HOW TO RUN TRAP

TRAP requires Tandem Repeats Finder (TRF - http://tandem.bu.edu/trf/trf.html) and is compatible with versions 3.21 and 4.00 of the software. Please remember to have an installed version of TRF on your server before running TRAP.

Feature table files generated by TRAP can be easily inspected and edited using Artemis (version 7). You can download Artemis from the Sanger Institute web site at the address http://www.sanger.ac.uk/ Software/Artemis/. GFF files can be visualized on Artemis and edited on Apollo (http://www.fruitfly.org/annot/apollo/), a genome annotation viewer and editor.

1. Running TRAP with parameters specified on the command line:

To run TRAP, you should type the following command:

```
trap.pl –i <input_prefix>
```

Note that `–i` is the only mandatory argument. It specifies the prefix name of the TRF output files, including the relative or complete path. TRF generates a series of output files, all with the same prefix, that is, the name of original input file. The `input_prefix` parameter is this prefix. For instance, if TRF was originally run on the file `assembly_20_05_05`, the output files will have names like assembly`_20_05_05.s1.2.5.7.80.10.25.1000.1.html`. TRAP only requires the user to type the file prefix.

In addition, the following arguments and flags can also be specified on the command:

- `–od <output_directory_name>`: name of the output directory to be created, that will store TRAP's output files. If the output directory name is not specified, `TRAP_dir#` will be used by default. If TRAP determines that `TRAP_dir1` already exists, it will create a directory named `TRAP_dir2`, `TRAP_dir3`, and so forth. If the specified directory exists, files will be potentially overwritten.
- `–of <output_file_name>`: name of the prefix of the output files created by TRAP. If the prefix file name is not specified, `TRAP_file#` will be used by default. If TRAP determines that `TRAP_file1` already exists, it will create files named `TRAP_ file2`, `TRAP_file3`, and so forth. If the specified file exists, it will be overwritten.
- `–min <integer>`: defines the minimum period size of the repeats selected by TRAP (default = `1`). The value should be an integer. If, for instance, one specifies 4, repeat *loci* containing the sequences TATATATA or CAGCAGCAGCAG will not be reported by TRAP.
- `–max <integer>`: defines the maximum period size of the repeats selected by TRAP (default = `2000`). The value should be an integer. Only repeat *loci* presenting period sizes in the range defined by `–min` and `–max` will be selected and stored by TRAP. Because TRF has an upper period size limit of 2,000, TRAP only accepts `–max` values up to this limit, otherwise an error will be reported.

- `-cpmin <integer>`: defines the minimum number of repetitive units for a *locus* to be reported by TRAP (default = `2`).
- `-cpmax <integer>`: defines the maximum copy number of repetitive units for a locus to be reported by TRAP (default = `no upper limit`).
- `-fs <integer>`: defines the minimum size of the 5' and 3' regions flanking the repeat locus. Only repeat loci presenting flanking regions above the value specified in this parameter will be reported by TRAP. Because TRF has an upper flanking region size limit of 500 bp, TRAP also accepts –fs values up to this limit. This parameter is especially useful if one intends to use TRAP's output for primer design purposes.
- `-id <integer>`: defines the minimum accepted percentage of matches between adjacent repeat units overall, for a repeat locus to be reported by TRAP.
- `-seq <nucleotide_sequence_of_repeat_unit>`: defines the nucleotide sequence of the repeat unit, as a criterion for TRAP to report a repeat loci. TRAP takes into account different reading frames and the reverse complementary sequences of the selected repeat sequence. Thus, by specifying CTG as the nucleotide sequence, one selects repeats containing the following sequences: CTG, TGC, GCT, CAG, AGC and GCA.
- `-tbf <option>`: defines the format of the summary and complete tables generated by TRAP.
- Summary tables list all repeats sorted by increasing period size classified in increasing order, according to the period size. Information depicted on the tables includes the total number of repeat *loci*, total number of repeat units and the average number of repeat units per locus. At the end of the summary table, TRAP reports the total number of repeat loci, repeat bases and repeat units. In addition, regions containing redundant repeats (see parameter `–rr` below) are identified and a calculus of the number of repeat bases excluding redundancy is also displayed.

  Complete tables report all repeat loci grouped according to the repeat unit sequence. In this case, TRAP also takes into account different reading frames and the reverse complementary sequences of the repeat sequences. Information depicted on these tables includes the total number of bases, total number of repeat *loci*, total number of repeat units, average number of repeat units per *locus*, period size and repeat sequence. Sorting criterion for complete tables is defined by the user, according to the parameter `-sort <option>` (see below).

  Options for `–tbf` parameter:

  - `html –` generates output files in HTML format
  - `csv –` generates output files in comma-separated value (CSV) format
  - `html+csv –` creates both HTML and CSV output files
- `-sort <option>:` defines the sorting criterion for displaying the repeat *loci* selected by TRAP on the complete tables.
  - `bases –` sorts the tables according to the total number of repeat bases (default)
  - `loci -` sorts the tables according to the total number of repeat *loci*
  - `units -` sorts the tables according to the total number of repeat units
  - `average -` sorts the tables according to the average number of repeat units per *locus*
  - `size -` sorts out the tables according to the period size of the repeat units

- `-trf`: creates output files that resemble the original TRF output HTML files, but displaying only the repeat *loci* selected by TRAP, according to user defined specifications.
- `-ff`: generates a FASTA format text file for primer design application. For each selected repeat *locus,* TRAP generates a sequence composed by the nucleotide sequence of the 5' flanking region, a stretch of 20 Ns replacing the original repetitive bases, and the 3' flanking region (the 20Ns are just for indicative purposes).
- `-con`: generates a FASTA text file containing the nucleotide consensus sequences of the repeat unit present in each *locus*. The FASTA header specifies the sequence name and the coordinates of the repeat *locus*.
- `-ft`: creates a feature table file for each input sequence containing repeat *loci* selected by TRAP. This is useful for automated annotation of repetitive *loci*.
- `-gff`: creates a GFF file for each input sequence containing repeat *loci* selected by TRAP. This is useful for automated annotation of repetitive *loci*.
- `-rr`: generates a redundancy report of the selected repeats.

  While the redundancy issue has been partially solved in the current version of TRF, nested repeats are frequently reported for the same *loci* or on overlapping coordinates. For example, a *locus* containing the sequence CAGCAG**CAT**CAGCAG**CAT**CAGCAG**CAT**, could be interpreted as presenting a consensus repeat unit of three bases CAG. On the other hand, TRF could also report an alternative 9 bp long repeat unit presenting the sequence CAGCAG**CAT**. In this example, the same *locus* could be reported as having either 9 copies of a CAG repeat unit or 3 copies of a CAGCAG**CAT** unit. TRAP always identifies such occurrences of nested repeats and subtracts the redundant bases from the calculus of the overall repetitive content of a genome.

  If `–rr` option is used, TRAP will generate HTML files reporting the coordinates of the redundant regions and their respective nested repeats.
- `-help`: prints a help screen

The example below shows how to invoke TRAP specifying all parameters on the command line:

```
trap.pl –i example.fasta –od example_directory –of
example_file –min 2 –max 6 –cpmin 2 –cpmax 20 –seq CAG –fs 100
–id 70 –tbf html+csv –sort size –con -ft -gff
```

- This command will run TRAP using files previously generated by TRF and containing the prefix `example.fasta`. TRAP output will be stored at the `example_directory` directory, and output files will contain the prefix `example_file`. TRAP will select only repeat *loci* containing repeats with period sizes in the range of 2 to 6, with a repeat copy number varying from 2 to 20, with a CAG (and relatives – see `–seq` parameter above) sequence, presenting flanking regions equal to or longer than 100 bp, and whose repeat units present a minimum of 70% of matches between adjacent repeat units overall.
- TRAP will generate HTML and CSV table files sorted according to the period size of the repeats. In addition, TRAP will also generate a FASTA text file containing all the repeat *consensi*, and annotation files (feature table and GFF) that can be loaded onto Artemis and Apollo annotation editors.

2. Running TRAP with parameters specified in a configuration file:

Configuration files have the advantage of serving as a documentation of the processing. Also, storing a configuration file makes it easier to run TRAP again in the future using the same parameters. The configuration file has the added advantage of being easier to read, since parameter names are longer and tend to be self-explanatory.

To run trap using configuration file parameters, you need to use the following command line:

```
trap.pl –c <configuration_file_name>
```

where <configuration_file_name> specifies the name of a file describing the parameters. To each command line parameter described above, there is a corresponding configuration file parameter. We present below a table associating command line parameters to their corresponding configuration file parameters.

Parameter name on command line	Parameter name on configuration file
-i	input_prefix
-od	output_directory
-of	output_file_prefix
-min	minimum_period_size
-max	maximum_period_size
-cpmin	minimum_copy_number
-cpmax	maximum_copy_number
-seq	define_repeat_sequence
-fs	minimum_flanking_region_size
-id	minimum_match_percentage
-tbf	table_format
-sort	sort_field
-con	create_consensus_file
-ft	create_feature_table
-gff	create_gff
-ff	create_flanking_region_file
-trf	create_trf_like_file
-rr	create_redundancy_report

The configuration file is a simple list of entries in the format

```
<parameter name> = <parameter value>
```

The order in which the parameters are specified is free.

As an example, let's create a configuration file for the following command:

```
trap.pl –i example.fasta –od example_directory –of
example_file –min 2 –max 6 –cpmin 2 –cpmax 20 –seq CAG –fs 100
–id 70 –tbf html+csv –sort size –con -ft -gff
```

In this case, the final format of the configuration file would be:

```
input_prefix = example.fasta
output_directory = example_directory
output_file_prefix = example_file
minimum_period_size = 2
maximum_period_size = 6
minimum_copy_number = 2
maximum_copy_number = 20
define_repeat_sequence = CAG
minimum_flanking_region_size = 100
minimum_match_percentage = 70
table_format = html+csv
sort_field = size
create_consensus_file = yes
create_feature_table = yes
create_gff = yes
create_flanking_region_file = no
create_trf_like_file = no
create_redundancy_report = no
```