June, 2001

T−COFFEE (1.35)

MOCCA

User Documentation and F.A.Q.

Cedric Notredame (cedric.notredame@igs.cnrs−mrs.fr)

WARNING

This program comes with no warranty. The code should not be modified and/or redistributed without the permission of the authors.

lalign2list is a modified version of the code by Huang and Miller, it is distributed thanks to the permission given by the authors. It should not be redistributed without their permission.

ADDRESS

Please send comments, bug reports to:
cedric.notredame@igs.cnrs−mrs.fr

DESCRIPTION

T−Coffee is a package for making multiple protein sequence alignments. The current implementation makes use of several facilities taken from publicly available packages.

−The tree reading/computing routines are taken from the ClustalW Package, courtesy of Julie Thompson, Des Higgins and Toby Gibson (Thompson, Higgins, Gibson, 1994, 4673−4680,vol. 22, Nucleic Acid Research).

−The implementation of the algorithm for aligning two sequences in linear space is taken from Myers and Miller, in CABIOS, 1988, 11−17, vol. 1), and adapted from the ClustalW package.

Table of Contents

INTRODUCTION

T−Coffee is a multiple sequence alignment program. Multiple sequence alignment programs are meant to align a set of sequences previously gathered using other programs such as blast, fasta, sw ...

The main characteristic of T−Coffee is that it will allow you to combine results obtained with several alignment methods. For instance if you have an alignment coming from ClustalW, an other alignment coming from Dialign, and a structural alignment of some of your sequences, T−Coffee will combine all that information and produce a new multiple sequence having the best agreement with all these methods.

By default, T−Coffee will compare all your sequences two by two, producing a global alignment and a series of local alignments (using lalign). The program will then combine all these alignments into a multiple alignment.

INSTALLATION

1−decompress distribution.tar.z

uncompress distribution.tar.Z
or
gunzip distribution.tar.gz

2−untar distribution.tar

tar −xvf distribution.tar

3−This will create the distribution directory with the following structure:

distribution/bin
distribution/doc/t_coffee_doc.ps,
t_coffee_doc.pdf, t_coffee_paper.tar in pdf
distribution/t_coffee_source
distribution/lalign
distribution/example
distribution/html

4−Install ClustalW (check before if it is not already installed). You can obtain ClustalW from the E.B.I. Web server (www.ebi.ac.uk) or from the LBM in Strasbourg.

5−Indicate the address and the name of ClustalW on your system:

−either edit the file define_header.h, in the section PROGRAM PATH

#define CLUSTALW_4_TCOFFEE "path/name_of_clustalw"

−or set the global variable CLUSTALW_4_TCOFFEE to "path/name_of_clustalw":

setenv CLUSTALW_4_TCOFFEE "path/name_of_clustalw"

Note: to be permanent, this last change must be set in your login file.

6−go into the main directory and type:

./install

7− ’Installation of t_coffee Successful’ must appear on your screen to indicate a proper completion.

8−add the bin folder to your path, for instance:

set path = ($path . <address of the t_coffee bin folder>)

Note This **must** be added to your login file. An alternative is to move all the binary files into your own bin.

QUICK START

T−COFFEE:

Write your sequences in the same file (Swiss−prot, Fasta or Pir) and type.

t_coffee <your sequences>

This will output two files:

<your sequences>.**aln**
A multiple alignment in a format similar to ClustalW, that can be read by most programs.

<your sequences>.**dnd**
A dendrogram in phylip format

MOCCA

Write your sequences in the same file (Swiss−prot, Fasta or Pir) and type.

mocca <your sequences>

This will output one files (*<your sequences>.mocca_lib)* and start an interactive menu. If you run again mocca, it will start looking for that file and read it, or will generate it if it does not exist.

OVERVIEW

T−Coffee is a versatile tool for making multiple sequence alignments. Its aim is to combine heterogeneous sources of information. The current implementation has been especially designed for combining local and global alignments, respectively from ClustalW and Lalign. It can also be used to combine a series of alternative multiple sequence alignments of the same sequences.

Efforts have been made to ensure full compatibility between ClustalW and T−COFFEE. For instance, this means that a script using ClustalW is likely to work alike using T−Coffee.

The first step in the T−Coffee process is the gathering of the pairwise alignments. A collection of such alignments is called a library. Once the libraries have been computed, they can be pooled, extended and used to compute a multiple sequence alignment.

Mocca is a special mode of T−Coffee that allows you to extract a series of repeats from a single sequence or a set of sequences.

RECENT MODIFICATIONS

For a detailed log of the modifications, please refer to the todo.txt file that is in the folder documentation.

−the ability to use various default parameter files (−t_coffee_defaults, −dali_defaults)
−the ability to use align profiles −profile[01/03/01]
−the ability to ’clean’ the unaligned bits of the alignment −clean_aln[01/03/01]
−a sequence weighting scheme (see −seq_weight, −outseqweight)[16/11/00]
−a new method for aligning cDNA (see method: fast_cdna_pair) [12/11/00]
−a new flag for choosing the evaluation mode (−evaluate_mode) [15/09/00]

T−COFFEE REFERENCE MANUAL

T_coffee is the main module. It aims at producing multiple sequence alignments from a set of sequences. It can be used for the following tasks:

−Making a multiple alignment.
−Evaluating a multiple alignment.
−Reformatting sequences and alignments.
−Reconciliation of slightly different sequence.
−Extracting repeats (Mocca)
−cleaning cDNAs.

INPUT

No Flag

if no flag is used <*your sequence*> must be the first argument. See format for further information.

t_coffee <your sequences>

In such a context, −run_name is set to <your_sequence>

−infile

For compatibility with ClustalW, it is also possible to indicate the sequences with the flag infile:

t_coffee −infile=<your sequences>

In such a context, −infile is set to <your sequences>

Note that valid formats for the sequence include alignments. Gaps are automatically reset.

−type

Usage: −type=[DNA, PROTEIN]
Default: −type=<automatically set>

Sets the type of the sequences. If omitted, the type is guessed automatically. This flag is compatible with ClustalW

−parameters

Indicates a file containing extra parameters. Parameters read that way superseed all the other parameters of the command line. For instance:

```
Parameter.file:

     −in sequences.pep fast_pair
     −output msf_aln
```

Please, Note that this parameter file can ONLY contain valid parameters. Comments are not allowed. Parameters passed this way will be checked like normal parameters.

Used with:

t_coffee −parameters=parameter.file

Will cause t_coffee to apply the fast_pair method only to the sequences contained in sequences.pep. If you wish, you can also pipe these arguments into t_coffee, by naming the parameter file stdin:

<your script> | t_coffee −parameters=stdin

for instance:

echo 'sequences.pep' | t_coffee

−t_coffee_defaults

Usage: −t_coffee_defaults=<file_name>
Default: not used.

This flag tells the program to use some default parameter file for t_coffee. The format of that file is the same as the one used with −aparameters, the name of that file can be indicated in three different ways:

1 *−t_coffee_defaults=myfile* will cause the default parameters to be read from myfile.

2 *−t_coffee_defaults*
If TCOFFEE_DEFAULTS is set to a filename, that file will be used, otherwise the program will try to use ~/.t_coffee_defaults.

Order:

−parameters > prompt parameters > −t_coffee_defaults > −dali_defaults

−dali_defaults

Usage: −dali_defaults=<file_name>
Default: not used.

−dali_defaults indicates that t_coffee will use the default dali parameters (hard coded). If a file is indicated as well (−dali_default=file_name) the parameters indicated in this file will be used instead. The format is the same as the one indicated before.

ALIGNMENT COMPUTATION

−in

Usage: −in=[<P,S,A,L,M,X><name>,]
Default: −in=Mlalign_id_pair,Mclustalw_pair

Five types of parameters may be expected:

P for pdb structure,
S for sequence,
M for method,
L for library,
A for (multiple)Alignments
X for substitution matrices.
R for profiles (guven in a legal mult. Aln. Format)

For instance:

−in=Ssequences1.seq,Aalignment1.aln,Aalignment2.msf,Mlalign_id_pair,Llibrary.lib

Will trigger the following chain of events:

1−Sequences will be gathered in all the sequence, alignments and library files. Format recognition is automatic. These sequences will be pooled together. Duplicates will be removed. For instance in the above case, the total set of sequences will be made of sequences contained in sequences1.seq, alignment1.aln, alignment2.msf and library.lib, plus the sequences initially gathered by −infile.

2−alignments alignment1.aln and alignment2.msf will be read and turned into libraries. Another library will be produced by applying the method lalign_id_pair to the set of sequences previously obtained (1). The final library used for the alignment will be the combination of all this information.

The identifiers (S, M, X...) may be omitted. But it is safer to use them, since the program may sometimes be wrong in guessing some file format.

Note as well the following rules:

1−The order in which sequences, methods, alignments and libraries are fed in is irrelevant.

2−There is no need for each element (A, S, L) to contain the same sequences.

3−Each file must contain only one copy of each sequence.

4−If two files (for instance two alignments) contain different versions of the same sequence due to indel, a new sequence will be reconstructed and used instead:

```
aln 1:
hgab1  AAAAABAAAAA

aln2:
hgab1  AAAAAAAAAACCC
```

will cause the program to reconstruct the following sequence

```
hgab1  AAAAABAAAAACCC
```

This can be useful if you are trying to combine several runs of blast, or structural information where residues may have been deleted. However substitutions are forbidden. If two sequences with the same name cannot be reconcilliated, they will cause the program to exit with an information message.

4−The method describer can either be built in (clustalw_aln, clustalw_pair, lalign_id_pair) or be a file describing the method to be used. The exact syntax is provided in part 4 of this manual.

5−If the method is a substitution matrix (X) then no other type of information should be provided. For instance:

```
t_coffee <your sequences> −in=Xpam250mt  −gapopen=−10
−gapext=−1
```

means that a progressive alignment will be carried out on the sequences in seqfile. The procedure is very similar to Pileup. In this context, appropriate gap penalties should be provided. The matrices are in the file source/matrices.h. Add−Hoc matrices can also be provided by the user (see the matrices format section at the end of this manual).

−profile

Usage: −profile=[<name>,] maximum of 200 profiles.
Default: no default

This flag allows to input alignment and treat them as a single sequences. That way , it is possible to make a multiple alignment of profiles. In the current implementation, the profiles are replaced with a consencus sequence and treated as normal sequences by the rest of the T−Coffee strategy.

In the future we plan to have a better way of handling profiles, but in the meantime, this solution is reasonnably effective for protein families that have a reasonnable level of conservation.

If your profile contains two distant sub−groups you may prefer spliting it in two more homogenous multiple alignments.

When provided with the −in flag, profiles should be preceded with the letter R.

−profile1

Usage: −profile1=[<name>] , one name only
Default: no default

This option is similar to the previous one and was provided for compatibility with ClustalW.

−profile2

Usage: −profile2=[<name>] , one name only
Default: no default

This option is similar to the previous one and was provided for compatibility with ClustalW.

−do_normalise

Usage: −do_normalise=<0 or a positive value>
Default:−do_normalise=1000

When using a value different from 0, this flag sets the score of the highest scoring pair to 1000.

−extend

Usage: −extend=<0,1 or a positive value>
Default:−extend=1

When turned on, this flag indicates that the library extension should be carried out when performing the multiple alignment. If extend is set to 0, the extension is not made, if it is set to 1, the extension is made on all the pairs in the library. If the extension is set to another positive value, the extension is only carried out on pairs having a weight value superior to the specified limit.

−dp_mode

Usage: −dp_mode=<string>
Default: −dp_mode=cfast_fair_wise

Indicates the type of dynamic programming used by the program:

gotoh_pair_wise: implementation of the gotoh algorithm (quadratic in memory and time)

myers_miller_pair_wise: implementation of the Myers and Miller dynamic programming algorithm (quadratic in time and linear in space). This algorithm is recommended for very long sequences. It is about 2 time slower than gotoh. It only accepts *tg_mode=1*.

fasta_pair_wise: implementation of the fasta algorithm. The sequence is hashed, looking for *ktuples* words. Dynamic programming is only carried out on the *ndiag* best scoring diagonals. This is much faster but less accurate than the two previous.

cfasta_pair_wise: c stands for checked. It is the same algorithm. The dynamic programming is made on the *ndiag* best diagonals, and then on the 2*ndiags, and so on until the scores converge. Complexity will depend on the level of divergence of the sequences, but will usually be L*log(L), with an accuracy comparable to the two first mode (this was checked on BaliBase).

−ktuple

Usage: −ktuple=<value>
Default: −ktuple=1 or 2

Indicates the ktuple size for cfasta_pair_wise dp_mode and fasta_pair_wise. It is set to 1 for proteins, and 2 for DNA. The alphabet used for protein is not the 20 letter code, but a mildly degenerated version, where some residues are grouped under one letter, based on physicochemical properties:

rk, de, qh, vilm, fy (the other residues are not degenerated).

This alphabet is set with the flag −sim_matrix=vasiliky.

−ndiag

Usage: −ndiag=<value>
Default: −ndiag=0

Indicates the number of diagonals used by the fasta_pair_wise algorithm. When set to 0, n_diag=Log (length of the smallest sequence).

−diag_mode

Usage: −diag_mode=<value>
Default: −diag_mode=0

Indicates the manner in which diagonals are scored during the fasta hashing.

0 indicates that the score of a diagonal is equal to the sum of the scores of the exact matches it contains.

1 indicates that this score is set equal to the score of the best uninterrupted segment

1 can be useful when dealing with fragments of sequences.

−sim_matrix

Usage: −sim_matrix=<string>
Default: −sim_matrix=vasiliky

Indicates the manner in which the amino acid is being degenerated when hashing. All the substitution matrix are acceptable. Categories will be defined as sub−group of residues all having a positive substitution score (they can overlap).
If you wish to keep the non degenerated amino acid alphabet, use −sim_matrix=idmat

−matrix

This flag is provided for compatibility with ClustalW. Setting *−matrix=blosum* is equivalent to *−in=Xblosum62mt*, *−matrix=pam* is equivalent to *in=Xpam250mt*. Apart from this, the rules are similar to those applying when declaring a matrix with the −in=X flag.

−gapopen

Usage: −gapopen=<value>
Default: −gapopen=0

Indicates the penalty applied for opening a gap. The penalty must be negative. If you provide a positive value, it will automatically be turned into a negative number. We recommend a value of 10 with pam matrices, and a value of 0 when a library is used.

−gapext

Usage: −gapext=<value>
Default: −gapext=0

Indicates the penalty applied for extending a gap.

−cosmetic_penalty

Usage: −cosmetic_penalty=<value>
Default: −cosmetic_penalty=100

Indicates the penalty applied for opening a gap. This penalty is set to a very low value. It will only have an influence on the portions of the alignment that are unalignable. It will not make them more correct, but only more pleasing to the eye (

i.e. Avoid stretches of lonely residues).
The cosmetic penalty is automatically turned off if a substitution matrix is used rather than a library.

−tg_mode

Usage: −tg_mode=<0, 1, or 2>
Default: −tg_mode=1

0: indicates that terminal gaps must be panelized with a gapopen and a gapext penalty.
1: indicates that terminal gaps must be penalized only with a gapext penalty
2: indicates that terminal gaps must not be penalized.

−weight

Usage: −weight=<winsimN, sim or sim_<matrix_name or matrix_file> or integer value>
Default:−weight=sim

Weight defines the way alignments are weighted when turned into a library.

winsimN indicates that the weight assigned to a given pair will be equal to the percent identity within a window of 2N+1 len centerd on that pair. For instance winsim10 defines a window of 10 residues around the pair being considered.

sim indicates that the weight equals the average identity within the match residues.

sim_matrix_name indicates the average identity with two residues regarded as identical when their substitution value is positive. The valid matrices names are in *matrices.h (pam250mt)* .Matrices not found in this header are considered to be filenames. See the format section for matrices. For instance, *−weight=sim_pam250mt* indicates that the grouping used for similarity will be the set of classes with positive substitutions. Other groups include

sim_clustalw_col (categories of clustalw marked with :)
sim_clustalw_dot (categories of clustalw marked with .)

Value indicates that all the pairs found in the alignments must be given the same weight equal to value. This is useful when the alignment one wishes to turn into a library must be given a pre−specified score (for instance if they come from a structure super−imposition program). Value is an integer:

−weight=1000

Note: Weight only affects methods that return an alignment to T−Coffee, such as ClustalW. On the contrary, the version of Lalign we use here returns a library where weights have already been applied and are therefore insensitive to the −weight flag.

−seq_weight

Usage: −seq_weight=<t_coffee or <file_name>>
Default: −seq_weight=t_coffee

These are the individual weights assigned to each sequence. The t_coffee weights try to compensate the bias in consistency caused by redundancy in the sequences.

sim(A,B)=%similarity between A and B, between 0 and 1.
weight(A)=1/sum(sim(A,X)^3)

Weights are normalized so that their sum equals the number of sequences, they are applied onto the primary library in the following manner:

res_score(Ax,By)=Min(weight(A), weight(B))*res_score(Ax, By)

These are very simple weights. Their main goal is to prevent a single sequence present in many copies to dominate the alignment.

Note: the library output by −out_lib is the un−weighted library.
Note: Weights can be output using the −outseqweight flag.
Note: You can provide your own set of weights (see the format section). Sequences whose weight is not provided will be set to 1.

−seq_to_align

Usage: −seq_to_align=<file_name>
Default: No file (align all the sequences).

You may not wish to align all the sequences brought in by the **−in** flag. Supplying the seq_to_align flag allows for this, the file is simply a list of names in Fasta format:

>*sequence1*
>*sequence2*
>*sequence3*

However, note that library extension will be carried out on all the sequences.

USING THE STRUCTURE IN THE ALIGNMENT.

It is possible to use t_coffee to compute multiple structural alignments. To do so, ensure that you have the perl script extract_from_pdb installed as well as sap.

t_coffee −in struc1.pdb struc2.pdb struc3.pdb sap_pair

Will combine the pairwise alignments produced by SAP. There are currently two methods that can be interfaced with t_coffee:

sap_pair: that uses the sap algorithm.

By default, the computation will be made only on the first chain contained in the pdb file. If your structure is an NMR structure, you are advised to provide the program with one structure only.

If you wish to align only a portion of the structure, you should extract it yourself from the pdb file, using extract_from_pdb.

You can provide t_coffee with a mixture of sequences and structure. When comparing two sequences, it will automatically switch to the ’fast_pair’ mode if both pdb structures are not available.

The pdb structure must be in the directory where computation is made. If these structures live somewhere else on your disk, copy them or alias them locally.

MOCA: LOCAL MULTIPLE ALIGNMENTS

It is possible to compute multiple local alignments, using the moca routine. MOCA is a routine that allows extracting all the local alignments that show some similarity with another predefined fragment.

’mocca’ is a perl script that calls t−coffee and provides it with the appropriate parameters.

−domain

Usage: −domain
Default: not set

This flag indicates that t_coffee will run using the domain mode. All the sequences will be concatenated, and the resulting sequence will be compared to itself using lalign_rs_s_pair mode (lalign of the sequence against itself using keeping the lalign raw score). This step is the most computer intensive, and it is advisable to save the resulting file.

t_coffee −in <your sequences> lalign_rs_s_pair −out_lib=<lib name> −domain −start=100 −len=50

Will use the fragment 100−150 on the concatenated sequences, as a template for the extracted repeats. The extraction will only be made once. The library will be placed in the file <lib name>.

If you want, you can test other coordinates for the repeat, such as *t_coffee −in <your lib> −domain −start=100 −len=60*

This run will use the fragment 100−160, and will be much faster because it does not need to recompute the lalign library.

−start

Usage: −start=<int value>
Default: not set

This flag indicates the starting position of the portion of sequence that will be used as a template for the repeat extraction. The value assumes that all the sequences have been concatenated, and is given on the resulting sequence.

−len

Usage: −len=<int value>
Default: not set

This flag indicates the length of the portion of sequence that will be used as a template.

−scale

Usage: −scale=<int value>
Default: −scale=−100

This flag indicates the value of the threshold for extracting the repeats. The actual threshold is equal to:
motif_len*scale
In order to increase the sensibility, you can increase the scale (i.e. −50).

−domain_interactive

Usage: −domain_interactive
Default: not set

This will generate an interactive domain extraction session:
t_coffee −in <your lib> −domain −start=100 −len=60

```
TOLB_ECOLI_212_262    211 SKLAYVTFESGR--SALVIQTLANGAVRQV-ASFPRHNGAPAFSPDGSKLAFA  261
TOLB_ECOLI_165_218    164 TRIAYVVQTNGGQFPYELRVSDYDGYNQFVVHRSPQPLMSPAWSPDGSKLAYV  217
TOLB_ECOLI_256_306    255 SKLAFALSKTGS--LNLYVMDLASGQIRQV-TDGRSNNTEPTWFPDSQNLAFT  305
TOLB_ECOLI_307_350    306 -------DQAGR--PQVYKVNINGGAPQRI-TWEGSQNQDADVSSDGKFMVMV  349
TOLB_ECOLI_351_393    350 -------SNGGQ--QHIAKQDLATGGV-QV-LSSTFLDETPSLAPNGTMVIYS  392
                        1           *             *    :          .   .:.  :      53
```

```
        MENU: Type Letter Flag[number] and Return: ex |10
        |x      -->Set     the START to x
        >x      -->Set     the LEN   to x
        Cx      -->Set     the sCale to x
        Sname   -->Save    the  Alignment
        Bx      -->Save    Goes back x it
        return  -->Compute the  Alignment
        X       -->eXit
```

```
[ITERATION   1] [START=211] [LEN= 50] [SCALE=−100]      YOUR CHOICE:
```

For instance, to set the length of the domain to 40, type:

```
[ITERATION   1] [START=211] [LEN= 50] [SCALE=−100]      YOUR CHOICE:>40[return]
[return]
```

Which will generate:

```
TOLB_ECOLI_212_252    211 SKLAYVTFESGRSALVIQTLANGAVRQVASFPRHNGAPAF  251
TOLB_ECOLI_256_296    255 SKLAFALSKTGSLNLYVMDLASGQIRQVTDGRSNNTEPTW  295
TOLB_ECOLI_300_340    299 QNLAFTSDQAGRPQVYKVNINGGAPQRITWEGSQNQDADV  339
TOLB_ECOLI_344_383    343 KFMVMVSSNGGQQHIAKQDLATGGV−QVLSSTFLDETPSL  382
TOLB_ECOLI_387_427    386 TMVIYSSSQGMGSVLNLVSTDGRFKARLPATDGQVKFPAW  426
                        1   :     :     :           ::         .     40



        MENU: Type Letter Flag[number] and Return: ex |10
        |x      −−>Set     the START to x
        >x      −−>Set     the LEN   to x
        Cx      −−>Set     the sCale to x
        Sname   −−>Save    the  Alignment
        Bx      −−>Save    Goes back x it
        return  −−>Compute the  Alignment
        X       −−>eXit

[ITERATION   3] [START=211] [LEN= 40] [SCALE=−100]      YOUR CHOICE:
```

If you want to indicate the coordinates, relative to a specific sequence, type:

|<seq_name>:start

Type S<your name> to save the current alignment, and extract a new motif.

Type X when you are done.

TREE COMPUTATION AND OUTPUT

−newtree

Usage: −newtree=<tree file>
Default: No file specified
Format: Phylips tree format

Indicates the name of the new tree to compute. The default will be <sequence_name>.dnd, or <run_name.dnd>.

−usetree

Usage: −usetree=<tree file>
Default: No file specified
Format: Phylips tree format

This flag indicates that rather than computing a new dendrogram, t_coffee can use a pre−computed one. The tree files are in phylips format and compatible with

ClustalW. In most cases, using a pre−computed tree will halve the computation time required by t_coffee. It is also possible to use trees output by ClustalW or Phylips.

−tree_mode

Usage: −tree_mode=<slow, fast, very_fast>
Default: very_fast

This flag indicates the method used for computing the dendrogram.

Slow: the chosen dp_mode using the extended library,
fast: The fasta dp_mode using the extended library.
very_fast: The fasta dp_mode using pam250mt.

−quicktree

Usage: −quicktree

This flag is kept for compatibility with ClustalW. It indicates that:

−tree_mode=very_fast.

ALIGNMENT OUTPUT: Formats and File names

−File names

stdin
stdout
stderr
no
/dev/null

stdout, *stderr* and *stdin* are valid filenames. They cause the corresponding file to be output in stderr or stdout, for an input file, stdin causes the program to requests the corresponding file through pipe. No causes a suppression of the output, as does /dev/null.

−outfile

Usage: −outfile=<out_aln file,default,no>
Default:−out_aln=default

Indicates the name of the alignment output by t_coffee. If the default is used, the alignment is named *<your sequences>.aln*

−output

Usage: −output=<format1,format2,...>
Default:−output=clustalw

Indicates the format used for outputting the −outfile.
Supported formats are:

clustalw_aln, clustalw	: ClustalW format.
gcg, msf_aln	: Msf alignment.
pir_aln	: pir alignment.
fasta_aln	: fasta alignment.
phylip	: Phylip format.
pir_seq	: pir sequences (no gap).
fasta_seq	: fasta sequences (no gap).

As well as:

score_html	: causes the output to be a reliability plot in HTML
score_pdf	: idem in PDF (if ps2pdf is installed on your system).
score_ps	: idem in postscript.

More than one format can be indicated:

−output=clustalw,gcg, score_html

−evaluate_mode

Usage: −evaluate_mode=<mode>
Default: −evaluate_mode=t_coffee_fast

This flag indicates the mode used to normalize the t_coffee score when computing the reliability score.

t_coffee_fast: use an estimation of the maximum extended score (the same for every position.

t_coffee_non_extended: the score of each residue is the ratio between the sum of its non extended scores with the column and the sum of all its possible non extended scores.

t_coffee_slow: for each residue, normalize the score with the proper maximum extended score. This computation takes as much time as the computation of the alignment.

−case

Usage: −case=<upper or lower>

Default:−case=<upper>

Triggers the choice of the case for the output.

−cpu

Usage: −cpu=<value>
Default:−cpu=0

Indicates the cpu time (micro seconds) that must be added to the t_coffee computation time.

−out_lib

Usage: −out_lib=<name of the library,default,no>
Default:−out_lib=default

Sets the name of the library output. Default implies <run_name>.tc_lib

−outseqweight

Usage: −outseqweight=<name of the file containing the weights applied>
Default: −outseqweight=no

−outorder

Usage: −outorder=<input or aligned>
Default:−outorder=input

Sets the name of the library output. Default implies <run_name>.tc_lib

−seqnos

Usage: −seqnos=<on or off>
Default:−seqnos=off

Causes the output alignment to contain residue numbers at the end of each line:

```
T−COFFEE

seq1 aaa−−−aaaa−−−−−−−−aa 9
seq2 a−−−−−aa−−−−−−−−−−−a 4

seq1 a−−−−−−−−−−−−−−−−−a 11
seq2 aaaaaaaaaaaaaaaaaaa 19

     ...
```

−clean_aln

Usage: −clean_aln

Default:−clean_aln

This flag causes T−Coffee to post−process the multiple alignment. Residues that have a reliability score smaller or equal to −clean_threshold (as given by an evaluation that uses −clean_evaluate_mode) are realigned to the rest of the alignment. Residues with a score higher than the threshold constitute a rigid framework that cannot be altered.

The cleanning algorithm is greedy. It starts from the top left segment of low constitency residues and works its way left to right, top to bottom along the alignment. You can require this operation to be carried out for several cycles using the −clean_iterations flag.

The rationale behind this operation is mostly cosmetic. In order to ensure a decent looking alignment, the gop is set to −20 and the gep to −1. There is no penalty for terminal gaps, and the matrix is blosum62mt.

Note1: Gaps are always considered to have a reliability score of 0.

−clean_threshold

Usage: −clean_threshold=<value between 0 and 9>
Default:−clean_aln=1

See −clean_aln for details.

−clean_iteration

Usage: −clean_iteration=<value between 1 and >
Default:−clean_iteration=1

See −clean_aln for details.

−clean_evaluation_mode

Usage: −clean_iteration=<evaluation_mode >
Default:−clean_iteration=t_coffee_non_extended

Indicates the mode used for the evaluation that will indicate the segments that should be realigned. See −evaluation_mode for the list of accepetd modes.

GENERIC OUTPUT

−run_name

Usage: −run_name=<your run name>
Default: no default set

This flag causes the prefix <*your sequences*> to be replaced by <*your run name*> when renaming the default files.

−quiet

Usage: −quiet=<stderr,stdout or file name or nothing.
Default:−quiet=stderr

Redirects the standard output to either a file. −quiet on its own redirect the output to /dev/null.

−align

Indicates that the program must produce the alignment. This flag is here for compatibility with ClustalW.

−convert

Indicates that the program must not compute the alignment but simply convert all the sequences, alignments and libraries into the format indicated with −output. This flag can also be used if you simply want to compute a library (i.e. You have an alignment and you want to turn it into a library).

USING NEW AND EXISTING METHODS

Methods With An Established T−Coffee Interface

Some packages already have an interface with t_coffee, these include:

align_pdb:	ALIGN_PDB_4_TCOFFEE
sap:	SAP_4_TCOFFEE
lalign2list:	LALIGN_4_TCOFFEE
clustalw:	CLUSTALW_4_TCOFFEE

If these programs are installed on your system and you want t_coffee to use a specific version:

setenv CLUSTALW_4_TCOFFEE <path to your version>

Built in methods methods can be requested using the following names:

fast_pair

Makes a global fasta style pairwise alignment. For proteins, matrix=blosum62mt, gep=−1, gop=−10, ktup=2. For DNA, matrix=idmat (id=10), gep=−1, gop=−20, ktup=5. Each pair of residue is given a score function of the weighting mode defined by −weight.

slow_pair

Identical to fast pair, but does a full dynamic programming, using the myers and miller algorithm. This method is recommended if your sequences are distantly related.

ifast_pair

Makes a global fasta alignmnet using the previously computed pairs as a library. ‘i‘ stands for iterative. Each pair of residue is given a score function of the weighting mode defined by −weight.

align_pdb_pair

Uses the align_pdb routine to align two structures. The pairwise scores are those returnes by the align_pdb program. If a structure is missing, fast_pair is used instead. Each pair of residue is given a score function defined by align_pdb.

sap_pair

Uses sap to align two structures. Each pair of residue is given a score function defined by sap. You must have sap installed on your system to use this method.

clustalw_pair

Uses clustalw (default parameters) to align two sequences. Each pair of residue is given a score function of the weighting mode defined by −weight.

clustalw_aln

Makes a multiple alignment using ClustalW and adds it to the library. Each pair of residue is given a score function of the weighting mode defined by −weight.

lalign_rs_pair

Uses the output of lalign2list. Each pair of residue is given a score equal to the lalign raw score it comes from.

lalign_id_pair

Same as lalign_rs_pir, but using the level of identity as a weight.

lalign_id_m_pair

Same as above, but does the alignment both way (m stands for miror).

lalign_s_pair

Same as above but does also the self comparison (s stands for self). This is needed when extracting repeats. The weights used that way are based on identity.

lalign_rs_s_pair

Same as above but does also the self comparison (s stands for self). This is needed when extracting repeats. The weights used that way are based on identity.

Matrix

Amy matrix can be requested. Simply indicate as a method the name of the matrix preceded with an X (i.e. Xpam250mt). If you indicate such a matrix, all the other methods will simply be ignored.

fast_cdna_pair

This method computes the pairwise alignment of two cDNA sequences. It is a fast_pair alignment that only takes into account the amino−acid similarity and uses different penalties for amino−acid insertions and frameshifts.

This alignment is turned into a library where matched nucleotides receive a score equql to the average level of identity at the amino−acid level.

This mode is intended to clean cDNA obtained from ESTs, or to align pseudo−genes.

To request a method, see the −in flag. For instance, if you wish to request the use of fast_pair and lalign_id_pair (the current default):

t_coffee −in <your sequences> Mfast_pair Mlalign_id_pair

The order is irrelevant.

Methods Without An Established T−Coffee Interface.

If the method you wish to use is not supported, the simplest solution is to generate yourself the pairwise/multiple alignments, in fasta, ClustalW, msf or Pir format and feed them into t_coffee using the *−in* flag:

−in Aalignment1 Aalignment2…..

Note1 This method forces you to use the same weighting scheme for each alignment and the rest of the libraries generated on the fly. This weighting scheme is based on pairwise sequence identity.

Note2 Alignments do not need to contain all the sequences or the same sequences, nor do they need to have the same starts and ends. However, names must be consistent.

Note3 Default methods are reset when you explicitely use −in, if you wish to keep using fast_pair and lalign_id_pair, you need to indicate these methods explicitely:

−in .... Mfast_pair, Mlalign_id_pair

Generate Your Own Libraries

This is suitable if you have local alignments, or very detailed information about your potential residue pairs, or if you want to use a very specific weighting scheme. You will need to generate your own libraries, using the format described in the last section.

Note1 You can have up to 200 libraries. They do not need to contain the same sequences.

Making a New Method File

If you have a method available (e.g. pileup), write a perl script that will contain your favorite parameters, and some input/output facilities so that the script can be ran using:

Your_pileup_script.perl −in sequence_file −out aln.

In and Out must be legal formats (see last section). You will also need to provide a configuration file for the method:

```
EXECUTABLE      S     Your_pileup_script.perl
ALN_MODE        S     pairwise or multiple
IN_FLAG         S     −in
OUT_FLAG        S     −out
DEFAULT         S     list of default parameters
```

Save this file as pileup_aln_method and use the −in flag:

−in=Mpileup_aln_method

Your script will be called by t_coffee as follows:

pileup.script <list of default parameters> −in=<tmp seq> −out=<tmp aln>

The following file indicates some of the parameters that can be passed to t_coffee when executing. Lines starting with a ’*’ sign will be ignored.

```
*********************************************************************************************
*
*       Master File for Incorporating new methods in T−Coffee
*       Cedric Notredame 23/02/01
*
*********************************************************************************************

*EXECUTABLE
*name of the executable
EXECUTABLE      S       clustalw

*ALN_MODE
*       pairwise   −>Half Everything Vs Everything excepty self   [(n2−n)/2 pw aln]
*       m_pairwise −>     Everything Vs Everything excepty self   [n2−n     pw aln]
*       s_pairwise −>Half Everything Vs Everything including self [n2−(n2−n)/2 pw aln
*       multiple   −>All the sequences in one go
ALN_MODE        S       pairwise

*OUT_MODE
*mode of the outout:
*     aln −> alignmnent File
*     list−> List file (librairie)
*     fL −> Internal Function returning a List (Librairie)
*     fA −> Internal Function returning an Alignmnent
OUT_MODE        S        aln

*IN_FLAG
*flag indicating the name of the in−coming sequences
*IN_FLAG S no_name −>no flag
IN_FLAG         S       −INFILE=

*OUT_FLAG
*flag indicating the name of the out−coming data
*OUT_FLAG S no_name −>no flag
OUT_FLAG        S       −OUTFILE=

*DEFAULT
*list of the default fixed parameters sent to the EXECUTABLE
DEFAULT         S        −OUTORDER=INPUT −NEWTREE=core −align −gapopen=−15

*The script passes to the system is in the following order
*<EXECUTABLE> <IN_FLAG> <OUT_FLAG> <DEFAULT>
```

FAQ

ABNORMAL TERMINATIONS and WRONG RESULTS.

Q: The program keeps crashing when I give my sequences

A: This may be a format problem. Try to reformat your sequences using any utility (readseq...). We recommend the Fasta format. If the problem persists, contact us.

Q: The default alignment is not good enough

A: see next question

Q: The alignment contains obvious mistakes

A: This happens with most multiple alignment procedures. However, wrong alignments are sometimes caused by a bugs or an implementation mistake. Please report the most unexpected results to the authors.

Q: The program is crashing

A: If you get the message:

FAILED TO ALLOCATE REQUIRED MEMORY

See the next question.
If the program crashes for some other reason, please check wether you are using the right syntax and if the problem persists get in touch with the authors.

Q: I am running out of memory

A: You can use a more accurate, slower and less memory hungry dynamic programming mode called myers_miller_pair_wise. Simply indicate the flag:

−dp_mode=myers_miller_pair_wise

To use an amount of memory as small as possible use the following settings:

t_coffee <your sequences> −in=slow_pair,lalign_id_pair −tree_mode=slow −dp_mode=myers_miller_pair_wise

If you keep running out of memory, you may want to stop using local information:

t_coffee <your sequences> −in=slow_pair −tree_mode=slow −dp_mode=myers_miller_pair_wise

Alternatively, to keep the alignment as good as possible, you may prefer working on a smaller test set, and/or remove low complexity regions.

INPUT/OUPUT And PARAMETRIZATION

Q: How many ways to pass parameters to t_coffee?

A: There are five layers of parameters in t_coffee, the lower layer are the hard coded defaults, then come −dali_defaults, then −t_coffee_defaults, then the prompt, then

−parameters. −parameters superseeds ALL the others (i.e. It is the strongest).

t_coffee −t_coffee_defaults −in=<my file> lalign_id_pair −parameters=<param_file>

If TCOFFEE_DEFAULTS is not set, the file ~/.t_coffee_defaults will be read (if it exists).
The prompt (−in=my_file,lalign_id_pair) will superseed any previously set value, and finally the values read in −parameters will superseed any previously set value.

The spirit is the following:
1−If for whatever reason you are not happy with the t_coffee defaults, put yours in ~/.t_coffee default or any file indicated by TCOFFEE_DEFAULTS. This will be set ’forever’.

2−If for a given task there is a set of parameters you need to use often (but not ALWAYS), put them in a file with a specific name that you will name −parameters.

3− The flag −dali_defaults is meant for compatibility with DALI and it triggers some hard coded defaults.

Remember:

parameters > command_line > t_coffee_defaults > dali_defaults > hard coded defaults

and t_coffee_default can be set with TCOFFEE_DEFAULTS

Q: How can I change the default output format?

A:See the −output option, common output formats are:

−output=msf

Q: My sequences are slightly different between all the alignments.

A:It does not matter. T−Coffee will reconstruct a set of sequences that incorporates all the residues potentially missing in some of the sequences (see flag −in).

Q: Is it possible to pipe stuff out of t_coffee?

A: If instead of a name, you indicate stderr or stdout, the output will be redirected accordingly. For instance

t_coffee <your sequences> −outfile=stdout −out_lib=stdout

will output the tree (in new hampshire format) and the alignment to stdout.

Q: Is it possible to pipe stuff into t_coffee?

A: If as a file name, you specify stdin, the content of this file will be expected throught pipe:

cat <your sequence file> | t_coffee −infile=stdin

will be equivalent to

t_coffee <your sequence file>

If you do not give any argument to t_coffee, they will be expected to come from pipe:

your script | t_coffee −parameters=stdin

For instance:

echo '<your sequence file> −in clustalw_pair | t_coffee

Q: Can I read my parameters from a file?

A: see the −parameters flag that allows you to put your parameters in a file and to re−use them.

Q: I want to decide myself on the name of the output files!!!

A: Use the *−run_name* flag.

Q: I want to use the sequences in an alignment file

A: Simply fed your alignment, any way you like, but do not forget to append the prefix S for sequence:

−in=Syour_alignment.aln
or
−infile=Syour_alignment.aln

This means that the gaps will be reset and that the alignment you provide will not be considered as an alignment, but as a set of sequences.

Q: I only want to produce a library

A: use the −convert flag

t_coffee <your sequences> −in <method...> −out_lib <your name> −convert

You must not omit −out_lib, otherwise no library will be output.

Q: I want to concatenate two libraries

A: You cannot concatenate these files on their own. You will have to use t_coffee. Assume you want to combine tc_lib1.tc_lib and tc_lib2.tc_lib.

```
t_coffee −in Ltc_lib1.tc_lib Ltc_lib2.tc_lib −convert
−out_lib=tot_lib.tc_lib
```

You can also make an alias with:

```
alias concatenate_library t_coffee −convert −out_lib =stdout
−in
```

and concatenating will become:

```
concatenate_library tc_lib1.tc_lib tc_lib2.tc_lib >
tot_lib.tc_lib
```

Q: What happens to the gaps when an alignment is fed to T−Coffee

A: when you feed in an alignment with −infile, it is regarded as a set of sequences, and gaps are removed. When you feed it throught −in, if you say nothing, it is recognized as an alignment and used to build the library. If you indicate it with the S identifier:

```
−in Smy_alignment.aln
```

It will be seen as a sequence file, even if it has an alignment format (gaps will be removed).

Q: I do not want to align ALL the sequences.

A: Use the −seq_to_align flag. All the sequences will be USED for computing the librrary, but only a subset will be aligned, using information contained in ALL the sequences.

Q: I cannot print the html graphic display!!!

A: This is a problem that has to do with your browser. Instead of requesting the score_html output, request the score_ps output that can be read using ghostview:

```
−output=score_ps
or
−output=score_pdf
```

Note that the latest versions of Internet Explorer and Netscape now allow the user to print the HTML display. Do not forget to request Background printing.

Q: I want to output an html file and a regular file

A: see the next question

Q: I would like to output more than one alignment format at the same time

A: The flag −output accepts more than one parameter. For instance,

```
−output=clustalw,score_html,score_ps, msf
```

will output four alignment files in the corresponding formats. Alignments' names will have the format name as an extension.

ALIGNMENT COMPUTATION

Q: I do not want to compute the alignment.

A:use the −convert flag

```
t_coffee your_seq.aln −convert −output=gcg
```

will read the .aln and will turn it into a .msf alignment.

Q: I would like to force some residues to be aligned.

See *I want to build my own libraries.*

Q: I would like to use structural alignments.

See the section Using structures in Multiple Sequence Alignments, or see the question *I want to build my own libraries.*

Q: I want to build my own libraries.

A: Turn your alignment into a library, forcing the residues to have a very good weight, using structure:

```
t_coffee alignment −convert −weight=10000
−out_lib=good_lib.tc_lib
```

The value 10000 is simply a high value that should make it more likely for the substitution found in your alignment to reoccur in the final alignment. This will produce the library good_lib.tc_lib that you can later use when aligning all the sequences:

```
t_coffee Sall_the_sequences.pep Lgood_lib.tc_lib
```

will output all_the_sequences.aln

If you only want some of these residues to be aligned, or want to give them individual weights, you will have to edit the library file yourself. A value of N*N * 1000 (N being the number of sequences) usually ensure the respect of a constraint.

Q: I want to use my own tree!!!!

A: Use the *−usetree=<your own tree>* flag.

Q: I want to align coding DNA

A: use the fasta_cdna_pair method that compares two cDNA using the best reading frame and taking frameshifts into account.

Q: There are duplicates or quasi−duplicates in my set

A: If you can remove them, this will make the program run faster, otherwise, the t_coffee scoring scheme should be able to avoid over−weighting of sur−represented sequences.

ALIGNMENT EVALUATION

Q: How good is my alignment?

A: see *what is the reliability story?*

Q: What is that reliability story?

A: T−Coffee can provide you with a measure of consistency among all the methods used. You can produce such an output using:

t_coffee your_seq.pep −output=score_html

This will compute your_seq.score_html that you can view using netscape. An alternative is to use score_ps or score_pdf that can be viewed using ghostview or acroread, score_ascii will give you an alignment that can be parsed as a text file.

Q: Can I evaluate alignments NOT produced with T−Coffee?

A: Yes. You may have an alignment produced from any source you like. To evaluate it do:

t_coffee your_alignment −in=your_library −score −output=score_html

If you have no library available, the library will be computed on the fly (this can take some time, depending on the size of your sample). To monitor the progress in a situation where the default library is being built, use:

t_coffee your_alignment −score −output=score_html −quiet=stdout

Q: I am aligning sequences with long regions of very good overlapp

A: Increase the ktuple size (up to 4 or 5 for DNA) and up to 3 for proteins.

```
−ktuple=3
```

This will speed up the program. It can be very useful, especially when aligning ESTs.

Q: T−Coffee is changing the names of my sequences!!!!

A: T−Coffee’ handling of names is consistent with Clustalw, see Sequence Name Handling in the Format section.

MAKING A T−COFFEE SERVER

In the html directory, you will find a tcoffee.html. It is an interface to tcoffee4html.cgi, a perl script able to run t_coffee. In order to install this, put tcoffee.html in your public_html directory and put tcoffee4html.cgi in your cgi−bin. Edit the top of both files: The ’Customize’ part in the cgi and the <form > tag in the html. Successful implementation will depend on various parameters, such as your web server, file permissions and so on.

Pleas, consider these files as a source of inspiration rather than a ready−made solution.

Common Problems

Output of the .dnd file.

A common source of error when running a server: T−Coffee MUST output the .dnd file because it re−reads it to carry out the progressive alignment. By default T−Coffee outputs this file in the directory where the process is running. If the T−Coffee process does not have permission to write in that directory, the computation will abort...
To avoid this, simply specify the name of the output tree:

```
−newtree=<writable file (usually in /tmp)>
```

Chose the name so that two processes may not over−write each other dnd file.

Permissions

The t_coffee process MUST be allowed to write in some scratch area, even when it is ran by Mr nobody... Make sure the /tmp/ partition is not protected.

Other Programs

T−Coffee may call various programs while it runs (lalign2list by defaults). Make sure your process knows where to find these executables.

Help about the use of the Html version is available on line.

FORMATS

Parameter files

Parameter files used with −parameters, −t_coffee_defaults, −dali_defaults... Must contain a valid parameter string where line breaks are allowed. These files cannot contain any comment, the recommended format is one parameter per line:

```
<parameter name>=<value1>,<value2>....
<parameter name>=.....
```

Sequence Name Handling

Sequence name handling is meant to be fully consistent with ClustalW (Version 1.75). This implies that in some cases the names of your sequences may be edited when coming out of the program. four rules apply:

1−Names that do contain spaces, for instance:
>seq1 human_myc
will be turned into
>seq1
it is your responsibility to make sure that the names you provide are not ambiguous after such an editing. This editing is consistent with Clustalw (Version 1.75)

2−Some non alphabetical characters are replaced with underscores. These are: ’;:()’
Other characters are legal and will be kept unchanged. This editing is meant to keep in line with Clustalw (Version 1.75).

3−One character is not legal: ’>’.

4−Name length must be below 100 characters.

Automatic Format Recognition

Most common formats are automatically recognized by t_coffee. See −in and the next section for more details. If your format is not recognized, use readseq or clustalw to switch to another format. We recommend Fasta.

Structures

PDB format is recognized by T−Coffee, only if the extract_from_pdb is also installed (that Perl module comes along with the t_coffee distribution). extract_from_pdb is a small module that can be used on its own to extract information from pdb files.

Sequences

Sequences can come in the following formats: fasta, pir, swiss−prot, clustal aln, msf aln and t_coffee aln. These formats are the one automatically recognized. Please replace the ’*’ sign sometimes used for stop codons with an X.

Alignments

Alignments can come in the following formats: msf, ClustalW, Fasta, Pir and t_coffee. The t_coffee format is very similar to the ClustalW format, but slightly more flexible. Any interleaved format with sequence name on each line will be correctly parsed:

```
<empy line>   [Facultative]n
<line of text>  [Required]
<line of text>                 [Facultative]n
<empty line>                   [Required]
<empty line>                   [Facultative]n
<seq1 name><space><seq1>
<seq2 name><space><seq2>
<seq3 name><space><seq3>
<empty line>                   [Required]
<empty line>                   [Facultative]n
<seq1 name><space><seq1>
<seq2 name><space><seq2>
<seq3 name><space><seq3>
<empty line>                   [Required]
<empty line>                   [Facultative]n
```

An empty line is a line that does NOT contain amino−acid. A line that contains the ClustalW annotation (.:*) is empty.

Spaces are forbidden in the name. When the alignment is being read, non character signs are ignored in the sequence field (such as numbers, annotation…).

Libraries

```
<nseq>
<seq1 name> <seq1 length> <seq1>
<seq2 name> <seq2 length> <seq2>
<seq3 name> <seq3 length> <seq3>
!Comment
(!Comment)n
#1 2
<x=residue x of seq 1> <y=residue y of seq 2>
12 13 99  (this indicates that residue 12 of seq 0 has a match score of 99 with residue 13 of
seq1)
12 14 70
15 16 56
```

#1 3
12 13 99
12 14 70
15 16 56
CPU <cpu time required for the compuation of the list=x> (OPTIONAL)
! **SEQ_1_TO_N**

Note 1: There is a space between the ! And SEQ_1_TO_N
Note 2: The last line (! SEQ_1_TO_N) indicates that:

Sequences are numbered from **1 to N**
Residues are numbered from **1 to N**

If this line is omitted, the following numbering will be expected and assumed:

Sequences are numbered from **0 to N−1**
Residues are numbered from **1 to N**

Note 3: the residues do not need to be sorted, and neither do the sequences. The same pair can appear several times in the library. For instance:

#0 1
12 13 99
#0 2
15 16 99
#0 1
12 14 70

would be legal.

Substitution matrices.

If the required substitution matrix is not available, write your own in a file using the following format:

$
v1
v2 v3
v4 v5 v6
..
$

v1, v2... are integers, possibly negatives.
the order of the amino acids is: ABCDEFGHIKLMNQRSTVWXYZ, which means that v1 is the substitution value for A vs A, v2 for A vs B, v3 for B vs B, v4 for A vs C and so on.

Sequences Weights

Create your own weight file, using the −seq_weight flag:

```
seq_name1 v1
seq_name2 v2
...
```

No duplicate. Sequences not included in the set of sequences provided to t_coffee will be ignored. Order is free. V1 is a float. Unweighted sequences will see their weight set to 0.

KNOWN PROBLEMS

1− The implementation of lalign used here is not symmetrical (i.e. the output obtained with seq1 vs seq2 is not identical to the output obtained with seq2 vs seq1). This means that sometimes, when the sequences are very divergent, t−coffee can become sensitive to the order in which the sequences are presented. To avoid this, it is possible to replace the method *lalign_id_pair* with *lalign_id_m_pair*, that does lalign both way (seq1 vs seq2 and seq2 vs seq1) and pools the results.

2−Nucleotides sequences with long stretches of Ns will cause problems to lalign, especially when using Mocca. To avoid any problem, filter these nucleotides out before running mocca.

3−Stop codons are sometimes coded with ’*’ in protein sequences. This will cause the program to abort. Please replace the ’*’ signs with an X.