Manual

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Cédric Notredame

T-Coffee User Guide and Reference Manual

T-Coffee User Guide
(Version 3.18, September 2005)

License and Terms of Use

T-Coffee is distributed under the Gnu Public License

Please make sure you have agreed with the terms of the license attached to the package before using the T-Coffee package or its documentation. T-Coffee is a freeware open source distributed under a GPL license. This means that there is no restriction to its use, either in an academic or a non academic environment.

T-Coffee code can be re-used freely

Our philosophy is that code is meant to be re-used, including ours. No permission is needed, although we are always happy to receive pieces of improved code.

Addresses and Contacts

Contributors

T-coffee is developed by a dedicated team that includes

Cédric Notredame

Olivier Poirot

Fabrice Armougom

Sebastien Moretti

Addresses

We are always very eager to get some user feedback. Please do not hesitate to drop us a line at: cedric.notredame@europe.com The latest updates of T-Coffee are always available on: http://igs-server.cnrs-mrs.fr/~cnotred. On this address you will also find a link to some of the online T-Coffee servers, including Tcoffee@igs: http://igs-server.cnrs-mrs.fr/Tcoffee/

T-Coffee can be used to automatically check if an updated version is available, however the program will not update automatically, as this can cause endless reproducibility problems.

```
EXCL: t_coffee –update
```

Citations

It is important that you cite T-Coffee when you use it. Citing us is (almost) like giving us money: it helps us convincing our institutions that what we do is useful and that they should keep paying our salaries and delivering Donuts to our offices from time to time (Not that they ever did it, but it would be nice anyway).

Cite the server if you used it, otherwise, cite the original paper from 2000 (No, it was never named "T-Coffee 2000").

Notredame C, Higgins DG, Heringa J. Related Articles, Links

T-Coffee: A novel method for fast and accurate multiple sequence alignment.
J Mol Biol. 2000 Sep 8;302(1):205-17.
PMID: 10964570 [PubMed - indexed for MEDLINE]

Other useful publications include:

T-Coffee

Claude JB, Suhre K, Notredame C, Claverie JM, Abergel C. Related Articles, Links

CaspR: a web server for automated molecular replacement using homology modelling.
Nucleic Acids Res. 2004 Jul 1;32(Web Server issue):W606-9.
PMID: 15215460 [PubMed - indexed for MEDLINE]

Poirot O, Suhre K, Abergel C, O'Toole E, Notredame C. Related Articles, Links

3DCoffee@igs: a web server for combining sequences and structures into a multiple sequence alignment.
Nucleic Acids Res. 2004 Jul 1;32(Web Server issue):W37-40.
PMID: 15215345 [PubMed - indexed for MEDLINE]

O'Sullivan O, Suhre K, Abergel C, Higgins DG, Notredame C. Related Articles, Links

3DCoffee: combining protein sequences and structures within multiple sequence alignments.
J Mol Biol. 2004 Jul 2;340(2):385-95.
PMID: 15201059 [PubMed - indexed for MEDLINE]

Poirot O, O'Toole E, Notredame C. Related Articles, Links

Tcoffee@igs: A web server for computing, evaluating and combining multiple sequence alignments.
Nucleic Acids Res. 2003 Jul 1;31(13):3503-6.
PMID: 12824354 [PubMed - indexed for MEDLINE]

Notredame C. Related Articles, Links

Mocca: semi-automatic method for domain hunting.
Bioinformatics. 2001 Apr;17(4):373-4.
PMID: 11301309 [PubMed - indexed for MEDLINE]

Notredame C, Higgins DG, Heringa J. Related Articles, Links

T-Coffee: A novel method for fast and accurate multiple sequence alignment.
J Mol Biol. 2000 Sep 8;302(1):205-17.
PMID: 10964570 [PubMed - indexed for MEDLINE]

Notredame C, Holm L, Higgins DG. Related Articles, Links

COFFEE: an objective function for multiple sequence alignments.
Bioinformatics. 1998 Jun;14(5):407-22.
PMID: 9682054 [PubMed - indexed for MEDLINE]

Mocca

Notredame C. Related Articles, Links

Mocca: semi-automatic method for domain hunting.

Bioinformatics. 2001 Apr;17(4):373-4.
PMID: 11301309 [PubMed - indexed for MEDLINE]

CORE

http://igs-server.cnrs-mrs.fr/~cnotred/Publications/Pdf/core.pp.pdf

Other Contributions

Some pieces of code from other packages have been incorporated within the T-Coffee package. These include:

-The Sim algorithm of Huang and Miller that given two sequences computes the N best scoring local alignments.

-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 was adapted from Myers and Miller, in CABIOS, 1988, 11-17, vol. 1)

-Various techniques and algorithms have been implemented. Whenever relevant, the source of the code/algorithm/idea is indicated in the corresponding function.

-64 Bits compliance was implemented by Benjamin Sohn, Performance Computing Center Stuttgart (HLRS), Germany

-Prof David Jones (UCL) reported and corrected the PDB1K bug (now t_coffee/sap can align PDB sequences longer than 1000 AA).

What Is T-COFFEE ?

What is T-Coffee?

Before going deep into the core of the matter, here are a few words to quickly explain some of the things T-Coffee will do for you.

What does it do?

T-Coffee is a multiple sequence alignment program: given a set of sequences previously gathered using database search programs like BLAST, FASTA or Smith and Waterman, T-Coffee will produce a multiple sequence alignment. **To use T-Coffee you must already have your sequences ready**.

What can it align?

T-Coffee will align DNA and protein sequences alike, although it does better at aligning proteins than nucleic acids. It will be able to use structural information for protein sequences with a known structure. We recently introduced a new mode that makes T-Coffee able to accurately align large datasets.

How can I use it?

T-Coffee is not an interactive program. It runs from your UNIX or Linux command line and you must provide it with the correct parameters. If you do not like typing commands, here is the simplest available mode where T-Coffee only needs the name of the sequence file:

```
EXCL: t_coffee sample_seq1.fasta
```

Installing and using T-Coffee requires a minimum acquaintance with the Linux/Unix operating system. If you feel this is beyond your computer skills, we suggest you use one of the available online servers.

Is T-Coffee different from ClustalW?

According to several benchmarks, T-Coffee appears to be more accurate than ClustalW. Yet, this increased accuracy comes at a price: T-Coffee is slower than Clustal (about N times).

If you are familiar with ClustalW, or if you run a ClustalW server, you will find that we have made some efforts to ensure as much compatibility as possible between ClustalW and T-COFFEE. Whenever it was relevant, we have kept the flag name and the flag syntax of ClustalW. Yet, you will find that T-Coffee also has many extra possibilities…

If you want to align closely related sequences, T-Coffee can also be used in a fast mode, much faster than ClustalW, and about as accurate (T-Coffee -very_fast) This mode is especially useful to align long sequences.

What T-Coffee Can and Cannot do for you ...

IMPORTANT: All the files mentioned here (sample_seq...) can be found in the example directory of the distribution.

(NOT) Fetching Sequences

T-Coffee will NOT fetch sequences for you: you must select the sequences you want to align before hand. We suggest you use any BLAST server and format your sequences in FASTA so that T-COFFEE can use them easily.

Aligning Sequences

Making accurate multiple alignments of DNA, RNA or Protein sequences.

Combining Alignments

T-Coffee allows 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 alignment having the best agreement with all these methods (see the FAQ for more details)

```
EXCL: t_coffee –
in=Asample_aln1.aln,Asample_aln2.aln,Asample_aln3.aln –
outfile=combined_aln.aln
```

Evaluating Alignments

You can use T-Coffee to measure the reliability of your Multiple Sequence alignment. If you want to find out about that, read the FAQ or the documentation for the -output flag.

```
EXCL: t_coffee –infile=sample_aln1.aln –special_mode=evaluate
```

Combining Sequences and Structures

One of the latest improvements of T-Coffee is to let you combine sequences and structures, so that your alignments are of higher quality. You need to have sap package installed to fully benefit of this facility.

```
EXCL: t_coffee 3d.fasta -special_mode=3dcoffee
```

Using this mode will cause T-Coffee to automatically identify the target corresponding to your sequence as indicated by an NCBI BLAST. T-Coffee then obtains the required PDB sequences from RCSB. However, if you are also using –template_file, the program will use the template you specified and the corresponding files on your disk.

All these network based operations are carried out using wget. If wget is not installed on your system, you can get it for free from (www.wget.org). To make sure wget is installed on your system, type

```
EXCL: which wget
```

Identifying Occurrences of a Motif: Mocca

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. In other words, if you know the coordinates of one copy of a repeat, you can extract all the other occurrences. If you want to use Mocca, simply type:

```
EXCL: t_coffee -other_pg mocca sample_seq1.fasta
```

The program needs some time to compute a library and it will then prompt you with an interactive menu. Follow the instructions.

How Does T-Coffee works

If you only want to make a standard multiple alignments, you may skip these explanations. But if you want to do more sophisticated things, these few indications may help before you start reading the doc and the papers.

When you run T-Coffee, the first thing it does is to compute a library. The library is a list of pairs of residues that *could* be aligned. It is like a Xmas list: you can ask anything you fancy, but it is down to Santa to assemble a collection of Toys that won't get him stuck at the airport, while going through the metal detector.

Given a standard library, it is not possible to have all the residues aligned at the same time because all the lines of the library may not agree. For instance, line 1 may say

```
Residue 1 of seq A with Residue 5 of seq B,
```

and line 100 may say

```
Residue 1 of seq A with Residue 29 of seq B,
```

Each of these constraints comes with a weight and in the end, the T-Coffee algorithm tries to generate the multiple alignment that contains constraints whose sum of weights yields the highest score. In other words, it tries to make happy as many constraints as possible (replace the word constraint with, friends, family members, collaborators… and you will know exactly what we mean).

You can generate this list of constraints however you like. You may even provide it yourself, forcing important residues to be aligned by giving them high weights (see the FAQ). For your convenience, T-Coffee can generate (this is the default) its own list by making all the possible global pairwise alignments, and the 10 best local alignments associated with each pair of sequences. Each pair of residues observed aligned in these pairwise alignments becomes a line in the library.

Yet be aware that nothing forces you to use this library and that you could build it using other methods (see the FAQ). In protein language, T-COFEE is synonymous for freedom, the freedom of being aligned however you fancy (It is with that sort of statements that I got elected Chief Tryptophan Officer in some previous life).

Installation

Standard Installation

1-decompress distribution.tar.gz

```
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.pdf,t_coffee_doc.html

distribution/t_coffee_source

distribution/example

distribution/html
```

4-go into the main directory and type:

```
./install
```

You will know the installation proceeded completely with the mention:

```
Installation of t_coffee Successful
```

5-add the bin folder to your path:

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

Note: The latest t_coffee distribution (2.15 and higher) is self contained and only requires one executable. You may still require external modules (sap, blast, ClustalW) if you wish to use another mode than the default.

Note: When updating, make sure to remove the old distribution and any associated program from your path.

6-If you have PDB installed:

Assuming you have a standard PDB installation in your file system

```
setenv PDB_DIR pdb_dir/structures/all/
```

Note: This must be added to your login file.

Extended Installation and other Packages

By default, T-Coffee does not require any other package than those included in the distribution. However, depending on your needs, you may want to install some of the following:

Package	Function
ClustalW	can interact with t_coffee
wget	3DCoffee Automatic Downloading of Structures Remote use of the Fugue server
sap	structure/structure comparisons (obtain it from W. Taylor, NIMR-MRC).
Blast	www.ncbi.nih.nlm.gov

Once the package is installed, make sure make sure that the executable is on your path, so that t_coffee ca find it automatically.

Quick Start

IMPORTANT: All the files mentionned here (sampe_seq...) can be found in the example directory of the distribution.

T-COFFEE

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

```
EXCL: t_coffee sample_seq1.fasta
```

This will output two files:

```
sample_seq1.aln: your Multiple Sequence Alignment

sample_seq1.dnd: The Guide tree (newick Format)
```

IMPORTANT: If you are trying to align Nucleic Acid, use –special_mode=dna.

```
EXCL: t_coffee -in sample_dnaseq1.fasta -special_mode=dna
```

MOCCA

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

```
EXCL: t_coffee -other_pg mocca sample_seq1.fasta
```

This command output one files (*<your sequences>.mocca_lib)* and starts an interactive menu.

Recent Modifications

-Use of @ as a separator when specifying methods parameters

-The most notable modifications have to do with the structure of the input. From version 2.20, all files must be tagged to indicate their nature (A: alignment, S: Sequence, L: Library…). We are becoming stricter, but that's for your own good…

Another important modification has to do with the flag -matrix: it now controls the matrix being used for the computation

Manual

This manual is at a very preliminary stage of redaction and will only show you how to do the very basic with T-Coffee. In order to solve a more specific problem, or answer a query, we suggest you first go through the FAQ to see of your problem has been addressed, read it and then read carefully the documentation associated with corresponding flags. Of course, we also welcome queries and do our best to provide answers and clues in a timely manner.

Using T-Coffee

Standard Alignments

T-Coffee can align sequences, structures and profiles. The default mode when using t_coffee is:

```
EXCL: t_coffee sample_seq1.fasta
```

It is also possible to combine sequences from various sources:

```
EXCL: t_coffee sample_seq1.fasta sample_seq2.fasta
```

Or even, sequences coming from sequences and alignment files:

```
EXCL: t_coffee sample_seq1.fasta,Ssample_aln2.aln
```

Note: the 'S' identifier tells the program to use the alignment as a collection of unaligned sequences.

Alignment Combination

It is possible to combine several alignments into one final alignment,

```
EXCL: t_coffee -in Asample_aln1_1.aln,Asample_aln1_2.aln -
outfile=combined_aln.aln
```

Note the 'A' identifier that tells the program to keep the sequences aligned.

Aligning Nucleic Acids

Nucleic acid sequences are difficult to align and T.-Coffee is not especially well gifted. However, if you want to give it a try, you are advised to use the following command line:

```
EXCL: t_coffee sample_dnaseq1.fasta -special_mode dna
```

This special mode triggers the use of slow_pair4dna and lalign_id_pair4dna, that use lower gap estension penalties and the identity matrix. If you would rather use your own matrix, use:

```
t_coffee sample_dnaseq1.fasta -in
Mlalign_id_pair4dna@EP@MATRIX@idmat
```

Where you should replace idmat with your own matrix, in BLAST format (see the format section).

Aligning Sequences and Structures

Assuming some structures are associated with your sequences, it is possible to align these sequences while using associated structural information. The easiest way to do this is to use 3dcoffee.

Aligning Sequences and Profiles

T-Coffee can make multiple profile alignments. In this context, the alignments are treated as single sequences and aligned to one another in a progressive fashion. Currently, we only support profiles under the form of standard multiple sequence alignments. The profile must either be entered via the –profile flag:

```
EXCL: t_coffee -profile sample_aln1.aln,sample_aln2.aln -
outfile=combined_profiles.aln
```

It is also possible to read the profile via the –in flag, as long as they are preceded with the ‘R’ identifier.

```
EXCL: t_coffee Ssample_seq1.fasta,Rsample_aln2.aln -outfile
seqprofile_aln
```

All the internal methods should support profiles. External methods do not support profiles (unless specified otherwise).

Using Structures (Or templates) Within Profiles

If your profiles contain structures, you can make sure these will be used during the computatiuon by specifying the 3dcoffee special mode:

```
EXCL: t_coffee  Rsample_profile1.aln,Rsample_profile2.aln -
special_mode=3dcoffee -outfile=aligned_prf.aln
```

Note that when providing a collection of templates, the program will use the –template_file flag to look for templates within the sequences AND within the profiles associated with some sequences.

Using New and Existing Methods

Although, it does not necessarily do so explicitly, T-Coffee always end up combining libraries. Libraries are collections of pairs of residues. Given a set of libraries, T-Coffee makes an attempt to assemble the alignment with the highest level of consistence. You can think of the alignment as a timetable. Each library pair would be a request from students or teachers, and the job of T-Coffee would be to assemble the time table that makes as many people as possible happy…

In T-Coffee, methods replace the students/professors as constraints generators. These methods can be any standard/non standard alignment methods that can be used to generate alignments (pairwise, most of the time). These alignments can be viewed as collections of constraints that must be fit within the final alignment. Of course, the constraints do not have to agree with one another…

This section shows you what are the vailable method in T-Coffee, and how you can add your own methods, either through direct parameterization or via a perl script.

Using Methods Integrated in T-Coffee

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>
```

LIST OF INTERNAL METHODS

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	
islow_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. The Library used for the computation is the one computed before the method is used. The resullt is therefore dependant on the order in methods and library are set via the –in flag.*
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. [UNSUPORTED]*

lalign_id_pair	*Same as lalign_rs_pir, but using the level of identity as a weight.*
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.*
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, and a standard fast progressive alignment will be computed. If you want to change the substitution matrix used by the methods, use the –matrix flag.*
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.*

LIST OF EXTERNAL METHODS

The following methods are external. They correspond to packages developped by other groups that you may want to run within T-Coffee. We are very open to extending these options and we welcome any request to ad an extra interface:

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_msa	*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.*
probcons_pair	*Probcons package, install the latest version from: http://probcons.stanford.edu/.*
probcons_msa	*idem.*
muscle_pair	*Muscle package, install the latest version from: http://www.drive5.com/muscle/ .*
muscle_msa	*idem.*
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.*
fugue_pair	*Uses fugue to align a structure and a sequence.Fugue does not need to be installed, the call is made through wget* ***[Unsupported]***

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):

```
EXCL: t_coffee -in Ssample_seq1.fasta,Mfast_pair,Mlalign_id_pair
```

MODFYING THE PARAMETERS

It is possible to modify on the fly the parameters of hard coding methods:

```
EXCL: t_coffee sample_seq1.fasta -in
slow_pair@EP@MATRIX@pam250mt@GOP@-10@GEP@-1
```

EP stands for Extra parameters. These parameters will superseed any other parameters.

Integrating External Methods

DIRECT ACCESS TO EXTERNAL METHODS

A special method exists in T-Coffee that can be used to invoke any existing program:

```
EXCL: t_coffee sample_seq1.fasta -in=Mem@clustalw@pairwise
```

In this context, Clustalw is a method that can be ran with the following command line:

```
method -infile=<infile> -outfile=<outfile>
```

Clustalw can be replaced with any method using a similar syntax. If the program you want to use cannot be run this way, you can either write a perl wrapper that fits the bill or write a tc_method file adapted to your program (cf next section).

This special method (em, external method) uses the following syntax:

```
Em@<method>@<aln_mode:pairwise¦s_pairwise|multiple>
```

CUSTOMIZING AN EXTERNAL METHOD (WITH PARAMETERS) FOR T-COFFEE

T-Coffee can run external methods, using a *tc_method* file that can be used in place of an established method. Two such files are incorporated in T-Coffee. You can dump them and customize them according to your needs:

For instance if you have ClustalW installed, you can use the following file to run the

```
EXCL: t_coffee -other_pg unpack_clustalw_method.tc_method

EXCL: t_coffee -other_pg unpack_generic_method.tc_method
```

The second file (generic_method.tc_method) contains many hints on how to customize your new method. The first file is a very straightforward example on how to have t_coffee to run Clustalw with a set of parameters you may be interested in:

```

*TC_METHOD_FORMAT_01
```

```
**************clustalw_method.tc_method*********
EXECUTABLE  clustalw
ALN_MODE          pairwise
IN_FLAG           -INFILE=
OUT_FLAG          -OUTFILE=
OUT_MODE          aln
PARAM       -gapopen=-10
SEQ_TYPE          S
*************************************************
```

This configuration file will cause T-Coffee to emit the following system call:

```
clustalw –INFILE=tmpfile1 –OUTFILE=tmpfile2 –gapopen=-10
```

Note that ALN_MODE instructs t_coffee to run clustalw on every pair of sequences (cf generic_method.tc_method for more details).

The tc_method files are treated like any standard established method in T-Coffee. For instance, if the file *clustalw_method.tc_method* is in your current directory, run:

```
EXCL: t_coffee sample_seq1.fasta –in=Mclustalw_method.tc_method
```

Managing a collection of method files

It may be convenient to store all the method files in a single location on your system. By default, t_coffee will go looking into the directory ~/.t_coffee/methods/. You can change this by either modifying the METHODS_4_TCOFFEE in define_headers.h (and recompile) or by modifying the envoronement variable METHODS_4_TCOFFEE.

Advanced Method Integration

It may sometimes be difficult to customize the program you want to use through a tc_method file. In that case, you may rather use an external perl_script to run your external application. This can easily be achieved using the generic_method.tc_method file.

```
*TC_METHOD_FORMAT_01
**************generic_method.tc_method*********
EXECUTABLE  tc_generic_method.pl
ALN_MODE          pairwise
IN_FLAG           -infile=
OUT_FLAG          -outfile=
OUT_MODE          aln
PARAM       -method clustalw
PARAM       -gapopen=-10
SEQ_TYPE          S
*************************************************
* Note: &bsnp can be used to for  white spaces
```

When you run this method:

```
EXCL: t_coffee sample_seq1.fasta –in=Mgeneric_method.tc_method
```

T-Coffee runs the script tc_generic_method.pl on your data. It also provides the script with parameters. In this case –method clustalw indicates that the script should run clustalw on your data. The script tc_generic_method.pl is incorporated in t_coffee. Over the time, this script will be the place where novel methods will be integrated

will be used to run the script *tc_generic_method.pl*. The file tc_generic_method.pl is a perl file, automatically generated by t_coffee. Over the time this file will make it possible to run all available methods. You can dump the script using the following command:

```
EXCL: t_coffee -other_pg=unpack_tc_generic_method.pl
```

Note: If there is a copy of that script in your local directory, that copy will be used in place of the internal copy of T-Coffee.

The Mother of All method files…

```
*TC_METHOD_FORMAT_01
******************generic_method.tc_method*************
*
*       Incorporating new methods in T-Coffee
*       Cedric Notredame 17/04/05
*
*******************************************************
*This file is a method file
*Copy it and adapt it to your need so that the method
*you want to use can be incorporated within T-Coffee
*******************************************************
*                  USAGE                              *
*******************************************************
*This file is passed to t_coffee via –in:
*
*     t_coffee –in Mgeneric_method.method
*
*     The method is passed to the shell using the following
*call:
*<EXECUTABLE><IN_FLAG><seq_file><OUT_FLAG><outname><PARAM>
*
*Conventions:
*<FLAG_NAME>      <TYPE>           <VALUE>
*<VALUE>:   no_name     <=> Replaced with a space
*<VALUE>:   &nbsp <=> Replaced with a space
*
*******************************************************
*                  EXECUTABLE                         *
*******************************************************
*name of the executable
*passed to the shell: executable
*
EXECUTABLE  tc_generic_method.pl
*
*******************************************************
*                  ALN_MODE                           *
*******************************************************
*pairwise   ->all Vs all (no self )[(n2-n)/2aln]
*m_pairwise ->all Vs all (no self)[n^2-n]^2
*s_pairwise ->all Vs all (self): [n^2-n]/2 + n
*multiple   ->All the sequences in one go
*
ALN_MODE          pairwise
*
*******************************************************
*                  OUT_MODE                           *
```

```
*******************************************************
* mode for the output:
*External methods:
* aln -> alignmnent File (Fasta or ClustalW Format)
* lib-> Library file (TC_LIB_FORMAT_01)
*Internal Methods:
* fL -> Internal Function returning a Lib (Librairie)
* fA -> Internal Function returning an Alignmnent
*
OUT_MODE          aln
*
*******************************************************
*                  IN_FLAG                              *
*******************************************************
*IN_FLAG
*flag indicating the name of the in coming sequences
*IN_FLAG S no_name ->no flag
*IN_FLAG S &nbsp–in&nbsp -> “ –in “
*
IN_FLAG           -infile=
*
*******************************************************
*                  OUT_FLAG                            *
*******************************************************
*OUT_FLAG
*flag indicating the name of the out-coming data
*same conventions as IN_FLAG
*OUT_FLAG   S no_name ->no flag
*
OUT_FLAG          -outfile=
*
*******************************************************
*                  SEQ_TYPE                            *
*******************************************************
*G: Genomic, S: Sequence, P: PDB, R: Profile
*Examples:
*SEQTYPE    S     sequences against sequences (default)
*SEQTYPE    S_P   sequence against structure
*SEQTYPE    P_P   structure against structure
*SEQTYPE    PS    mix of sequences and structure
*
SEQ_TYPE    S
*
*******************************************************
*                  PARAM                               *
*******************************************************
*Parameters sent to the EXECUTABLE
*If there is more than 1 PARAM line, the lines are
*concatenated
*
PARAM -method clustalw
PARAM   -OUTORDER=INPUT -NEWTREE=core -align -gapopen=-15
*
*******************************************************
*                  END                                 *
*******************************************************
```

Creating Your Own T-Coffee Libraries

If the method you want to use is not integrated, or impossible to integrate, you can generate your own libraries, either directly or by turning existing alignments into libraries. You may also want to precompute your libraries, in order to combine them at your convenience.

Using Pre-Computed Alignments

If the method you wish to use is not supported, or if you simply have the alignments, the simplest thing to do 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:

```
EXCL: t_coffee -in=Asample_aln1_1.aln,Asample_aln1_2.aln -
outfile=combined_aln.aln
```

Customizing the Weighting Scheme

The previous integration 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 global pairwise sequence identity. If you want to use a more specific weighting scheme with a given method, you should either:

generate your own library (cf next section)

convert your aln into a lib, using the –weight flag:

```
EXCL: t_coffee -in Asample_aln1.aln -out_lib=test_lib.tc_lib -
lib_only -weight=sim_pam250mt

EXCL: t_coffee -in=Asample_aln1.aln,Ltest_lib.tc_lib -
outfile=outaln

EXCL: t_coffee -
in=Asample_aln1_1.aln,Asample_aln1_2.aln,Mfast_pair,Mlalign_id_pai
r -outfile=out_aln
```

Generating 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.

You may also want to pre-compute your libraries in order to save them for further use. For instance, in the following example, we generate the local and the global libraries and later re-use them for combination into a multiple alignment.

```
EXCL: t_coffee sample_seq1.fasta -in Mslow_pair -out_lib
slow_pair_seq1.tc_lib -lib_only

EXCL: t_coffee sample_seq1.fasta -in Mlalign_id_pair -out_lib
lalign_id_pair_seq1.tc_lib -lib_only
```

Once these libraries have been computed, you can then combine tem at your convenience in a single MSA. Of course you can decide to only use the local or the global library

```
EXCL: t_coffee sample_seq1.fasta -in Llalign_id_pair_seq1.tc_lib
Lslow_pair_seq1.tc_lib
```

Frequently Asked Questions

IMPORTANT: All the files mentionned here (sampe_seq...) can be found in the example directory of the distribution.

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.

A: Your sequences may not be recognized for what they really are. Normally T-Coffee recognizes the type of your sequences automatically, but if it fails, use:

```
EXCL: t_coffee sample_seq1.fasta -type=PROTEIN
```

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 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 whether 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:

```
EXCL: t_coffee sample_seq1.fasta -special_mode low_memory
```

Note that this mode will be much less time efficient than the default, although it may be slightly more accurate. In practice the parameterization associate with special mode turns off every memory expensive heuristic within T-Coffee. For version 2.11 this amounts to

```
EXCL: t_coffee  sample_seq1.fasta -in=Mslow_pair,Mlalign_id_pair -
distance_matrix_mode=slow -dp_mode=myers_miller_pair_wise
```

If you keep running out of memory, you may also want to lower –maxnseq, to ensure that t_coffee_dpa will be used.

Input/Output Control

Q: How many Sequences can t_coffee handle

A: T-Coffee is limited to a maximum of 50 sequences. Above this number, the program automatically switches to a heuristic mode, named DPA, where DPA stands for Double Progressive Alignment.

DPA is still in development and the version currently shipped with T-Coffee is only a beta version.

Q: How many ways to pass parameters to t_coffee?

A: See the section well behaved parameters

Q: How can I change the default output format?

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

```
EXCL: t_coffee sample_seq1.fasta -output=msf,fasta_aln
```

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: Specify stderr or stdout as output filename, the output will be redirected accordingly. For instance

```
EXCL: t_coffee sample_seq1.fasta -outfile=stdout -out_lib=stdout
```

This instruction 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:

```
EXCL: cat sample_seq1.fasta | t_coffee -infile=stdin
```

will be equivalent to

```
EXCL: t_coffee sample_seq1.fasta
```

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

```
EXCL: cat sample_param_file.param  | t_coffee -parameters=stdin
```

For instance:

```
EXCL: echo -in=Ssample_seq1.fasta,Mclustalw_pair | t_coffee -
parameters=stdin
```

Q: Can I read my parameters from a file?

A: See the well behaved parameters section.

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

A: Use the *-run_name* flag.

```
EXCL: t_coffee sample_seq1.fasta -run_name=guacamole
```

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:

```
EXCL: t_coffee Ssample_aln1.aln

EXCL: t_coffee -infile=Ssample_aln1.aln

EXCL: t_coffee -in=Ssample_aln1.aln,Mslow_pair,Mlalign_id_pair -
outfile=outaln
```

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 –lib_only flag

```
EXCL: t_coffee sample_seq1.fasta -out_lib=sample_lib1.tc_lib -
lib_only
```

Please, note that the previous usage supersedes the use of the –convert flag. Its main advantage is to restrict computation time to the actual library computation.

Q: I want to turn an alignment into a library

A: use the –lib_only flag

```
EXCL: t_coffee -in=Asample_aln1.aln -out_lib=sample_lib1.tc_lib -
lib_only
```

It is also possible to control the weight associated with this alignment (see the –weight section).

```
 EXCL: t_coffee -in=Asample_aln1.aln -out_lib=sample_lib1.tc_lib -
lib_only -weight=1000
```

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.

```
EXCL: t_coffee -in Lsample_lib1.tc_lib Lsample_lib2.tc_lib -
lib_only -out_lib=sample_lib3.tc_lib
```

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

A: An alignment is ALWAYS considered as a library AND a set of sequences. If you want your alignment to be considered as a library only, use the S identifier.

```
EXCL: t_coffee Ssample_aln1.aln -outfile=outaln
```

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

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:

```
EXCL: t_coffee sample_seq1.fasta -output=score_ps
```

or

```
EXCL: t_coffee sample_seq2.fasta -output=score_pdf
```

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,

```
EXCL: t_coffee sample_seq1.fasta -
output=clustalw,score_html,score_ps,msf
```

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

Note: you need to have the converter ps2pdf installed on your system (standard under Linux and cygwin). The latest versions of Internet Explorer and Netscape now allow the user to print the HTML display *Do not forget to request Background printing.*

Alignment Computation

Q: Can t_coffee align Nucleic Acids ???

A: yes it can, but you must use the –special_mode=dna:

```
EXCL: t_coffee sample_dnaseq1.fasta –special_mode dna
```

Q: I do not want to compute the alignment.

A: use the -convert flag

```
EXCL: t_coffee sample_aln1.aln -convert -output=gcg
```

This command will read the .aln file and turn it into an .msf alignment.

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

If you want to brutally force some residues to be aligned, you may use as a post processing, the force_aln function of seq_reformat:

```
EXCL: t_coffee –other_pg seq_reformat –in sample_aln4.aln –action
+force_aln seq1 10 seq2 15
```

```
EXCL: t_coffee –other_pg seq_reformat –in sample_aln4.aln –action
+force_aln sample_lib4.tc_lib02
```

sample_lib4.tc_lib02 is a T-Coffee library using the tc_lib02 format:

```
*TC_LIB_FORMAT_02

SeqX resY ResY_index SeqZ ResZ ResZ_index
```

The TC_LIB_FORMAT_02 is still experimental and unsupported. It can only be used in the context of the force_aln function described here.

Given more than one constraint, these will be applied one after the other, in the order they are provided. This greedy procedure means that the Nth constraint may disrupt the (N-1)th previously imposed constraint, hence the importance of forcing the constraints in the right order, with the most important coming last.

We do not recommend imposing hard constraints on an alignment, and it is much more advisable to use the soft constraints provided by standard t_coffee libraries (cf. building your own libraries section)

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:

```
EXCL: t_coffee -in Asample_seq1.aln -weight=1000 -
out_lib=sample_seq1.tc_lib -lib_only
```

The value 1000 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 *sample_aln1.tc_lib* that you can later use when aligning all the sequences:

```
EXCL: t_coffee -in Ssample_seq1.fasta Lsample_seq1.tc_lib -outfile
sample_seq1.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 or use the –force_aln option (cf FAQ: I would like to force some residues to be aligned). 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.

```
EXCL: t_coffee sample_seq1.fasta -usetree=sample_tree.dnd
```

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.

```
EXCL: t_coffee sample_seq4.fasta -in Mcdna_fast_pair
```

Notice that in the resulting alignments, all the gaps are of modulo3, except one small gap in the first line of sequence hmgl_trybr. This is a framshift, made on purpose. You can realign the same sequences while ignoring their coding potential and treating them like standard DNA:

```
EXCL: t_coffee sample_seq4.fasta
```

Note: This method has not yet been fully tested and is only provided "as-is" with no warranty.

Q: I do not want to use all the possible pairs when computing the library

Q: I only want to use specific pairs to compute the

library

A: Simply write in a file the list of sequence groups you want to use:

```
EXCL: t_coffee sample_seq1.fasta -in=Mclustalw_pair,Mclustalw_msa
-lib_list=sample_list1.lib_list -outfile=test
```

```
***************sample_list1.lib_list****
2 hmgl_trybr hmgt_mouse
2 hmgl_trybr hmgb_chite
2 hmgl_trybr hmgl_wheat
3 hmgl_trybr hmgl_wheat hmgl_mouse
***************sample_list1.lib_list****
```

Note: Pairwise methods (slow_pair…) will only be applied to list of pairs of sequences, while multiple methods (clustalw_aln) will be applied to any dataset having more than two sequences.

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 over-represented sequences.

Using Structures and Profiles

Q: Can I align sequences to a profile with T-Coffee?

A: Yes, you simply need to indicate that your alignment is a profile with the R tag..

```
EXCL: t_coffee sample_seq1.fasta Rsample_aln2.aln -outfile tacos
```

Q: Can I align sequences Two or More Profiles?

A: Yes, you, simply tag your profiles with the letter R and the program will treat them like standard sequences.

```
EXCL: t_coffee Rsample_aln1.fasta Rsample_aln2.aln -outfile tacos
```

Q: Can I align two profiles according to the structures they contain?

A: Yes. As long as the structure sequences are named according to their PDB identifier

```
EXCL: t_coffee  Rsample_profile1.aln,Rsample_profile2.aln -
special_mode=3dcoffee -outfile=aligne_prf.aln
```

Q: T-Coffee becomes very slow when combining sequences and structures

A: This is true. By default the structures are feteched on the net, using RCSB. The problem arises when T-Coffee looks for the structure of sequences WITHOUT structures. One solution is to install PDB locally. In that case you will need to set two environement variables:

```
setenv PDB_DIR="directory containing the pdb structures"
setenv NO_REMOTE_PDB_DIR=1
```

Interestingly, the observation that sequences without structures are those that take the most time to be checked is a reminder of the strongest rational argument that I know of against torture: any innocent would require the maximum amount of torture to establish his/her innocence, which sounds...ahem...strange., and at least inneficient. Then again I was never struck by the efficiency of the Bush administration...

Alignment Evaluation

Q: How good is my alignment?

A: see what is the color index?

Q: What is that color index?

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

```
EXCL: t_coffee sample_seq1.fasta -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.

A book chapter describing the CORE index is available on:

http://igs-server.cnrs-mrs.fr/~cnotred/Publications/Pdf/core.pp.pdf

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:

```
EXCL: t_coffee -infile=sample_aln1.aln -in=Lsample_aln1.tc_lib -
special_mode=evaluate
```

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

```
EXCL: t_coffee -infile=sample_aln1.aln -special_mode evaluate
```

Q: Can I Compare Two Alignments?

A: Yes. You can treat one of your alignments as a library and compare it with the second alignment:

```
EXCL: t_coffee -infile=sample_aln1_1.aln -in=Asample_aln1_2.aln -
special_mode=evaluate
```

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

```
EXCL: t_coffee -infile=sample_aln1.aln -special_mode evaluate
```

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

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

```
EXCL: t_coffee sample_seq1.fasta -ktuple=3
```

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

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

A: If there is no duplicated name in your sequence set, T-Coffee's handling of names is consistent with Clustalw, (Cf Sequence Name Handling in the Format section). If your dataset contains sequences with identical names, these will automatically be renamed to:

```
************************
>seq1
>seq1
************************
>seq1
>seq1_1
************************
```

Warning: The behaviour is undefined when this creates two sequence with a similar names.

Reference Manual

This reference manual gives a list of all the flags that can be used to modify the behavior of T-Coffee. For your convenience, we have grouped them according to their nature. To display a list of all the flags used in the version of T-Coffee you are using (along with their default value), type:

```
EXCL: t_coffee
```

Or

```
EXCL: t_coffee -help
```

Or

```
EXCL: t_coffee -help -in
```

Or any other parameter

Well Behaved Parameters

Separation

You can use any kind of separator you want (i.e. ,; <space>=). The syntax used in this document is meant to be consistent with that of ClustalW. However, in order to take advantage of the automatic filename compleation provided by many shells, you can replace "=" and "," with a space.

Posix

T-Coffee is not POSIX compliant.

Entering the right parameters

There are many ways to enter parameters in T-Coffee, see the -parameter flag in

Parameters Priority

In general you will not need to use these complicated parameters. Yet, if you find

yourself typing long command lines on a regular basis, it may be worth reading this section.

One may easily feel confused with the various manners in which the parameters can be passed to t_coffee. The reason for these many mechanisms is that they allow several levels of intervention. For instance, you may install t_coffee for all the users and decide that the defaults we provide are not the proper ones… In this case, you will need to make your own t_coffee_default file.

Later on, a user may find that he/she needs to keep re-using a specific set of parameters, different from those in t_coffee_default, hence the possibility to write an extra parameter file: parameters. In summary:

-parameters > prompt parameters > -t_coffee_defaults > -special_mode

This means that *-parameters* supersede all the others, while parameters provided via *-special mode* are the weakest.

Parameters Syntax

No Flag

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

```
EXCL: t_coffee sample_seq1.fasta
```

Which is equivalent to

```
EXCL: t_coffee Ssample_seq1.fasta
```

When you do so, **sample_seq1** is used as a name prefix for every file the program outputs.

-parameters

Usage: -parameters=parameters_file

Default: no parameters file

Indicates a file containing extra parameters. Parameters read this way behave as if they had been added on the right end of the command line that they either supersede(one value parameter) or complete (list of values). For instance, the following file (parameter.file) could be used

```
*******sample_param_file.param********
      -in=Ssample_seq1.fasta,Mfast_pair
      -output=msf_aln
**************************************
```

Note: This is one of the exceptions (with –infile) where the identifier tag (S,A,L,M…) can be omitted. Any dataset provided this way will be assumed to be a sequence (S). These exceptions have been designed to keep the program compatible with ClustalW.

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

Used with:

```
EXCL: t_coffee -parameters=sample_param_file.param
```

Will cause t_coffee to apply the fast_pair method onto to the sequences contained in sample_seq.fasta. If you wish, you can also pipe these arguments into t_coffee, by naming the parameter file "stdin" (as a rule, any file named stdin is expected to receive its content via the stdin)

```
cat sample_param_file.param  | t_coffee -parameters=stdin
```

-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 -parameters. The file used is either:

1. <file name> if a name has been specified
2. **~/.t_coffee_defaults** if no file was specified
3. The file indicated by the environment variable **TCOFFEE_DEFAULTS**

-special_mode

Usage: -special_mode= hard coded mode

Default: not used.

It indicates that t_coffee will use some hard coded parameters. These include:

quickaln: very fast approximate alignment

dali: a mode used to combine dali pairwise alignments

evaluate: defaults for evaluating an alignment

3dcoffee: runs t_coffee with the 3dcoffee parameterization

dna: runs t_coffee with appropriate parameters

Other modes exist that are not yet fully supported

-score [Deprecated]

Usage: -score

Default: not used

Toggles on the evaluate mode and causes t_coffee to evaluates a precomputed alignment provided via **-infile=<alignment>**. The flag **-output** must be set to an appropriate format (i.e. -output=score_ascii, score_html or score_pdf). A better default parameterization is obtained when using the flag **-special_mode=evaluate.**

-evaluate

Usage: -evaluate

Default: not used

Replaces –score. This flag toggles on the evaluate mode and causes t_coffee to evaluates a pre-computed alignment provided via **-infile=<alignment>**. The flag **-output** must be set to an appropriate format (i.e. -output=score_ascii, score_html or score_pdf).

The main purpose of –evaluate is to let you control every aspect of the evaluation. Yet it is advisable to use pre-defined parameterization: **special_mode=evaluate.**

```
EXCL: t_coffee –infile=sample_aln1.aln -special_mode=evaluate

EXCL: t_coffee –infile=sample_seq1.aln –in  Lsample_lib1.tc_lib –
special_mode=evaluate
```

-convert [cw]

Usage: -convert

Default: turned off

Toggles on the conversion mode and causes T-Coffee to convert the sequences, alignments, libraries or structures provided via the **-infile** and **-in** flags. The output format must be set via the **-output** flag. 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).

This flag is ClustalW compliant.

-do_align [cw]

Usage: -do_align

Default: turned on

Special Parameters

-version

Usage: -version

Default: not used

Returns the current version number

-check_configuration

Usage: -check_configuration

Default: not used

Checks your system to determine whether all the programs T-Coffee can interact with are installed.

-cache

Usage: -cache=<use, update, ignore, <filename>>

Default: -cache=use

By default, t_coffee stores in a cache directory, the results of computationally expensive (structural alignment) or network intensive (BLAST search) operations.

-update

Usage: -update

`Default: turned off`

Causes a ***wget*** access that checks whether the t_coffee version you are using needs updating.

-full_log

Usage: -full_log=<filename>

`Default: turned off`

Causes t_coffee to output a full log file that contains all the input/output files.

-other_pg

Usage: -other_pg=<filename>

`Default: turned off`

Some rumours claim that Tetris is embedded within T-Coffee and could be ran using some special set of commands. We wish to deny these rumours, although we may admit that several interesting reformatting programs are now embedded in t_coffee and can be ran through the –other_pg flag.

```
EXCL: t_coffee -other_pg=seq_reformat

EXCL: t_coffee -other_pg=unpack_all

EXCL: t_coffee -other_pg=unpack_extract_from_pdb
```

Input

Sequence Input

-infile [cw]

To remain compatible with ClustalW, it is possible to indicate the sequences with this flag

```
EXCL: t_coffee -infile=sample_seq1.fasta
```

Note: Common multiple sequence alignments format constitute a valid input format.

Note: T-Coffee automatically removes the gaps *before* doing the alignment. This behaviour is different from that of ClustalW where the gaps are kept.

-in (Cf –in from the Method and Library Input section)

-get_type

Usage: -get_type

`Default: turned off`

Forces t_coffee to identify the sequences type (PROTEIN, DNA).

-type [cw]

Usage: -type=DNA ¦ PROTEIN¦ DNA_PROTEIN

`Default: -type=<automatically set>`

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

Note: In case of low complexity or short sequences, it is recommended to set the type manually.

-seq

Usage: -seq=[<P,S><name>,]

`Default: none`

-seq is now the recommended flag to provide your sequences. It behaves mostly like the -in flag.

-seq_source

Usage: -seq_source=<ANY or _LS or LS >

`Default: ANY.`

You may not want to combine all the provided sequences into a single sequence list. You can do by specifying that you do not want to treat all the –in files as potential sequence sources.

-seq_source=_LA indicates that neither sequences provided via the A (Alignment) flag or via the L (Library flag) should be added to the sequence list.

-seq_source=S means that only sequences provided via the S tag will be considered. All the other sequences will be ignored.

Note: This flag is mostly designed for interactions between T-Coffee and T-CoffeeDPA (the large scale version of T-Coffee).

Structure Input

-pdb

Usage: -pdb=<pdbid1>,<pdbid2>…[Max 200]

`Default: None`

Reads or fetch a pdb file. It is possible to specify a chain or even a sub-chain:

```
PDBID(PDB_CHAIN)[opt] (FIRST,LAST)[opt]
```

Tree Input

-usetree

Usage: -usetree=<tree file>

```
Default: No file specified

Format: newick tree format (ClustalW Style)
```

This flag indicates that rather than computing a new dendrogram, t_coffee must 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, Phylips and any other program.

Methods and Library Input

The -in Flag and its Identifier TAGS

<-in> is the real grinder of T-Coffee. Sequences, methods and alignments all pass through so that T-Coffee can turn it all into a single list of constraints (the library). Everything is done automatically with T-Coffee going through each file to extract the sequences it contains. The methods are then applied to the sequences. Pre-compiled constraint list can also be provided. Each file provided via this flag must be preceded with a symbol (Identifier TAG) that indicates its nature to T-Coffee.

P	**pdb structure**
S	**for sequences (use it as well to treat an MSA as unaligned sequences)**
M	**Methods used to build the library**
L	**Pre-computed T-Coffee library**
A	**Multiple Alignments that must be turned into a Library**
X	**Substitution matrices.**
R	**Profiles. This is a legal multiple alignments that will be treated as single sequences (the sequences it contains will not be realigned).**

-in

Usage: -in=[<P,S,A,L,M,X><name>,]

```
Default: -in=Mlalign_id_pair,Mclustalw_pair
```

See the box for an explanation of the -in flag. The following argument passed via -in

```
EXCL: t_coffee -
in=Ssample_seq1.fasta,Asample_aln1.aln,Asample_aln2.msf,Mlalign_id
_pair,Lsample_lib1.tc_lib -outfile=outaln
```

This command will trigger the following chain of events:

1-Gather all the sequences

Sequences within all the provided files are pooled together. Format recognition is automatic. Duplicates are removed (if they have the same name). Duplicates in a single file are only tolerated in FASTA format file, although they will cause sequences to be renamed.

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-Turn alignments into libraries

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.

Note as well the following rules:

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

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

3-No Duplicate: Each file should contain only one copy of each sequence. Duplicates are only allowed in FASTA files but will cause the sequences to be renamed.

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

```
aln 1:hgab1   AAAAABAAAAA

aln 2:hgab1   AAAAAAAAAACCC
```

will cause the program to reconstruct and use 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 merged, they will cause the program to exit with an information message.

5-Methods: The method describer can either be built in (See ### for a list of all the available methods) or be a file describing the method to be used. The exact syntax is provided in part 4 of this manual.

6-Substitution Matrices: If the method is a substitution matrix (X) then no other type of information should be provided. For instance:

```
EXCL: t_coffee sample_seq1.fasta -in=Xpam250mt  -gapopen=-10  -
gapext=-1
```

This command results in a progressive alignment 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 Input

-profile

Usage: -profile=[<name>,] maximum of 200 profiles.

Default: no default

This flag causes T-Coffee to treat multiple alignments as a single sequences, thus making it possible to make multiple profile alignments. The profile-profile alignment is controlled by -profile_mode and -profile_comparison. When provided with the **-in** flag, profiles must be preceded with the letter R.

```
EXCL: t_coffee –profile sample_aln1.aln,sample_aln2.aln –
outfile=profile_aln

EXCL: t_coffee –in
Rsample_aln1.aln,Rsample_aln2.aln,Mslow_pair,Mlalign_id_pair –
outfile=profile_aln
```

Note that when using –template_file, the program will also look for the templates associated with the profiles, even if the profiles have been provided as templates themselves (however it will not look for the template of the profile templates of the profile templates…)

-profile1 [cw]

Usage: -profile1=[<name>], one name only

Default: no default

Similar to the previous one and was provided for compatibility with ClustalW.

-profile2 [cw]

Usage: -profile1=[<name>], one name only

Default: no default

Similar to the previous one and was provided for compatibility with ClustalW.

Alignment Computation

Library Computation: Methods

-lalign_n_top

Usage: -lalign_n_top=<Integer>

Default: -lalign_n_top=10

Number of alignment reported by the local method (lalign).

-align_pdb_param_file

Unsuported

-align_pdb_hasch_mode

Unsuported

Library Computation: Extension

-lib_list [Unsupported]

Usage: -lib_list=<filename>

Default:unset

Use this flag if you do not want the library computation to take into account all the possible pairs in your dataset. For instance

Format:

```
2 Name1 name2
2 Name1 name4
3 Name1 Name2 Name3…
```

(the line 3 would be used by a multiple method).

-do_normalise

Usage: -do_normalise=<0 or a positive value>

Default:-do_normalise=1000

Development Only

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

Development Only

When turned on, this flag indicates that the library extension should be carried out when performing the multiple alignment. If **-extend =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.

-extend_mode

Usage: -extend=<string>

Default:-extend=very_fast_triplet

Warning: Development Only

Controls the algorithm for matrix extension. Available modes include:

relative_triplet	Unsupported
g_coffee	Unsupported
g_coffee_quadruplets	Unsupported
fast_triplet	Fast triplet extension
very_fast_triplet	slow triplet extension, limited to the **-max_n_pair** best sequence pairs when aligning two profiles
slow_triplet	Exhaustive use of all the triplets
mixt	Unsupported
quadruplet	Unsupported
test	Unsupported

matrix	Use of the matrix **-matrix**
fast_matrix	Use of the matrix **-matrix**. Profiles are turned into consensus

-max_n_pair

Usage: -max_n_pair=<integer>

```
Default:-extend=10
```

```
Development Only
```

Controls the number of pairs considered by the **-extend_mode**=very_fast_triplet. Setting it to 0 forces all the pairs to be considered (equivalent to **-extend_mode**=slow_triplet).

-seq_name_for_quadruplet

Usage: Unsupported

-compact

Usage: Unsupported

-clean

Usage: Unsupported

-maximise

Usage: Unsupported

-do_self

Usage: Flag -do_self

Default: No

This flag causes the extension to carried out within the sequences (as opposed to between sequences). This is necessary when looking for internal repeats with Mocca.

-seq_name_for_quadruplet

Usage: Unsupported

-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 length centered on that pair. For instance winsim10 defines a window of 10 residues around the pair being considered. This gives its own weight to each residue in the output library. In our hands, this type of weighting scheme has not provided any significant improvement over the standard sim value.

```
EXCL: t_coffee sample_seq1.fasta -weight=winsim10 -
out_lib=test.tc_lib
```

sim indicates that the weight equals the average identity within the sequences containing the matched 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.

```
EXCL: t_coffee sample_seq1.fasta -weight=winsim10 –
out_lib=test.tc_lib
```

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:

```
EXCL: t_coffee sample_seq1.fasta -weight=1000 –out_lib=test.tc_lib
```

Tree Computation

-distance_matrix_mode

Usage: -distance_matrix_mode=<slow, fast, very_fast>

Default: very_fast

This flag indicates the method used for computing the distance matrix (distance between every pair of sequences) required for the computation of 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 blosum62mt.
ktup	Ktup matching (Muscle kind)
aln	Read the distances on a precomputed MSA

-quicktree [CW]

Usage: -quicktree

Description: Causes T-Coffee to compute a fast approximate

guide tree

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

```
EXCL: t_coffee sample_seq1.fasta –distance_matrix_mode=very_fast
```

```
EXCL: t_coffee sample_seq1.fasta –quicktree
```

Pair-wise Alignment Computation

Controlling Alignment Computation

Most parameters in this section refer to the alignment mode fasta_pair_wise and cfatsa_pair_wise. When using these alignment modes, things proceed as follow:
1-Sequences are recoded using a degenerated alphabet provided with <-sim_matrix>
2-Recoded sequences are then hashed into ktuples of size <u><-ktup></u>
3-Dynamic programming runs on the <u><-ndiag></u> best diagonals whose score is higher than <u><-diag_threshold></u>, the way diagonals are scored is controlled via <u><-diag_mode></u> .
4-The Dynamic computation is made to optimize either the library scoring scheme (as defined by the <u>-in</u> flag) or a substitution matrix as provided via the <u>-matrix</u> flag. The penalty scheme is defined by <u>-gapopen</u> and <u>-gapext</u>. If <u>-gapopen</u> is undefined, the value defined in <u>-cosmetic_penalty</u> is used instead.
5-Terminal gaps are scored according to <u>-tg_mode</u>

-dp_mode

Usage: -dp_mode=<string>

`Default: -dp_mode=cfasta_fair_wise`

This flag indicates the type of dynamic programming used by the program:

```
EXCL: t_coffee sample_seq1.fasta -dp_mode myers_miller_pair_wise
```

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 times slower than gotoh and only accepts *tg_mode=1or 2 (i.e. gaps penalized for opening).*

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. This mode is controlled by the parameters **-ktuple, -diag_mode and -ndiag**

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). This mode is controlled by the parameters **-ktuple, -diag_mode and –ndiag**

Note: Users may find by looking into the code that other modes with fancy names exists (viterby_pair_wise…) Unless mentioned in this documentation, these modes are not supported.

-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 can be a degenerated version, set with **<u>-sim_matrix.</u>**.

-ndiag

Usage: -ndiag=<value>

`Default: -ndiag=0`

Indicates the number of diagonals used by the *fasta_pair_wise* algorithm (cf **-dp_mode**). When **-ndiag=0**, n_diag=Log (length of the smallest sequence)+1.

When –ndiag and –diag_threshold are set, diagonals are selected if and only if they fulfill both conditions.

-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 (useful when dealing with fragments of sequences).

-diag_threshold

Usage: -diag_threshold=<value>

`Default: -diag_threshold=0`

Sets the value of the threshold when selecting diagonals.

0: indicates that –ndiag should be used to select the diagonals (cf –ndiag section).

-sim_matrix

Usage: -sim_matrix=<string>

`Default: -sim_matrix=vasiliky`

Indicates the manner in which the amino acid alphabet is degenerated when hashing in the fasta_pairwise dynamic programming. Standard ClustalW matrices are all valid. They are used to define groups of amino acids having positive substitution values. In T-Coffee, the default is a 13 letter grouping named Vasiliky, with residues grouped as follows:

```
rk, de, qh, vilm, fy (other residues kept alone).
```

This alphabet is set with the flag **-sim_matrix=vasiliky**. In order to keep the alphabet non degenerated, **-sim_matrix=idmat** can be used to retain the standard alphabet.

-matrix [CW]

Usage: -matrix=<blosum62mt>

`Default: -matrix=blosum62mt`

The usage of this flag has been modified from previous versions, due to frequent mistakes in its usage. This flag sets the matrix that will be used by alignment methods within t_coffee (slow_pair, lalign_id_pair). It does not affect external methods (like clustal_pair, clustal_aln…).

-nomatch

Usage: -nomatch=<positive value>

`Default: -nomatch=0`

Indicates the penalty to associate with a match. When using a library, all matches are positive or equal to 0. Matches equal to 0 are unsupported by the library but non-penalized. Setting nomatch to a non-negative value makes it possible to penalize these null matches and prevent unrelated sequences from being aligned (this can be useful when the alignments are meant to be used for structural modeling).

-gapopen

Usage: -gapopen=<negative value>

`Default: -gapopen=0`

Indicates the penalty applied for opening a gap. The penalty must be negative. If no value is provided when using a substitution matrix, a value will be automatically computed.

-gapext

Usage: -gapext=<negative value>

`Default: -gapext=0`

Indicates the penalty applied for extending a gap (cf **-gapopen**)

-fgapopen

Unsupported

-fgapext

Unsupported

-cosmetic_penalty

Usage: -cosmetic_penalty=<negative value>

`Default: -cosmetic_penalty=-50`

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: terminal gaps penalized with ***-gapopen*** + ***-gapext*len***

1: terminal gaps penalized with a ***-gapext*len***

2: terminal gaps unpenalized.

Weighting Schemes

-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 use your own weights (see the format section).

Multiple Alignment Computation

-msa_mode

Usage: -msa_mode=<tree,graph,precomputed>

`Default: -evaluate_mode=tree`

Unsupported

-profile_comparison

Usage: -profile_mode=<fullN,profile>

`Default: -profile_mode=full50`

The profile mode flag controls the multiple profile alignments in T-Coffee. There are two instances where t_coffee can make multiple profile alignments:

1-When N, the number of sequences is higher than **–maxnseq,** the program switches to its multiple profile alignment mode (t_coffee_dpa).

2-When MSAs are provided via the **–profile** flag or via **–profile1** and **–profile2**.

In these situations, the –profile_mode value influences the alignment computation, these values are:

–profile_comparison =profile, the MSAs provided via –profile are vectorized and the function specified by –profile_comparison is used to make profile profile alignments. In that case, the complexity is NL^2

-profile_comparison=fullN, N is an integer value that can omitted. *Full* indicates that given two profiles, the alignment will be based on a library that includes every possible pair of sequences between the two profiles. If N is set, then the library will be restricted to the N most similar pairs of sequences between the two profiles, as judged from a measure made on a pairwise alignment of these two profiles.

-profile_mode

Usage: -profile_mode=<cw_profile_profile, muscle_profile_profile, multi_channel>

`Default: -profile_mode=cw_profile_profile`

When **–profile_comparison=profile**, this flag selects a profile scoring function.

Alignment Post-Processing

-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 cleaning algorithm is greedy. It starts from the top left segment of low constituency 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.

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

Note: The use of the cleaning option can result in memory overflow when aligning large sequences,

-clean_threshold

Usage: -clean_threshold=<0-9>

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 accepted modes.

-iterate

Usage: -iterate=<integer>

`Default: -iterate=0`

Sequences are extracted in turn and realigned to the MSA. If iterate is set to -1, each sequence is realigned, otherwise the number of iterations is set by –iterate.

CPU Control

Multithreading

-multi_thread [NOT Supported]

Usage: -multi_thread=<N>

`Default: 0`

Specifies that the program should be used in multithreading mode. N specifies the number of processors available.

```
EXCL: t_coffee sample_seq2.fasta -multi_thread 4
```

If you are using a quadriprocessor

Limits

-mem_mode

Usage: deprecated

-ulimit

Usage: -ulimit=<value>

`Default: -ulimit=0`

Specifies the upper limit of memory usage (in Megabytes). Processes exceeding this limit will automatically exit. A value 0 indicates that no limit applies.

-maxlen

Usage: -maxlen=<value, 0=nolimit>

`Default: -maxlen=1000`

Indicates the maximum length of the sequences.

Aligning more than 100 sequences with DPA

-maxnseq

Usage: -maxnseq=<value, 0=nolimit>

`Default: -maxnseq=50`

Indicates the maximum number of sequences before triggering the use of t_coffee_dpa.

-dpa_master_aln

Usage: -dpa_master_aln=<File, method>

`Default: -dpa_master_aln=NO`

When using dpa, t_coffee needs a seed alignment that can be computed using any appropriate method. By default, t_coffee computes a fast approximate alignment.

A pre-alignment can be provided through this flag, as well as any program using the following syntax:

```
your_script -in <fasta_file> -out <file_name>
```

-dpa_maxnseq

Usage: -dpa_maxnseq=<integer value>

`Default: -dpa_maxnseq=30`

Maximum number of sequences aligned simultaneously when DPA is ran. Given the tree computed from the master alignment, a node is sent to computation if it controls more than –**dpa_maxnseq** OR if it controls a pair of sequences having less than –**dpa_min_score2** percent ID.

-dpa_min_score1

Usage: -dpa_min_score1=<integer value>

`Default: -dpa_min_score1=95`

Threshold for not realigning the sequences within the master alignment. Given this alignment and the associated tree, sequences below a node are not realigned if none of them has less than –**dpa_min_score1** % identity.

-dpa_min_score2

Usage: -dpa_min_score2

`Default: -dpa_min_score2`

Maximum number of sequences aligned simultaneously when DPA is ran. Given the tree computed from the master alignment, a node is sent to computation if it controls more than –**dpa_maxnseq** OR if it controls a pair of sequences having less than –**dpa_min_score2** percent ID.

-dap_tree [NOT IMPLEMENTED]

Usage: -dpa_tree=<filename>

`Default: -unset`

Guide tree used in DPA. This is a newick tree where the distance associated with each node is set to the minimum pairwise distance among all considered sequences.

Using Structures

Generic

-special_mode

Usage: -special_mode=3dcoffee

`Default: turned off`

Runs t_coffee with the 3dcoffee mode (cf next section).

-check_pdb_status

Usage: -check_pdb_status

`Default: turned off`

Forces t_coffee to run extract_from_pdb to check the pdb status of each sequence. This can considerably slow down the program.

3D Coffee: Using SAP

It is possible to use t_coffee to compute multiple structural alignments. To do so, ensure that you have the sap program installed.

```
EXCL: t_coffee –in=struc1.pdb,struc2.pdb,struc3.pdb,Msap_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

align_pdb: uses a t_coffee implementation of sap, not as accurate.

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 **t_coffee –other_pg extract_from_pdb** or any pdb handling program.

You can provide t_coffee with a mixture of sequences and structure. In this case, you should use the special mode:

```
EXCL: t_coffee -special_mode 3dcoffee -seq 3d_sample3.fasta -
template_file template_file.template
```

Using/finding PDB templates for the Sequences

-template_file

Usage: -template_file =

<filename,

SCRIPT_scriptame,

SELF_TAG

SEQFILE_TAG_filename,

no>

`Default: no`

This flag instructs t_coffee on the templates that will be used when combining several types of information. For instance, when using structural information, this file will indicate the structural template that corresponds to your sequences. The identifier T indicates that the file should be a FASTA like file, formatted as follows. There are several ways to pass the templates:

1-File name

This file contains the sequence/template association it uses a FASTA-like format, as follows:

```
><sequence name> _P_ <pdb template>
><sequence name> _G_ <gene template>
><sequence name> _R_ <MSA template>
```

Each template will be used in place of the sequence with the appropriate method. For instance, structural templates will be aligned with sap_pair and the information thus generated will be transferred onto the alignment.

Note the following rule:

-Each sequence can have one template of each type (structural, genomics…)

-Each sequence can only have one template of a given type

-Several sequences can share the same template

-All the sequences do not need to have a template

The type of template on which a method works is declared with the SEQ_TYPE parameter in the method configuration file:

SEQ_TYPE	S: a method that uses sequences
SEQ_TYPE	PS: a pairwise method that aligns sequences and structures
SEQ_TYPE	P: a method that aligns structures (sap for instance)

There are 3 tags identifying the template type:

_P_	Structural templates: a pdb identifier OR a pdb file
_G_	Genomic templates: a protein sequence where boundary amino-acid have been recoded with (o:0, i:1, j:2)
_R_	Profile Templates: a file containing a multiple sequence alignment

More than one template file can be provided. There is no need to have one template for every sequence in the dataset.

_P_, _G_, and _R_ are known as **template TAGS**

2-SCRIPT_<scriptname>

Indicates that filename is a script that will be used to generate a valid template file. The script will run on a file containing all your sequences using the following syntax:

```
scriptname –infile=<your sequences> -

outfile=<template_file>
```

It is also possible to pass some parameters, use @ as a separator (i.e. the @ will be turned into a space):

```
SCRIPT_myscript@-val1=val2@
```

<u>3-SELF_TAG</u>

TAG can take the value of any of the known TAGS (_S_, _G_, _P_). SELF indicates that the original name of the sequence will be used to fetch the template:

```
EXCL: t_coffee 3d_sample2.fasta –template_file SELF_P_
```

The previous command will work because the sequences in 3d_sample3 are named

<u>4-SEQFILE_TAG_filename</u>

Use this flag if your templates are in filename, and are named according to the sequences. For instance, if your protein sequences have been recoded with Exon/Intron information, you should have the recoded sequences names according to the original:

```
SEQFILE_G_recodedprotein.fasta
```

-struc_to_use

Usage: -struc_to_use=<struc1, struc2…>

`Default: -struc_to_use=NULL`

Restricts the 3Dcoffee to a set of pre-defined structures.

Multiple Local 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/-mocca

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.

```
EXCL: t_coffee -in Ssample_seq1.fasta,Mlalign_rs_s_pair -
out_lib=sample_lib1.mocca_lib -domain -start=10 -len=50
```

This instruction 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

```
EXCL: t_coffee -in sample_lib1.mocca_lib -domain -start=10 -len=60
```

This run will use the fragment 100-160, and will be much faster because it does not need to re-compute 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

Increase the scale ⇔Increase sensitivity ⇔ More alignments(i.e. -50).

-domain_interactive [Examples]

Usage: -domain_interactive

`Default: unset`

Launches an interactive mocca session.

```
EXCL: t_coffee -in Lsample_lib3.tc_lib,Mlalign_rs_s_pair -domain -
start=100 -len=60
```

```
TOLB_ECOLI_212_26                        211      SKLAYVTFESGR--SALVIQTLANGAVRQV-
ASFPRHNGAPAFSPDGSKLAFA
TOLB_ECOLI_165_218    164 TRIAYVVQTNGGQFPYELRVSDYDGYNQFVVHRSPQPLMSPAWSPDGSKLAYV
TOLB_ECOLI_256_306    255 SKLAFALSKTGS--LNLYVMDLASGQIRQV-TDGRSNNTEPTWFPDSQNLAFT
TOLB_ECOLI_307_350    306 -------DQAGR--PQVYKVNINGGAPQRI-TWEGSQNQDADVSSDGKFMVMV
TOLB_ECOLI_351_393    350 -------SNGGQ--QHIAKQDLATGGV-QV-LSSTFLDETPSLAPNGTMVIYS
                        1           *             *    :          .   .:.  :

        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.

Output Control

Generic

Conventions Regarding Filenames

stdout, *stderr*, *stdin, no, /dev/null* 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*.

Identifying the Output files automatically

In the t_coffee output, each output appears in a line:

```
##### FILENAME <name> TYPE <Type> FORMAT <Format>
```

Alignments

-outfile

Usage: -outfile=<out_aln file,default,no>

`Default:-outfile=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_ascii	: causes the output of a reliability flag
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:

```
EXCL: t_coffee sample_seq1.fasta -output=clustalw,gcg, score_html
```

A publication describing the CORE index is available on:
http://igs-server.cnrs-mrs.fr/~cnotred/Publications/Pdf/core.pp.pdf

-outseqweight

Usage: -outseqweight=<filename>

```
Default: not used
```

Indicates the name of the file in which the sequences weights should be saved..

-case

Usage: -case=<keep,upper,lower>

```
Default: -case=keep
```

Instructs the program on the case to be used in the output file (Clustalw uses upper case). The default keeps the case and makes it possible to maintain a mixture of upper and lower case residues.

If you need to change the case of your file, you can use seq_reformat:

```
EXCL: t_coffee -other_pg seq_reformat -in sample_aln1.aln -action
+lower -output clustalw
```

-cpu

Usage: deprecated

-outseqweight

Usage: -outseqweight=<name of the file containing the weights applied>

Default: -outseqweight=no

Will cause the program to output the weights associated with every sequence in the dataset.

-outorder [cw]

Usage: -outorder=<input OR aligned OR filename>

`Default:-outorder=input`

Sets the order of the sequences in the output alignment: **-outorder=input** means the sequences are kept in the original order. **-outorder=aligned** means the sequences come in the order indicated by the tree. This order can be seen as a one-dimensional projection of the tree distances. **–outdorder=<filename>**Filename is a legal fasta file, whose order will be used in the final alignment.

-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
```

Libraries

Although, it does not necessarily do so explicitly, T-Coffee always end up combining libraries. Libraries are collections of pairs of residues. Given a set of libraries, T-Coffee makes an attempt to assemble the alignment with the highest level of consistence. You can think of the alignment as a timetable. Each library pair would be a request from students or teachers, and the job of T-Coffee would be to assemble the time table that makes as many people as possible happy…

-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

-lib_only

Usage: -lib_only

```
Default: unset
```

Causes the program to stop once the library has been computed. Must be used in conjunction with the flag **–out_lib**

Trees

-newtree

Usage: -newtree=<tree file>

```
Default: No file specified
```

Indicates the name of the file into which the guide tree will be written. The default will be <sequence_name>.dnd, or <run_name.dnd>. The tree is written in the parenthesis format known as newick or New Hampshire and used by Phylips (see the format section).

Do NOT confuse this guide tree with a phylogenetic tree.

Reliability Estimation

CORE Computation

The CORE is an index that indicates the consistency between the library of piarwise alignments and the final multiple alignment. Our experiment indicate that the higher this consistency, the more reliable the alignment. A publication describing the CORE index can be found on:

http://igs-server.cnrs-mrs.fr/~cnotred/Publications/Pdf/core.pp.pdf

-evaluate_mode

Usage: -evaluate_mode=<t_coffee_fast,t_coffee_slow,t_coffee_non_extended >

```
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: Normalization is made using the highest score in the MSA. This evaluation mode was validated and in our hands, pairs of residues with a score of 5 or higher have 90 % chances to be correctly aligned to one another.

t_coffee_slow: Normalization is made using the library. This usually results in lower score and a scoring scheme more sensitive to the number of sequences in the dataset. Note that this scoring scheme is not any more slower, thanks to the implementation of a faster heuristic algorithm.

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.

These modes will be useful when generating colored version of the output, with the **–output** flag:

```
EXCL: t_coffee sample_seq1.fasta –evaluate_mode t_coffee_slow –
output score_ascii, score_html

EXCL: t_coffee sample_seq1.fasta –evaluate_mode t_coffee_fast –
```

```
output score_ascii, score_html

EXCL: t_coffee sample_seq1.fasta -evaluate_mode
t_coffee_non_extended -output score_ascii, score_html
```

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 output files.

-quiet

Usage: -quiet=<stderr,stdout,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 [CW]

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

Building a Server

We maintain a T-Coffee server (igs-server.cnrs-mrs.fr/Tcoffee/). We will be pleased to provide anyone who wants to set up a similar service with the sources

Common Problems when setting up servers

CACHE Directory

T-Coffee needs a cache directory where it stores temporary files, caches alignments and all sort of other messy things. For a normal user, this cache is ususally build in $HOME/.t_coffee/... Yet in the case of a server, such permissions may not be availale. You can therefore redirect the cache by setting the environement variable CACHE_4_TCOFFEE to some mpore suitable value in your scratch area.

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.

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. Five rules apply:

Naming Your Sequences the Right Way

1-No Space
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-No Strange Character
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-> is NEVER legal (except as a header token in a FASTA file)

4-Name length must be below 100 characters, although 15 is recommended for compatibility with other programs.
5-Duplicated sequences will be renamed (i.e. sequences with the same name in the same dataset) are allowed but will be renamed according to their original order.
When sequences come from multiple sources via the –in flag, consistency of the

renaming is not guaranteed. You should avoid duplicated sequences as they will cause your input to differ from your output thus making it difficult to track data.

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. T-Coffee uses extract_from_pdb (cf –other_pg flag). extract_from_pdb is a small embeded 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…).

Note: a different number of lines in the different blocks will cause the program to crash or hang.

Libraries

T-COFFEE_LIB_FORMAT_01

This is currently the only supported format.

```
!<space> TC_LIB_FORMAT_01
<nseq>
<seq1 name> <seq1 length> <seq1>
<seq2 name> <seq2 length> <seq2>
<seq3 name> <seq3 length> <seq3>
!Comment
(!Comment)n
#Si1 Si2
Ri1 Ri2 V1 (V2, V3)
#1 2
12 13 99 (12/0 vs 13/1, weight 99)
12 14 70
15 16 56
#1 3
12 13 99
12 14 70
15 16 56
!<space>SEQ_1_TO_N
```

Si1: index of Sequence 1

Ri1: index of residue 1 in seq1

V1: Integer Value: Weight

V2, V3: optional values

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 and residues are numbered from 1 to N, unless the token SEQ_1_TO_N is omitted, in which case the sequences are numbered from 0 to N-1, and residues are from 1 to N.

Residues do not need to be sorted, and neither do the sequences. The same pair can appear several times in the library. For instance, the following file would be legal:

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

T-COFFEE_LIB_FORMAT_02

A simpler format is being developed, however it is not yet fully supported and is only mentioned here for development purpose.

```
! TC_LIB_FORMAT_02
#S1 SEQ1 [OPTIONAL]
#S2 SEQ2 [OPTIONAL]
...
!comment [OPTIONAL]
S1 R1 Ri1 S2 R2 Ri2 V1 (V2 V3)
```

```
...
```

S1, S2: name of sequence 1 and 2

SEQ1: sequence of S1

Ri1, Ri2: index of the residues in their respective sequence

R1, R2: Residue type

V1, V2, V3: integer Values (V2 and V3 are optional)

Value1, Value 2 and Value3 are optional.

Library List

These are lists of pairs of sequences that must be used to compute a library. The format is:

```
<nseq> <S1> <S2>
2 hamg2 globav
3 hamgw hemog singa
...
```

Substitution matrices.

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

ClustalW Style [Deprecated]

```
$
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.

BLAST Format [Recommended]

```
# BLAST_MATRIX FORMAT\n
# ALPHABET=AGCT
 A G C T
A 0 1 2 3
G 0 2 3 4
C 1 1 2 3
...
```

The alphabet can be freely defined

Sequences Weights

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

```
# SINGLE_SEQ_WEIGHT_FORMAT_01
seq_name1 v1
```

```
seq_name2 v2
...
```

No duplicate allowed. Sequences not included in the set of sequences provided to t_coffee will be ignored. Order is free. V1 is a float. Un-weighted sequences will see their weight set to 1.

Known Problems

1-Sensitivity to sequence order: It is difficult to implement a MSA algorithm totally insensitive to the order of input of the sequences. In t_coffee, robustness is increased by sorting the sequences alphabetically before aligning them. Beware that this can result in confusing output where sequences with similar name are unexpectedly close to one another in the final alignment.

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

3-Stop codons are sometimes coded with '*' in protein sequences. This will cause the program to crash or hang. Please replace the '*' signs with an X.

4-Results can differ from one architecture to another, due rounding differences. This is caused by the tree estimation procedcure. If you want to make sure an alignment is reproducible, you should keep the associated dendrogram.

Technical Notes

These notes are only meant for internal development.

Development

The following examples are only meant for internal development, and are used to insure stability from release to release

PROFILE2LIST

prf1: profile containing one structure

prf2: profile containing one structure

```
EXCL: t_coffee  Rsample_profile1.aln,Rsample_profile2.aln -
special_mode=3dcoffee -outfile=aligned_prf.aln
```

Command Line List

These command lines have been checked before every release (along with the other CL in this documentation:

-external methods;

```
   EXCL: t_coffee sample_seq1.fasta –
in=Mclustalw_pair,Mclustalw_msa,Mslow_pair –outfile=clustal_text
```

-fugue_client

```
EXCL: t_coffee –in Ssample_seq5.fasta Pstruc4.pdb Mfugue_pair
```

To Do…

-implement UPGMA tree computation

-implement seq2dpa_tree

-debug dpa

-Reconciliate sequences and template when reading the template

-Add the server command lines to the checking procedure