.0	= % of deletions in the query sequence (deleted bp)
7.2	= % of insertions in the query sequence (inserted bp)
chr1_Arabidopsis	= name of query sequence
1	= starting position of match in query sequence
105	= ending position of match in query sequence
(3499825)	= number of bases in query sequence past the ending position of match
C	= match is with the Complement of the repeat consensus sequence
ATREP18	= name of the matching interspersed repeat
DNA	= the class of the repeat, in this case a DNA transposon
(1142)	= number 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)
649	= starting position of match in repeat consensus sequence
561	= ending position of match in repeat consensus sequence
1	= 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.

Fig. 3. The meaning of each column in the map file.

```
>chr1_Arabidopsis
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNACCCGAAACCGGTTTCTCTGGTTGAAAATCATTGTGTATATAATGATAATTTTATCGTTTTTATGTAATTGCTTATTGTTGTGTGTAGATTTTTT
AAAAATATCATTTGAGGTCAATACAAATCCTATTTCTTGTGGTTTTCTTTCCTTCACTTAGCTATGGATGGTTTATCTTCATTTGTTATATTGGATACAA
GCTTTGCTACGATCTACATTTGGGAATGTGAGTCTCTTATTGTAACCTTAGGGTTGGTTTATCTCAAGAATCTTATTAATTGTTTGGACTGTTTATGTTT
GGACATTTATTGTCATTCTTACTCCTTTGTGGAAATGTTTGTTCTATCAATTTATCTTTTGTGGGAAAATTATTTAGTTGTAGGGATGAAGTCTTTCTTC
GTTGTTGTTACGCTTGTCATCTCATCTCTCAATGATATGGGATGGTCCTTTAGCATTTATTCTGAAGTTCTTCTGCTTGATGATTTTATCCTTAGCCAAA
AGGATTGGTGGTTTGAAGACACATCATATCAAAAAAGCTATCGCCTCGACGATGCTCTATTTCTATCCTTGTAGCACACATTTTGGCACTCAAAAAAGTA
TTTTTAGATGTTTGTTTTGCTTCTTTGAAGTAGTTTCTCTTTGCAAAATTCCTCTTTTTTTAGAGTGATTTGGATGATTCAAGACTTCTCGGTACTGCAA
AGTTCTTCCGCCTGATTAATTATCCATTTTACCTTTGTCGTAGATATTAGGTAATCTGTAAGTCAACTCATATACAACTCATAATTTAAAATAAAATTAT
GATCGACACACGTTTACACATAAAATCTGTAAATCAACTCATATACCCGTTATTCCCACAATCATATGCTTTCTAAAAGCAAAAGTATATGTCAACAATT
GGTTATAAATTATTAGAAGTTTTCCACTTATGACTTAAGAACTTGTGAAGCAGAAAGTGGCAACANNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNAAA
TTGAGAAGTCAATTTTATATAATTTAATCAAATAAATAAGTTTATGGTTAAGAGTTTTTTACTCTCTTTATTTTTCTTTTTCTTTTTGAGACATACTGAA
AAAAGTTGTAATTATTAATGATAGTTCTGTGATTCCTCCATGAATCACAT
```

Fig. 4. Example of a masked sequence. This sequence is the beginning of Arabidopsis chromosome 1.

Summary:

```
==================================================
file name: RM2_temp.fna_1298535356
sequences:            1
total length:    3499930 bp  (3499930 bp excl N/X-runs)
GC level:         37.31 %
bases masked:      91778 bp ( 2.62 %)
==================================================
               number of      length   percentage
               elements*    occupied  of sequence
--------------------------------------------------
Retroelements           24         7427 bp    0.21 %
   SINEs:                1          869 bp    0.02 %
   Penelope              0            0 bp    0.00 %
   LINEs:               12         4125 bp    0.12 %
    CRE/SLACS            0            0 bp    0.00 %
      L2/CR1/Rex         0            0 bp    0.00 %
      R1/LOA/Jockey      0            0 bp    0.00 %
      R2/R4/NeSL         0            0 bp    0.00 %
      RTE/Bov-B          0            0 bp    0.00 %
      L1/CIN4           12         4125 bp    0.12 %
   LTR elements:        11         2433 bp    0.07 %
      BEL/Pao            0            0 bp    0.00 %
      Ty1/Copia          8         1810 bp    0.05 %
      Gypsy/DIRS1        3          623 bp    0.02 %
        Retroviral       0            0 bp    0.00 %

DNA transposons         44        25896 bp    0.74 %
   hobo-Activator        9         4483 bp    0.13 %
   Tc1-IS630-Pogo        6         2074 bp    0.06 %
   En-Spm                1           67 bp    0.00 %
   MuDR-IS905           16        17023 bp    0.49 %
   PiggyBac              0            0 bp    0.00 %
   Tourist/Harbinger     7         1687 bp    0.05 %
   Other (Mirage,        0            0 bp    0.00 %
    P-element, Transib)

Rolling-circles          0            0 bp    0.00 %

Unclassified:           35        22581 bp    0.65 %

Total interspersed repeats:       55904 bp    1.60 %


Small RNA:               0            0 bp    0.00 %

Satellites:             13         4286 bp    0.12 %
Simple repeats:        273         9639 bp    0.28 %
Low complexity:        472        21949 bp    0.63 %
```

Fig. 5. Table of summary result. RepeatMasker sums the number of occurrences and the number of nucleotides of each "superfamily" detected. A superfamily is defined here by A.R.A Smith and corresponds to the definition of Wicker et al. (3).

4.2.3. Summary Results

RepeatMasker returns a .tbl file containing a plain text in the form of a table. The file summarizes the number of identified repeats clustered in superfamilies and sums the nucleotides that belong to each superfamily (Fig. 5). A superfamily is mainly defined as described by Wicker et al. (3). Because it is extremely difficult to distinguish which

repeat fragments are derived from the same insertion event of a transposable element, there is a slight overestimate of the copy numbers. There are also differences in the number of ''bases masked'' and the sum of the bases annotated in this summary file. For example, the fragments shorter than 10 bp are not annotated but are masked. Note that the classification is not based on the target size duplication sizes (for example, a TA for a Tc1/Mariner element), but on the proteins coded by the repeat or, whether these are not known in the classification, on the extremity matches (i.e., inverted terminal repeat (ITR) or long terminal repeat (LTR). I also noted the classification of some superfamilies are more precise than others, especially mammalian repeats are very well annotated. These differences are more precisely described in Subheading 5.4 about algorithm identification of repeats. We observed this classification was not updated at least since 2006, especially, it does not mention the Polinton/Marverick superfamily (4).

5. RepeatMasker Main Functions

Before describing how to use the options of RepeatMasker, the main functions are presented during a standard run (i.e., without option). DateRepeats and DupMasker are two binaries installed with RepeatMasker and not used in standard run. The user can use them separately.

Figure 6 shows the main PERL scripts involved in Repeat Masker execution and the chronological order they are used (the numbers adjacent to the arrows). The execution of RepeatMasker can be mainly split into seven different steps:

- Verify hit point to the valid alignment tool.
- Read and check the input sequence(s).
- Check the RepeatMasker library or the user transposable elements library.
- Split the sequence(s) into fragments and prepare a list of execution on the different fragments.
- Launch the alignment tool on the sequences.
- Change the search engine output into RepeatMasker standard output.
- Merge the fragment sequences and merge the fragmented hits of transposable elements.

5.1. RepeatMasker

The first task of RepeatMasker is to verify its own configuration. RepeatMasker loads and checks all binary paths found in the module RepeatMaskerConfig.pm (number 1 in Fig. 7). RepeatMasker

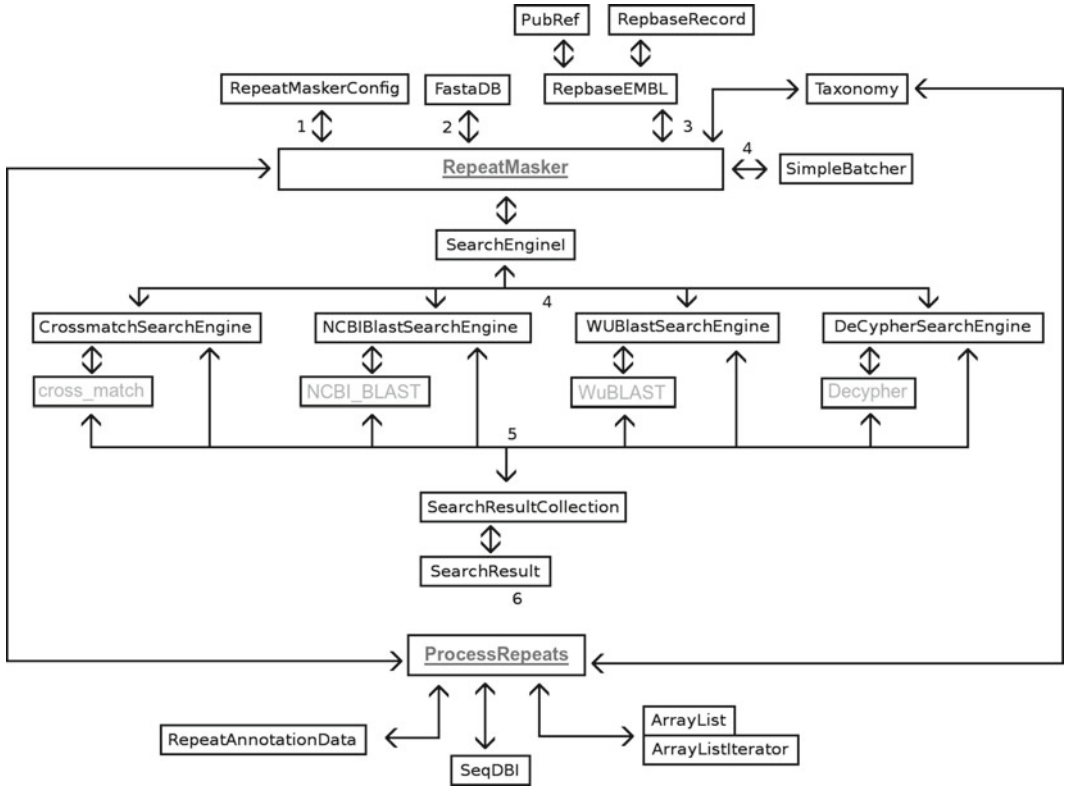

Fig. 6. Main organization of RepeatMasker binaries and libraries. These files are used in all RepeatMasker executions. The *arrows* correspond to the relationship between two files. The *numbers* indicate the main chronological order where the files are used. The binary files of RepeatMasker are written in *gray underline color*, the search engine binaries are written in *light gray color*.

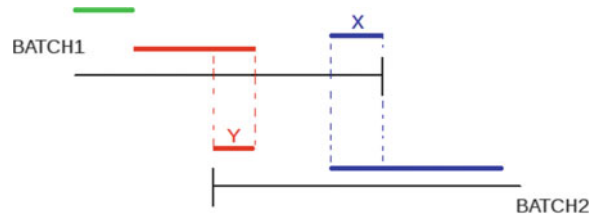

Fig. 7. Example of duplicate hits on two fragmented batch sequences.

initializes some parameters as the size of fragment that will run with the alignment tool and loads the date for creating the output directory for each run.

The second task of RepeatMasker is to check the input sequence (number 2 in Fig. 7). The FASTADB.pm module loads and checks many parameters: the number of FASTA sequences, the length, the number of G and C nucleotides (calculates the GC ratio in the FASTA sequence), and the number of X and/or N nucleotides.

The module checks whether the FASTA sequence is valid and splits the sequences into short overlapping fragments (their length depends on the search engine parameters). The length of the overlap is 2,000 bp.

The third task of RepeatMasker is to check the database file that contains the repeat elements (transposable elements and satellites; number 3 and 4 in Fig. 7). The user can use the RepeatMasker library downloaded from Genetic Information Research Institute (http://www.girinst.org) or can use its own library. If the user provides a custom database, RepeatMasker checks the format of the database. With the use of custom library, RepeatMasker cannot fill the summary table, except the value of GC percentage and the length of the sequence. If the user uses the RepeatMasker library, the RepbaseEMBL.pm, RepbaseRecord.pm and PubRef.pm modules extract the information about the repeats. Section 4.2 details the information extracted from the database. In this case, the user must specify to which species its sequences belong (option '-specie') or the taxonomy order, if RepeatMasker does not recognize the specified species. For example, the option '-specie bear' does not work because the beard genome is not recognized but the user can enter '-specie mammal' instead.

The fourth task of RepeatMasker creates and writes the list of command batch of each fragmented sequence (SimpleBatcher.pm) according to the search engine chosen. Five PERL modules convert the option of RepeatMasker into Search Engine options and write them in the command batch: SearchEngineI.pm, CrossmatchSearchEngine.pm, DeCypherSearchEngine.pm, NCBI BlastSearchEngine.pm, and WUBlastSearchEngine.pm. Their name is correlated to the search engine they control. Note that the search engine parameters cannot be changed directly by the user, but are calculated based on RepeatMasker parameters. For example, the minimal size of the exact words found by Decypher, NCBI-BLAST and WU-BLAST is 14., and WU-BLAST uses BLASTP with "`-warnings -T=1000000 -p=1 -hspmax=2000000000 -gi V=0 B=100000000 Q=12 R=2 W=14 S=30 gapS2=30 S2=15 X=30 gapX=60`" parameters.

After the runs of search engine, the fifth task of RepeatMasker is to convert the different outputs in a standard output: cross_match output (Figs. 2 and 3). The only task left to RepeatMasker is to run ProcessRepeats.

5.2. ProcessRepeats

ProcessRepeats follows the RepeatMasker application. It organizes and processes the results by assembling and sorting the fragmented repeats. Many repeat subfamilies are similar to each other and could be close in the sequence, thus the main difficulty of ProcessRepeats algorithm is to merge fragmented repeats and to assign appropriate subfamily names to the search engine output.

The main algorithm of ProcessRepeats is:

- Read the options
- Load the information data (specie, GC content…)
- Remove duplicated hit present in many fragment sequences
- Assemble the sequence fragmented by RepeatMasker
- Sort all repeats hits
- Join the fragmented repeats subject to some parameters
- Removing insignificant fragments
- Write the (optional) annotation output files

The first task of ProcessRepeats is to read and check the parameters users. These parameters will define the ProcessRepeats tasks. For example, the option '-gff' will write a gff output file and the option '-noint' will skip all tasks involving transposable elements.

The second task loads and checks the information about the query sequence and the library. For example, if the user works on Drosophila genomes and enter it with the '-specie' option, ProcessRepeats will not merge of the mammalian long interspersed nuclear elements (LINEs). The GC content (%) calculated by RepeatMasker is a very important information which determines the minimal threshold of many repeats and adjusts the Smith-Watermann score of each hit.

The third task of ProcessRepeats is to remove the duplicated hit present in two batch sequences (Fig. 7): when the same repeat is present in two hits on two batch files, ProcessRepeats removes the one on the edges of the batch file.

The fourth task is the most difficult. It consists in merging the fragmented repeats. This task is subdivided based on the A.R.A. Smith defined superfamilies: DNA transposons, short interspersed nuclear elements (SINEs), LTR retrotransposons, LINEs, and recombinant LINEs.

The fifth task removes the insignificant hits. The parameters and the information data allow ProcessRepeats to calculate the minimal threshold. The insignificant hits are those that have a Smith-Watermann score lower than this threshold. This threshold is different for each superfamily.

Finally, the last task of ProcessRepeats is to write the assembled repeats in the different output files according to the options.

5.3. DateRepeats

The goal of the script DateRepeats is to know if a repeat, present in one specific species, is expected to be present in another one. Actually, DateRepeats works only with the mammal species. DateRepeats takes a RepeatMasker .out file. The user must choose the species of the repeats in RepeatMasker .out file and choose the compared species. For example, if a user wants to know whether the repeats in the RepeatMasker map file from mouse genome are

also present in the rat and the human genome, it will be able to process as follows:

```
DateRepeats MouseRepeat.out -q mouse -c rat -c human
```

DateRepeats analyses the MouseRepeat.out file that contains mouse transposable elements and checks if they exist in rat and human genome. DateRepeats prints out the following output file:

```
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) + L1Mur2
 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
```

DateRepeats adds the two last columns for the rat and the human respectively. The X indicates that the repeat is expected to be present at orthologous sites, while the O predicts an absence.

5.4. DupMasker

DupMasker annotates segmental duplications in query sequence. It analyses this file against a library known segmental duplications. If the map file does not exist, DupMasker will launch RepeatMasker to create this file. DupMasker writes a '.duplicons' output file.

5.5. RepeatMasker Library

The user can use the RepeatMasker library downloaded from Genetic Information Research Institute (http://www.girinst.org) or can use its own library. The RepeatMasker library has an EMBL-like format. In addition to the sequence, the EMBL format gives some important information. Each lane starts with a 'two-letter code' that defines the type of line. An example from the library is written below:

```
ID AluJb repbase; DNA; PRI; 283 BP.
XX
AC .
XX
DT 20-AUG-1998 (Rel. 1, Created)
DT 20-AUG-1998 (Rel. 1, Last updated, Version 1)
XX
DE Alu-Jb subfamily - a consensus.
```

```
XX
KW   SINE1/7SL;   SINE;   Non-LTR   Retrotransposon;
Transposable Element;
KW Alu-J; Alu-Jb; AluJ; AluJb; Repetitive sequence.
XX
OS Primates
OC   Eukaryota;   Metazoa;   Chordata;   Craniata;
Vertebrata; Euteleostomi;
OC Mammalia; Eutheria; Euarchontoglires.
XX
RN [1]
RA Jurka J.;
RT   "Origin   and   evolution   of   Alu   repetitive
elements.";

RL Molecular Biology Intelligence Unit:The impact of
short
RL interspersed elements (SINEs) on the host genome
(ed. Richard J.
RL Maraia), R.G. Landes Company, Austin, pp.25-41
(1995).
XX
DR [1] (Consensus)
XX
SQ Sequence 283 BP; 59 A; 82 C; 98 G; 44 T; 0 other;
ggccgggcgc    ggtggctcac    gcctgtaatc    ccagcacttt
gggaggccga ggcgggagga 60
tcacttgagc    ccaggagttc    gagaccagcc    tgggcaacat
ggtgaaaccc cgtctctaca 120
aaaaatacaa    aaattagccg    ggcgtggtgg    cgcgcgcctg
tagtcccagc tactcgggag 180
gctgaggcag    gaggatcgct    tgagcccggg    aggtcgaggc
tgcagtgagc cgtgatcgcg 240
ccactgcact ccagcctggg cgacagagcg agaccctgtc tca 283
```

The first line, starting with the 'ID' identifier, gives the name of the repeats (here AluJb). This line also gives the size of the consensus sequence. The fourth line, starting with 'DT' (Date), gives the date when the repeat was available in Repbase database (5). The 'DE' line gives a short definition of the repeat and the KW (KeyWord) informs about the class and superfamily of the repeat. The 'OS' and the 'OC' lines give the organism where this repeat was first found and the taxonomy of this specie. 'RA', 'RT,' and 'RL' lines inform about the author and the article where it was first mentioned. The 'SQ' line indicates the size and the ratio of each nucleotide in the sequence. The following lines are the sequence itself.

5.6. Search Engine Software

5.6.1. AB-BLAST (Old WU-BLAST)

Basic Local Alignment Search Tool (BLAST) was developed by Stephen Altschul, Warren Gish, and David Lipman at the U.S. National Center for Biotechnology Information (NCBI), Webb Miller at the Pennsylvania State University, and Gene Myers at the University of Arizona (1). The software WU-BLAST, that is an alternative implementation of BLAST, was acquired from Washington University by Warren R. Gish. The right to license the software to the community was acquired by Advanced Biocomputing, LLC in 2009 (http://blast.advbiocomp.com/) (6). The user can use the old WU-BLAST software, but it is no more available on AB-BLAST Web site. AB-BLAST is free for academic uses. AB-BLAST uses the same core algorithm of WU-BLAST: the binaries kept the same name and the same options. For information, the main difference between AB-BLAST and WU-BLAST is the match/mismatch scores M=+1 N=−3 with gap penalties Q=7 R=2; whereas WU-BLAST uses M=+5 N=−4 with gap penalties Q=10 R=10 by default. This difference changes the hits score and could identify new repeats that the old version misses and vice versa.

5.6.2. RMBLAST

RMBlast is a RepeatMasker compatible version of the standard NCBI BLAST (7). It is a free software that can be downloaded at the RepeatMasker Web site (http://www.repeatmasker.org/RMBlast.html). RMBlast supports some new features such as custom matrices (without KA-Statistics) and cross_match-like complexity adjusted scoring.

5.6.3. Cross_Match

Cross_Match was implemented by Phil Green (http://www.phrap.org/phredphrapconsed.html). This software is part of the Phred/Phrap/Consed package and it is free for academic use but the user must email the information requested in the academic user agreement to David Gordon and receive an email reply before downloading the search engine. Cross_Match is the search engine reference in output format, and some options are specific to it.

5.6.4. Decypher

DeCypher, so-called DeCypherSW, is a software that belongs to "TimeLogic biocomputing solutions." If it could be found in RepeatMasker configuration, it is not listed on RepeatMasker Web site as a search engine tool for RepeatMasker.

6. Options

RepeatMasker's author classifies the options in five categories: Species, Contamination, Masking, Output, and Speed options. I will precise which options are also present on WEBRepeatMasker.

```
-h(elp) Detailed help
```

This option prints all options present in the RepeatMasker command line.

6.1. Species Options

The two following options select the custom library or the part of RepeatMasker library using by RepeatMasker. The goal of these two options is limited to the number of unnecessary alignment against the input sequence. For example, it is useless to search for human Alu in plant genomes.

```
-species <query species>
Specify the species or clade of the input sequence. The
species name must be a valid NCBI Taxonomy Database
species name and be contained in the RepeatMasker
repeat database. Some examples are:
-species human
-species rattus
-species "ciona savignyi"
Other commonly used species:
mammal, carnivore, rodentia, rat, cow, pig, cat, dog,
chicken, fugu,danio, "ciona intestinalis" drosophila,
anopheles, elegans, diatoaea, artiodactyl, arabidopsis,
rice, wheat, and maize
```

This option is present on WEBRepeatMasker. The user must choose the species (or the clade) to which the input data belongs. For example, if the input data is a human chromosome, the user can type "human," "homo sapiens," or "mamma." RepeatMasker

does not distinguish between uppercase and lowercase words. The user can use multiple words to define a species like "homo sapiens." Quotation marks must be used before and after the multiple words. The PERL script queryRepeatDatabase.pl in the util directory checks whether the specie's name that the user enters is available in RepeatMasker library.

```
-lib [filename]
Allows use of a custom library (e.g. from another
species)
```

The user can enter a custom library with this option, especially if the studied specie is not covered by RepeatMasker. The library sequence must be in FASTA format. Information of the class repeat can be added after the repeat name: `>repeatname#class`. The symbol '#' indicates the name of the repeat class is after the repeat name. The user can also combine the custom sequence and RepeatMasker library to create a custom library.

6.2. Contamination Options

By default, RepeatMasker looks for bacterial insertion sequences (IS elements). Here, the options control the research of contamination. I noticed the search of IS elements generally takes less than one second per fragment sequence.

```
-no_is
Skips bacterial insertion element check
```

This option skips the research of IS elements in the query sequence. The search of interspersed repeats does not change.

```
-no_is
Only clips E coli insertion elements out of FASTA and
.qual files
```

This option limits the run of RepeatMasker to check only the IS elements, it does not search the other repeats.

```
-no_is
Clips IS elements before analysis (default: IS only
reported)
```

This option removes the IS elements found in the input sequence before the standard runs of RepeatMasker. It writes out a '.withoutIS' file that correspond to the unmasked query sequence without the IS elements. Note the coordinates of repeat elements in the map file correspond to the withoutIS file and not to the original query.

```
-rodspec
Only checks for rodent specific repeats (no
repeatmasker run)
-primspec
Only checks for primate specific repeats (no
repeatmasker run)
```

These two previous options check the contamination of rodent repeats in primate and vice versa. As notified below, there is no RepeatMasker run against the library. There is no option to check the contamination of non-mammal species. These options are mainly created by A.R.A Smith to check if the sequence belongs to rodent species or primate species.

6.3. Masking Options

These options change all standard output files: the map file changes according to the options, and the masked file and the summary file change according to the map file.

```
-cutoff [number]
Sets cutoff score for masking repeats when using -
lib (default 225)
```

For each repeat hit, a Swith-Watermann score is calculated (Column 1 in Fig. 3). Usually, the hit written in the map file has a Swith-Watermann score higher that a threshold. This value equals 225 by default. Using a local library (option '-lib'), the user can choose this option and change the threshold value. Decreasing the value increases the number the false matches and vice versa. The author stipulates that below 200, the result contains false matches and up to 250 the results give only real matches. The user must know that the low-complexity region in repeats sequences such as non-autonomous elements increase false matches.

```
-nolow /-low
Does not mask low_complexity DNA or simple repeats
```

This option removes the low-complexity DNA matches from all output files.

```
-nolow /-low
Only masks low complex/simple repeats (no interspersed
repeats)
```

On the contrary, this option removes all matches excepted to be low-complexity DNA matches.

```
-alu
Only masks Alus (and 7SLRNA, SVA and LTR5)(only for
primate DNA)
```

The Alu option removes all repeat from the map file excepted Alu families hits. This options only works with the primate sequence, i.e., when the user tapes the option '-specie' followed by human, primate, monkey …

```
-div [number]

Masks only those repeats < x percent diverged from

consensus seq
```

Each line in the map file contains the divergence percentage from consensus sequence (second column in Figs. 2 and 3). This option creates a user's threshold for the maximal value of divergence between the hit and the corresponding consensus sequence in the library (Repbase or custom).

```
- norna

Does not mask small RNA (pseudo) genes
```

By default, RepeatMasker screens the input sequence to look for the small polIII transcribed RNAs (mostly transfer RNA (tRNAs) and small nuclear RNA (snRNAs)) because they are very similar to SINE elements. This option does not mask the identified pseudogenes and small RNA genes.

6.4. Output Options

These options do not change RepeatMasker or the map file runs but adds new output file or the format of the output files.

```
-a(lignments)

Writes alignments in .align output file
```

This option creates a supplementary output file. All alignment hits from the search engine are saved in a '.align' file (Fig. 8). 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. Note it exists some differences between the alignment file and the map file. The map file is produced by ProcessRepeats that the main task is to defragment the original map file and the alignment file is created from the original map file: the difference between them comes from the defragmented hits.

```
-inv

Alignments are presented in the orientation of the

repeat (with option -a)
```

This option works only with the previous option. By default, the alignment file keeps the orientation of the query sequence. With this option, the alignment has the same orientation than that of the consensus repeat in the library

```
-x

Returns repetitive regions masked with Xs rather than

Ns
```

```
Matrix = 20p35g.matrix
Transitions / transversions = 1.29 (9 / 7)
Gap_init rate = 0.03 (3 / 92), avg. gap size = 1.33 (4 / 3)

   273  14.51 0.00 7.22  chr1_Arabidopsis        2     105 (3499825) C ATREP18       DNA                (375)   1416    1320

  chr1_Arabidop        2 CCTAAACCCTAAACCCTAAACCCTAAACCTCTGAATCCTTAATCCCTAAA 51
                                                 v -  i      v  -
C ATREP18#DNA       1416 CCTAAACCCTAAACCCTAAACCCTAAAGC-CTAAATCCTAAA-CCCTAAA 1369

  chr1_Arabidop       52 TCCCTAAATCTTTAAATCCTACATCCATGAATCCCTAAATACCTAATTCC 101
                         -       i   -   i    v -- v v     v  i i  -   vi
C ATREP18#DNA       1368 -CCCTAAACCTT-AAACCCTAAA--CTTTAATCCATAGACAC-TAAGCCC 1324

  chr1_Arabidop      102 CTAA 105
                         i
C ATREP18#DNA       1323 TTAA 1320


Matrix = Unknown
Transitions / transversions = 0.50 (1 / 2)
Gap_init rate = 0.00 (0 / 42), avg. gap size = 0.0 (0 / 0)

   267   7.89 0.00 0.00  chr1_Arabidopsis     4291    4328 (3495602) C (TA)n        Simple_repeat         (1)    179     142

  chr1_Arabidop     4291 ATATATATATATATATATATATGTGGATATATATATAT 4328
                                           i iv
C (TA)n#Simple_      179 ATATATATATATATATATATATATATATATATATATAT 142
```

Fig. 8. An example of optional alignment file of the query with the matching repeats. The alignments are in the cross_match/SWAT format, whereby mismatches rather than matches are indicated: transitions with an i and transversions with a v.

This option replaces the N character in masked file by X character.

```
-xsmall
```

```
returns repetitive regions in lowercase (rest capitals)
```

```
rather than masked
```

It does not use the N or X characters for the repeat hit but writes in lowercase for the repeats and in uppercase otherwise.

```
-small
```

```
Returns complete .masked sequence in lower case
```

This option writes out the masked sequence in lower case instead of uppercase.

```
-poly
```

```
Reports simple repeats that may be polymorphic (in
```

```
file.poly)
```

Creates a new output file that contains the list of polymorphic microsatellites. The extension name is '.polyout'. The polymorphic microsatellites correspond to the (X)n satellites in map file. For example, (C)n and (TA)n hit present in Fig. 2 are present in the '.polyout' 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
output
```

The '-xm', '-ace,' and '-gff' options create an additional output file in cross match, ACeDB, and Gene Feature Finding format respectively.

```
-u
Creates an additional annotation file not processed by
ProcessRepeats
```

This option creates an additional '.ori.out' file before the ProcessRepeats run. The difference between this new file and the previous map file mainly comes from the fragmented repeats: the fragmented ones correspond to many lines in this file and only one line after the ProcessRepeats run. The '.ori.out' also contains the repeat hits removed by ProcessRepeats: for example, a small hit included between a fragmented repeat should be removed. Then the user can compare and/or check the defragmented repeat hit in the query sequence.

```
-fixed
Creates an (old style) annotation file with fixed width
columns
```

It creates another map file with fixed width column. Regardless of the repeats query name, it simplifies the creation of generic scripts that can parse this file but this option can shorten a long name and create ambiguous names.

```
-no_id
Leaves out final column with unique ID for each element
(was default)
```

Since September 2000, a column displaying a unique number (ID) for each repeat is printed by default. A fragmented single element has the same number that allows better interpretation of the data. This option removes the ID column.

```
-e(xcln)
Calculates repeat densities (in .tbl) excluding runs of
>=20 N/Xs in the query
```

This option calculates the summary table file (.tbl), excluding the runs of equals or more consecutive N or X characters. This option cans change the proportion of TEs families in sequences containing a lot of N characters (for example draft sequences).

```
-noisy
Prints search engine progress report to screen
(defaults to .stderr file)
```

The option '-noisy' prints a progress report on the user's screen instead of writing a .stderr file.

```
-dir [directory name]
Writes output to this directory (default is query file
directory,
"-dir ." will write to current directory).
```

By default, the output files are written in the directory that contains the query sequence. With this option, RepeatMasker writes the output files in the user directory. Note that the directory is not created by RepeatMasker but it just checks if it exists. Note that during the runs, RepeatMasker creates a temporary directory under the format 'RM XXX.Date', where XXX is the random number and Date is the date the user runs RepeatMasker. This temporary directory is deleted in the end of the run.

6.5. Speed and Search Options

These options change indirectly the options of the search engine tools or the way ProcessRepeat assemble and store the repeats before writing the output files.

```
-e(ngine) [crossmatch|wublast|abblast|ncbi|decypher]
Use an alternate search engine to the default.
```

In the configuration step the user chooses a default search engine and may choose other search engines. This option selects the search engine from the list of configured search engines and use the selected ones for this RepeatMasker run. Note that it is possible to select NCBI BLAST, even if it is not possible to select NCBI BLAST in automatic configuration (see paragraph 3).

```
-pa(rallel) [number]
The number of processors to use in parallel (only works
for batch files or sequences over 50 kb)
```

This option determines the number of simultaneous search engine runs. The limit number of parallel is 16.

```
-s Slow search; 0-5% more sensitive, 2-3 times slower
than default
-q Quick search; 5-10% less sensitive, 2-5 times faster
than default
-qq Rush job; about 10% less sensitive, 4->10 times
faster than default
```

The three previous options (-s, -q, and -qq) change the speed of RepeatMasker runs. The increase of speed depends on the input sequence and the library sequence. The scale of speed increase given by the author was verified in my experience. The speed comes from the parameters given to the search engine. In fact, these three options change the minimal size of the initial word (called seed in BLAST algorithm) used by the search engine: a search engine, like BLAST, looks for an exact word of defined size contained in the library sequence on a sequence query. When this exact word (seed) is found, the algorithm tries to extend the seed by allowing gap and mutations with the query sequence. The size of the seed depends on the search engine and the parameter given by the user. For more explanations, read the Altschul et al. algorithm (1). The more the seed size increases, the less the algorithm finds the seed in the query sequence. For example, the seed size in RepeatMasker script is 9 for normal speed, 8 for slow search, 10 for quick search, and 11 for the rush job option. These values correspond to the default repeats, and they change according to the type of repeats.

As stated by the author, the sensitivity of RepeatMasker decreases according to the speed options. The loss of sensitivity depends on the type of repeats. More precisely, it depends on the repeat age. A transposable element generally accumulates mutations when it becomes older. These mutations decrease the size of the words that are the exact copy with the consensus sequence. As a consequence, an old family of transposable is generally more difficult to find with a larger seed than a younger family.

```
-frag [number]
Maximum sequence length masked without fragmenting
(default 60000,300000 for DeCypher)
```

This option changes the size of batch fragment. The minimal size of the fragment must be twofold longer than the size of the overlap fragment. This corresponds to 4,000 nucleotides. A smaller fragment can improve the detection of transposable elements when the fragment contains large regions of DNA with significantly different GC contents. The maximal size of the fragment depends on the amount of memory used. A larger fragment decreases the error of defragmentation hits by ProcessRepeat. However, if your computer does not have sufficient memory, RepeatMasker will

redo the failed search with a longer seed that will decrease the sensitivity of your research (see above).

```
-gc [number]
Use matrices calculated for 'number' percentage
background GC level
-gccalc
RepeatMasker calculates the GC content even for batch
files/small seqs
```

These two parameters change the matrices used by RepeatMasker to identify the transposable elements. Normally, RepeatMasker calculates the average GC content (%) of the query sequence and uses the corresponding matrices (see paragraph 7) to identify the repeats. In some cases, transposable elements have a different GC content that the rest of the genome sequence and the RepeatMasker algorithm uses this difference to detect the repeats. The -gc [number]' option forces RepeatMasker to use a defined GC percent instead of the calculated GC content. The different CG contents that can be applied by RepeatMasker vary from 35 to 53%. The 'gccalc' option calculates the GC content of each fragment instead of the whole query sequence.

```
-nopost
Do not postprocess the results of the run ( i.e. call
ProcessRepeats). NOTE: This option should only be used
when ProcessRepeats will be run manually on the
results.
```

This option skips the ProcessRepeats step. The fragmented hits are not assembled with this option.

References

1. Altschul SF, *et al.* (1997) Gapped Blast and Psi-Blast: a new generation of protein database search programs. J Nucleic Acids Res 25: 3389–402
2. Benson G (1999) Tandem repeats finder: a program to analyze DNA sequences. Nucl Acids Res 27:573–580
3. Wicker T, et al. (2007) A unified classification system for eukaryotic transposable elements. Nat Rev Genet 8:973–82
4. Kapitonov VV, Jurka J (2006) Self-synthesizing DNA transposons in eukaryotes. Proc Natl Acad Sci USA 103:4540–4545
5. Jurka J, et al. (2005) Repbase Update, a database of eukaryotic repetitive elements. Cytogenet Genome Res 110, 462–467
6. Gish W (1996–2009) http://blast.advbiocomp.com
7. Johnson M, et al. (2008) NCBI BLAST: a better web interface. Nucl Acids Res 36:W5-W9

http://www.springer.com/978-1-61779-602-9

Mobile Genetic Elements
Protocols and Genomic Applications
Bigot, Y. (Ed.)
2012, XI, 308 p., Hardcover
ISBN: 978-1-61779-602-9
A product of Humana Press